**package** com.auth\_api.spring\_jwt\_auth.config;  
  
**import** com.auth\_api.spring\_jwt\_auth.filter.JwtFilter;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.context.annotation.Bean;  
**import** org.springframework.context.annotation.Configuration;  
**import** org.springframework.security.authentication.AuthenticationManager;  
**import** org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;  
**import** org.springframework.security.config.annotation.web.builders.HttpSecurity;  
**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
**import** org.springframework.security.config.http.SessionCreationPolicy;  
**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
**import** org.springframework.security.web.SecurityFilterChain;  
**import** org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;  
  
@Configuration  
@EnableWebSecurity  
**public class** SecurityConfig  
{  
 **private final** JwtFilter jwtFilter;  
  
 @Autowired  
 **public** SecurityConfig(JwtFilter jwtFilter){  
 **this**.jwtFilter = jwtFilter;  
 }  
  
 @Bean  
 **public** SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) **throws** Exception{  
  
// CSRF -> Cross-Site Request Forgery  
 httpSecurity  
 .csrf(customizer->customizer.disable())  
 .authorizeHttpRequests(  
 request -> request  
 .requestMatchers(  
 "/api/v1/auth/login",  
 "/api/v1/auth/register",  
 "/api/v1/auth/test",  
 "/api/v1/user/test"  
 ).permitAll() // try adding array here  
 .anyRequest().authenticated()  
 )  
 .sessionManagement(session->session.sessionCreationPolicy(SessionCreationPolicy.*STATELESS*))  
 .addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.**class**);  
  
  
 **return** httpSecurity.build();  
 }  
  
 @Bean  
 **public** BCryptPasswordEncoder bCryptPasswordEncoder(){  
 **return new** BCryptPasswordEncoder();  
 }  
  
 @Bean  
 **public** AuthenticationManager authenticationManager(AuthenticationConfiguration authenticationConfiguration) **throws** Exception {  
 **return** authenticationConfiguration.getAuthenticationManager();  
 }  
  
}

--------------------------------------------------------------------------------------------------------------------------------------

**package** com.auth\_api.spring\_jwt\_auth.constants;  
  
**public enum** Roles  
{  
 *ADMIN*,  
 *USER*}

**package** com.auth\_api.spring\_jwt\_auth.controller;  
  
**import** com.auth\_api.spring\_jwt\_auth.entity.User;  
**import** com.auth\_api.spring\_jwt\_auth.service.AuthService;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.web.bind.annotation.\*;  
  
@RestController  
@RequestMapping("/api/v1/auth")  
**public class** AuthController {  
  
 **private final** AuthService authService;  
  
 @Autowired  
 **public** AuthController(AuthService authService){  
 **this**.authService=authService;  
 }  
   
 @GetMapping("/test")  
 **public** String test(){  
 **return** "AuthController Endpoint test";  
 }  
 @PostMapping("/login")  
 **public** ResponseEntity<String> userLogin(@RequestBody User user){  
 **return** ResponseEntity.*ok*(authService.verifyUser(user));  
 }  
  
 @PostMapping("/register")  
 **public** ResponseEntity<User> userRegistration(@RequestBody User user){  
 **return** ResponseEntity.*ok*(authService.newUserRegistration(user));  
 }  
}

**package** com.auth\_api.spring\_jwt\_auth.controller;  
  
**import** com.auth\_api.spring\_jwt\_auth.entity.User;  
**import** com.auth\_api.spring\_jwt\_auth.service.UserService;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.web.bind.annotation.\*;  
  
**import** java.util.List;  
  
@RestController  
@RequestMapping("/api/v1/user")  
**public class** UserController {  
  
 **private final** UserService userService;  
 @Autowired  
 **public** UserController(UserService userService){  
 **this**.userService=userService;  
 }  
 @GetMapping("/test")  
 **public** String test(){  
 **return** "UserController Endpoint test";  
 }  
 @PatchMapping("/update/{id}")  
 **public** ResponseEntity<User> updateUser(@PathVariable **long** id, @RequestBody User user){  
 **return** ResponseEntity.*ok*(userService.updateUser(id,user));  
 }  
  
 @DeleteMapping("/delete/{id}")  
 **public void** deleteUser(@PathVariable **long** id){  
 userService.deleteUser(id);  
 }  
  
 @GetMapping("/all-users")  
 **public** ResponseEntity<List<User>> getAllUsers(){  
 **return** ResponseEntity.*ok*(userService.getAllUsers());  
 }  
  
 @GetMapping("/id/{id}")  
 **public** ResponseEntity<User> getUserByUserId(@PathVariable **long** id){  
 **return** ResponseEntity.*ok*(userService.getUserByUserId(id));  
 }  
  
 @GetMapping("/username/{username}")  
 **public** ResponseEntity<User> getUserByUsername(@PathVariable String username){  
 **return** ResponseEntity.*ok*(userService.getUserByUsername(username));  
 }  
}

**package** com.auth\_api.spring\_jwt\_auth.dao;  
**import** com.auth\_api.spring\_jwt\_auth.entity.User;  
**import** com.auth\_api.spring\_jwt\_auth.repo.UserRepo;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.stereotype.Component;  
  
**import** java.util.List;  
**import** java.util.Optional;  
@Component  
**public class** UserDao {  
 **private final** UserRepo userRepo;  
 @Autowired  
 **public** UserDao(UserRepo userRepo){  
 **this**.userRepo=userRepo;  
 }  
 **public** User save(User user) {  
 **return** userRepo.save(user);  
 }  
 **public void** delete(**long** id) {  
 Optional<User> userOptional = userRepo.findById(id);  
 userRepo.delete(userOptional.get());  
 }  
 **public** List<User> retrieveAllUsers() {  
 **return** userRepo.findAll();  
 }  
 **public** User retrieveUserById(**long** id) {  
 Optional<User> userOptional = userRepo.findById(id);  
 **return** userOptional.get();  
 }  
 **public** User retrieveUserByUsername(String username) {  
 Optional<User> userOptional = userRepo.findByUsername(username);  
 **return** userOptional.get();  
 }  
}

**package** com.auth\_api.spring\_jwt\_auth.entity;  
  
**import** jakarta.persistence.\*;  
  
**import** java.util.List;  
  
@Entity  
@Table(name = "User")  
**public class** User  
{  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 @Column(name = "Id")  
 **private long** id;  
  
 @Column(name = "first\_name")  
 **private** String firstName;  
  
 @Column(name = "last\_name")  
 **private** String lastName;  
  
 @Column(name = "username")  
 **private** String username;  
  
 @Column(name = "password")  
 **private** String password;  
  
 @Column(name = "role")  
 **private** String userRole;  
  
 **public** User(){}  
  
 **public** User(String firstName, String lastName, String username, String password){  
 **this**.firstName=firstName;  
 **this**.lastName=lastName;  
 **this**.username=username;  
 **this**.password=password;  
 }  
  
 **public long** getId() {  
 **return** id;  
 }  
  
 **public void** setId(**long** id) {  
 **this**.id = id;  
 }  
  
 **public** String getFirstName() {  
 **return** firstName;  
 }  
  
 **public void** setFirstName(String firstName) {  
 **this**.firstName = firstName;  
 }  
  
 **public** String getLastName() {  
 **return** lastName;  
 }  
  
 **public void** setLastName(String lastName) {  
 **this**.lastName = lastName;  
 }  
  
 **public** String getUsername() {  
 **return** username;  
 }  
  
 **public void** setUsername(String username) {  
 **this**.username = username;  
 }  
  
 **public** String getPassword() {  
 **return** password;  
 }  
  
 **public void** setPassword(String password) {  
 **this**.password = password;  
 }  
  
 **public** String getUserRole() {  
 **return** userRole;  
 }  
  
 **public void** setUserRole(String userRole) {  
 **this**.userRole = userRole;  
 }  
  
 @Override  
 **public** String toString() {  
 **return** "User{" +  
 "id=" + id +  
 ", firstName='" + firstName + '\'' +  
 ", lastName='" + lastName + '\'' +  
 ", username='" + username + '\'' +  
 ", password='" + password + '\'' +  
 ", userRole='" + userRole + '\'' +  
 '}';  
 }  
}

**package** com.auth\_api.spring\_jwt\_auth.filter;  
  
**import** com.auth\_api.spring\_jwt\_auth.jwt.JwtService;  
**import** com.auth\_api.spring\_jwt\_auth.serviceImpl.CustomUserDetailsService;  
**import** jakarta.servlet.FilterChain;  
**import** jakarta.servlet.ServletException;  
**import** jakarta.servlet.http.HttpServletRequest;  
**import** jakarta.servlet.http.HttpServletResponse;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.context.ApplicationContext;  
**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
**import** org.springframework.security.core.context.SecurityContextHolder;  
**import** org.springframework.security.core.userdetails.UserDetails;  
**import** org.springframework.security.core.userdetails.UserDetailsService;  
**import** org.springframework.security.web.authentication.WebAuthenticationDetailsSource;  
**import** org.springframework.stereotype.Component;  
**import** org.springframework.web.filter.OncePerRequestFilter;  
  
**import** java.io.IOException;  
  
@Component  
**public class** JwtFilter **extends** OncePerRequestFilter {  
  
 **private final** JwtService jwtService;  
 **private final** UserDetailsService userDetailsService;  
  
 @Autowired  
 **public** JwtFilter(JwtService jwtService, UserDetailsService userDetailsService) {  
 **this**.jwtService = jwtService;  
 **this**.userDetailsService = userDetailsService;  
 }  
  
 @Override  
 **protected void** doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) **throws** ServletException, IOException {  
 String authHeader = request.getHeader("Authorization");  
 String token = **null**;  
 String username = **null**;  
  
 **if** (authHeader != **null** && authHeader.startsWith("Bearer ")) {  
 **try** {  
 token = authHeader.substring(7);  
 username = jwtService.extractUsernameFromToken(token);  
 } **catch** (Exception e) {  
 logger.warn("Token extraction failed: " + e.getMessage());  
 }  
 }  
  
 **if** (username != **null** && SecurityContextHolder.*getContext*().getAuthentication() == **null**) {  
 UserDetails userDetails = userDetailsService.loadUserByUsername(username);  
  
 **if** (jwtService.validateToken(token, userDetails)) {  
 UsernamePasswordAuthenticationToken authenticationToken =  
 **new** UsernamePasswordAuthenticationToken(userDetails, **null**, userDetails.getAuthorities());  
  
 authenticationToken.setDetails(**new** WebAuthenticationDetailsSource().buildDetails(request));  
 SecurityContextHolder.getContext().setAuthentication(authenticationToken);  
 }  
 }  
  
 filterChain.doFilter(request, response);  
 }  
}

**package** com.auth\_api.spring\_jwt\_auth.jwt;  
  
**import** io.jsonwebtoken.Claims;  
**import** io.jsonwebtoken.Jwts;  
**import** io.jsonwebtoken.security.Keys;  
**import** org.springframework.security.core.userdetails.UserDetails;  
**import** org.springframework.stereotype.Service;  
  
**import** javax.crypto.KeyGenerator;  
**import** javax.crypto.SecretKey;  
**import** java.security.NoSuchAlgorithmException;  
**import** java.util.\*;  
**import** java.util.function.Function;  
  
@Service  
**public class** JwtService {  
 **private final** String secretKey;  
  
 **public** JwtService() {  
 **try** {  
 KeyGenerator keyGen = KeyGenerator.*getInstance*("HmacSHA256");  
 SecretKey key = keyGen.generateKey();  
 secretKey = Base64.*getEncoder*().encodeToString(key.getEncoded());  
 } **catch** (NoSuchAlgorithmException exception) {  
 **throw new** RuntimeException(exception.getMessage());  
 }  
 }  
  
 **public** String generateToken(String username) {  
 Map<String, Objects> claims = **new** HashMap<>();  
  
 **return** Jwts.*builder*()  
 .setClaims(claims)  
 .setSubject(username)  
 .setIssuedAt(**new** Date(System.*currentTimeMillis*()))  
 .setExpiration(**new** Date(System.*currentTimeMillis*() + 1000 \* 60 \* 30))  
 .signWith(getKey())  
 .compact();  
 }  
  
  
 **public** SecretKey getKey() {  
 **byte**[] keyBytes = Base64.*getDecoder*().decode(secretKey);  
 **return** Keys.*hmacShaKeyFor*(keyBytes);  
 }  
  
 **public** String extractUsernameFromToken(String token) {  
 **return** extractClaim(token, Claims::getSubject);  
 }  
  
 **public boolean** validateToken(String token, UserDetails userDetails) {  
 **final** String username = extractUsernameFromToken(token);  
// return (userDetails.getUsername().equals(username)&& !isTokenExpired(token));  
 **return** (username.equals(userDetails.getUsername()) && !isTokenExpired(token));  
 }  
  
 **private** <T> T extractClaim(String token, Function<Claims, T> claimResolver) {  
 **final** Claims claims = extractAllClaims(token);  
 **return** claimResolver.apply(claims);  
 }  
  
 **private** Claims extractAllClaims(String token) {  
 **return** Jwts.*parser*()  
 .setSigningKey(getKey())  
 .build()  
 .parseClaimsJws(token)  
 .getBody();  
 }  
  
 **private boolean** isTokenExpired(String token) {  
 **return** extractTokenExpiration(token).before(**new** Date());  
 }  
  
 **private** Date extractTokenExpiration(String token) {  
 **return** extractClaim(token, Claims::getExpiration);  
 }  
}

**package** com.auth\_api.spring\_jwt\_auth.repo;  
  
**import** com.auth\_api.spring\_jwt\_auth.entity.User;  
**import** org.springframework.data.jpa.repository.JpaRepository;  
**import** org.springframework.stereotype.Repository;  
  
**import** java.util.Optional;  
  
@Repository  
**public interface** UserRepo **extends** JpaRepository<User,Long> {  
 Optional<User> findByUsername(String username);  
}

**package** com.auth\_api.spring\_jwt\_auth.service;  
  
**import** com.auth\_api.spring\_jwt\_auth.entity.User;  
  
**public interface** AuthService  
{  
 User newUserRegistration(User user);  
  
// User userLogin(UserLogin userLogin);  
  
 String verifyUser(User user);  
}

**package** com.auth\_api.spring\_jwt\_auth.service;  
  
**import** com.auth\_api.spring\_jwt\_auth.entity.User;  
**import** org.springframework.http.ResponseEntity;  
  
**import** java.util.List;  
  
**public interface** UserService  
{  
 User saveUser(User user);  
 User updateUser(**long** id,User user);  
  
 **void** deleteUser(**long** id);  
  
 List<User> getAllUsers();  
  
 User getUserByUserId(**long** id);  
  
 User getUserByUsername(String username);  
}

**package** com.auth\_api.spring\_jwt\_auth.serviceImpl;  
  
**import** com.auth\_api.spring\_jwt\_auth.constants.Roles;  
**import** com.auth\_api.spring\_jwt\_auth.dao.UserDao;  
**import** com.auth\_api.spring\_jwt\_auth.entity.User;  
**import** com.auth\_api.spring\_jwt\_auth.jwt.JwtService;  
**import** com.auth\_api.spring\_jwt\_auth.service.AuthService;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.security.authentication.AuthenticationManager;  
**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
**import** org.springframework.security.core.Authentication;  
**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
**import** org.springframework.stereotype.Service;  
  
@Service  
**public class** AuthServiceImpl **implements** AuthService  
{  
 **private** UserDao userDao;  
 **private** BCryptPasswordEncoder passwordEncoder;  
  
 **private** AuthenticationManager authenticationManager;  
  
 **private** JwtService jwtService;  
 @Autowired  
 **public** AuthServiceImpl(  
 UserDao userDao,  
 BCryptPasswordEncoder passwordEncoder,  
 JwtService jwtService,  
 AuthenticationManager authenticationManager  
 )  
 {  
 **this**.userDao=userDao;  
 **this**.passwordEncoder=passwordEncoder;  
 **this**.jwtService=jwtService;  
 **this**.authenticationManager=authenticationManager;  
 }  
  
 @Override  
 **public** User newUserRegistration(User user){  
 user.setUserRole(Roles.*USER*.name());  
 user.setPassword(passwordEncoder.encode(user.getPassword()));  
 **return** userDao.save(user);  
 }  
  
  
 @Override  
 **public** String verifyUser(User user) {  
  
// check if the user is authenticated or not  
 Authentication authentication = authenticationManager.authenticate(**new** UsernamePasswordAuthenticationToken(user.getUsername(),user.getPassword()));  
  
// check if authentication is successful  
  
 **if**(authentication.isAuthenticated()){  
 String token = jwtService.generateToken(user.getUsername());  
 **return** token;  
 }  
  
 **return** "fail";  
 }  
}

**package** com.auth\_api.spring\_jwt\_auth.serviceImpl;  
  
**import** com.auth\_api.spring\_jwt\_auth.entity.User;  
**import** org.springframework.security.core.GrantedAuthority;  
**import** org.springframework.security.core.authority.SimpleGrantedAuthority;  
**import** org.springframework.security.core.userdetails.UserDetails;  
  
**import** java.util.Arrays;  
**import** java.util.Collection;  
  
**public class** CustomUserDetails **implements** UserDetails {  
  
 **private** User user;  
  
 **public** CustomUserDetails(User user){  
 **this**.user=user;  
 }  
  
 @Override  
 **public** Collection<? **extends** GrantedAuthority> getAuthorities() {  
 GrantedAuthority authority = **new** SimpleGrantedAuthority(user.getUserRole());  
 **return** Arrays.*asList*(authority);  
 }  
  
 @Override  
 **public** String getPassword() {  
 **return** user.getPassword();  
 }  
  
 @Override  
 **public** String getUsername() {  
 **return** user.getUsername();  
 }  
  
 @Override  
 **public boolean** isAccountNonExpired() {  
 **return true**;  
 }  
  
 @Override  
 **public boolean** isAccountNonLocked() {  
 **return true**;  
 }  
  
 @Override  
 **public boolean** isCredentialsNonExpired() {  
 **return true**;  
 }  
  
 @Override  
 **public boolean** isEnabled() {  
 **return true**;  
 }  
}

**package** com.auth\_api.spring\_jwt\_auth.serviceImpl;  
  
**import** com.auth\_api.spring\_jwt\_auth.dao.UserDao;  
**import** com.auth\_api.spring\_jwt\_auth.entity.User;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.security.core.userdetails.UserDetails;  
**import** org.springframework.security.core.userdetails.UserDetailsService;  
**import** org.springframework.security.core.userdetails.UsernameNotFoundException;  
**import** org.springframework.stereotype.Component;  
**import** org.springframework.stereotype.Service;  
  
@Service  
**public class** CustomUserDetailsService **implements** UserDetailsService {  
 **private** UserDao userDao;  
  
 **public** CustomUserDetailsService(){}  
 @Autowired  
 **public** CustomUserDetailsService(UserDao userDao){  
 **this**.userDao=userDao;  
 }  
  
 @Override  
 **public** UserDetails loadUserByUsername(String username) **throws** UsernameNotFoundException {  
 User user = userDao.retrieveUserByUsername(username);  
 **return new** CustomUserDetails(user);  
 }  
}

**package** com.auth\_api.spring\_jwt\_auth.serviceImpl;  
  
**import** com.auth\_api.spring\_jwt\_auth.dao.UserDao;  
**import** com.auth\_api.spring\_jwt\_auth.entity.User;  
**import** com.auth\_api.spring\_jwt\_auth.service.UserService;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.stereotype.Service;  
  
**import** java.util.List;  
  
@Service  
**public class** UserServiceImpl **implements** UserService {  
 **private** UserDao userDao;  
 **private** User dbUser;  
  
 @Autowired  
 **public** UserServiceImpl(UserDao userDao){  
 **this**.userDao =userDao;  
 }  
  
 **public** User saveUser(User user){  
 **return** userDao.save(user);  
 }  
  
 **public** User updateUser(**long** id,User user){  
 dbUser = userDao.retrieveUserById(id);  
  
 **if**(user.getFirstName()!=**null**){  
 dbUser.setFirstName(user.getFirstName());  
 }  
  
 **if**(user.getLastName()!=**null**){  
 dbUser.setLastName(user.getLastName());  
 }  
  
 **if**(user.getUsername()!=**null**){  
 dbUser.setUsername(user.getUsername());  
 }  
  
 **if**(user.getPassword()!=**null**){  
 dbUser.setPassword(user.getPassword());  
 }  
  
 **if**(user.getUserRole()!=**null**){  
 dbUser.setUserRole(user.getUserRole());  
 }  
  
 **return** userDao.save(dbUser);  
 }  
  
 **public void** deleteUser(**long** id){  
 userDao.delete(id); }  
  
 **public** List<User> getAllUsers(){  
 **return** userDao.retrieveAllUsers();  
 }  
  
 **public** User getUserByUserId(**long** id){  
 **return** userDao.retrieveUserById(id);  
 }  
  
 **public** User getUserByUsername(String username){  
 **return** userDao.retrieveUserByUsername(username);  
 }  
}

**package** com.auth\_api.spring\_jwt\_auth.util;  
  
**public record** UserLogin(String username, String password) {  
}

**package** com.auth\_api.spring\_jwt\_auth;  
  
**import** org.springframework.boot.SpringApplication;  
**import** org.springframework.boot.autoconfigure.SpringBootApplication;  
**import** org.springframework.cloud.client.discovery.EnableDiscoveryClient;  
  
@SpringBootApplication  
@EnableDiscoveryClient  
**public class** SpringJwtAuthApplication {  
  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(SpringJwtAuthApplication.**class**, args);  
 }  
  
}

spring.application.name=auth-service  
  
#Server Port  
server.port=0  
  
#MySQL Configuration  
spring.datasource.url=jdbc:mysql://localhost:3306/spring\_security\_jwt\_auth\_api?createDatabaseIfNotExist=true  
spring.datasource.username=root  
spring.datasource.password=root  
  
#JPA/Hibernate Configuration  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect  
  
#Eureka Client Config  
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/  
eureka.client.hostname=localhost

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.jwt\_auth\_microservices</groupId>  
 <artifactId>jwt\_auth\_microservices</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 </parent>  
  
 <groupId>org.auth\_service</groupId>  
 <artifactId>auth\_service</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 </dependency>  
  
  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt-api</artifactId>  
 <version>0.12.5</version>  
 </dependency>  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt-impl</artifactId>  
 <version>0.12.5</version>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt-jackson</artifactId>  
 <version>0.12.5</version> <!-- or jjwt-gson for Gson support -->  
 <scope>runtime</scope>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
 </dependency>  
  
 </dependencies>  
  
</project>