API Security Basic:

```
package com.example.apisecurity;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.ProviderManager;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.Customizer;
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.configurers.AbstractHttpConfigurer;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class SpringSecurityConfig {
    private final CustomUserDetailsService userDetailsService;

    public SpringSecurityConfig(CustomUserDetailsService userDetailsService) {
        this.userDetailsService = userDetailsService;
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

//    @Bean
//    public UserDetailsService userDetailsService(PasswordEncoder passwordEncoder) {
//        UserDetails adminUser = User.withUsername("admin")
//                .password(passwordEncoder.encode("admin"))
//                .roles("ADMIN")
//                .build();
//        UserDetails managerUser = User.withUsername("manager")
//                .password(passwordEncoder.encode("manager"))
//                .roles("MANAGER")
//                .build();
//
//        return new InMemoryUserDetailsManager(adminUser, managerUser);
//    }
```

```
    @Bean
    public AuthenticationManager authenticationManager() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(userDetailsService);
        authProvider.setPasswordEncoder(passwordEncoder());
        return new ProviderManager(authProvider);
    }
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
                .csrf(AbstractHttpConfigurer::disable)
                .authorizeHttpRequests(authorizationManagerRequestMatcherRegistry ->
                        authorizationManagerRequestMatcherRegistry.requestMatchers(HttpMethod.POST,
"/save").permitAll()
                                .requestMatchers(HttpMethod.GET, "/admin/**").hasRole("ADMIN")
                                .requestMatchers(HttpMethod.GET, "/manager/**").hasRole("MANAGER")
                                .anyRequest().authenticated()
                )
                .httpBasic(Customizer.withDefaults())
                .sessionManagement(session->
session.sessionCreationPolicy(SessionCreationPolicy.STATELESS));
        return http.build();
    }
}
```

```
package com.example.apisecurity;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.User;
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.Service;


@Service
public class CustomUserDetailsService implements UserDetailsService {
    @Autowired
    private UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        UserEntity user = userRepository.findByUsername(username)
                .orElseThrow(() -> new UsernameNotFoundException("User not found"));
        return User.builder()
                .username(username)
                .password(user.getPassword())
```

```
                .roles(user.getRole()).build();
    }


}
```

```
package com.example.apisecurity;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

@Entity
@Table(name = "users")
public class UserEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, nullable = false)
    private String username;

    @Column(nullable = false)
    private String password;

    @Column(nullable = false)
    private String role; // Example: "ROLE_USER" or "ROLE_ADMIN"

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    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 getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }
}
```