Заштита на веб апликации со Spring Security

Веб програмирање

Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Каде треба да се заштитат веб апликациите

- Заштита на мрежниот слој
  - Предефинирани IP адреси
  - Филтрирање на мрежни пакети со користење на firewall и листи за контрола на пристап
- Заштита на транспортен слој
  - Заштита на порти (http://nmap.org/ е алатка која овозможува да се прегледаат достапните порти на некој сервер)
    - https://shodan.io
    - Firewall кој ги ограничува достапните порти
  - Secure Sockets Layer (SSL)
    - **HTTPS**
- Заштита на ниво на оперативен систем
  - Доделување на минимални привилегии за функционирање на апликацијата
  - Изолација на апликациите (кога има повеќе апликации на ист сервер)
    - Контејнеризација
- **Заштита на апликациски слој**
  - Автентикација
  - Авторизација

Што треба да заштитиме?

- **Интегритет** - способноста да се осигураме дека информациите кои се пренесуваат до и примаат од веб апликацијата преку Интернет не се променети на кој било начин од неовластено лице.
- **Автентичност** - способноста да го идентификуваме идентитетот на лицето или субјектот кој пристапува на веб апликацијата.
- **Доверливост** - способноста да се осигураме дека пораките и податоците се достапни само за оние кои се овластени да ги гледаат. Приватност - способноста да се контролира употребата на личните информации.
- **Достапност** - способноста да се контролира кога може да се пристапи апликацијата.

Најчести напади

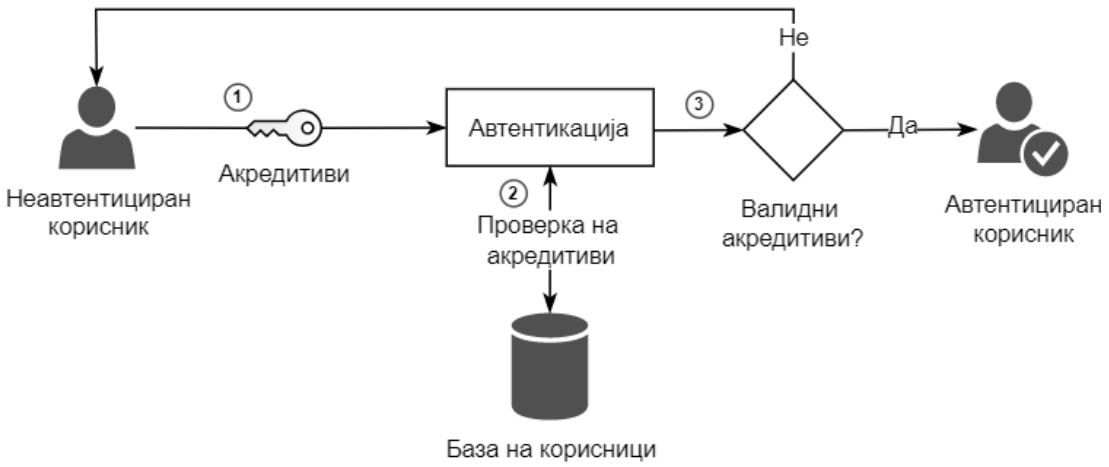
- **Вметнување на SQL**
  - Се нарушуваат **интегритетот** и **приватноста** на податоците во рамките на апликацијата.
  - Врши модификација на **комуникацијата со базата на податоци**, при што може да се променат податоците во неа, или пак да се добијат приватни информации кои не се дозволени за пристап за напаѓачот.
  - https://www.youtube.com/watch?v=_jKylhJtPmI
- **Cross-site scripting**
  - Се нарушуваат **интегритетот** и **приватноста** на податоците
  - Овој напад вметнува извршен код како податок, кој доколку се прикаже на друг корисник, може неовластено да пристапи до неговите податоци и да ги украде (нарушување на приватноста) или да ги промени (нарушување на интегритетот).
  - https://www.youtube.com/watch?v=L5l9lSnNMxg
- **Фалсификување барања меѓу страници (Cross-Site Request Forgery – CSRF)**
  - Се нарушува **автентикацијата** и **доверливоста** на податоците
  - Напаѓачот ќе ги искористи податоците кои се локално зачувани кај жртвата (колачиња за автентикација и авторизација) за да изврши акции без негово знаење
  - https://www.youtube.com/watch?v=vRBihr41JTo

Поими

- **Корисник (User)**
  - Концепт преку кој ги идентификуваме легитимните агенти на кои треба да им дозволиме пристап до системот
- **Акредитиви (Credentials)**
  - Начинот на кој корисникот може да докаже дека тој е оној кој што вели дека е
- **Улога (Role)**
  - Групирање на корисници со цел да се олесни процесот на доделување на дозволи за пристап до ресурсите
- **Ресурси (Resource)**
  - Она што сакаме да го заштитиме
- **Дозволи (Permissions)**
  - Дефинираат **кои ресурси** се дозволени или забранети за одредени **улоги** или **корисници**

Како да ги заштитиме веб апликациите?

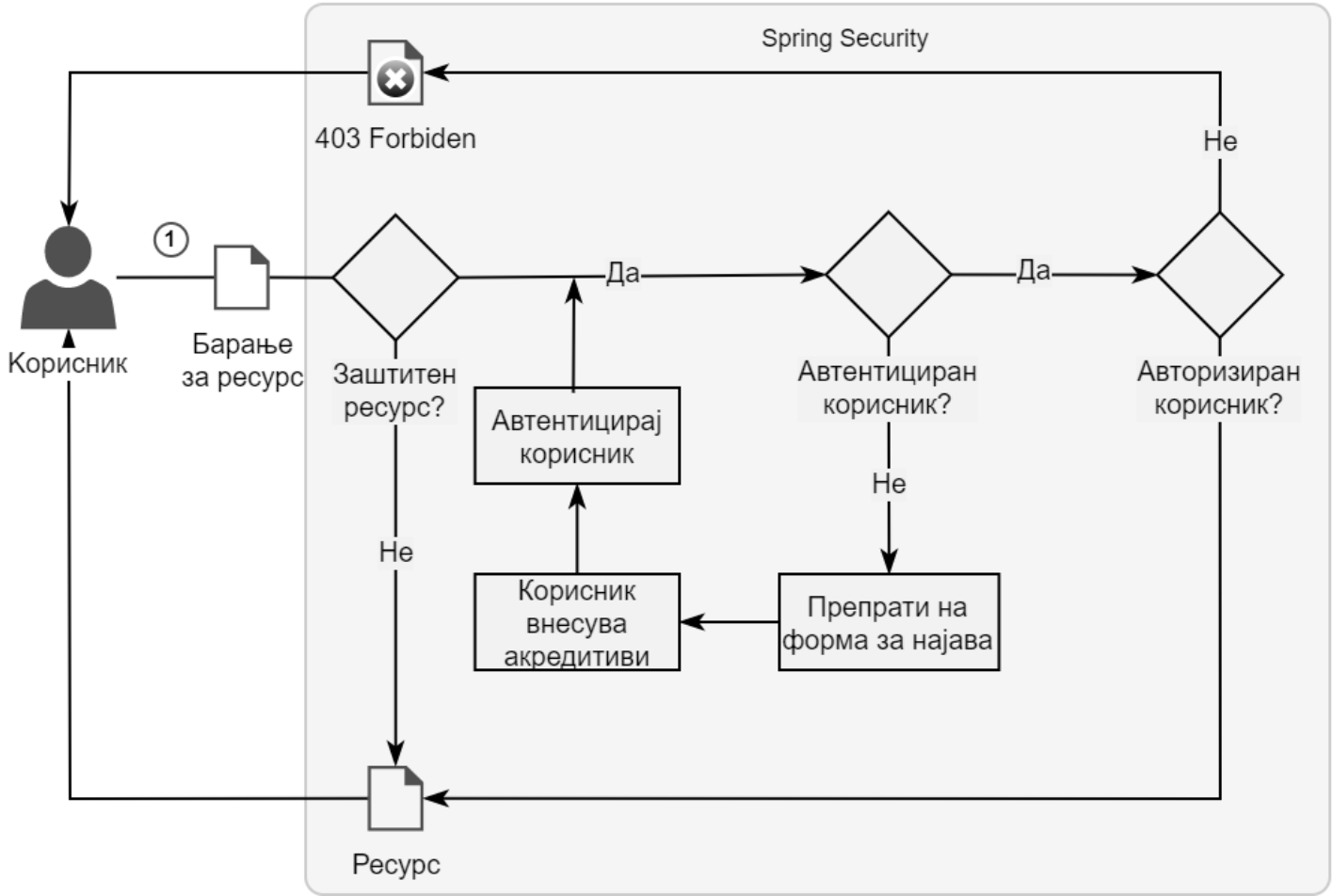
- **Автентикација**
  - Корисникот кажува кој е, односно му се претставува на системот
- **Авторизација**
  - Контролираме кој корисник какви интеракции има со ресурсите
- **Екнрипција**
  - Процес преку кој сензитивните информации ги правиме неразбирливи за напаѓачите



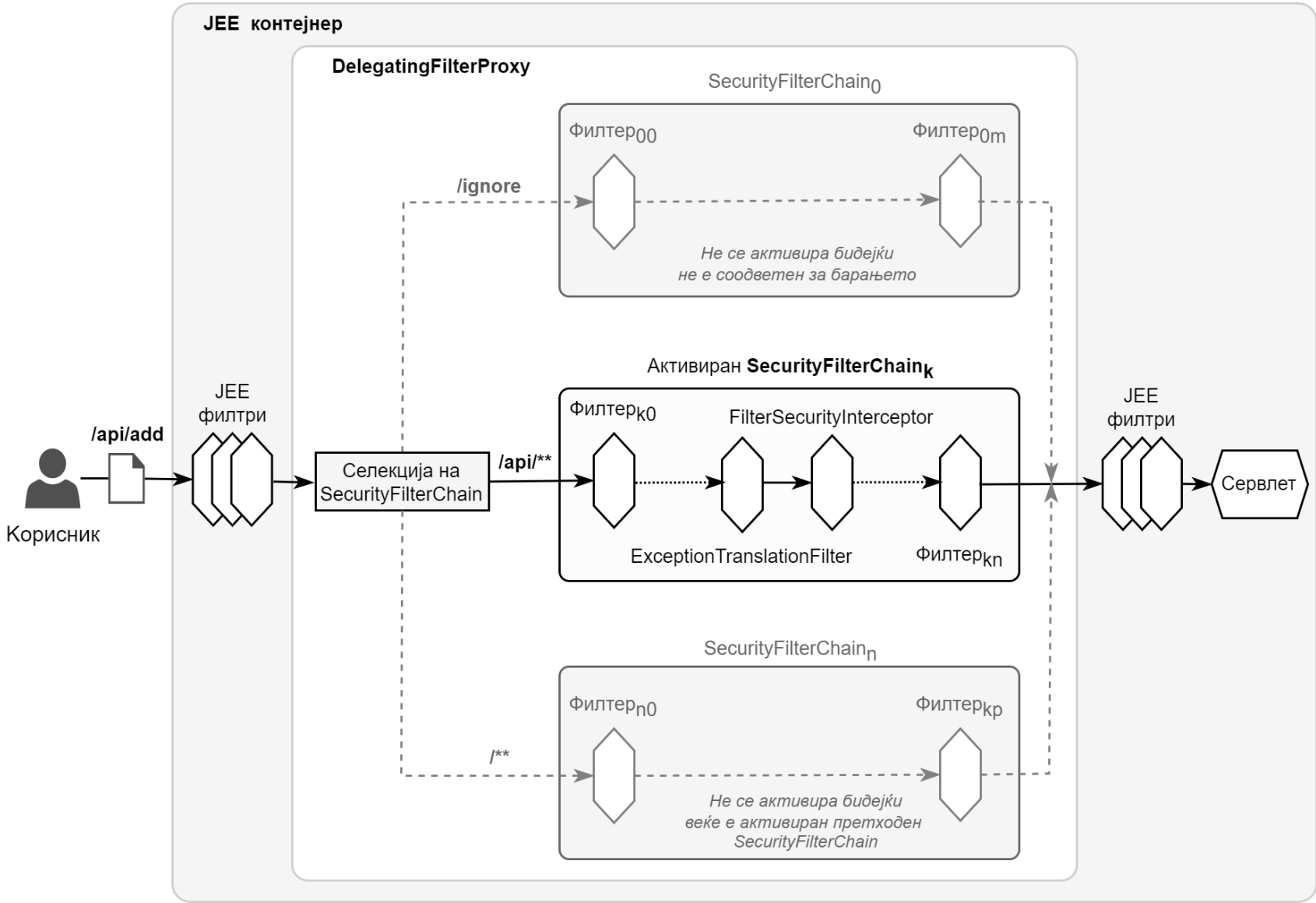
Заштита на веб апликации со користење на Spring Security

- Дополнителен слој изграден на околу веб апликациите, кој ги обвиткува специфичните влезни точки во бизнис логиката со одредени безбедносни правила.
- Конфигурабилни правила за авторизација на корисниците
- Заштита од начјестите напади
- Едноставни конфигурации за начинот на автентикација
  - Користи филтри со единствена одговорност за секоја функција

Преглед на заштита со Spring Security



Синџир со сигурносни филтри



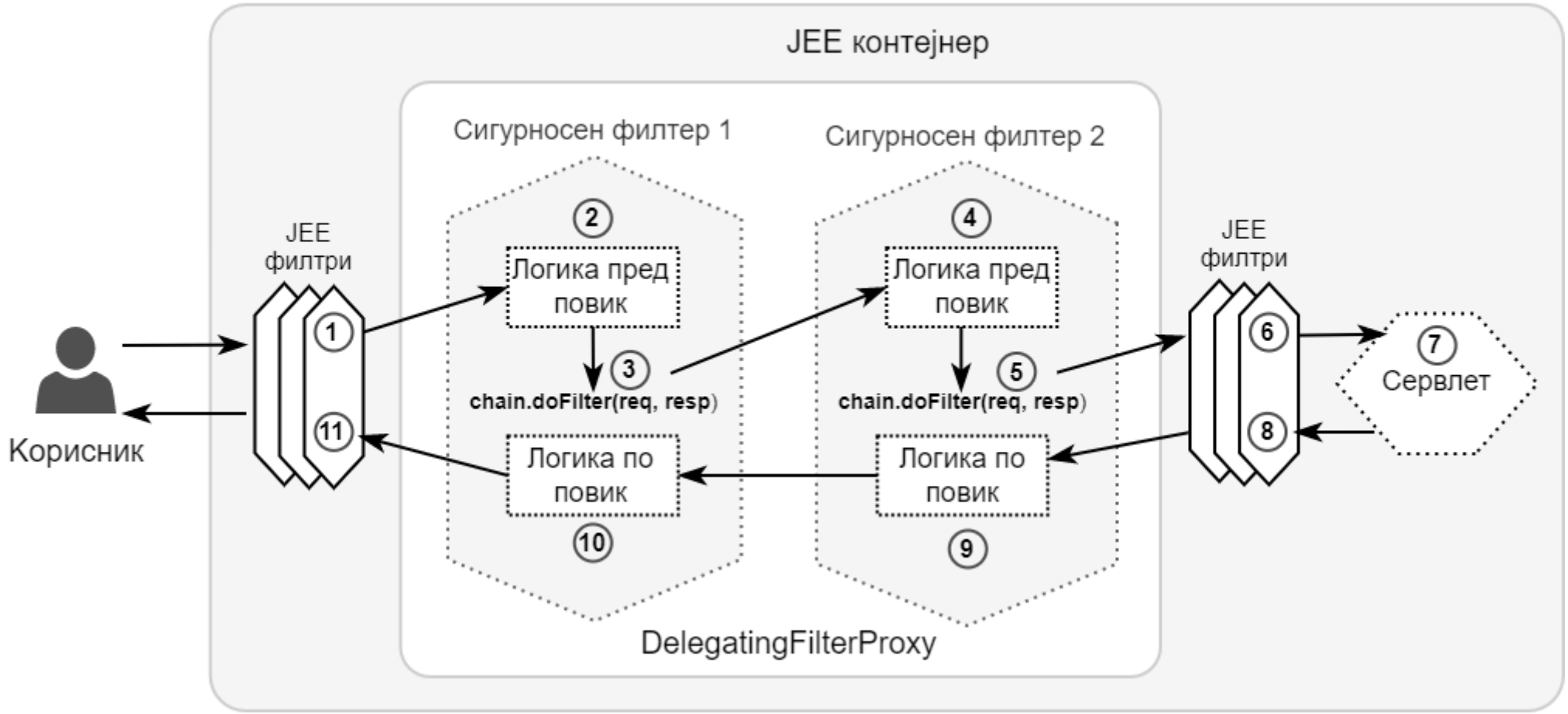
Код за вклучување на синџир со филтри за патеката /api/**

```
@Order(BASIC_AUTH_ORDER-10)
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http.securityMatcher("/api/**")
    // the rest of the configration
    return http.build();
}
```

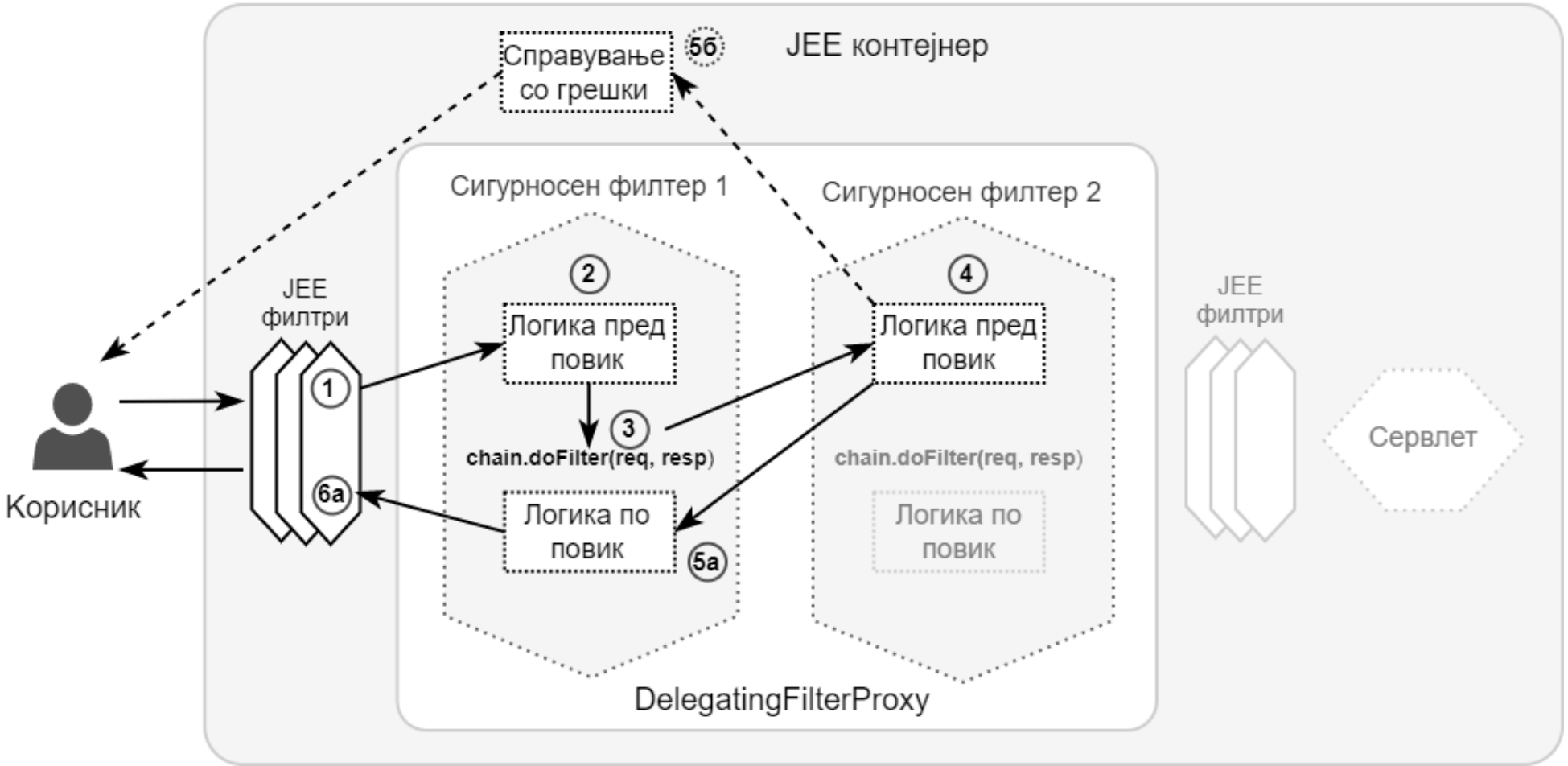
Филтри за заштита (Security Filters)

- **Помошни филтри**
  - Овие филтри се грижат за вчитување или зачувување на податоците кои се потребни во процесите на автентикација или авторизација.
- **Филтри за пресретнување на напади**
  - Овие филтри се грижат за превенирање на Cross Site Request Forgery и Cross Origin Request Sourcing нападите преку посебни филтри и за зголемување на сигурноста на апликацијата со додавање на сигурностни заглавја.
- **Филтри за автентикација**
  - Овие филтри се грижат за извршување на процесот на автентикација
- **Филтри за авторизација**
  - Претставник на оваа група е FilterSecurityInterceptor филтерот, кој проверува ресурсот е дозволен соодветниот корисник.

Повикување на сигурносните филтри во стандардно сценарио



Повикување на сигурносните филтри при исклучок или неисполнување на безбедносни услови

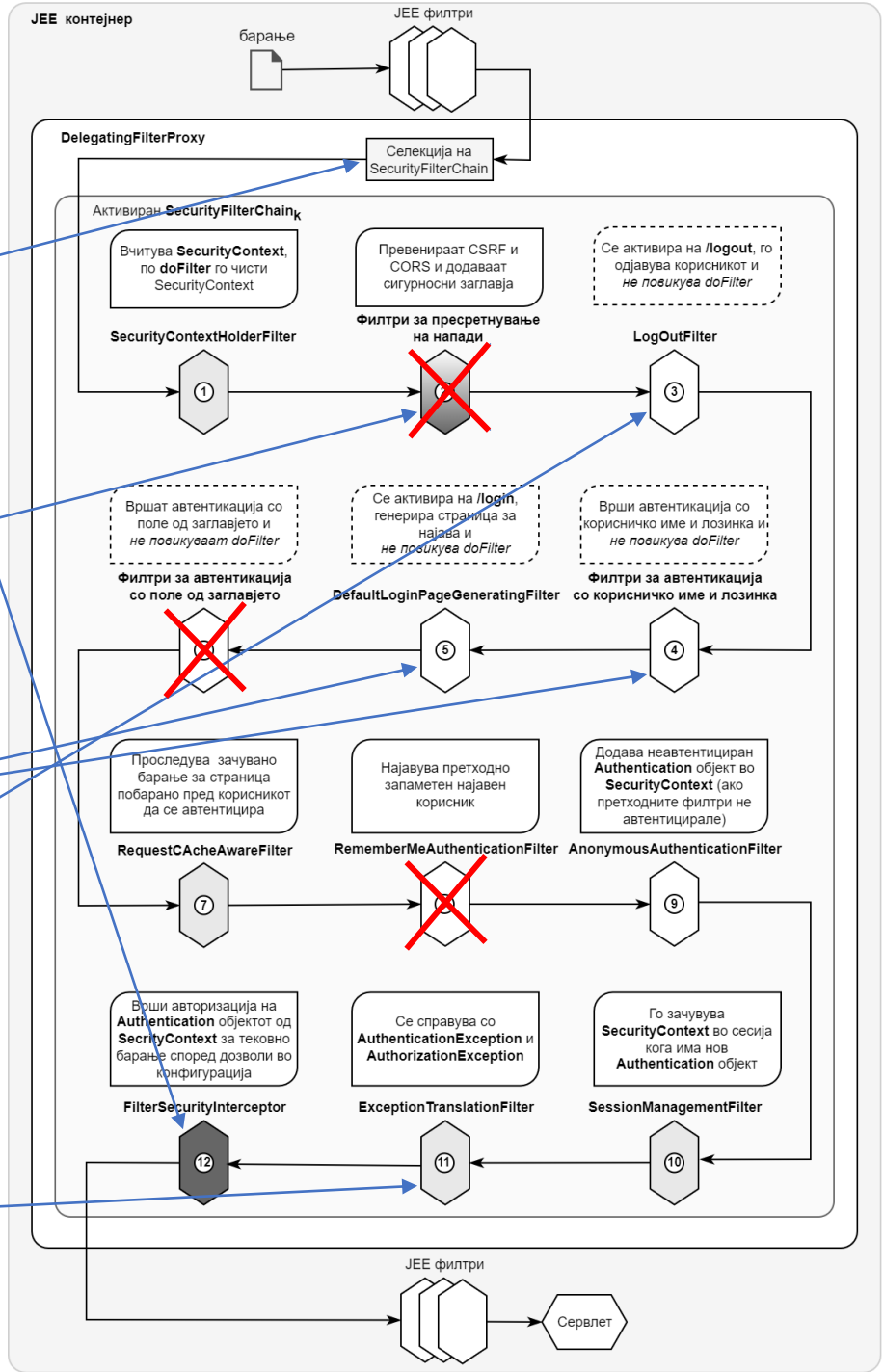


Конфигурирање на SecurityFilterChain

```java
@Configuration
public class WebSecurityConfig {
    @Bean
    public SecurityFilterChain configure(HttpSecurity http) throws Exception {
        http.securityMatcher("/**")
                .authorizeHttpRequests((requests) -> requests
                        .requestMatchers("/", "/home", "/register", "/products").permitAll()
                        .requestMatchers("/admin/**").hasRole("ADMIN")
                        .anyRequest().authenticated()
                )
                .cors(AbstractHttpConfigurer::disable)
                .formLogin(login -> login
                        .loginProcessingUrl("/do-login")
                        .usernameParameter("email")
                        .passwordParameter("secret")
                        .failureUrl( authenticationFailureUrl: "/login?error=BadCredentials")
                        .defaultSuccessUrl( defaultSuccessUrl: "/home",  alwaysUse: false)
                )
                .logout(logout -> logout
                        .logoutUrl("/logout")
                        .clearAuthentication(true)
                        .invalidateHttpSession(true)
                        .deleteCookies( ...cookieNamesToClear: "JSESSIONID")
                        .logoutSuccessUrl("/login")
                )
                .exceptionHandling(handling -> handling
                        .accessDeniedPage( accessDeniedUrl: "/access-denied")
                );
        return http.build();
    }
}
```

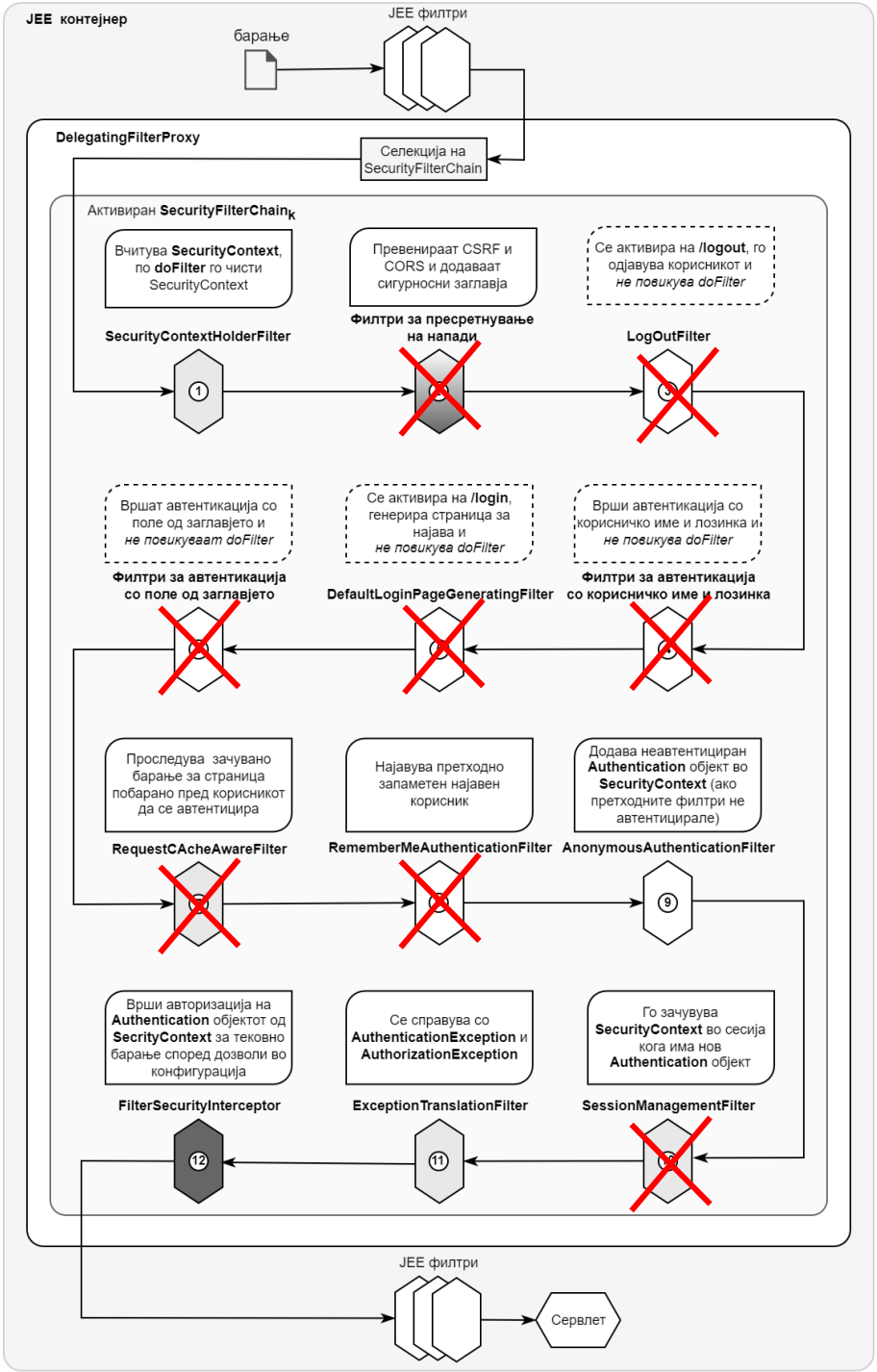
Преглед на сигурносните филтри од Spring Security модулот

```java
@Configuration
public class WebSecurityConfig {
    @Bean
    public SecurityFilterChain configure(HttpSecurity http) throws Exception {
        http.securityMatcher("/**")
                .authorizeHttpRequests((requests) -> requests
                        .requestMatchers("/", "/home", "/register", "/products").permitAll()
                        .requestMatchers("/admin/**").hasRole("ADMIN")
                        .anyRequest().authenticated()
                )
                .cors(AbstractHttpConfigurer::disable)
                .formLogin(login -> login
                        .loginProcessingUrl("/do-login")
                        .usernameParameter("email")
                        .passwordParameter("secret")
                        .failureUrl( authenticationFailureUrl: "/login?error=BadCredentials")
                        .defaultSuccessUrl( defaultSuccessUrl: "/home",  alwaysUse: false)
                )
                .logout(logout -> logout
                        .logoutUrl("/logout")
                        .clearAuthentication(true)
                        .invalidateHttpSession(true)
                        .deleteCookies( ...cookieNamesToClear: "JSESSIONID")
                        .logoutSuccessUrl("/login")
                )
                .exceptionHandling(handling -> handling
                        .accessDeniedPage( accessDeniedUrl: "/access-denied")
                );
        return http.build();
    }
}
```



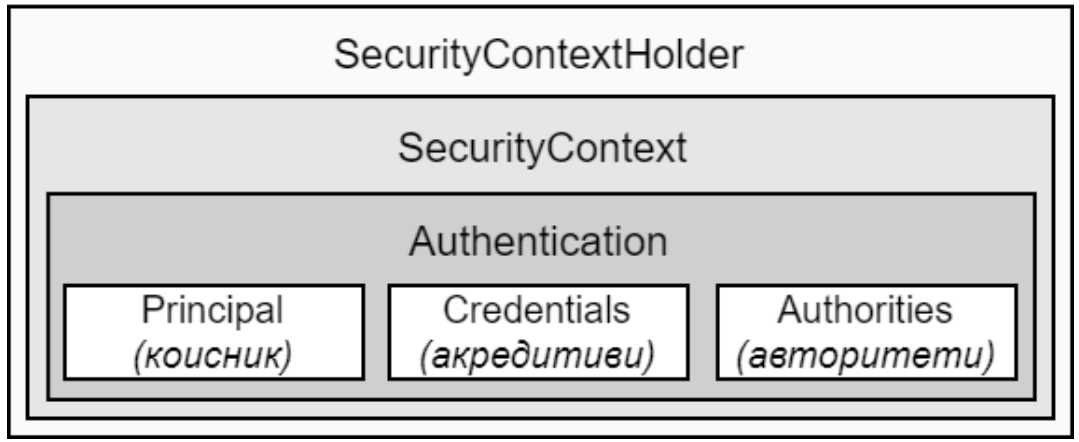
/admin – Активни филтри

- 2, 6, 8
  - Оневозможени од конфигурација
- 3, 4, 5
  - Не се совпаѓа патеката за активирање
- 7 - RequestCacheAwareFilter
  - Нема зачувано барање
- 10 – SessionManagementFilter
  - AnonymousAuthentication <=> нема автентикација
- Активни филтри
  - 1, 9, 12 и 11
    - **11** има само логика за извршување по процесирањето **на chain.doFilter()**

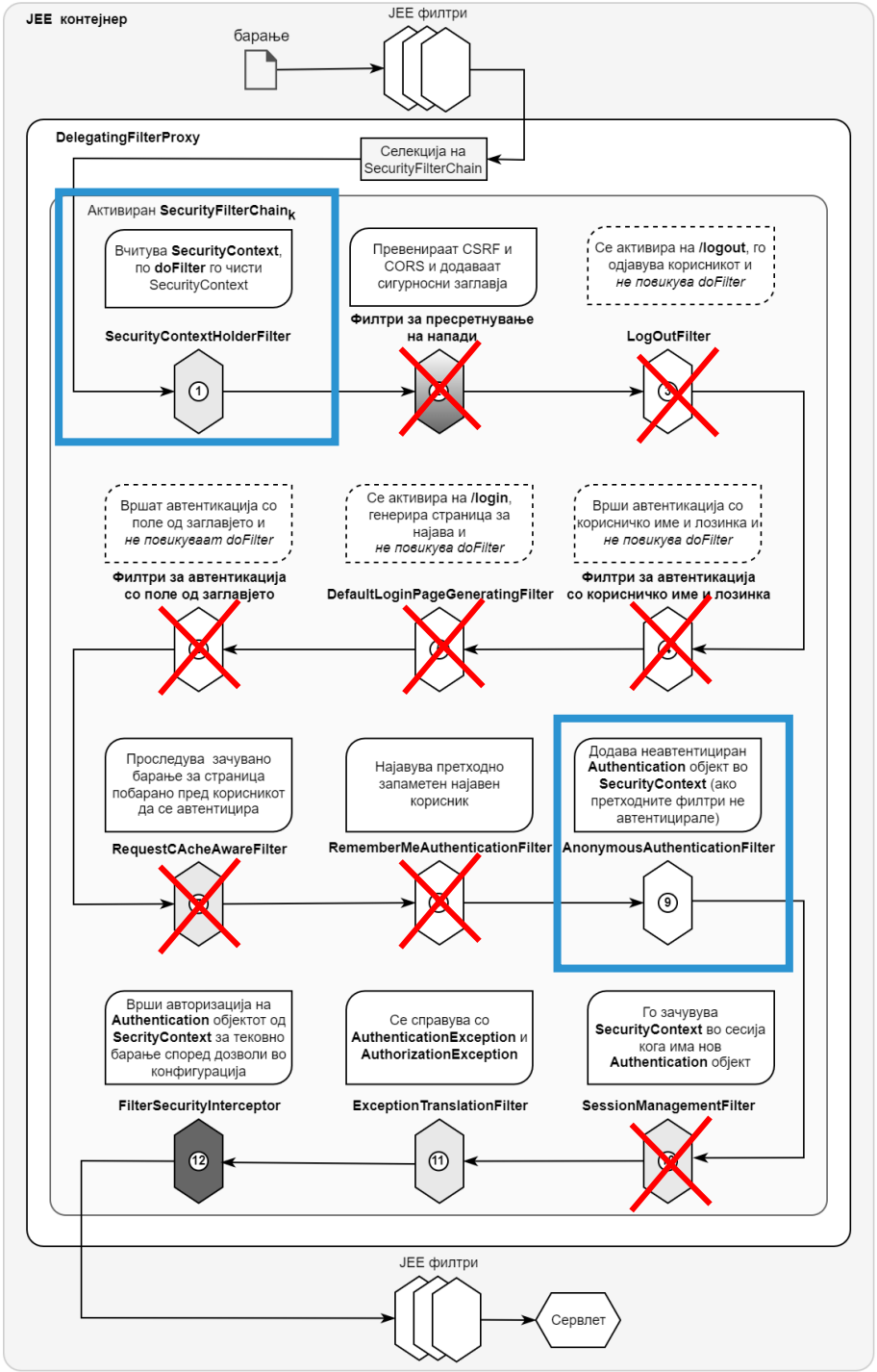


/admin – Процесирање 1, 9

- 1 – SecurityContextHolderFilter
  - Го иницијализира SecurityContextHolder
  - Предефинирано од Session

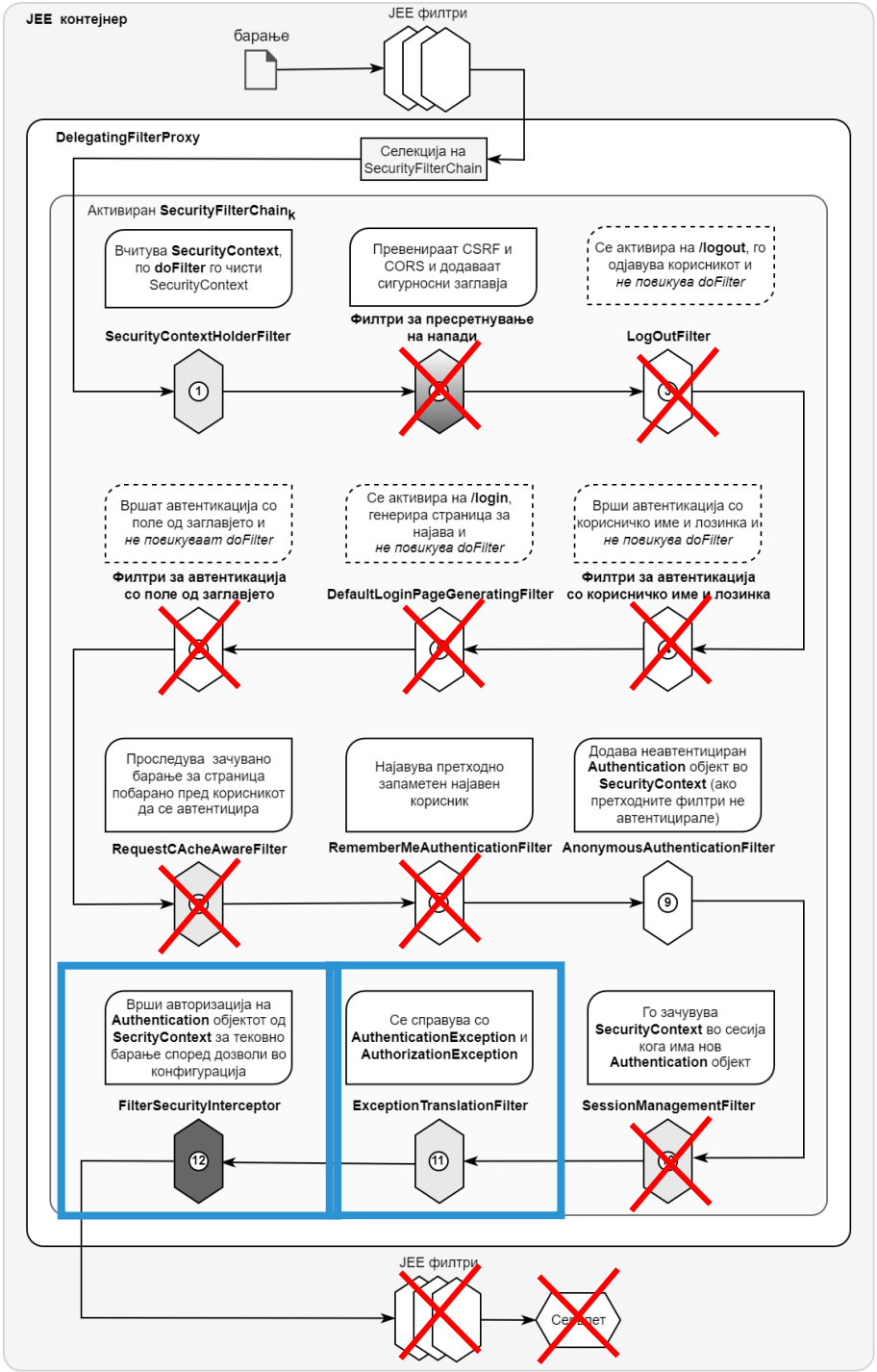


- 9 – AnonimousAuthenticationFilter
  - Креира Authentication објект кој означува дека имаме анонимен корисник

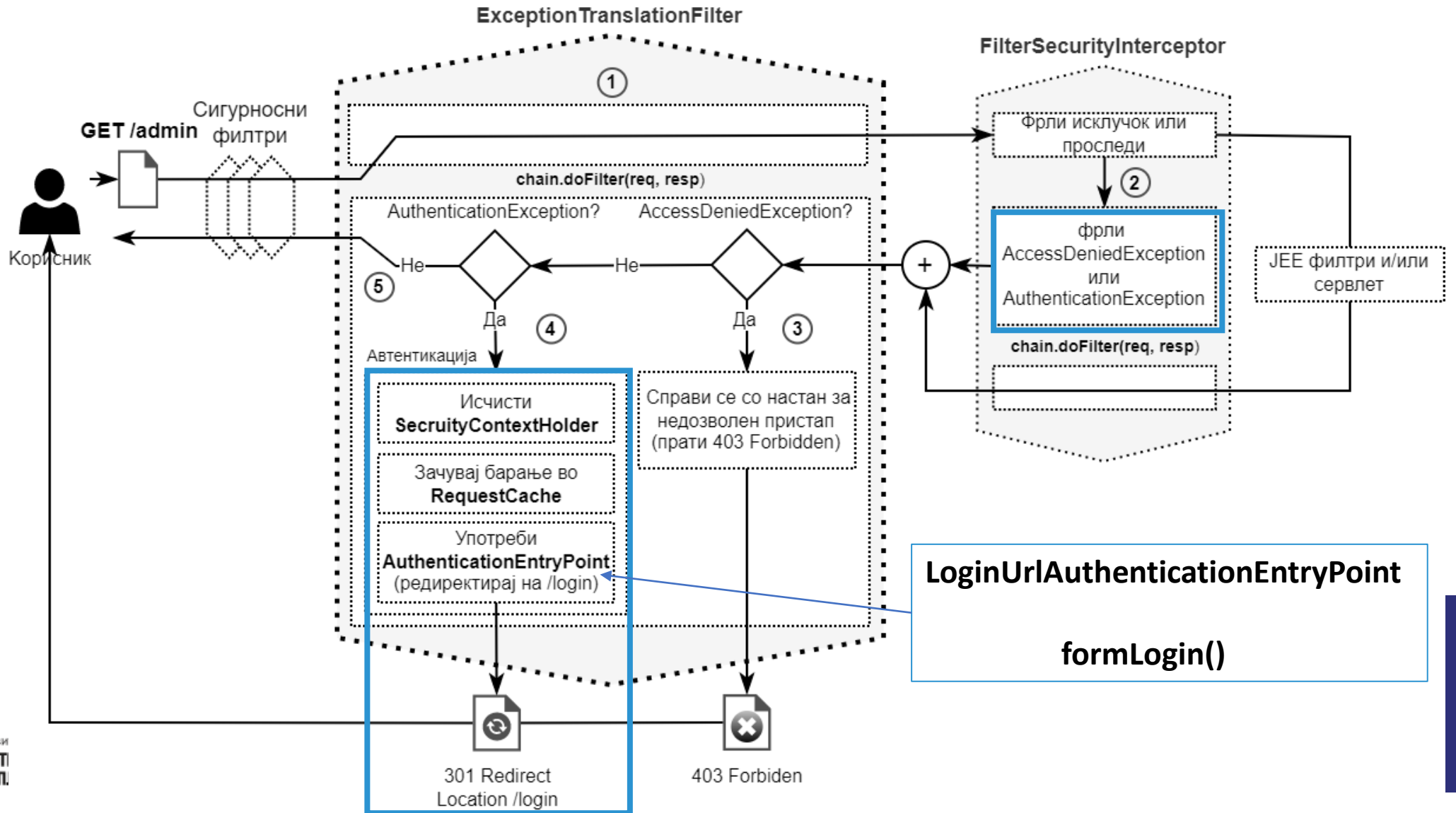


/admin – Процесирање 12 и 11

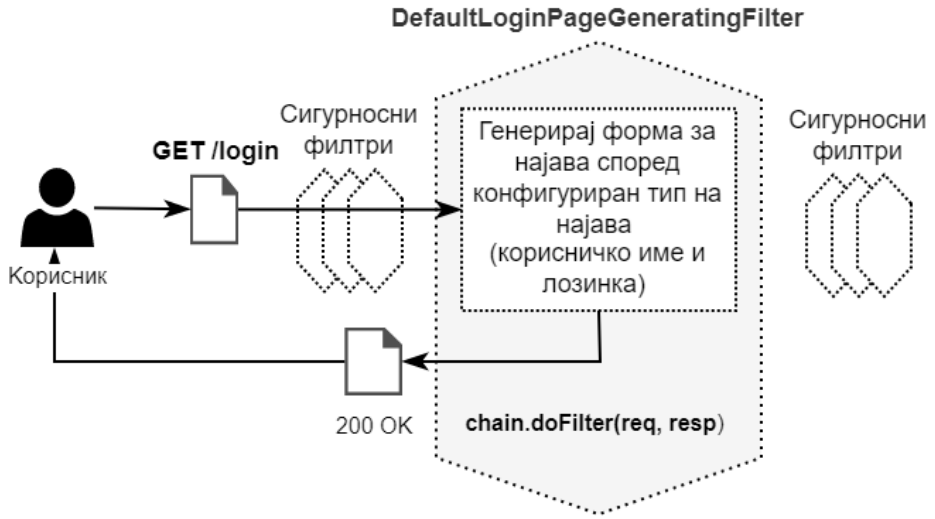
- 12 – **FilterSecurityInterceptor**
  - Проверува дали тековниот корисник има привилегии да пристапи до ресурси кои ги бара
  - за патеката ***/admin***, ќе се бара автентициран корисник со улога ***ADMIN***
  - Во сигурносниот контекст има **анонимен корисник** => **AuthenticationException**
- 11 – **ExceptionTranslationFilter**
  - Нема логика пред извршување на методот chain.doFilter()
  - Негова логика се извршува доколку некој од нредните филтри генерира исклучок
    - **AuthenticationException** и **AccessDeniedException** ги преведува во HTTP одговори

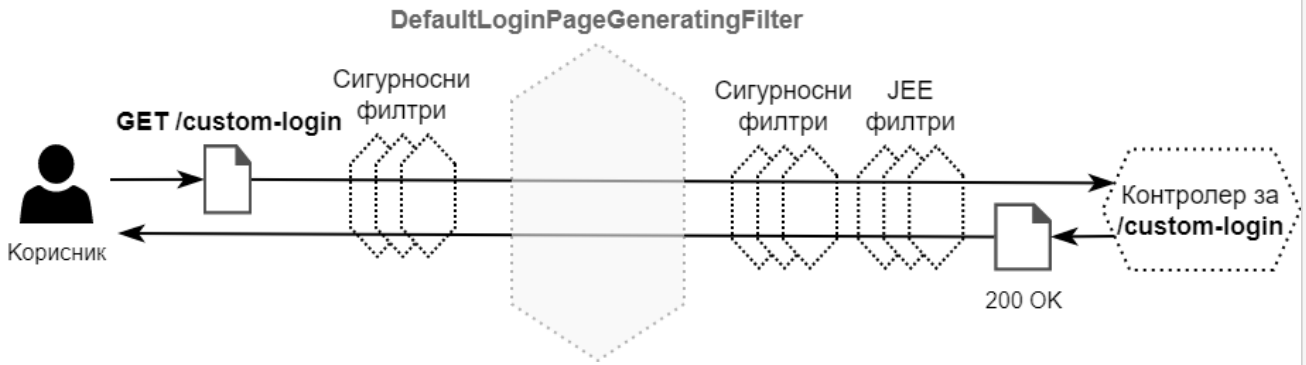


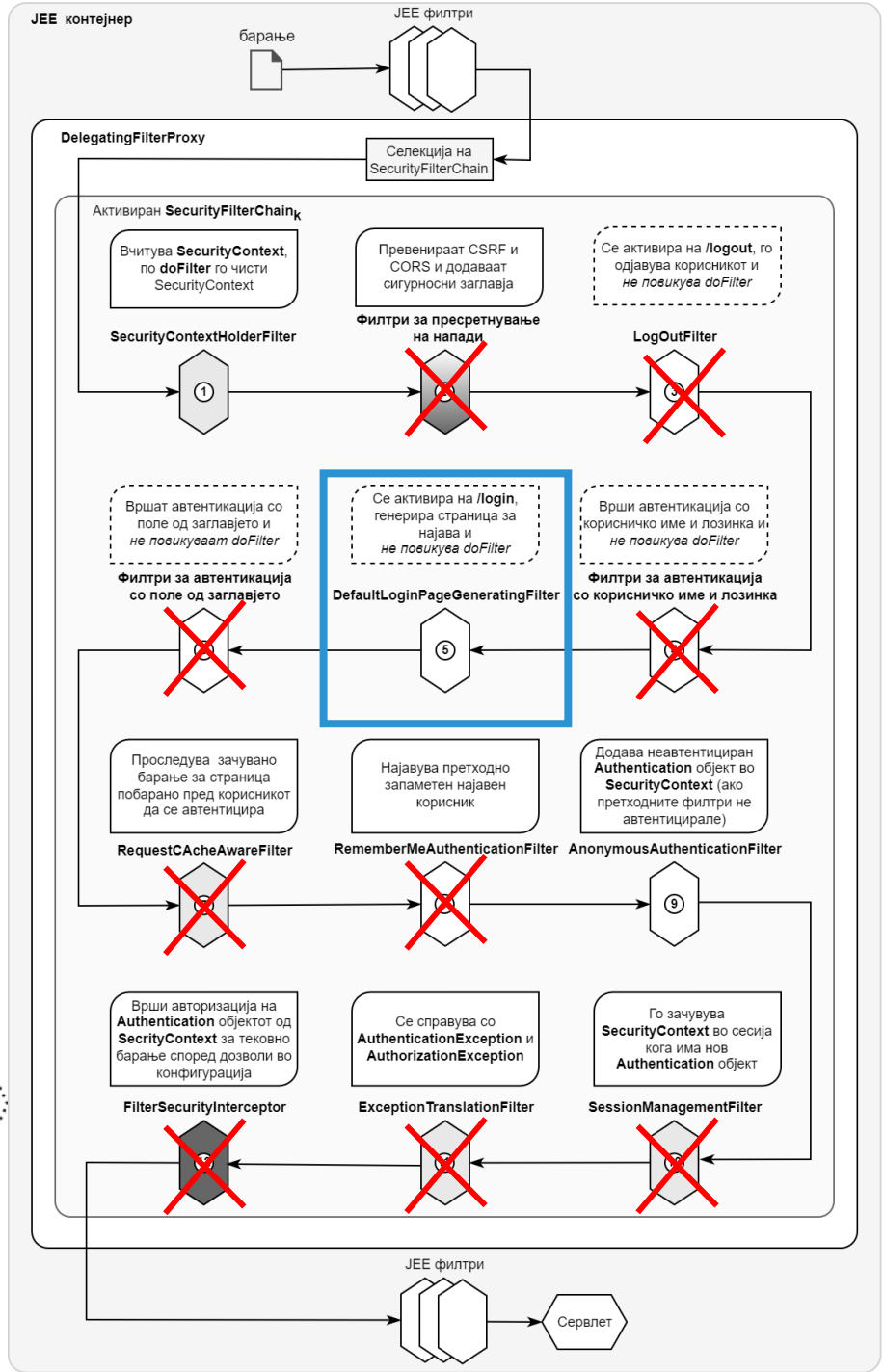
FilterSecurityInterceptor & ExceptionTranslationFilter



Редирекција кон /login

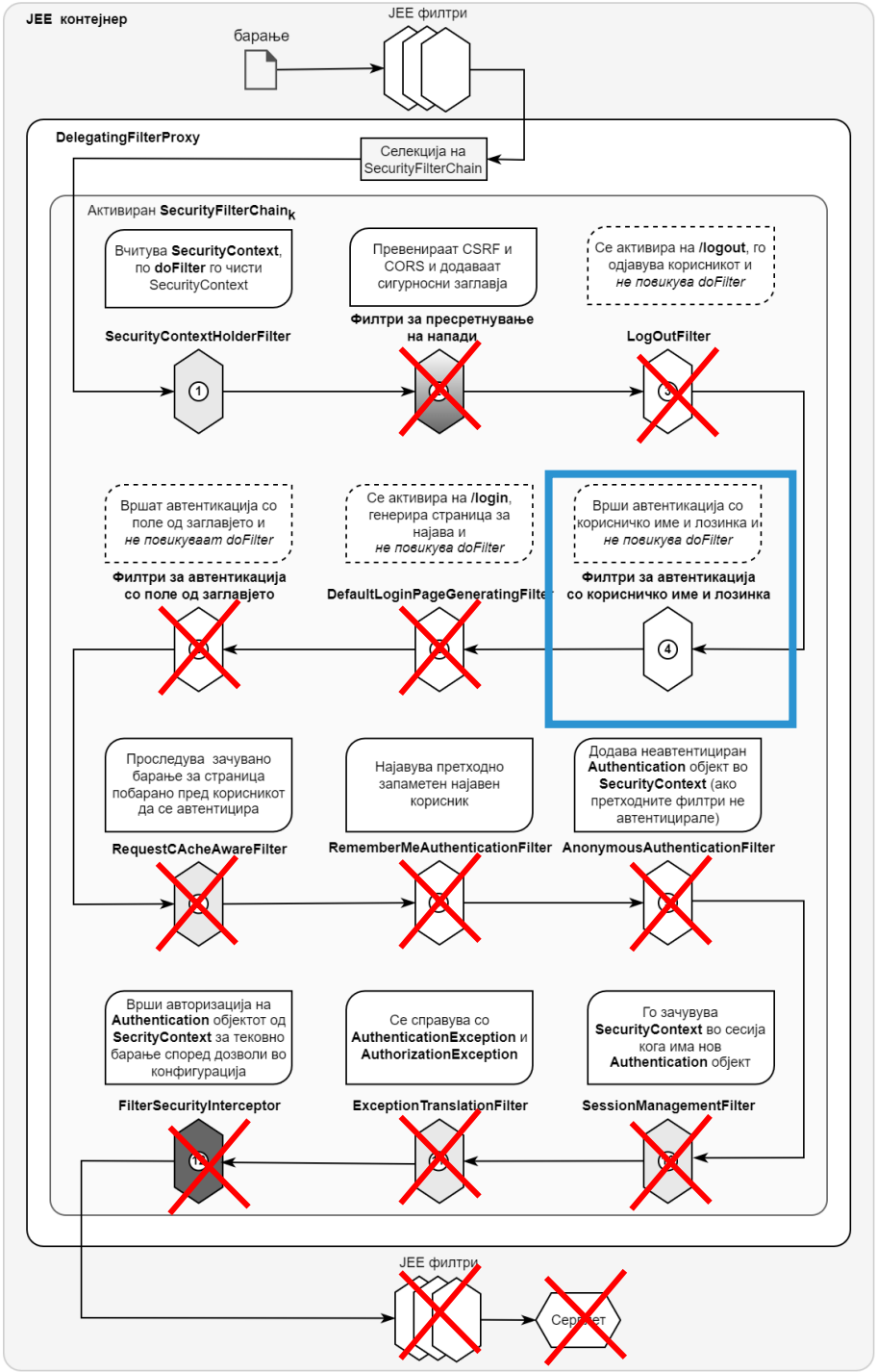




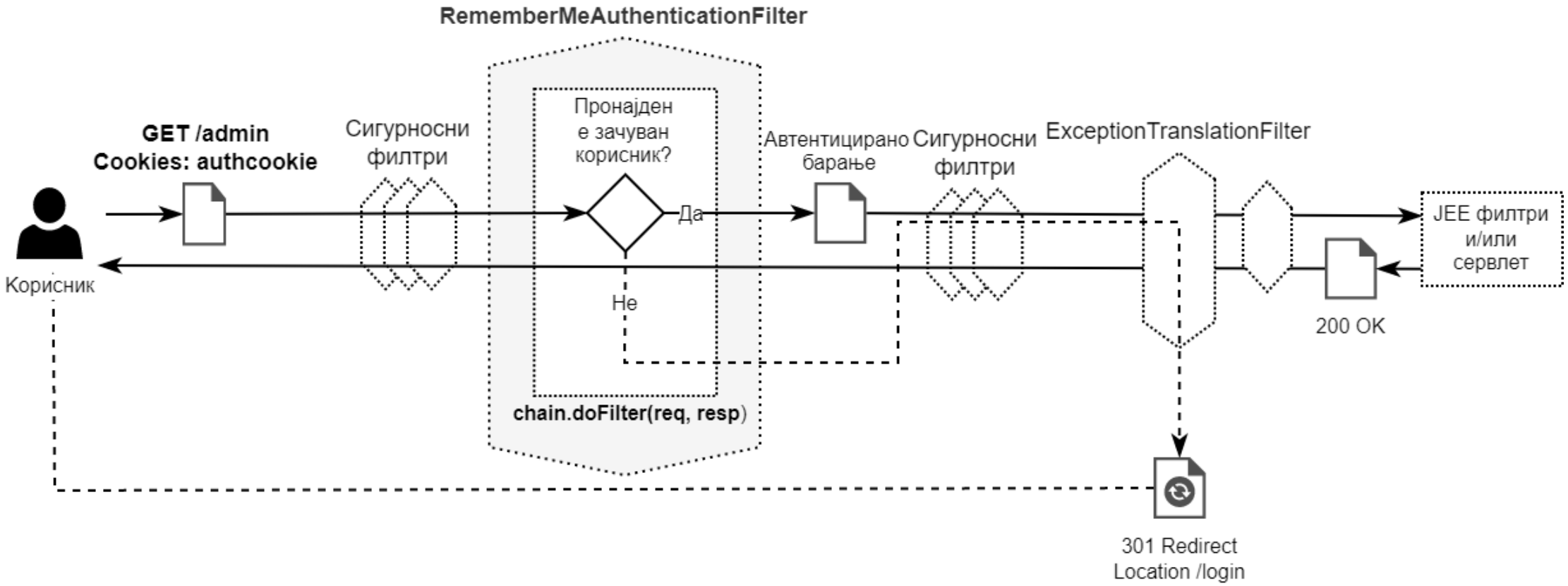


Автентикација со корисничко име и лозинка (POST /do-login)

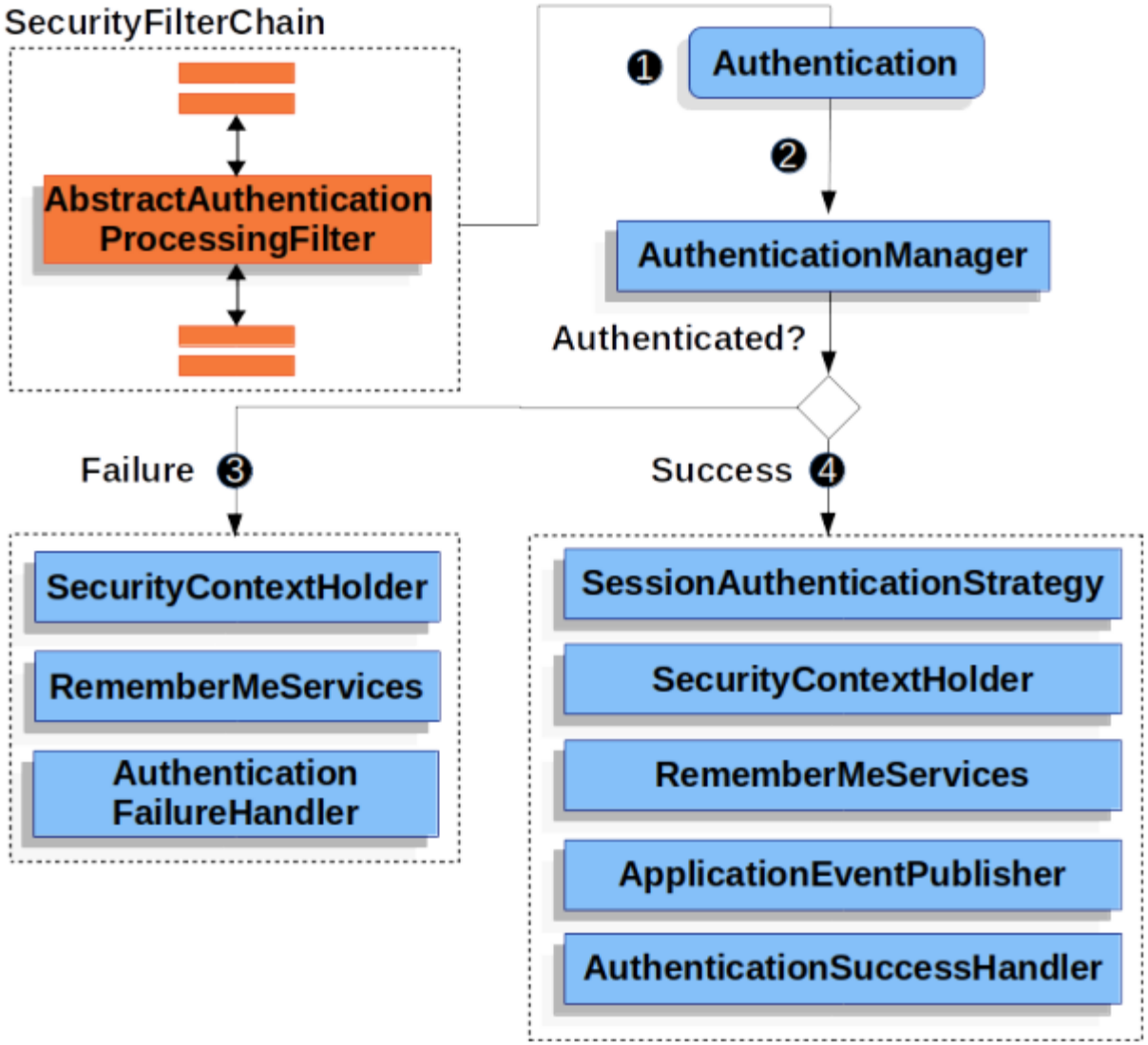
- **UsernamePasswordAuthenticationFilter** при успешна автентикација
  - Го поставува **Authentication** објектот во тековниот **SecurityContext**
  - Го зачувува тековниот сигурносен контекст преку SecurityContextRepository
    - предефинирано во сесијата на JEE контејнерот
  - Доколку е конфигурирано да се запомнат автентицираните корисници,
    - Се зачувува Authentication објектот со **RememberMeService**.
    - Во тековната конфигурација, нема да биде запаметен корисникот, бидејќи не е конфигуриран овој сервис.
  - Врши редирекција на страната за успешна најава, која може да се конфигурира со **defaultSuccessUrl("...", true/false)**



Барање кон /admin/products за запаметен корисник



Абстрактен тек на автентикација со Spring Security



Authentication Manager

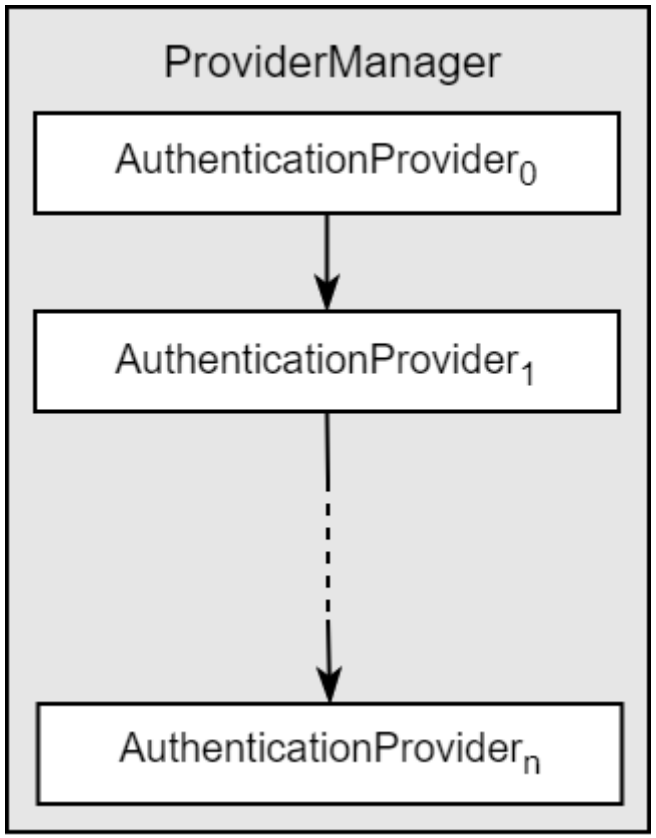
```java
public interface AuthenticationManager {
    5 implementations
    Authentication authenticate(Authentication authentication) throws AuthenticationException;
}
```

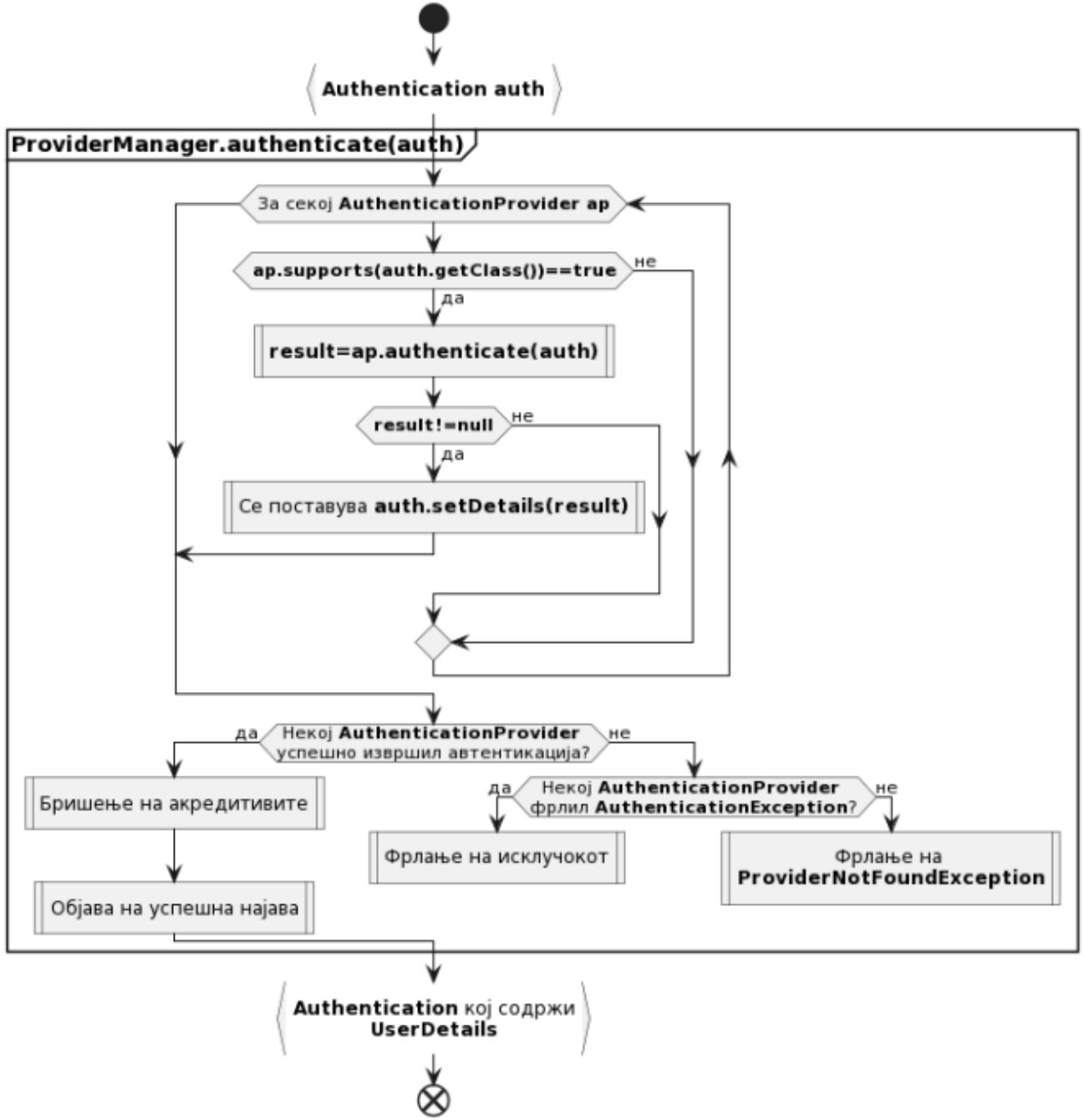
```java
public interface AuthenticationProvider {
    Authentication authenticate(Authentication authentication) throws
     ↪  AuthenticationException;
    boolean supports(Class<?> authentication);
}
```

```java
public interface UserDetailsService {
    UserDetails loadUserByUsername(String username) throws
     ↪  UsernameNotFoundException;
}
```

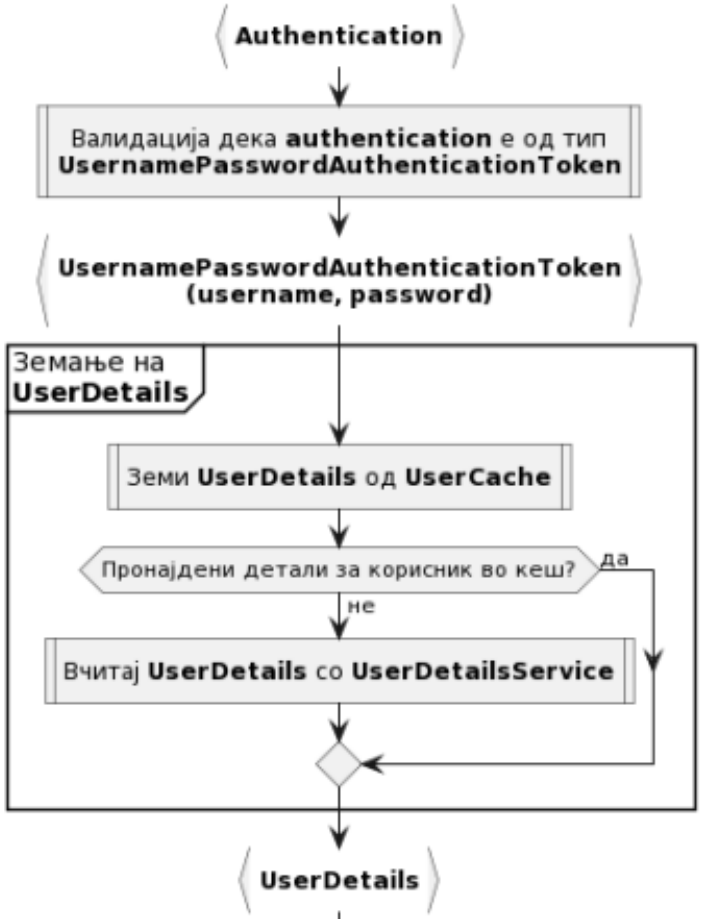
```java
public interface UserDetails extends Serializable {
    Collection<? extends GrantedAuthority> getAuthorities();
    String getPassword();
    String getUsername();
    boolean isAccountNonExpired();
    boolean isAccountNonLocked();
    boolean isCredentialsNonExpired();
    boolean isEnabled();
}
```

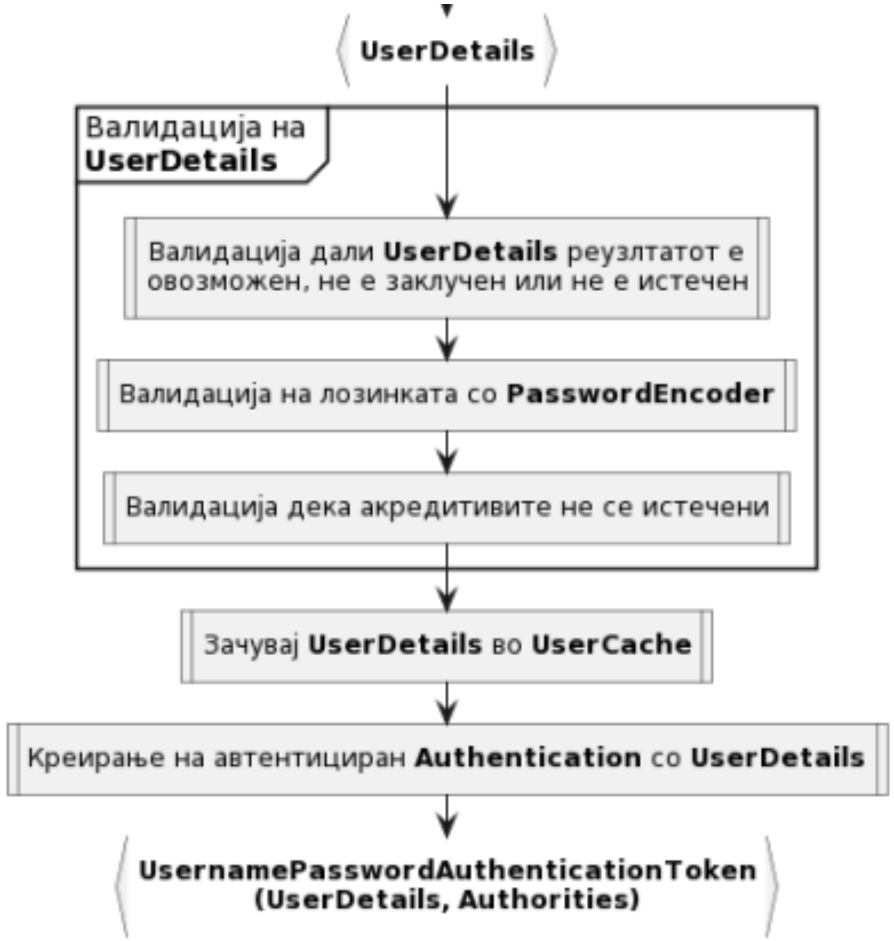
ProviderManager





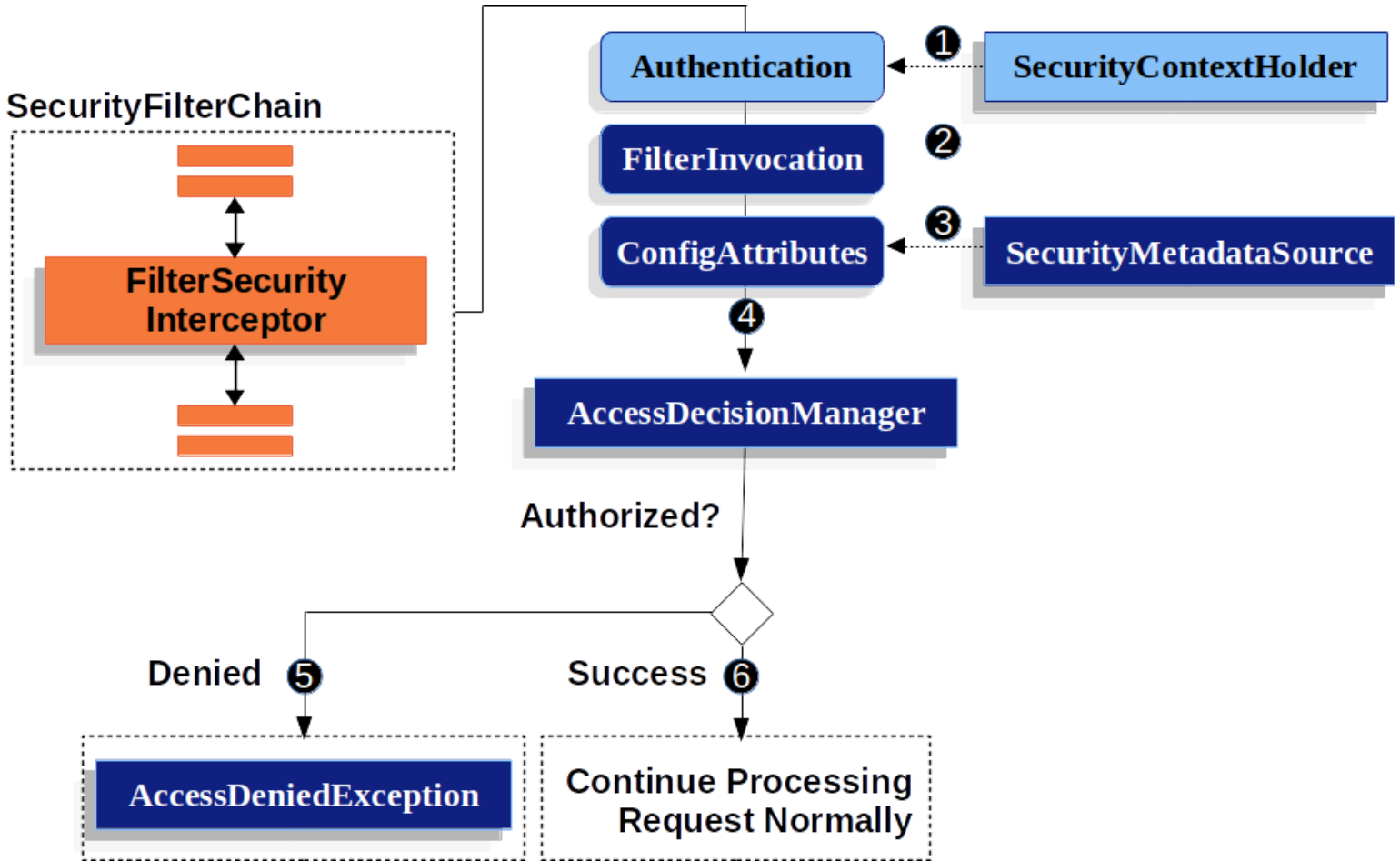
DaoAuthenticationProvider





Авторизација

```
.authorizeHttpRequests((requests) -> requests
        .requestMatchers("/", "/home", "/register", "/products").permitAll()
        .requestMatchers("/admin/**").hasRole("ADMIN")
        .anyRequest().authenticated()
)
```



Авторизација со Spring Security за веб ресурси

- **permitAll()** - дозволува пристап за сите
- **denyAll()** - не дозволува пристап на никој
- **authenticated()** - дозволува пристап само на автентицирани корисници (не се anonymous)
- **fullyAuthenticated()** - дозволува пристап на автентицираните корисници кои не се запаметени (не се добиени од RememberMeService)
- **anonymous()** - дозволува пристап само за не-автентицирани корисници
- **hasAuthority("...")** - дозволува пристап само за корисници кој во резултатите на Authentication.getAuthorities() методот ја содржат соодветната вредност

Авторизација со Spring Security за веб ресурси

- **hasAnyAuthority(...)** - дозволува пристап само за корисници кој во резултатите на Authentication.getAuthorities() методот содржат некоја од наведените вредности
- **hasRole("...")** - дозволува пристап само за корисници кој во резултатите на Authentication.getAuthorities() методот ја содржат соодветната вредност со префикс "ROLE_"
- **hasAnyRole(...)** - дозволува пристап само за корисници кој во резултатите на Authentication.getAuthorities() методот содржат некоја од наведените вредности со префикс "ROLE_"
- **hasIpAddress("...")** - дозволува пристап само за барања направени од дадена IP адреса
- **rememberMe()** - дозволува пристап само за корисници кои се запаметени, односно добиени со RememberMeService
- **access("...")** - Овозможува дефинирање на израз со SpEL

Заштита на ниво на методи

```
@Secured("ROLE_TELLER")
public Account post(Account account, double amount){ ... }

@Secured("IS_AUTHENTICATED_ANONYMOUSLY")
public Account readAccount(Long id) { ... }

@PreAuthorize("hasRole('USER')")
@PostAuthorize("returnObject.owner == authentication.name")
Account readAccount(Long id); { ... }

@PreAuthorize("#c.name == authentication.name")
public void doSomething(@P("c") Contact contact) { ... }

@PreAuthorize("hasRole('USER')")
@PostFilter("filterObject.name == authentication.name")
public List<Contact> getAll() { ... }
```

Преглед на Spring Security функционалности

- Spring Security е рамка која обезбедува
  - автентикација, авторизација и заштита од вообичаени напади.
  - конфигурабилна заштита на сите влезни точки во апликацијата.
  - конфигурираните правила за авторизација.
  - за веб апликациите се нудат предефинирани филтри кои подржуваат различни протоколи за автентикација и не заштитуваат од некои од најчестите напади.
  - се овозможуваат повеќе точки на проширување на овој модул
    - специфични сервиси,
    - овозможувачи за автентикација и
    - филтри
  - може да се имплементира уште понапредна заштита, во зависност од потребите.
- Накратко кажано, Spring Security модулот претставува де-факто стандард за заштита на Spring апликациите.