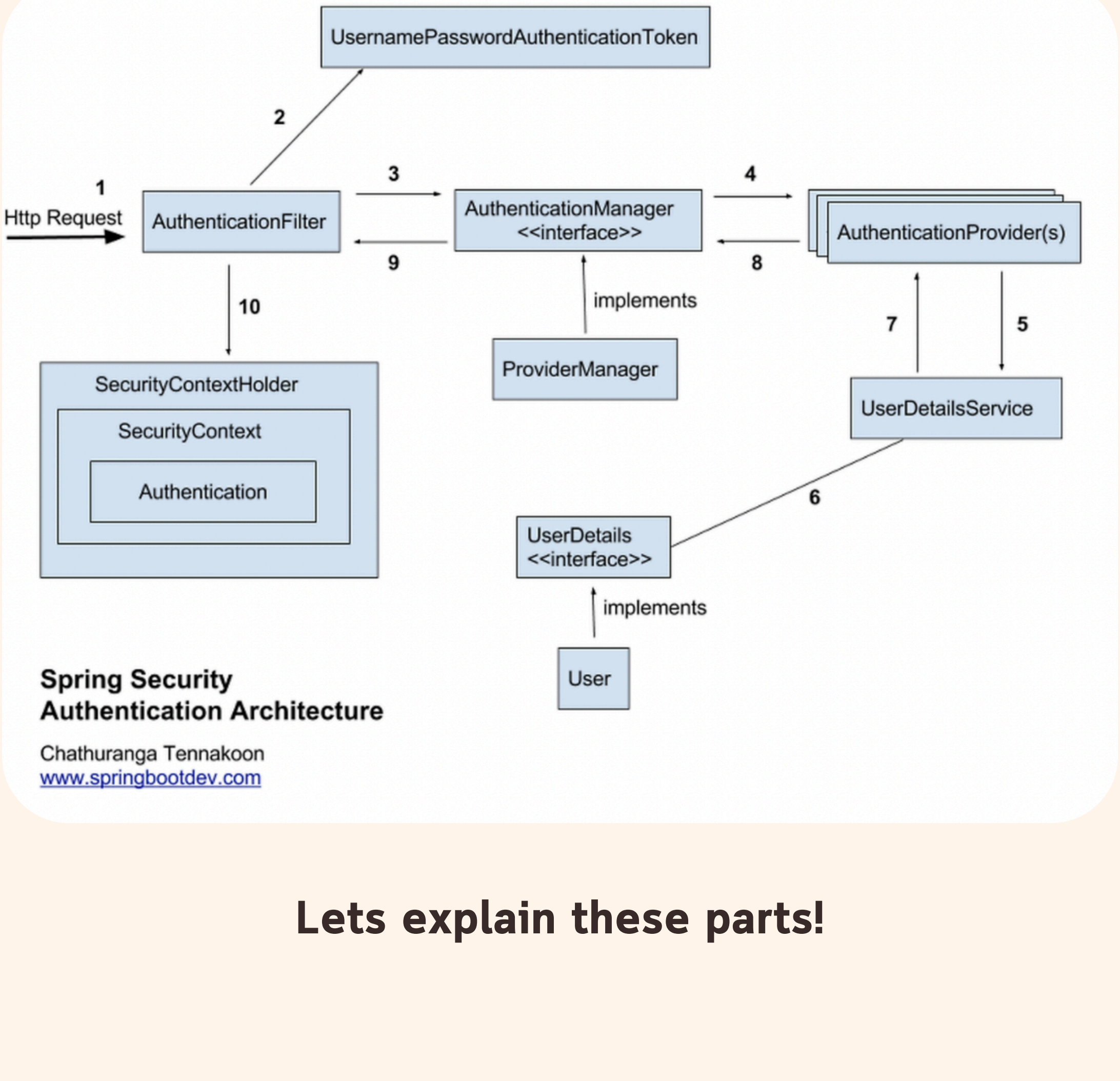
Authentication in Spring Security

Discover the built-in magic that comes with Spring Security when we want to implement **users' authentication** and therefore **authorization**.

@mauricioperez

02

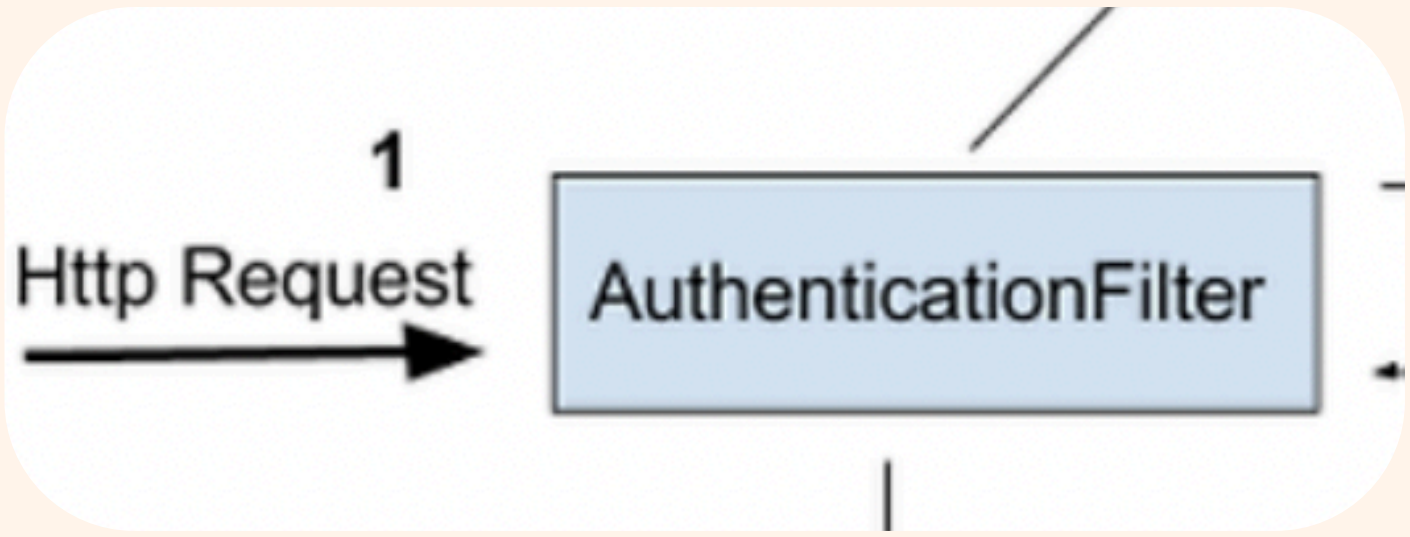
Reference Diagram



Lets explain these parts!

Authentication Filter

03



The request has arrived to the **AuthenticationFilter** which is found in the **SecurityFilterChain**.

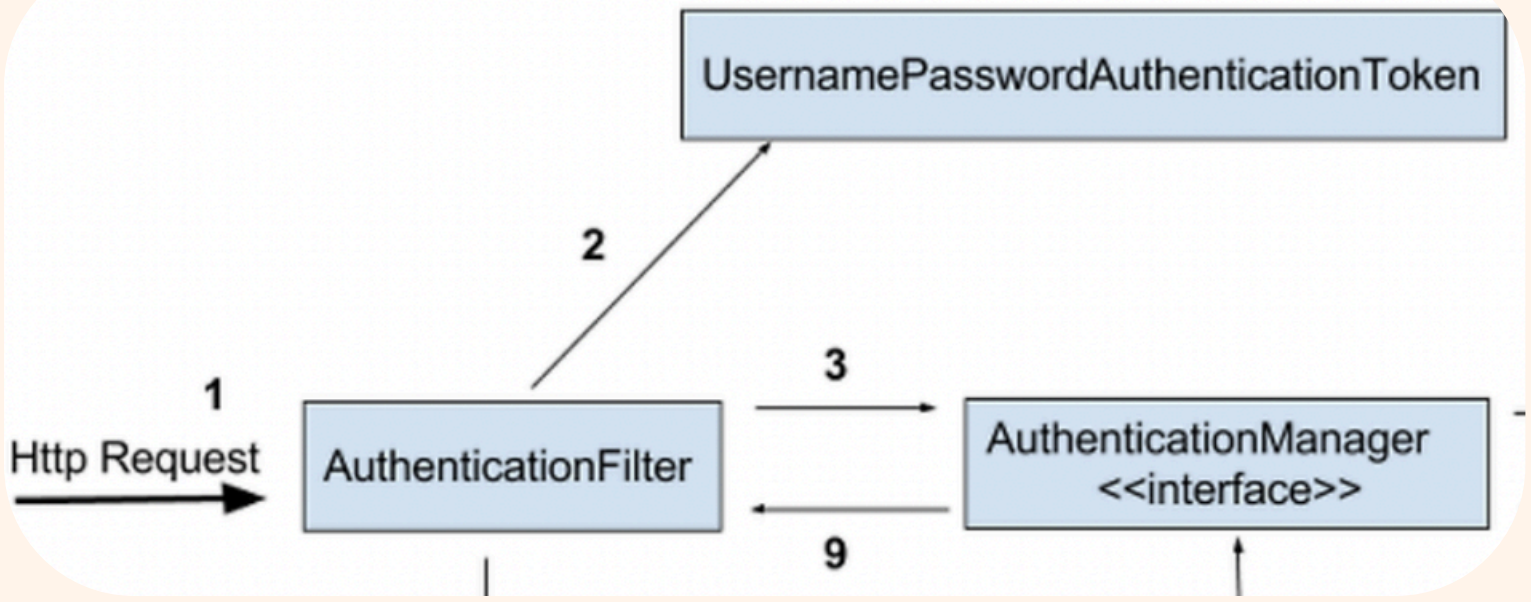
Do you remeber what **SecurityFilterChain** does?

This is the chain of rules that rigids the behaviour of Spring Security and actions that'll do to preserve security in our application.

In this case AuthenticationFilter is just another filter of the chain but now doing its work!

UsernamePassword AuthenticationToken

04



Step 2

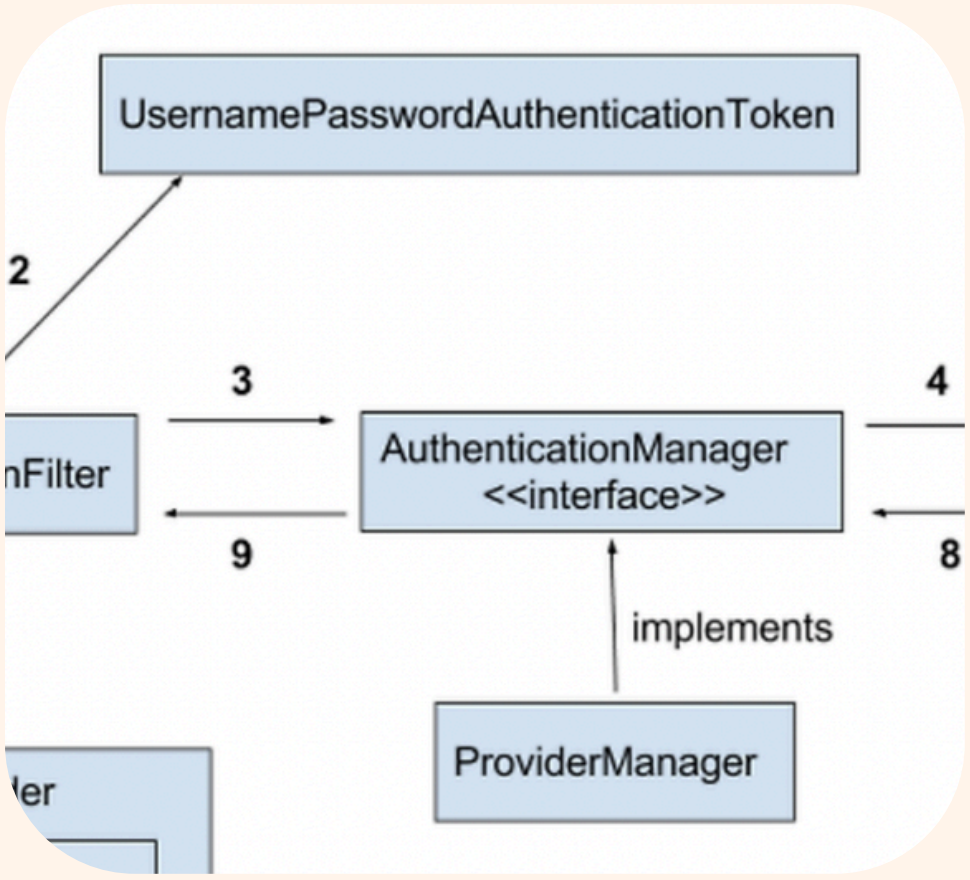
AuthenticationFilter requests a token to **UsernamePasswordAuthenticationToken**.

This is due to the authentication method we are choosing, in this case we chose the typical one: by using a username and a password.

This token is unique and will be able to identify the current process of user authentication

ProviderManager and AuthenticationManager

05



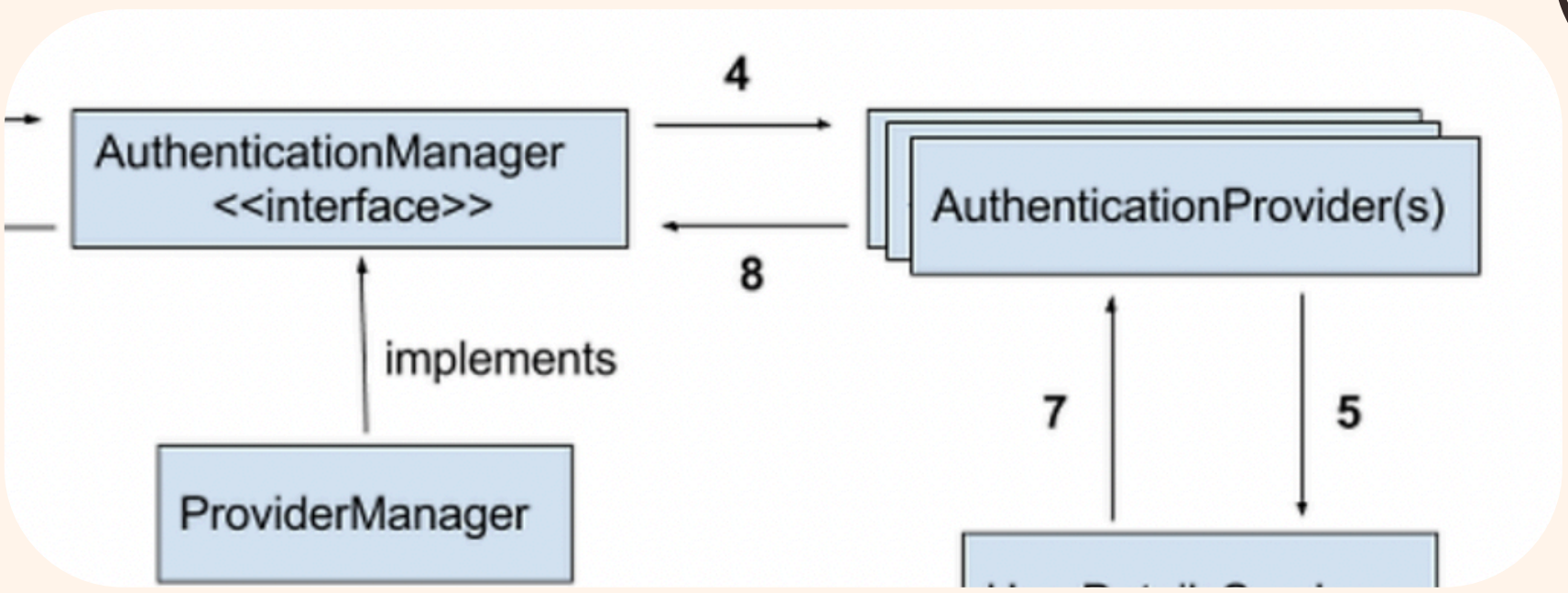
Step 3

The **AuthenticationFilter** passes the token generated to **ProviderManager** which implements the **AuthenticationManager** interface.

Authentication will be based on the Username and Password way and passes the token to the **authenticate** overridden method by **ProviderManager**.

Authentication Provider

06

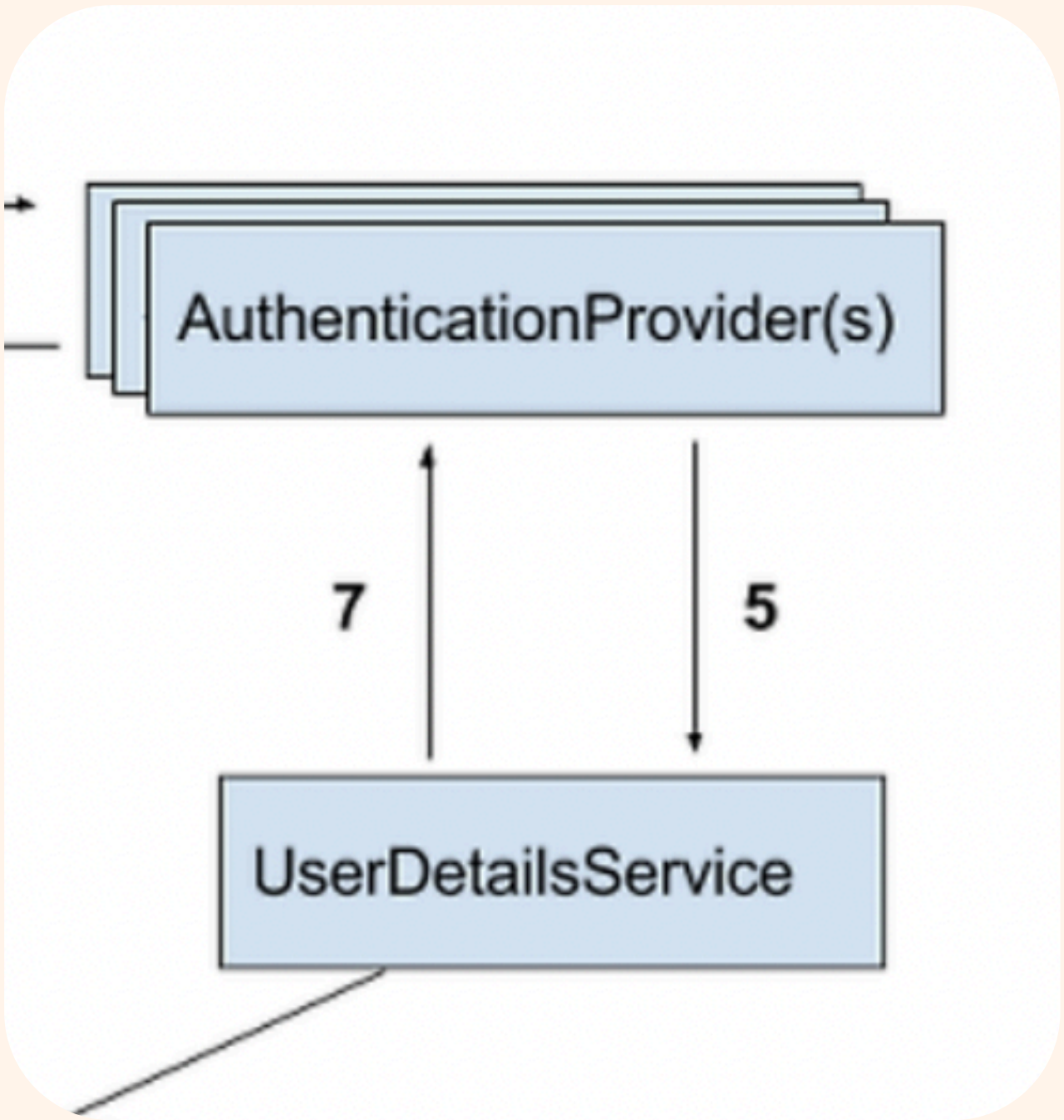


Step 4

ProviderManager asks for information that **AuthenticationProvider** has on its own based on what **UserDetailsService** provided.

UserDetailsService

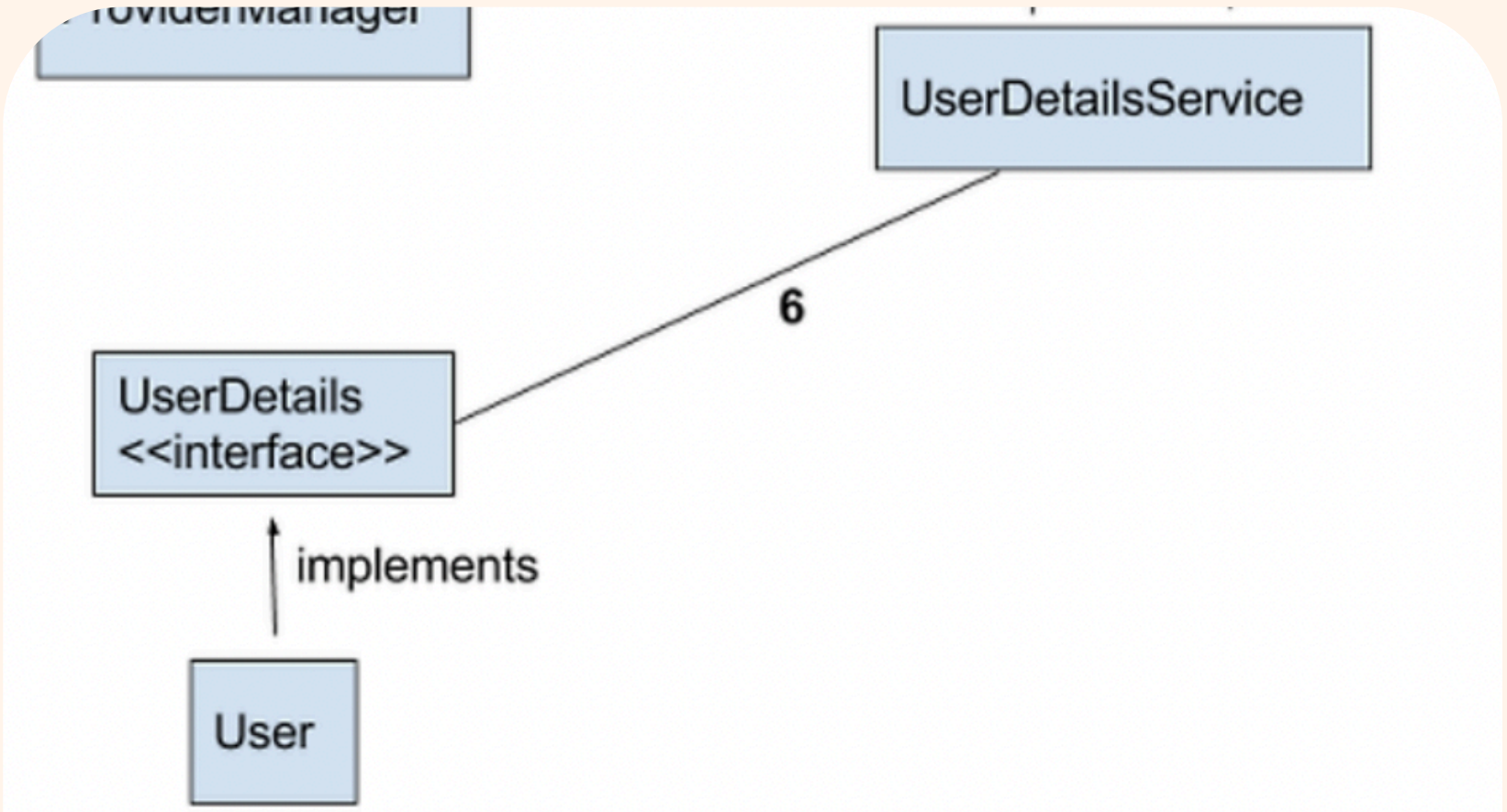
07



Step 5

AuthenticationProvider asks **UserDetailsService** to give him an object of the user in validation.

UserDetails



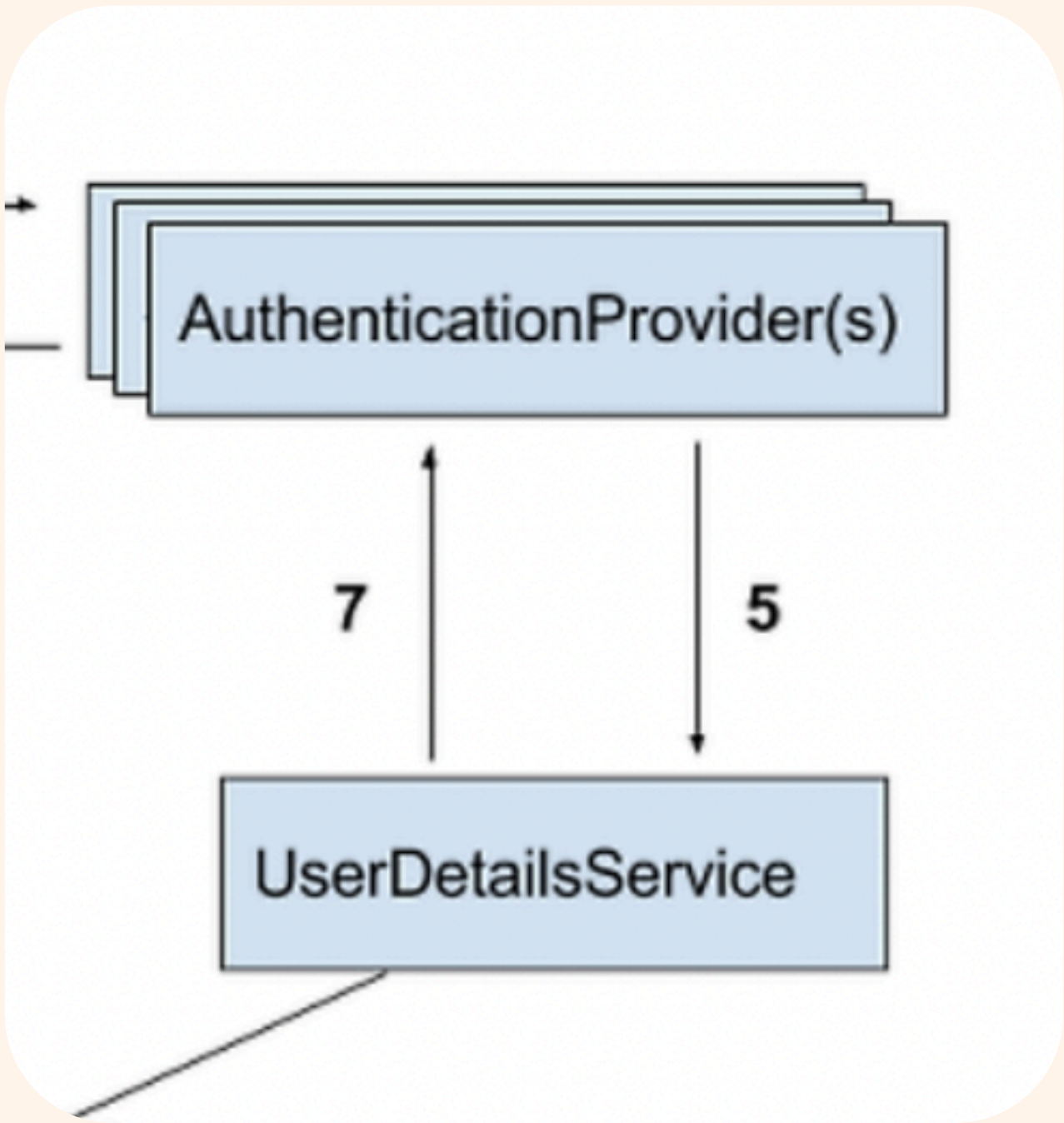
08

Step 6

The information is returned as **UserDetails** type of object, containing all the user information.

Again... Authentication Provider

09

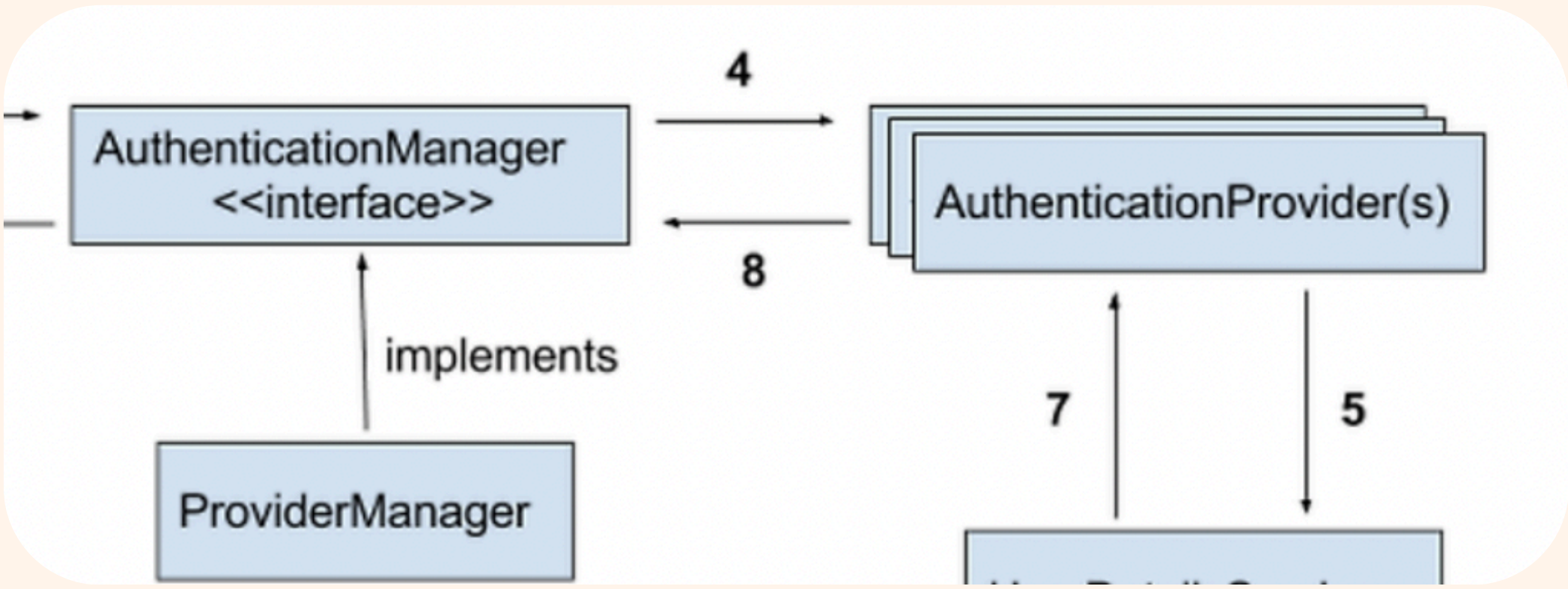


Step 7

The **UserDetails** object arrives to **AuthenticationProvider** and matches the information.

Again... ProviderManager

10

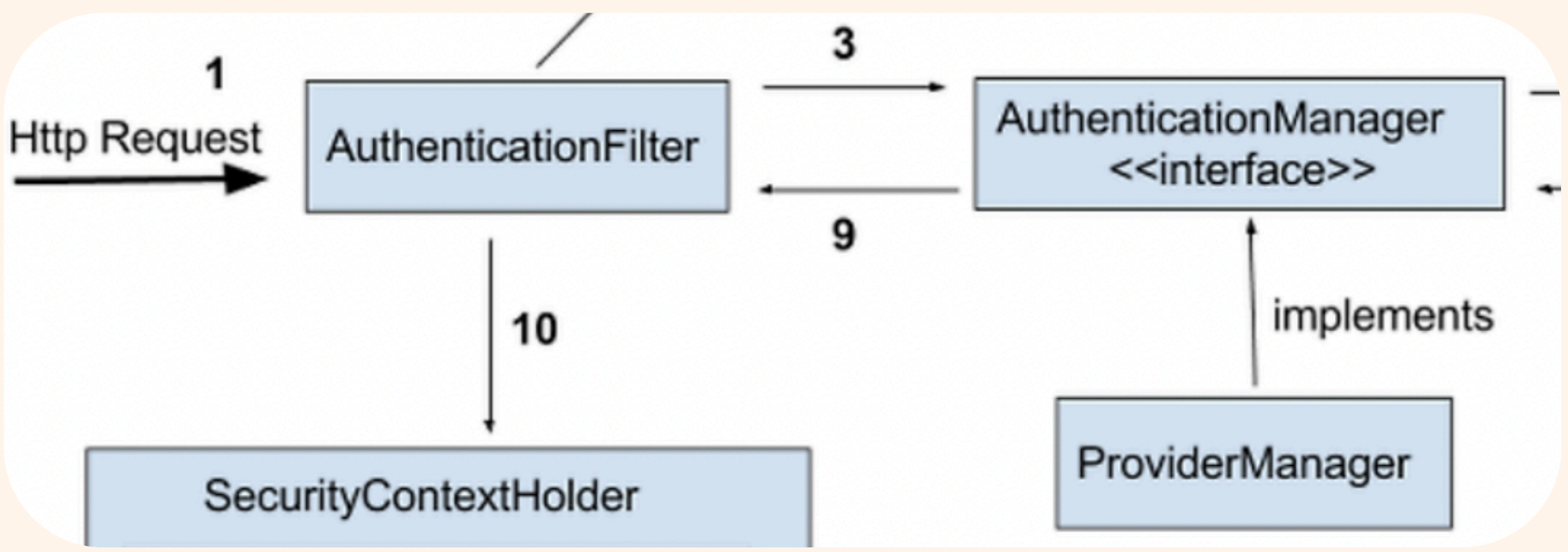


Step 8

If the information matches, then it authenticates and creates an **Authentication object** and the process flows again to **ProviderManager** giving him the new **Authentication object** just created by **AuthenticationProvider**.

Again... ProviderManager

11

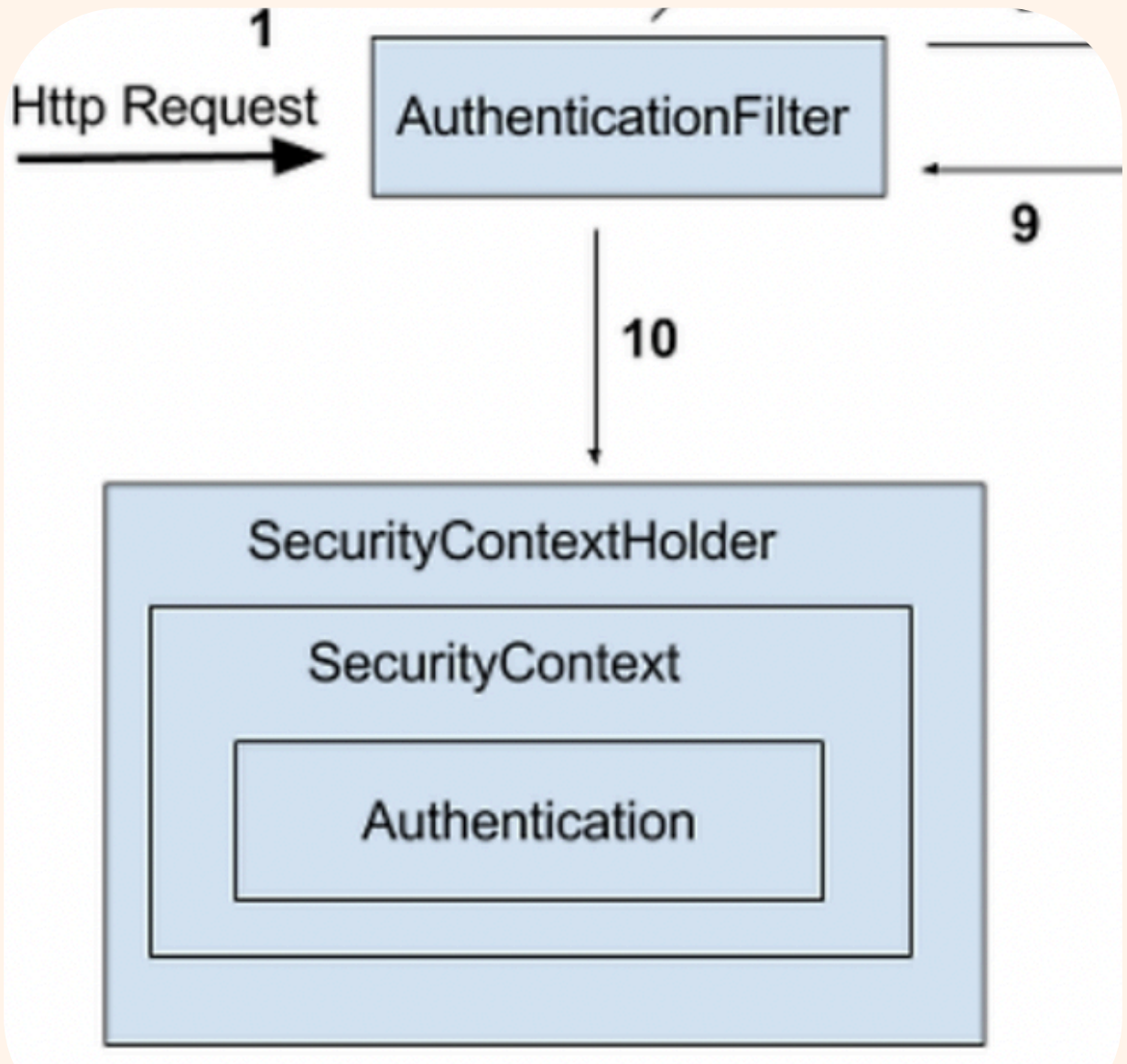


Step 9

The **Authentication object** is passed as a parameter to the **authenticate method** from the interface implementation **AuthenticationManager** and then passes the **Authenticate object** to the **AuthenticationFilter**.

Again... Authentication Filter

12



Step 10

AuthenticationFilter saves inside the SecurityContext the **Authentication object** just signed-up by **ProviderManager**.

Thank you!

Did you already know these concepts?

@mauricioperez

If you liked it, don’t hesitate to like and share this post!