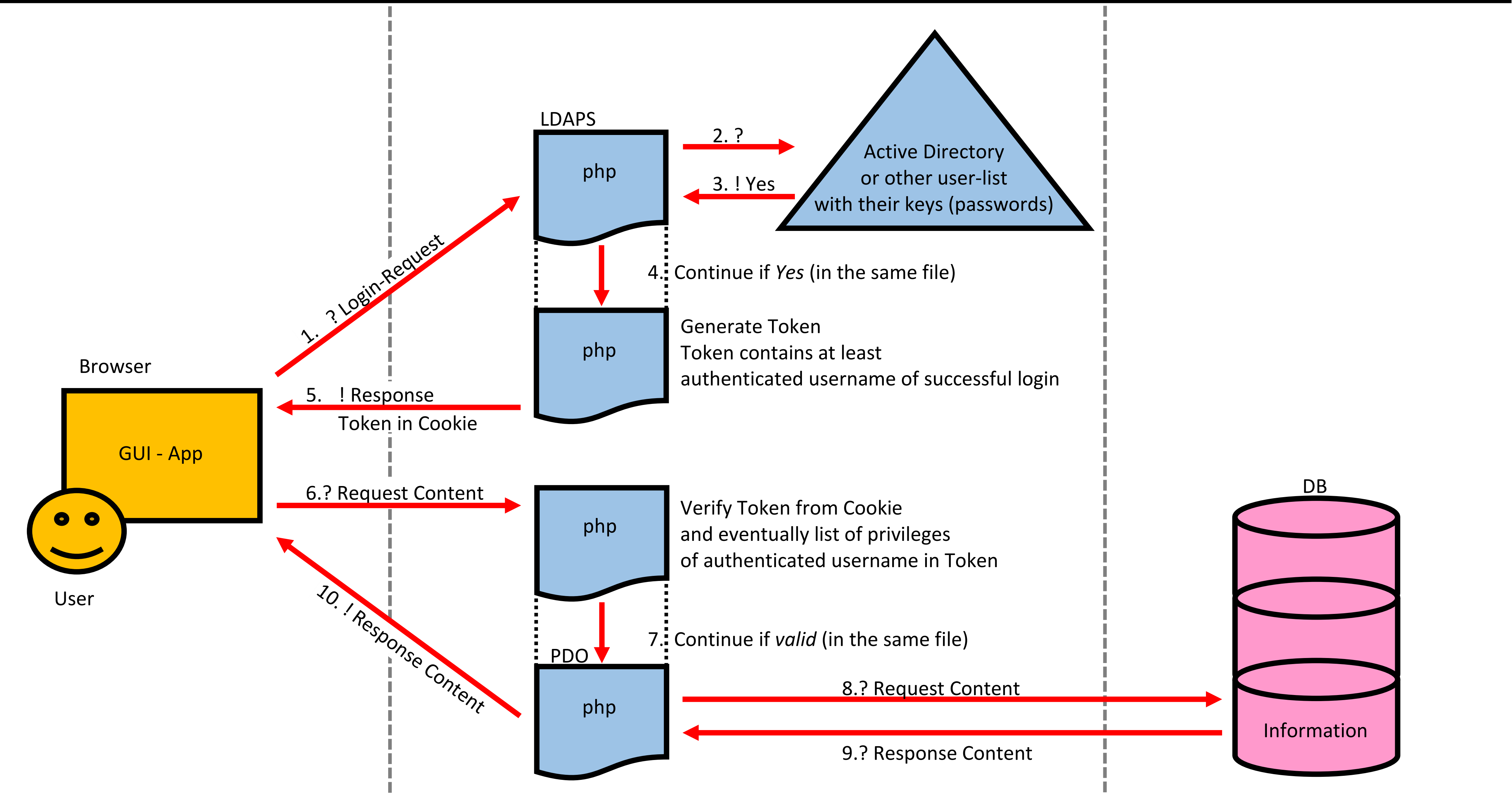
First Party | **Third Party** | **Second Party**



LDAPS

php

2. ?

3. ! Yes

Active Directory
or other user-list
with their keys (passwords)

4. Continue if *Yes* (in the same file)

php

Generate Token
Token contains at least
authenticated username of successful login

Browser

GUI - App

User

1. ? Login-Request

5. ! Response
Token in Cookie

6.? Request Content

php

Verify Token from Cookie
and eventually list of privileges
of authenticated username in Token

7. Continue if *valid* (in the same file)

PDO

php

8.? Request Content

9.? Response Content

10. ! Response Content

DB

Information

In other words:
View
Client
Stage
Workstation / PC / Desktop
Guest
People

In other words:
Controller
Worker
Backstage
Server
Reception
Delivery Service

In other words:
Model
Ressource
Story
Content
Hotelroom (TV, Bed, Shower)
Pizza or Newspaper …

Advice for deving authentication in a browser-app

On finding the right decision for the way of authentication in the mass of confusing offers on the net,
this elementary structure and authentication-flow (see above) should be understood.

It doesn't matter if it's

- *third-party trusted authentication*
- *standard network authentication*
- *secure ticketing protocol*
- *SAML*
- *OAuth*
- *KeyCloak*
- *Kerberos*
- *or others,*

or how many other names, signatures, and certificates they invent to be used
and how confusingly complex they are designed and described;
ALL different kinds of authentications (protocols, extensions, frameworks, or apps)
have ONE thing in common. They are grounded on the MVC-pattern.

The Model-View-Controller-Pattern is like a universal STANDARD for all programs or applications.
Even if we don't structure our code accordingly to follow this pattern.
In theory all apps follow this unavoidable pattern-rule.
This is why some say MVC is not a pattern; it is always a fact.

If this simple fact and the default-flow above are understood,
you will find it makes more sense to write your own code for your authentication,
instead of integrating another external dependency that regularly needs updates
and that could lead to hard-to-maintain and inflexible code in your apps.

Because all frameworks have their own strict rules to be followed,
that forces you to restructure your code.
Or it even might be that your new app does not has the right interface/API to be connected.
Then you would need to rewrite your app or still write your own authentication -flow.

BTW:

Yes, I know about the Single-Sign-On troubles.
From my point of view, this is the wrong direction.
Same-Sign-On is the right way. One key to open all doors.
But not at the same time.

Same-Sign-On means:
I log in to all my apps with my credentials registered in the Active Directory.
Different apps but same credentials. If I change my key (password),
then I can use the new key for all my apps without needing to change it too for every app I use.
Same key, different doors.

In reality this would be like this:
For my flat, I got one key to open:
the door to my apartment, the door to my garage,
the door to my garbage-container and the door to my cellar.
Why should I want to open all the doors at the same time if I enter my appartment?!

There might be some strange constellations where I permanently
switch between my apartment and my cellar when I clean up, or whatever,
but I don't see the point. Where is the trouble to open both doors,
and keep them open? Why should I need to open all the doors that I don't use?
I can’t be at two places at the same time. Some people claim they can.
I imagine a point on a time-line is crossing a point of a place-line (position)
but even if there was a wormhole to shorten the way to another point on the place-line,
me, myself and I (and that is always the same guy) can only be at one place at one time.

Single-Sign-On is a weak solution for a rarely fantastic claimed purpose of little use.
When I log in to my PC, I can use all the programs, but they all are „in the same room“.
That is a difference. The apps on the net spreaded all over the world are not in the same room.