Server-side
Web Development

Unit 01. Introduction to Web Applications.

Fidel Oltra, Ricardo Sánchez

IES Jaume II El Just
Tavernes de la Valldigna
Departament d'Informàtica
Curs 2024-25

Contents

1 Web applications.

A web application is a program that runs in a web browser. The application is usually stored on a remote server and delivered over the Internet to the browser. A web application can be stored too in a local server and running in the web browser through an intranet.

Some benefits of web applications are:

- Multiple users can access to the same application
- Don't need installation or additional software aside from the browser
- Access allowed through various platforms, browsers and operating systems

1.1 Web evolution.

Let's take a look to the historical evolution of the web, in order to understand how web applications work.

1.1.1 Web 1.0

Basically static pages. HTML and CSS.

1.1.2 Web 2.0

Dinamic pages. PHP, ASP, JSP and other programming languages. Interaction with the user.

1.1.3 Web 3.0

Decentralized applications. Semantic web. Internet Of Things. Blockchain technology.

2 Architecture of a web application.

The most common structure of a web application contains three levels: presentation, application and storage.

Presentation: the web browser. Interaction user-application. Interfaces. **Application**: what the application does, the dynamic level. Business logic. **Storage**: interaction with some kind of data storing, mainly a database.

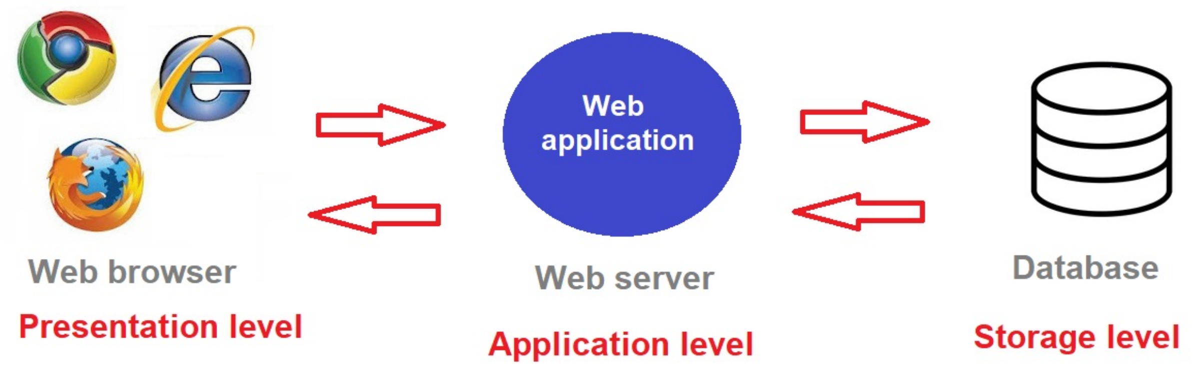


Figure 1: Levels of a web application

This three-levels structure is usually implemented through the so-called ***Client-Server Model***.

2.1 Client-Server model.

The Client-Server architecture is an application model that distributes the application tasks between service providers (servers) and service demand points (clients).

The client makes ***requests*** to the server, and the server sends ***responses*** to the client.

In a web application, the client is a web browser. The communication between the client and the server runs through the Internet/Intranet using the HTTP protocol. Usually the client is a local machine and the server is a remote machine, but both the client and the server can be on the same physical machine. While developing an application is quite usual to work with a local installed server.

2.1.1 Advantages and disadvantages

Advantages:

- Centralization: if one client turns off, everything keeps working

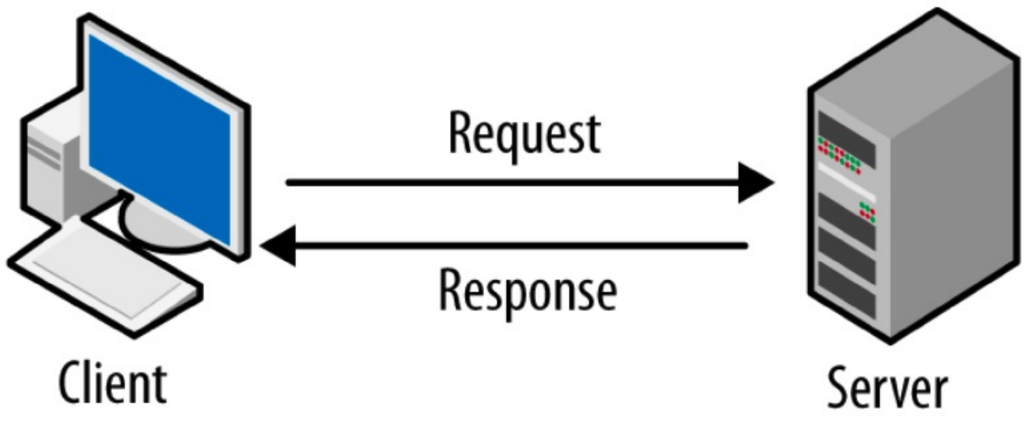


Figure 2: Client and server

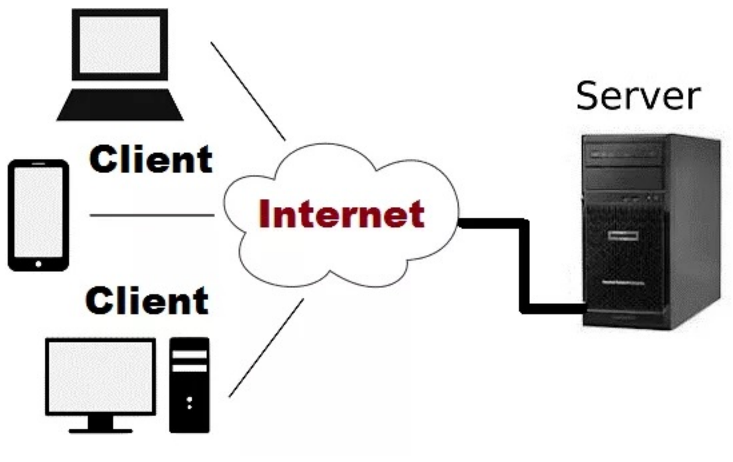


Figure 3: Some clients and one server

- Maintenance and scalability: is easy to change, add, remove or modify machines
- Security: transaction control, data encapsulation
- Optimization: client programs are simple and don’t require many resources

Disadvantages:

- Centralization: if the server turns off, the application stops working unless we have a distributed system
- Traffic: large number of simultaneous communications (requests and responses)
- Cost: server machines are needed with some specific software (application servers, database servers…)

2.1.2 Functions

Client functions:

- Starting requests and send them to the server
- Waiting for the server’s response
- Interacting with the user through an graphic interface

Server functions:

- Waiting for client requests
- Attending for the client requests
- Interacting with data stores
- Processing the results before sending them to the client
- Sending the results to the client in a format that the client understands

We tend to separate the application in two sections: one for the client side (***Front-end***) and one for the server side (***Back-end***).

2.2 Front-end.

The front-end development is the creation of the graphical user interface for the web application. This section is developed using code that the web browser can understand and execute: mainly HTML, CSS or Javascript.

In the front-end we have the applications menus, images, forms and many other visual (static or interactive) components.

2.3 Back-end.

In the back-end development we develop the business logic of the application, the access to the databases and the interaction with the front-end. In the back-end the application receives the client requests and, using scripting languages like PHP, Python, JSP and so on, performs the necessary tasks to prepare the corresponding response. This response must be provided to the client in a format that the web browser can interpret, so some kind of ***rendering*** is needed. Rendering means that the server generate an HTML page with the response to the client request.

3 Development of a web application.

To develop a web application, we need to merge languages that can be used to design both the static and dynamic part of the application. Usually we have ***client languages*** and ***server languages***.

3.1 Languages.

The **client-side** languages are HTML, CSS and Javascript, basically. These languages can be directly interpreted by the browser. We use client languages to design the graphic part of the application, forms, menus and so on.

The **server-side** languages are PHP, Node.js, Phyton, ASP and others. These languages run on the server-side/back-end of the application. They provide input validation, database access and output encoding. We use server languages to receive, validate and execute the client requests, to work with databases and eventually to process and generate the response to the client request. This response must be codified in a client language, usually HTML.

In practice we use both types (client-side and server-side) languages in the same application and even in the same script.

We can't see the source code of the server languages from our browser, only the client-side part and the results generated by the server-side part.

3.2 Environment. Tools.

A web application works in two environments:

- Development environment: while we are still developing the application
- Testing environment: while the final user is testing the application
- Production environment: when the application is finished, installed and ready to use

At the classroom we will work in a development environment. We will need:

- A web server. In our case, Apache
- A database server. In our case, MariaDB or MySQL
- An editor ready to work with server and client languages. We will work with VSCode
- A browser. Whatever you have in your computer

We need to choose our server language, too. We will use ***PHP***.

Although we could create a development environment in the cloud or in a virtual machine or container using tools like Docker, we will create a local development environment including both web and database servers using ***XAMPP***.

3.3 Creating a Local Web Server in a Linux host

We can install the local web server and the development tools either in a real machine or in a virtual machine. If we use a real machine we can take benefit of the speed of the computer. On the other hand, a virtual machine is more portable and useful in many cases (for example, using a Linux VM in a Windows host).

In addition, we're going to use a Linux operating system because of their stability and security. However, you can use a Windows or Mac operating system (in the next section we explain how to install a XAMPP server in a Windows host).

In this section we'll explain how to install Apache, PHP 8.1 and MySQL in a **Xubuntu Desktop 24.04**, but you can install it in whatever Ubuntu flavour you want.

3.3.1 Installing Apache and PHP

First, we install the Apache server and the PHP interpreter:

```
sudo apt update
sudo apt install apache2
sudo apt install php
```

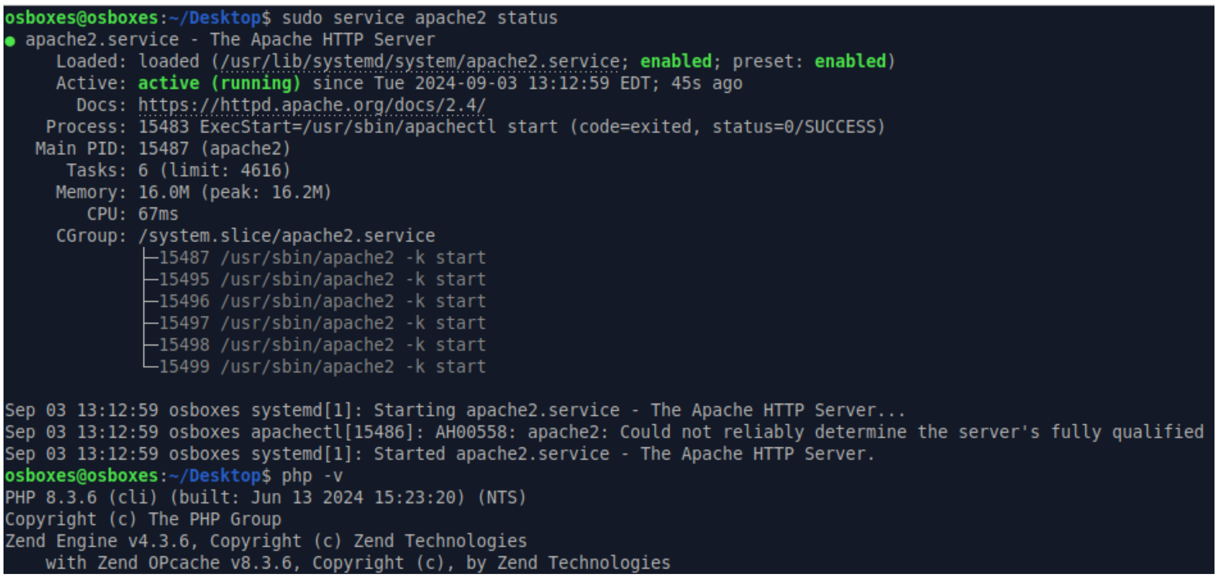
We can also install some common modules we'll use later:

```
sudo apt install php-mysql php-zip php-gd php-mbstring php-curl php-xml
↪   php-bcmath php-cgi php-pear
```

Done! We can check the status of both with:

```
sudo service apache2 status
php -v
```

Now, if we want to use directly the default folder of the system, we need to change permissions of the `/var/www/html` folder, where the web files are stored :



```
osboxes@osboxes:~/Desktop$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-09-03 13:12:59 EDT; 45s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 15483 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 15487 (apache2)
      Tasks: 6 (limit: 4616)
     Memory: 16.0M (peak: 16.2M)
        CPU: 67ms
     CGroup: /system.slice/apache2.service
             ├─15487 /usr/sbin/apache2 -k start
             ├─15495 /usr/sbin/apache2 -k start
             ├─15496 /usr/sbin/apache2 -k start
             ├─15497 /usr/sbin/apache2 -k start
             ├─15498 /usr/sbin/apache2 -k start
             └─15499 /usr/sbin/apache2 -k start

Sep 03 13:12:59 osboxes systemd[1]: Starting apache2.service - The Apache HTTP Server...
Sep 03 13:12:59 osboxes apachectl[15486]: AH00558: apache2: Could not reliably determine the server's fully qualified
Sep 03 13:12:59 osboxes systemd[1]: Started apache2.service - The Apache HTTP Server.
osboxes@osboxes:~/Desktop$ php -v
PHP 8.3.6 (cli) (built: Jun 13 2024 15:23:20) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.3.6, Copyright (c) Zend Technologies
    with Zend OPcache v8.3.6, Copyright (c), by Zend Technologies
```

Figure 4: Checking Apache and PHP installation

```
sudo chown -R xubuntu /var/www/html
```

Replace *xubuntu* by your username.

It is recommended to create a shortcut on the Desktop in order to access faster to the server folder. In Xubuntu we do that by browsing to the `/var/www/html` folder and right-clicking on it, then selecting *Send to -> Desktop (create link)*.

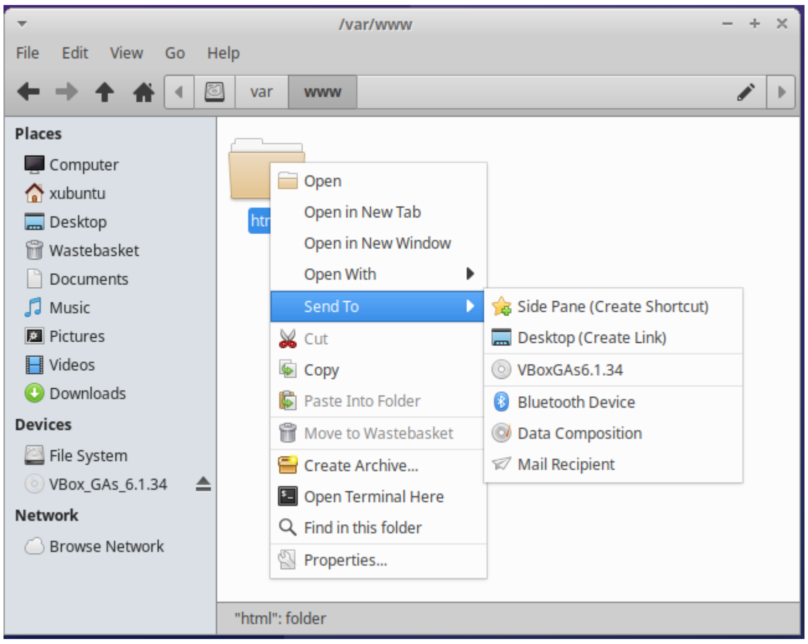


Figure 5: Creating a shortcut to /var/www/html

Alternatively, we can start a PHP local server on another folder, for instance in a folder in our home directory. This is useful in shared computers where we don't have full access to the whole system. To do that, open a terminal in your local web folder and type the next command:

```
php -S <host>:<port>
```

For example:

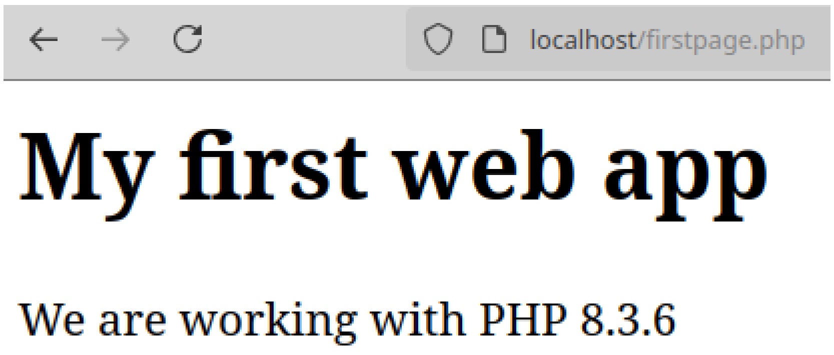
```
php -S localhost:8080
```

Now, you can type localhost:8080 in your browser in order to go to your webpage.

Assuming all the services are running properly, let's create our first web page with this content:

```
<h1>My first web app</h1>
<p>We are working with PHP version
<?php echo phpversion(); ?>
</p>
```

Save the file as **firstpage.php** in the root folder of the web server. Now type this URL in your browser: **`http://localhost/firstpage.php`**. You should see your first PHP page looking like this:



My first web app

We are working with PHP 8.3.6

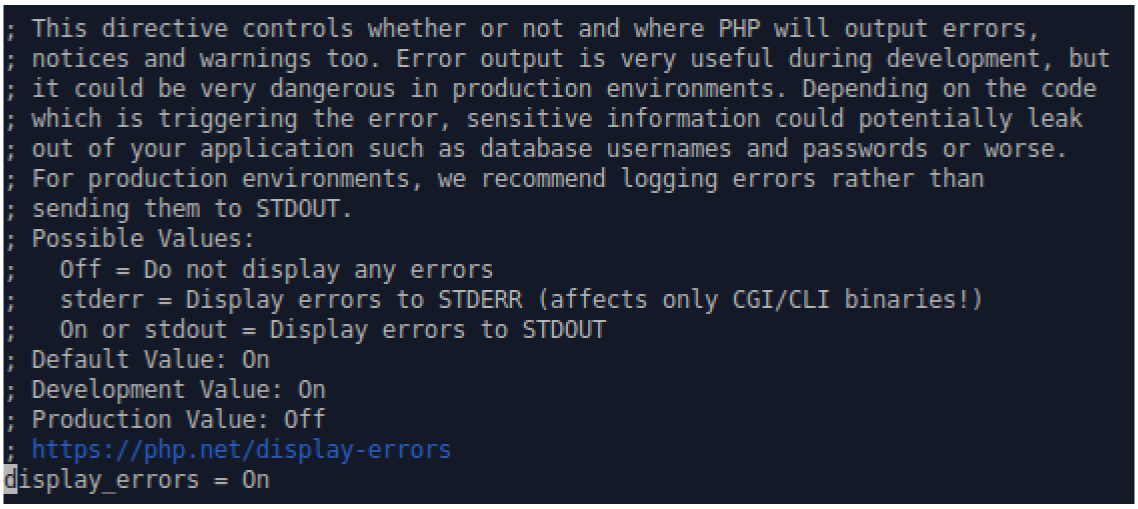
Figure 6: Your first PHP application

3.3.2 Displaying errors

We can configure our PHP installation to display programming errors, otherwise we can only see a blank page.

Edit the php configuration file that is located in */etc/php/8.1/apache2/php.ini*, search *display_errors = Off* and change Off to On. If you are using another version of PHP instead of the 8.1, the third folder is named by the version name.

```
sudo nano /etc/php/8.3/apache2/php.ini
```



```
; This directive controls whether or not and where PHP will output errors,
; notices and warnings too. Error output is very useful during development, but
; it could be very dangerous in production environments. Depending on the code
; which is triggering the error, sensitive information could potentially leak
; out of your application such as database usernames and passwords or worse.
; For production environments, we recommend logging errors rather than
; sending them to STDOUT.
; Possible Values:
;   Off = Do not display any errors
;   stderr = Display errors to STDERR (affects only CGI/CLI binaries!)
;   On or stdout = Display errors to STDOUT
; Default Value: On
; Development Value: On
; Production Value: Off
; https://php.net/display-errors
display_errors = On
```

Figure 7: php.ini display_errors configuration

To apply the change, restart the Apache service:

```
sudo systemctl restart apache2
```

3.3.3 Installing MySQL

We don't need the MySQL database management system until further units, but you can install it now if you want.

To install the MySQL server:

```
sudo apt update
sudo apt install mysql-server
```

Once installed, enter the server by typing:

```
sudo mysql
```

Now, we can create an admin user to manage the entire DBMS:

```
create user 'admin'@'localhost' identified by 'php';
grant all privileges on *.* to 'admin'@'localhost' with grant option;
```

With these commands, we are creating a user named *admin* with the password *php*

And, finally, exit the MySQL console:

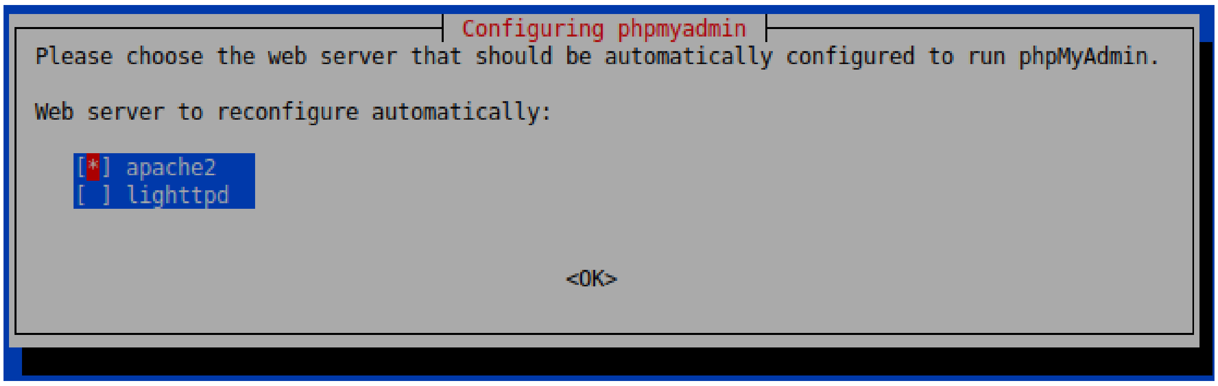
```
exit
```

Optionally, you can install **phpMyAdmin** if you want to administrate MySQL easily:

```
sudo apt update
sudo apt install phpmyadmin
```

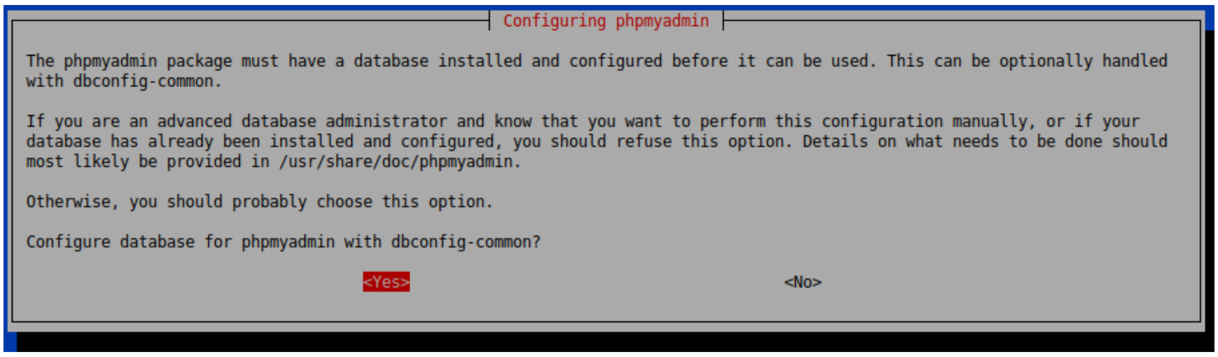
During the installation you have to choose *apache2* as server and *dbconfig-common* to configure the database. Remember to use the space bar to select options. Enter the password for phpmyadmin of your own choice or left it blank.

Once installed PhpMyAdmin, you can enter by typing on your browser **`http://localhost/phpmyadmin`** with the user *admin* and the password *php*.



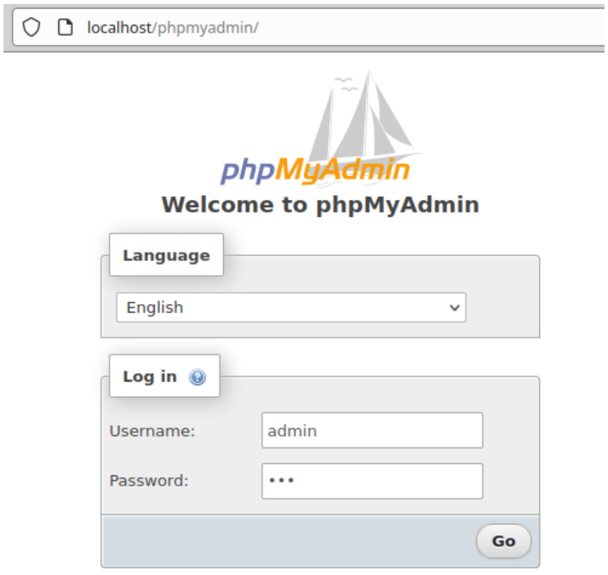
```
Configuring phpmyadmin
Please choose the web server that should be automatically configured to run phpMyAdmin.

Web server to reconfigure automatically:

    [*] apache2
    [ ] lighttpd

                                   <OK>
```

Figure 8: PhpMyAdmin installation: web server



```
Configuring phpmyadmin
The phpmyadmin package must have a database installed and configured before it can be used. This can be optionally handled
with dbconfig-common.

If you are an advanced database administrator and know that you want to perform this configuration manually, or if your
database has already been installed and configured, you should refuse this option. Details on what needs to be done should
most likely be provided in /usr/share/doc/phpmyadmin.

Otherwise, you should probably choose this option.

Configure database for phpmyadmin with dbconfig-common?

                  <Yes>                                  <No>
```

Figure 9: PhpMyAdmin installation: configure with dbconfig-common



localhost/phpmyadmin/

Welcome to phpMyAdmin

Language: English

Log in

Username: admin

Password: •••

Figure 10: PhpMyAdmin login

3.4 Creating a local web server using XAMPP.

XAMPP is a popular PHP development environment. It's free, open source and easy to install. We can get XAMPP packages for Windows, Linux and OS X. XAMPP contains MariaDB (a database server), PHP / Perl (server-side languages) and Apache (a web server).

To install XAMPP, we just need to download and start the installer. We can find XAMPP on the website https://www.apachefriends.org

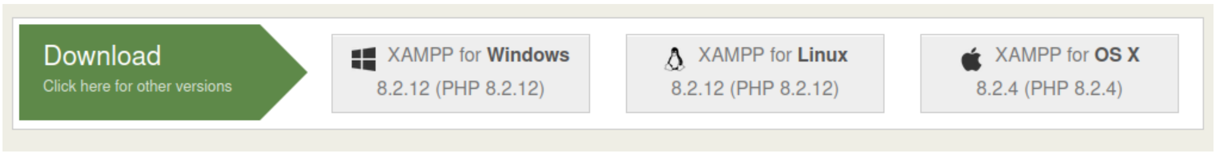


Figure 11: Downloading XAMPP

3.4.1 Windows installation

Once the installer has been downloaded, we will run the executable file just by double-clicking on it. First we will select which components to install. All the components are selected by default. We can click **Next** and go on with the installation.

Then we choose the folder (by default `c:\xampp`) and click **Next**.

Select your preferred language, deselect the Bitnami option, and finally confirm the installation.

The installation creates the folder **`C:\xampp`**. In this folder we can find some scripts to start and stop the different services: *apache_start*, *apache_stop*, *mysql_start* and *mysql_stop*, among others. We will find as well the XAMPP control panel (*xampp_control*) and a couple of scripts to start and stop the entire XAMPP platform (*xampp_start* and *xampp_stop*). If we run xampp-control we see the control panel. We can start, administrate, configure and stop every single service from this panel.

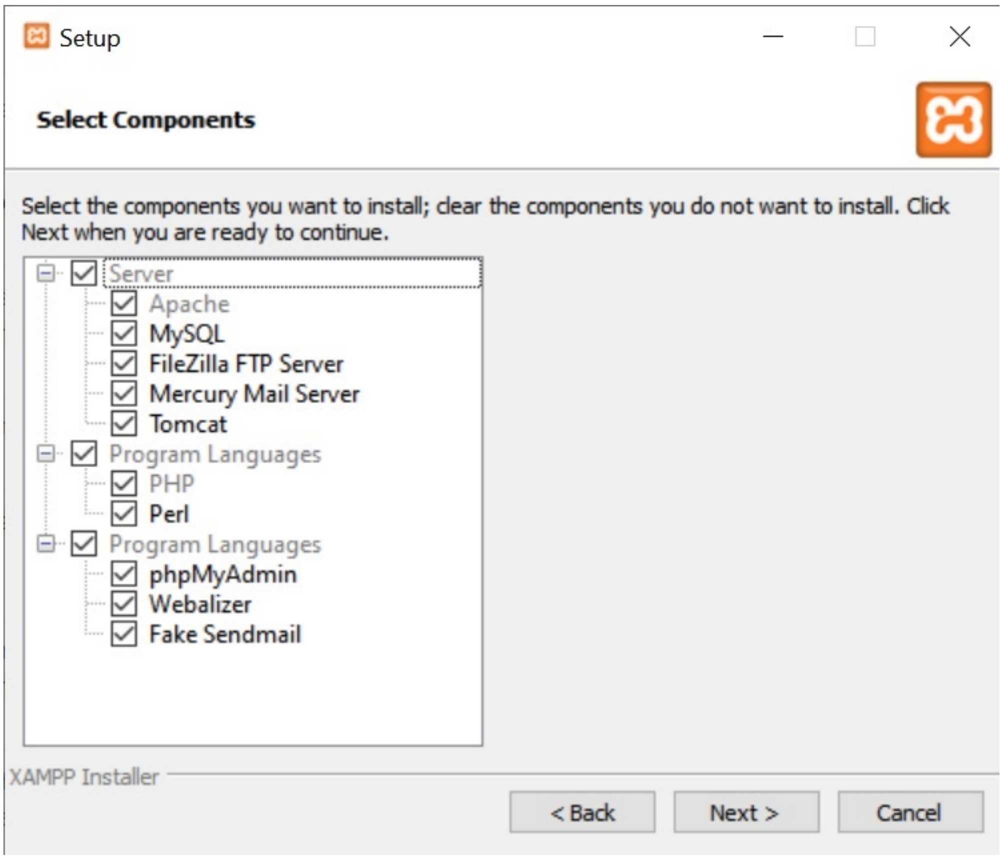
The root folder where we will place our web applications is **`c:\xampp\htdocs`**.

3.4.2 Linux installation

To run the installer in Linux, first we must change the permissions for the installer:

```
chmod 755 xampp-linux-*-installer.run
```

Now we can execute the installer:


Setup

Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [✓] Server
  - [✓] Apache
  - [✓] MySQL
  - [✓] FileZilla FTP Server
  - [✓] Mercury Mail Server
  - [✓] Tomcat
- [✓] Program Languages
  - [✓] PHP
  - [✓] Perl
- [✓] Program Languages
  - [✓] phpMyAdmin
  - [✓] Webalizer
  - [✓] Fake Sendmail

XAMPP Installer

Figure 12: Selecting components

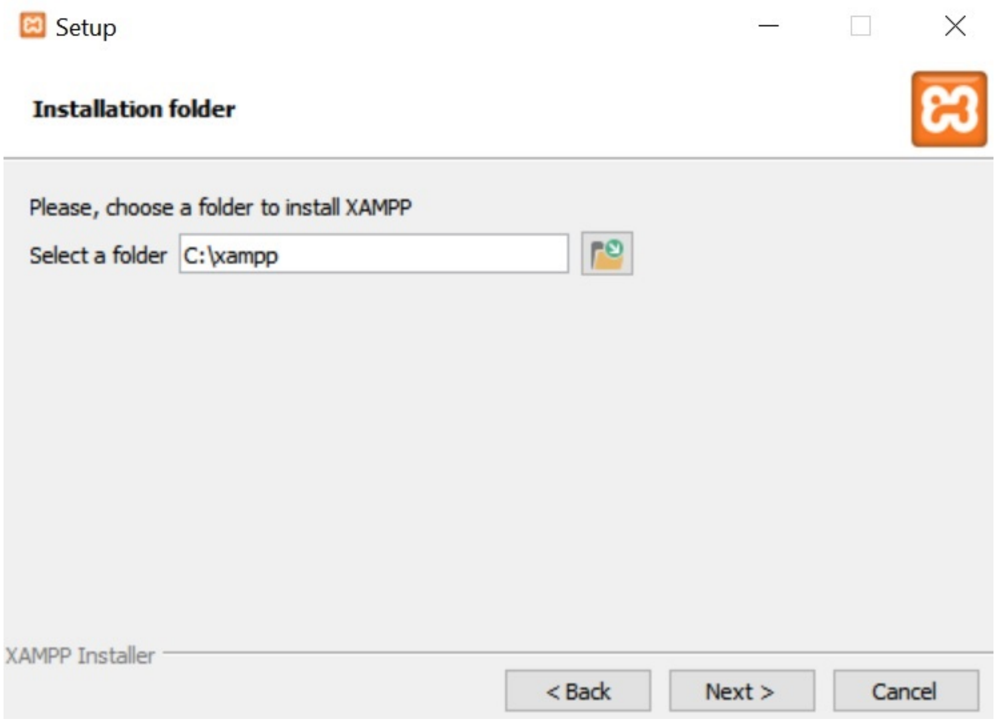
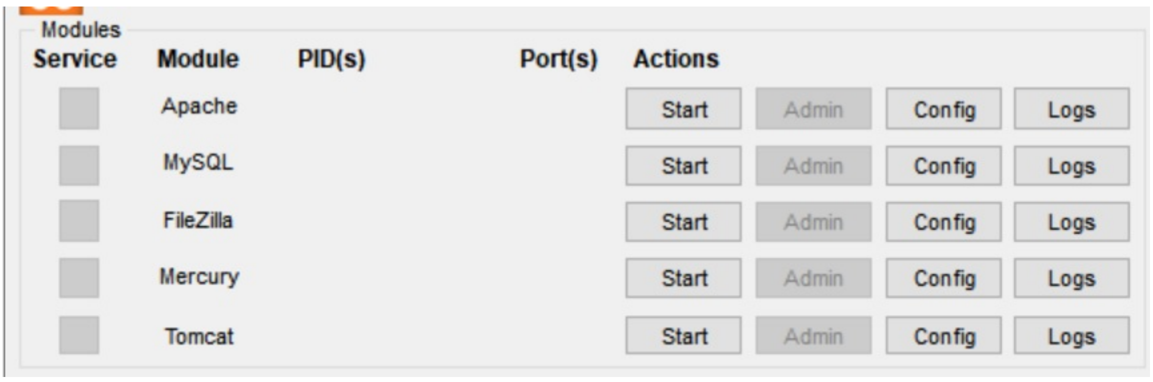


Figure 13: Choosing the XAMPP folder



Service	Module	PID(s)	Port(s)	Actions			
	Apache			Start	Admin	Config	Logs
	MySQL			Start	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

Figure 14: XAMPP control panel

```
sudo ./xampp-linux-*-installer.run
```

In Linux the XAMPP folder is **/opt/lampp**. We can launch all services by doing

```
sudo /opt/lampp/lampp start
```

And

```
sudo /opt/lampp/lampp stop
```

to stop them.

We can find the XAMPP configuration files at the folder **`/etc`**.

The root folder where we will place our web applications in Linux is **`/var/www/`** or **`/var/www/html`**.

3.4.3 Checking our installation

To check if the web server is working, run XAMPP and then, in your favourite browser, type the URL **`http://localhost`**. You should see the XAMPP welcome page.

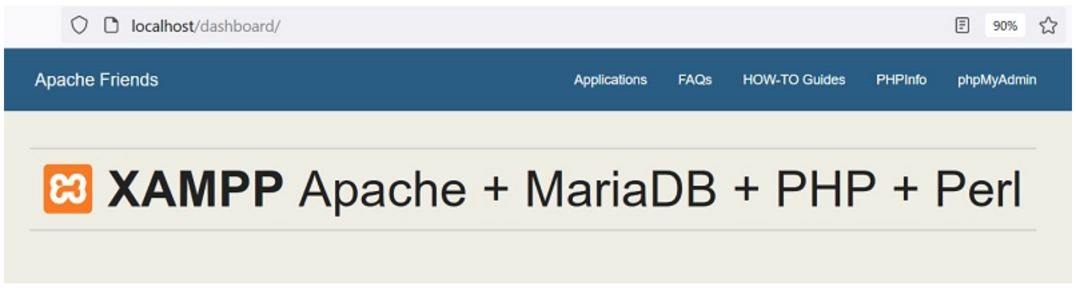
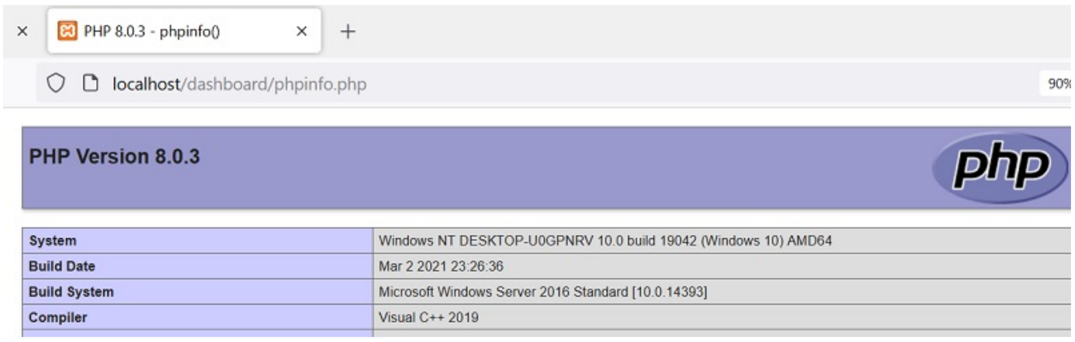


Figure 15: XAMPP welcome page

We can check if PHP is working by clicking the **`PHPinfo`** link in the menu.

If we launch the database server, we can check if it works by clicking the **`phpMyAdmin`** link in the same menu.

Keep in mind that, by default, we have a **root** user without password with admin privileges to work with the database server. It's a good practice to create a password for root.



PHP 8.0.3 - phpinfo()

localhost/dashboard/phpinfo.php

PHP Version 8.0.3

System	Windows NT DESKTOP-U0GPNRV 10.0 build 19042 (Windows 10) AMD64
Build Date	Mar 2 2021 23:26:36
Build System	Microsoft Windows Server 2016 Standard [10.0.14393]
Compiler	Visual C++ 2019

Figure 16: PHPinfo output

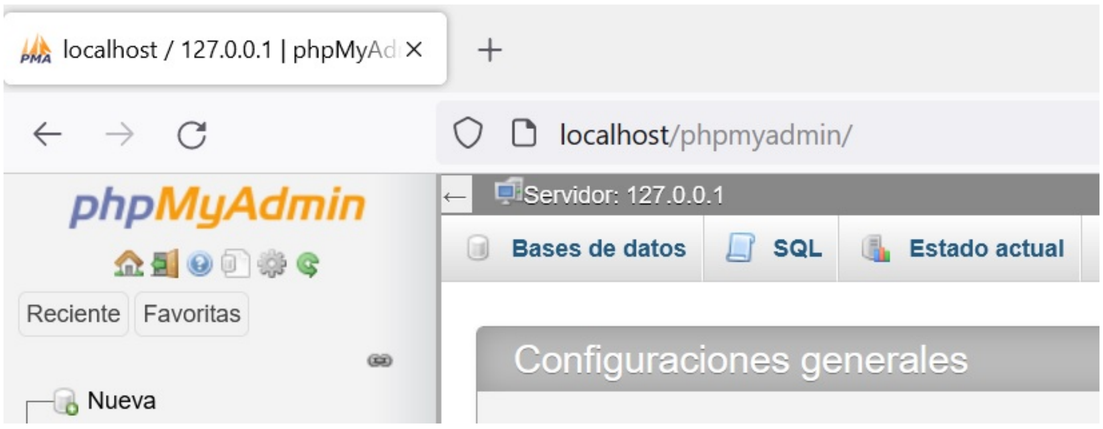


Figure 17: phpmyAdmin output

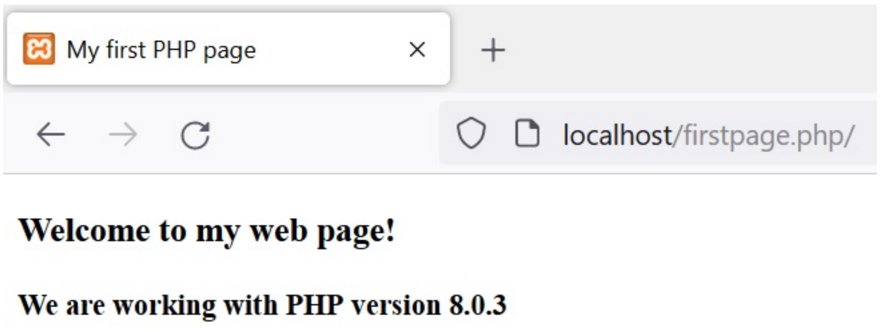
3.4.4 Our first web application using PHP

Assuming all the services are running properly, let's create our first web page with this content:

```
<!DOCTYPE html>
<html>
    <head>
        <title>My first PHP page</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <h3>Welcome to my web page!</h3>
        <h4>
        We are working with PHP version
        <?php echo phpversion(); ?>
        </h4>
    </body>
</html>
```

Save the file as **firstpage.php** in the root folder of the web server (`c:\xampp\htdocs` in Windows, `/var/www` or `/var/www/html` in Linux).

Now type this URL in your browser: **`http://localhost/firstpage.php`**. You should see your first PHP page looking like this:



Welcome to my web page!

We are working with PHP version 8.0.3

Figure 18: Your first PHP application

4 Code editors and IDEs

What's the difference within a **code editor** and a **IDE (Integrated Development Environment)**? A **code editor** is basically a text editor but it is also designed to help you write code. It helps you color your code and provides you more advanced tools to make coding easier for you. These features simplify and accelerate the process of editing and help you write better software programs by identifying problem areas and debugging code. There are a myriad of code editors out there, such as Atom, Sublime Text, Visual Studio Code, etc.

An **IDE**, short for **Integrated Development Environment**, is one of the most powerful programming tools that combine the different aspects of a computer program into a single GUI. It is a robust software environment that consolidates many of the functions like code creation, building and testing, together in a single framework, service or application. What it does is it simplifies the whole software development process by allowing management and deployment of multiple tools at once. An IDE is a text editor, a code editor, a debugger, compiler and more all under a single tool belt.

In this course we'll start with a code editor, **Visual Studio Code**, but we will introduce an IDE, **PhpStorm**, later. However, you can do the practices with the tool of your choice.

4.1 Installing VSCode

On **Linux** (Xubuntu 24.04), type in a terminal:

```
sudo snap install code --classic
```

On **Windows**, simply download the installer and run it:

https://code.visualstudio.com/download

Once installed,we're going to install some extensions in order to work better with PHP. Click on the *Extensions* button in the left bar and search for *"php"*. You'll see a lot of PHP extensions. The most useful are:

- **PHP Intelephense**: for code completion, classes navigation, methods signature help, formatting, etc.
- **PHP Namespace Resolver** by Mehedi Hassan: for working with namespaces.
- **PHP Getters & Setters** (CV fork): for adding getters and setters methods automatically.
- **PHP Refactor Tool**: allows us to refactor easily our project.
- **Project Manager**: generic tool for manage projects.

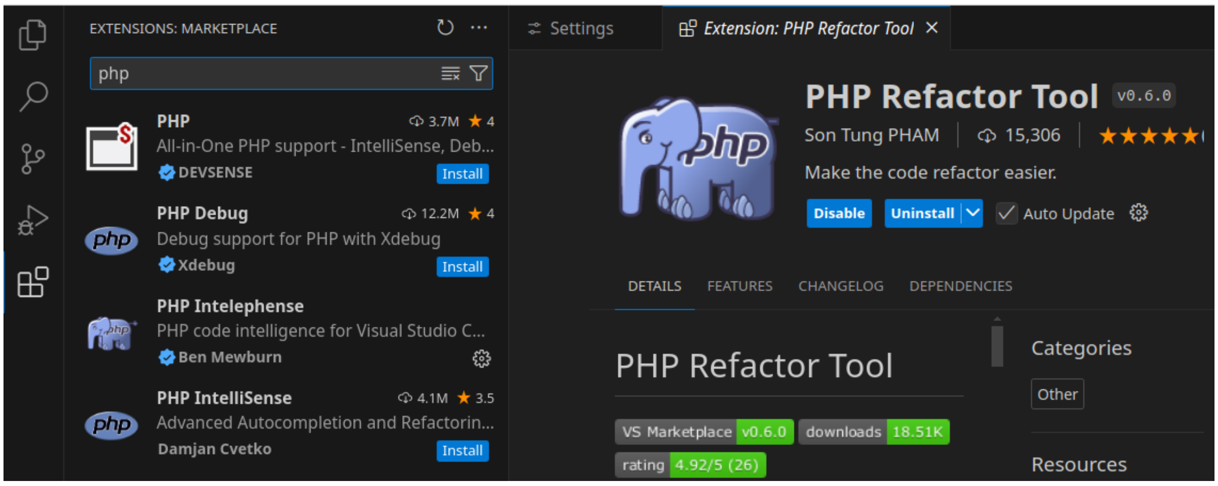

EXTENSIONS: MARKETPLACE
php
PHP 3.7M ★ 4
All-in-One PHP support - IntelliSense, Deb...
DEVSENSE Install
PHP Debug 12.2M ★ 4
Debug support for PHP with Xdebug
Xdebug Install
PHP Intelephense
PHP code intelligence for Visual Studio C...
Ben Mewburn
PHP IntelliSense 4.1M ★ 3.5
Advanced Autocompletion and Refactorin...
Damjan Cvetko Install
Settings
Extension: PHP Refactor Tool
PHP Refactor Tool v0.6.0
Son Tung PHAM 15,306
Make the code refactor easier.
Disable Uninstall Auto Update
DETAILS FEATURES CHANGELOG DEPENDENCIES
PHP Refactor Tool
VS Marketplace v0.6.0 downloads 18.51K
rating 4.92/5 (26)
Categories
Other
Resources

Figure 19: VSCode PHP extensions

4.2 Installing PhpStorm

The easiest way to install PhpStorm in Linux is via the JetBrains Toolbox App. However if you want to install only the IDE in a custom location, use the Standalone installation process.

To install the Toolbox App and the desired IDE in Xubuntu 24.04, follow the next instructions:

1. Download the **tarball `.tar.gz`** from the Toolbox App web page.
2. Extract the tarball to a directory that supports file execution.

   For example, you can extract it to the recommended `/opt` directory using the following command:

```
sudo tar -xzf jetbrains-toolbox-*.tar.gz -C /opt
```

3. Execute the **jetbrains-toolbox** binary from the extracted directory to run the Toolbox App.

```
sudo /opt/jetbrains-toolbox-*/jetbrains-toolbox
```

After you run the Toolbox App for the first time, it will automatically add the Toolbox App icon Toolbox App icon to the main menu.

4. Select the product that you want to install. To install a specific version, click App actions more and select **Available versions**.

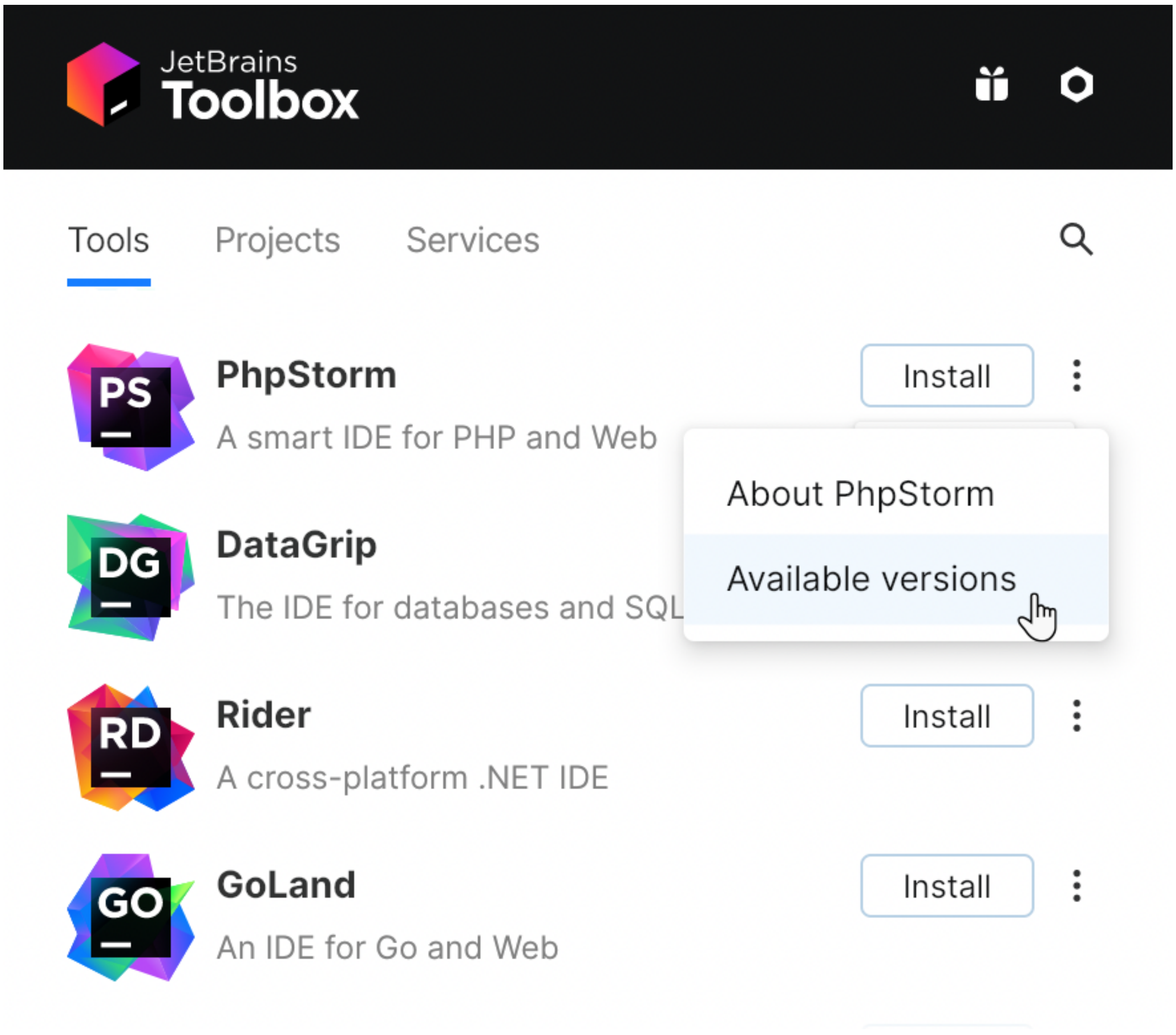


Figure 20: Toolbox App

Log in to your JetBrains Account from the Toolbox App, and it will automatically activate the available licenses for any IDE that you install.

To install the Toolbox App in **Windows**, simply download the installer `.exe` and run it.