

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

SISTEMA DE INFORMACIÓN WEB PARA LA OPTIMIZACIÓN DEL PROCESO DE GESTIÓN Y ADMINISTRACIÓN DE LOS LABORATORIOS DE INFORMÁTICA DE LA UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS - FACULTAD TECNOLÓGICA

MANUAL DEL PROGRAMADOR

Andrés Felipe Sánchez Osorio
Luis Alfredo Parra Rativa
Universidad Distrital Francisco José de Caldas
Facultad Tecnológica

Contenido

Lista de Figuras 1

INTRODUCCION 3

OBJETIVO 3

1. HERRAMIENTAS DE SOFTWARE 4

1.1 XAMPP 4

1.1.1 Definición de XAMPP 4

1.1.2 Instalación de XAMPP 4

2. EJECUCIÓN DEL APLICATIVO 14

2.1 Copia de proyecto a XAMPP 14

2.2 Activar servicios Apache y MySQL 15

2.3 Crear Base de Datos 19

2.4 Ingreso al aplicativo 23

3. ESTRUCTURA DEL PROYECTO 25

3.1 Uso de YAML en Symfony 25

3.2 Estructura de directorios 26

3.1.1 Directorio app/ 26

3.1.2 Directorio src/ 32

3.1.3 Directorio vendor/ 36

3.1.4 Directorio web/ 36

4. TECNOLOGIAS IMPORTANTES UTILIZADAS 38

4.1 Servidor 38

4.2 Cliente 38

Lista de Figuras

Figura 1. Comprobar servidor 4
Figura 2. Descargar XAMPP 5
Figura 3. Instalación XAMPP 6
Figura 4. Control cuentas de usuario 6
Figura 5. Instalación de XAMPP 7
Figura 6. Seleccionar paquetes 8
Figura 7. Seleccionar directorio 9
Figura 8. Instalación de XAMPP 10
Figura 9. Instalación de XAMPP 11
Figura 10. Progreso de instalación 12
Figura 11. Firewall de Windows 12
Figura 12. Finalizar instalación 13
Figura 13. Copiar proyecto 14
Figura 14. Ubicación del proyecto 15
Figura 15. Inicio de XAMPP 16
Figura 16. Control de XAMPP 17
Figura 17. Firewall de Windows 18
Figura 18. Activar servicios de XAMPP 19
Figura 19. Panel de phpMyAdmin 20
Figura 20. Crear nueva base de datos 20
Figura 21. Crear base de datos 21
Figura 22. Importar datos 21
Figura 23. Menú de importación 22
Figura 24. Lista de tablas 23
Figura 25. Página principal 24
Figura 26. Configuración del Sistema 24
Figura 27. Estructura de directorios 26
Figura 28. Estructura app/ 27
Figura 29. Estructura app/cache/ 27
Figura 30. Estructura app/config/ 28
Figura 31. Importación de config.yml 28
Figura 32. Parámetros de configuración 29
Figura 33. Archivo routing.yml 29
Figura 34. Archivos de configuración easyadmin 30
Figura 35. Archivo basic.yml de EasyAdmin 31
Figura 36. Estructura configuración de entidades 31
Figura 37. Estructura app/logs/ 32
Figura 38. Estructura app/Resources/ 32
Figura 39. Estructura src/ 33

Figura 40. Estructura src/Controller/ 33
Figura 41. Controlador 34
Figura 42. Estructura src/Entity/ 34
Figura 43. Inicio entidad Asignatura 35
Figura 44. Estructura src/EventListener/ 35
Figura 45. Estructura src/Form/ 35
Figura 46. Estructura src/Repository/ 36
Figura 47. Estructura vendor/ 36
Figura 48. Estructura web/ 37

INTRODUCCION

El presente manual del usuario programador, tiene como finalidad dar a conocer de una manera detallada y sencilla la estructura del sistema de información web para que un usuario programador en el futuro pueda modificar este sistema con mayor facilidad.

OBJETIVO

Este manual le permitirá entender la estructura de un proyecto con Symfony, como instalar el sistema en un entorno XAMPP, también le dará una guía sobre las tecnologías más importantes usadas en este proyecto.

Además brindar una descripción detallada y clara sobre el funcionamiento del sistema.

1. HERRAMIENTAS DE SOFTWARE

Para ejecutar el sistema de información web en un equipo de desarrollo local, hay que instalar el paquete XAMPP.

1.1 XAMPP

1.1.1 Definición de XAMPP

XAMPP es un paquete de instalación independiente de plataforma, software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MariaDB, PHP, Perl. Desde la versión "5.6.15", XAMPP cambió la base de datos de MySQL a MariaDB. El cual es un fork de MySQL con licencia GPL.

El programa se distribuye bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y Mac OS X.

1.1.2 Instalación de XAMPP

Nota: Antes de instalar un servidor de páginas web, es conveniente comprobar si no hay ya uno instalado. Para ello, es suficiente con abrir el navegador y escribir la dirección http://localhost. Si no se obtiene un mensaje de error es que hay algún servidor de páginas web instalado (ver figura 1).

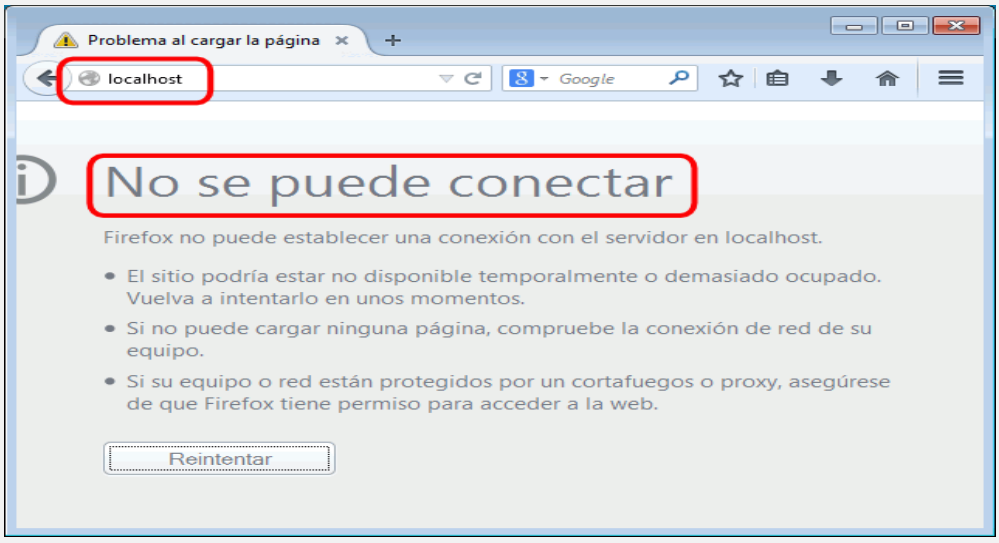


Figura 1. Comprobar servidor
Fuente: Autor

El primer paso es descargar el paquete de instalación el cual se puede obtener del siguiente link: https://www.apachefriends.org/es/download.html . Seleccionar el paquete según el sistema operativo (ver figura 2).



Figura 2. Descargar XAMPP
Fuente: Autor

Al poner en marcha el instalador de XAMPP muestra dos avisos:

- El primero aparece si en el ordenador hay instalado un antivirus (ver figura 3):

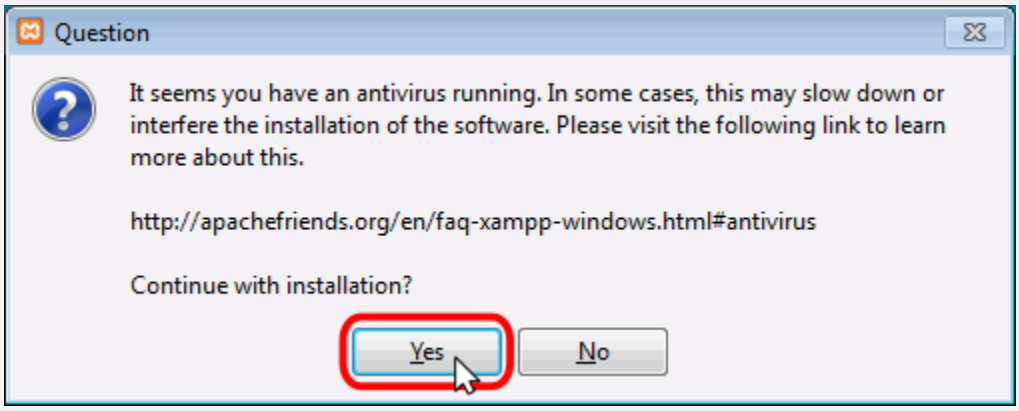


Figura 3. Instalación XAMPP
Fuente: Autor

- El segundo aparece si está activado el Control de Cuentas de Usuario y recuerda que algunos directorios tienen permisos restringidos (ver figura 4):

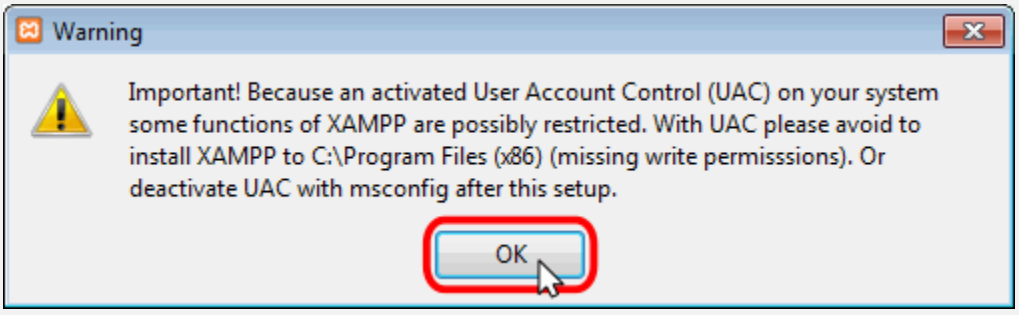


Figura 4. Control cuentas de usuario
Fuente: Autor

A continuación, se inicia el asistente de instalación. Para continuar, hay que hacer clic en el botón "**Next**" (ver figura 5):

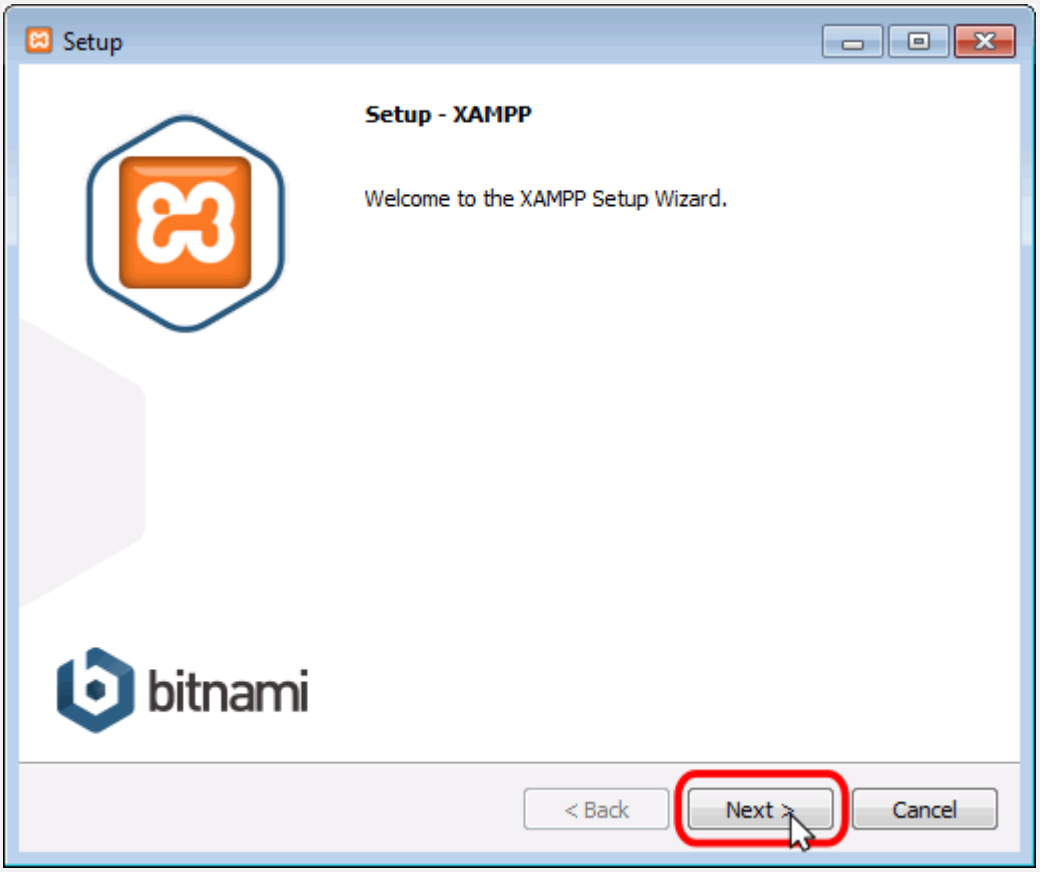


Figura 5. Instalación de XAMPP
Fuente: Autor

Los componentes mínimos que instala XAMPP son el servidor Apache y el lenguaje PHP, pero XAMPP también instala otros elementos. En la pantalla de selección de componentes puede elegirse la instalación o no de estos componentes, pero para poder ejecutar este proyecto se necesita al menos instalar MySQL y phpMyAdmin (ver figura 6):

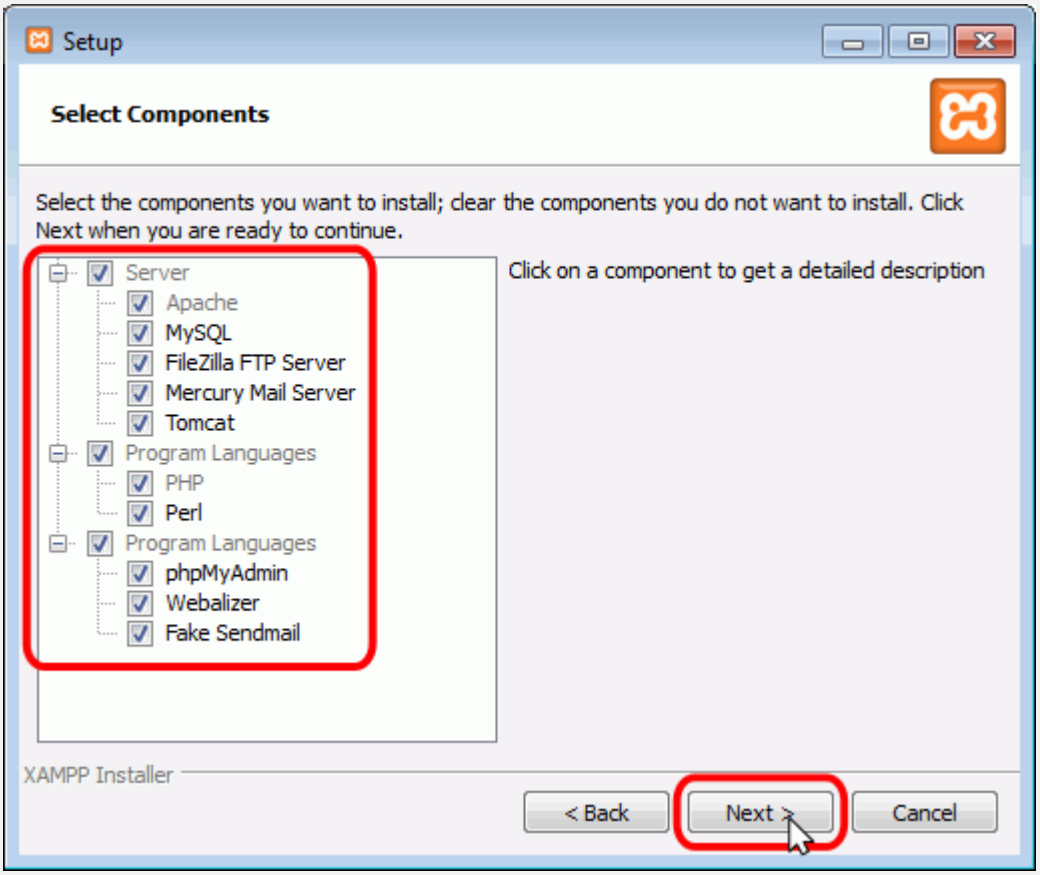


Figura 6. Seleccionar paquetes
Fuente: Autor

En la siguiente pantalla se puede elegir la carpeta de instalación de XAMPP. La carpeta de instalación predeterminada es C:\xampp. Si se quiere cambiar, hay que hacer clic en el icono de carpeta y seleccionar la carpeta donde se quiere instalar XAMPP. Para continuar la configuración de la instalación, hay que hacer clic en el botón "**Next**" (ver figura 7):

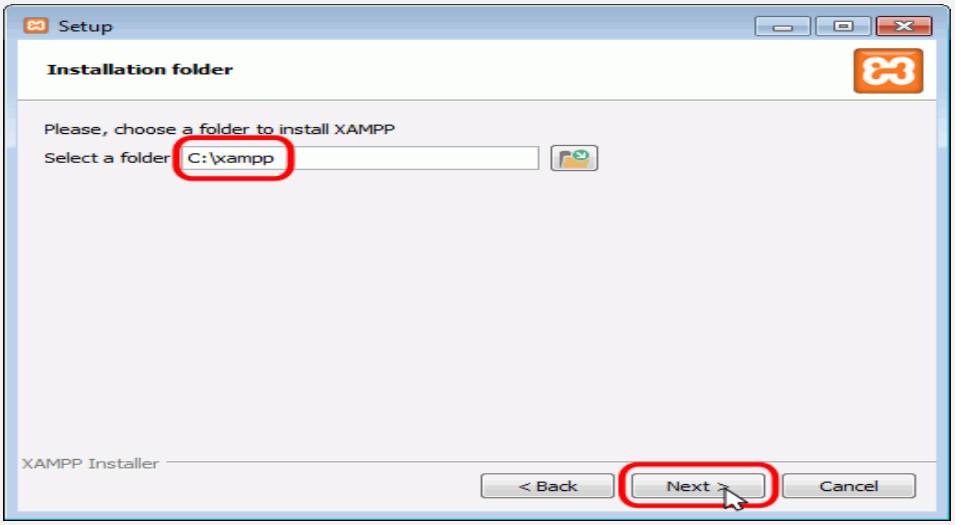


Figura 7. Seleccionar directorio
Fuente: Autor

La siguiente pantalla nos ofrece información sobre los instaladores de aplicaciones para XAMPP creados por Bitnami. Para que no se abra la página web de Bitnami, habría que desmarcar la casilla correspondiente (ver figura 8):

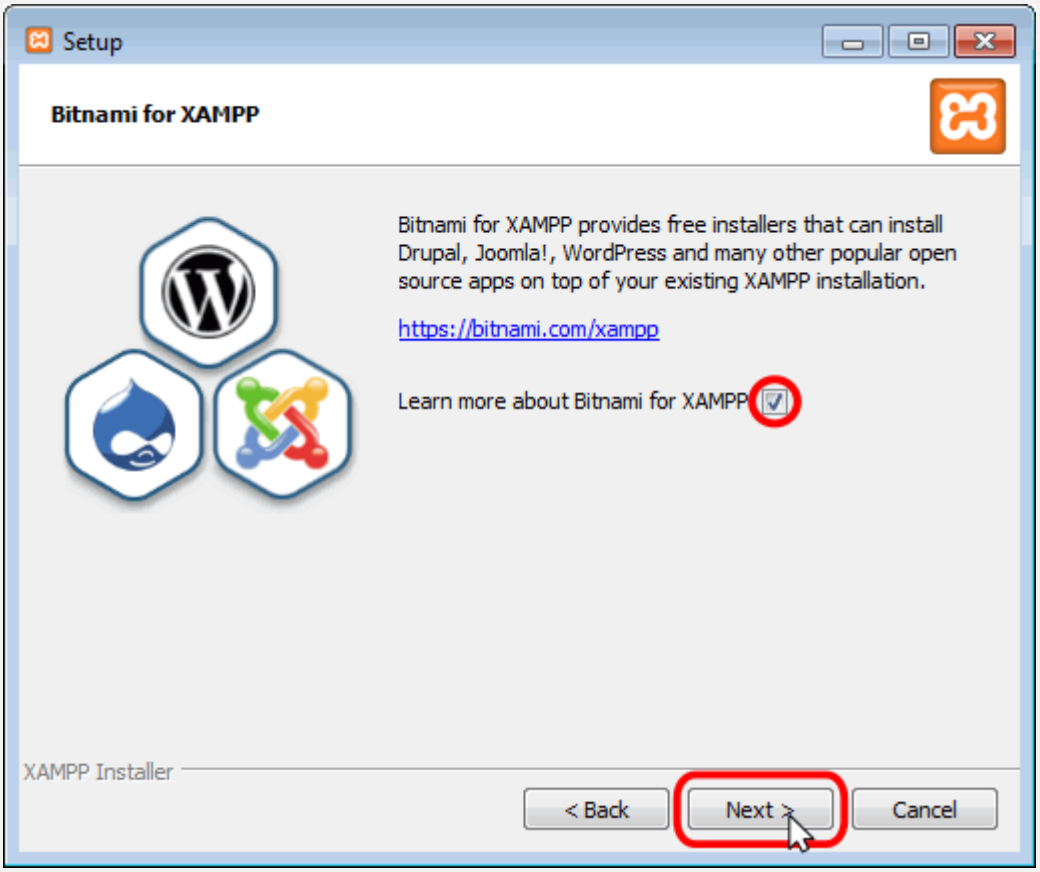


Figura 8. Instalación de XAMPP
Fuente: Autor

Para empezar la instalación de XAMPP, hay que hacer clic en el botón "**Next**" en la pantalla siguiente (ver figura 9):

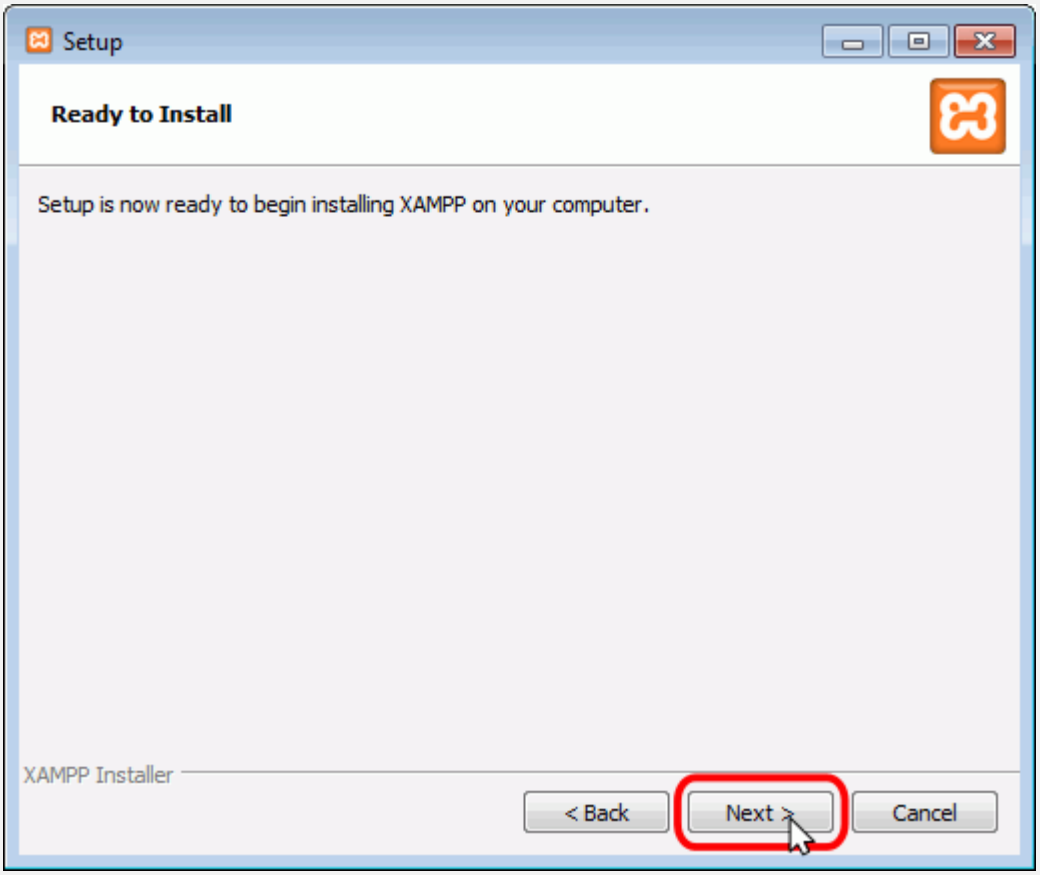


Figura 9. Instalación de XAMPP
Fuente: Autor

A continuación, se inicia el proceso de copia de archivos, que puede durar algunos minutos dependiendo del equipo host (ver figura 10):

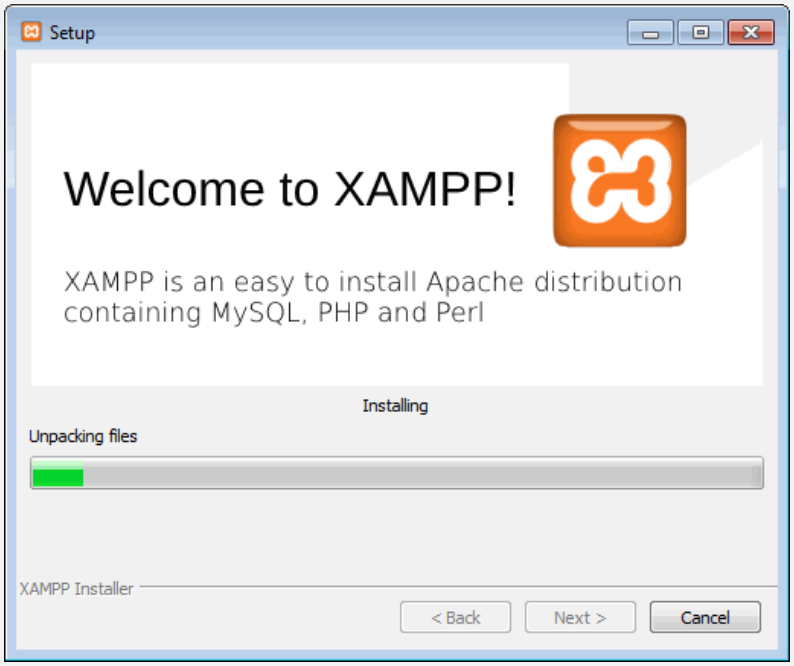


Figura 10. Progreso de instalación
Fuente: Autor

Durante la instalación, si en el ordenador no se había instalado Apache anteriormente, se mostrará un aviso del cortafuegos de Windows para autorizar a Apache para comunicarse en las redes domésticas o de trabajo, lo que debemos permitir haciendo clic en el botón "Permitir acceso" (ver figura 11):

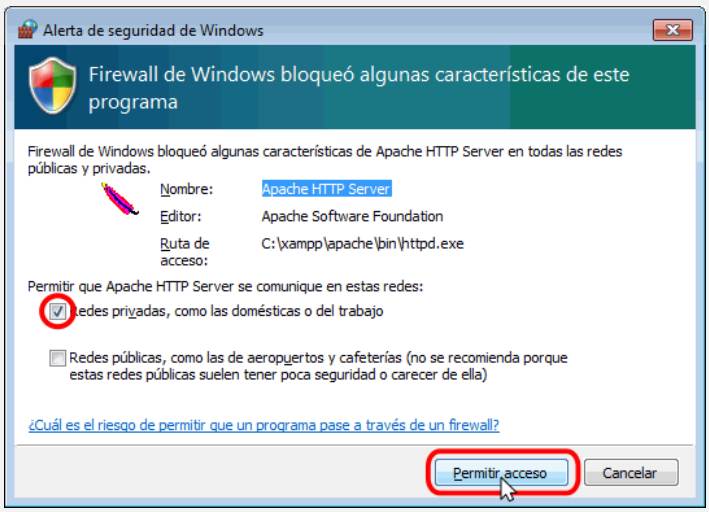


Figura 11. Firewall de Windows
Fuente: Autor

Una vez terminada la copia de archivos, se muestra la pantalla que confirma que XAMPP ha sido instalado. Hay que hacer clic en el botón "Finish". Para no abrir a continuación el panel de control de XAMPP habría que desmarcar la casilla correspondiente (ver figura 12):

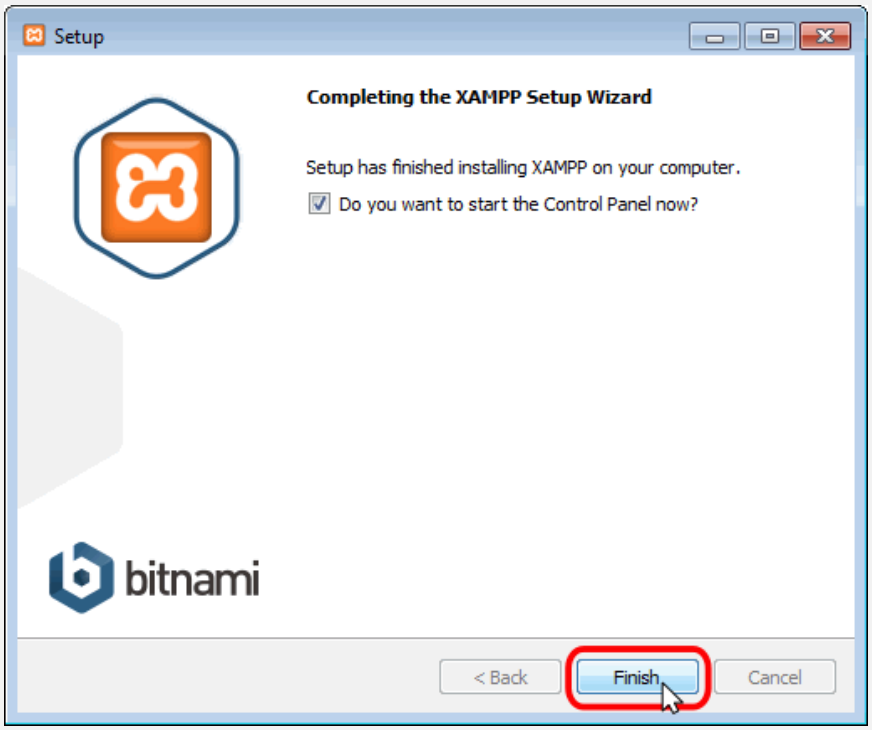


Figura 12. Finalizar instalación
Fuente: Autor

2. EJECUCIÓN DEL APLICATIVO

2.1 Copia de proyecto a XAMPP

Ingresar a la carpeta "**APLICATIVO**" dentro del DVD, luego a la capeta "**CÓDIGO FUENTE**" y copiar la carpeta "**labs**" (ver figura 13):

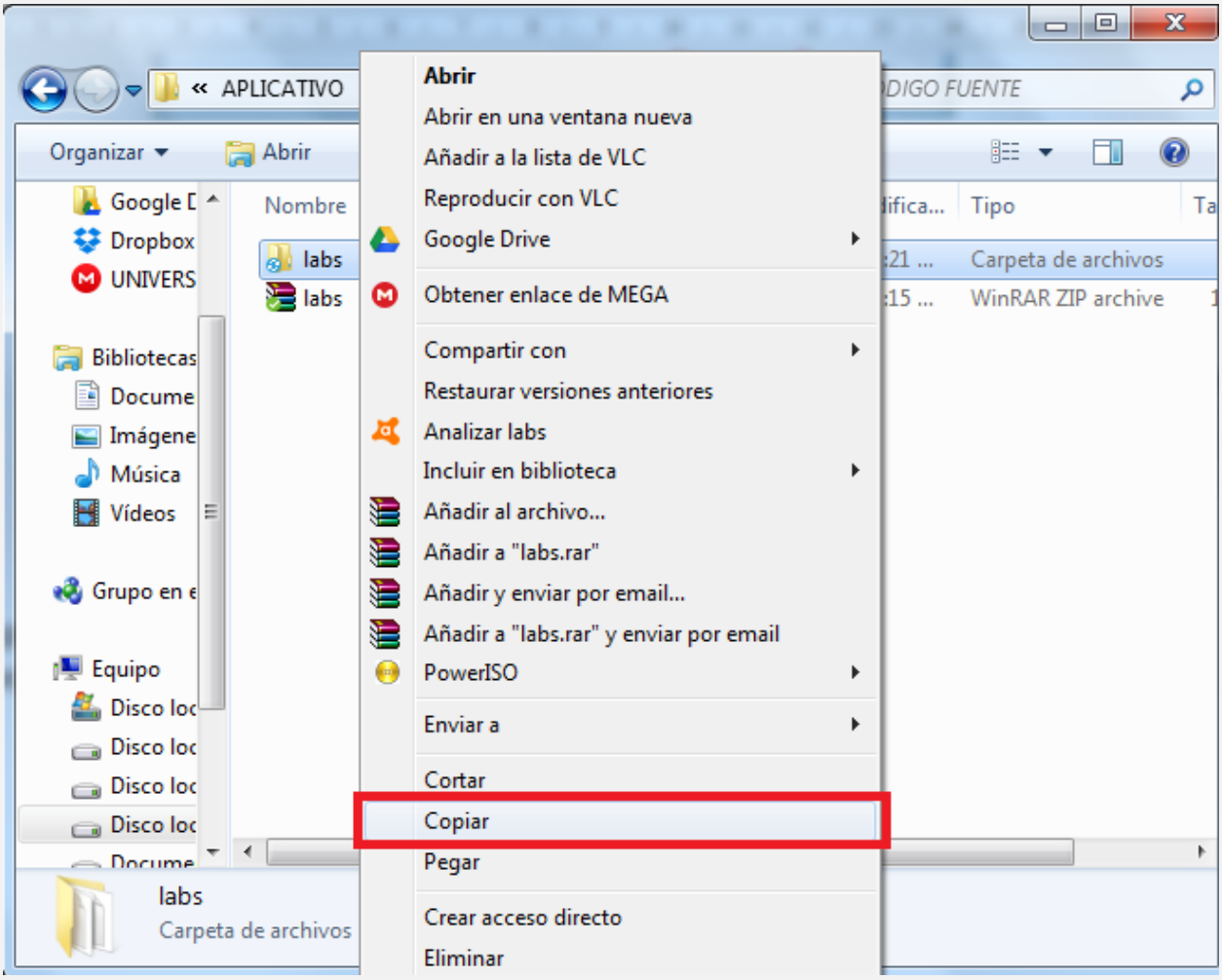


Figura 13. Copiar proyecto
Fuente: Autor

Dirigirse a la carpeta de instalación de XAMPP, por lo general **C:\xampp**, y pegamos el archivo "**labs**" dentro de la carpeta "**htdocs**" (ver figura 14):

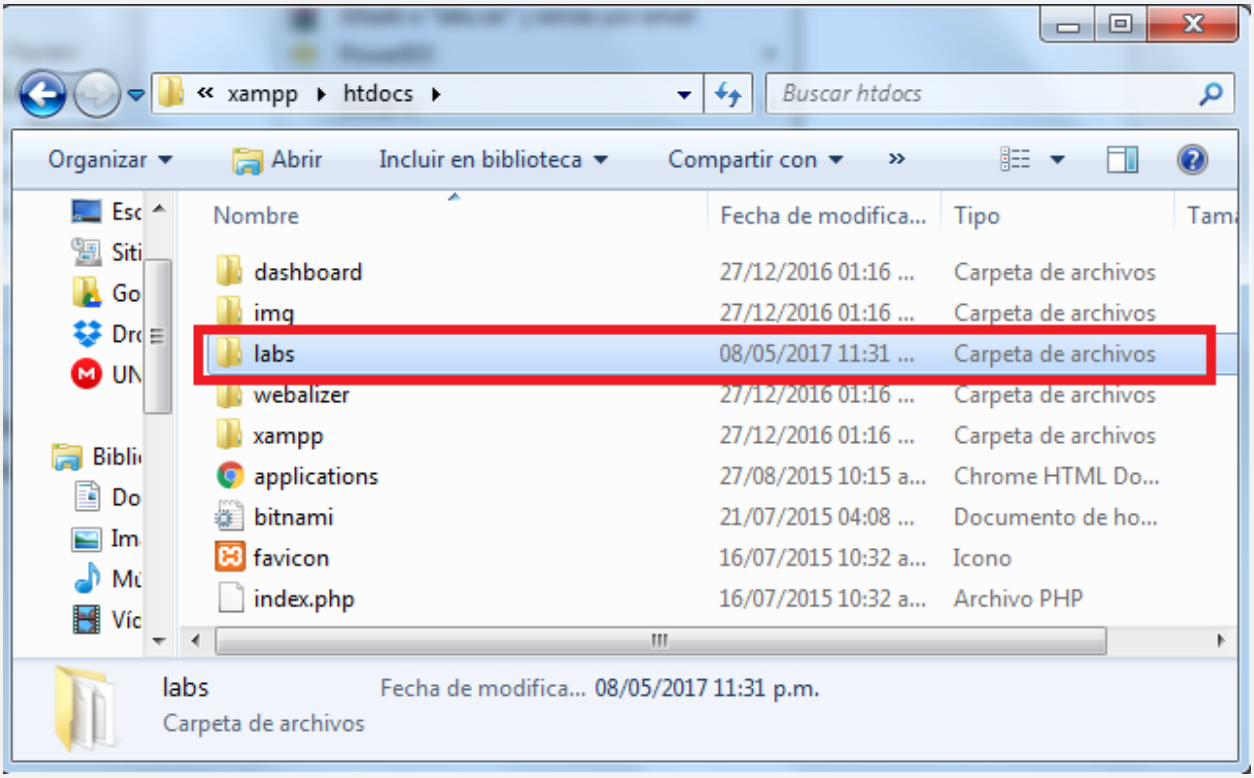


Figura 14. Ubicación del proyecto
Fuente: Autor

2.2 Activar servicios Apache y MySQL

Una vez copiada la carpeta "**labs**" en la carpeta "**htdocs**" de XAMPP, procedemos a iniciar los servicios Apache y MySQL. Para esto, nos dirigimos a la barra de búsqueda de Windows y buscamos "XAMPP Control Panel" y damos clic para ejecutar (ver figura 15):

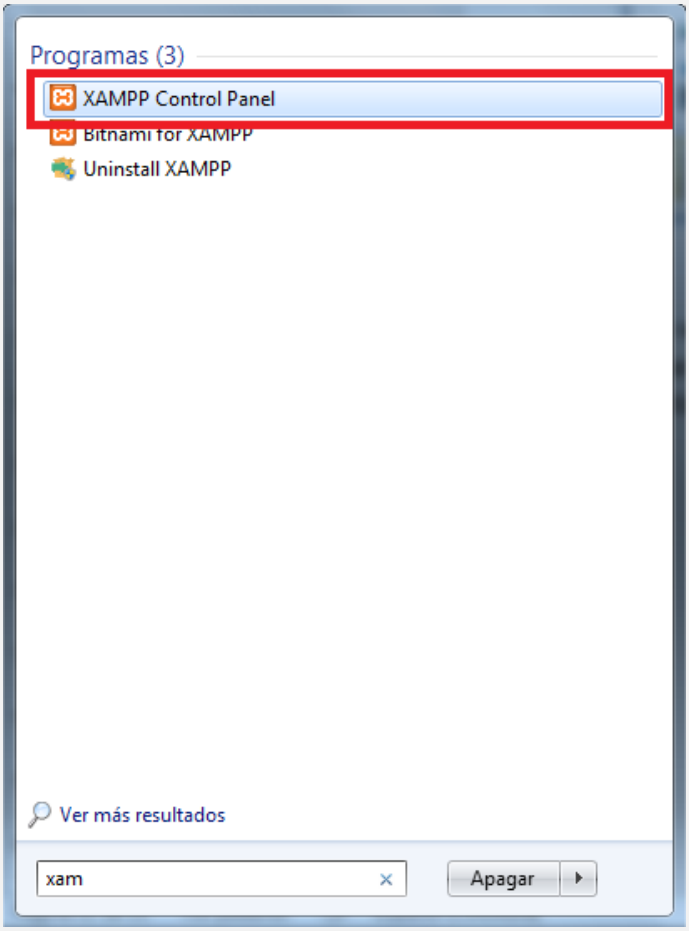


Figura 15. Inicio de XAMPP
Fuente: Autor

El panel de control que se muestra es el siguiente (ver figura 16):

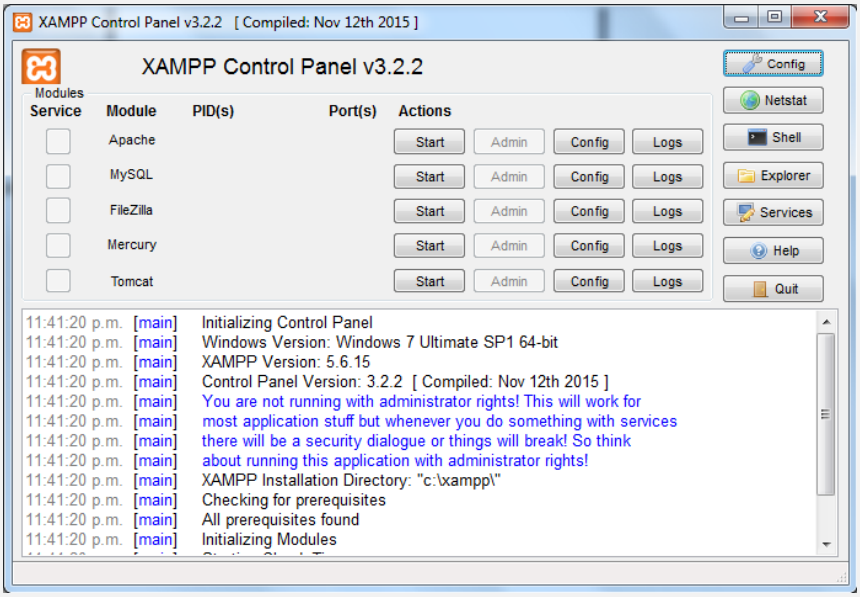


Figura 16. Control de XAMPP
Fuente: Autor

Cuando se pone en marcha por primera vez cualquiera de los servidores que instala XAMPP, el cortafuegos de Windows pide al usuario confirmación de la autorización. Por ejemplo, la primera vez que se pone en marcha Apache mediante el botón Start correspondiente. Para poder utilizarlo hace falta al menos autorizar el acceso en redes privadas (ver figura 17):

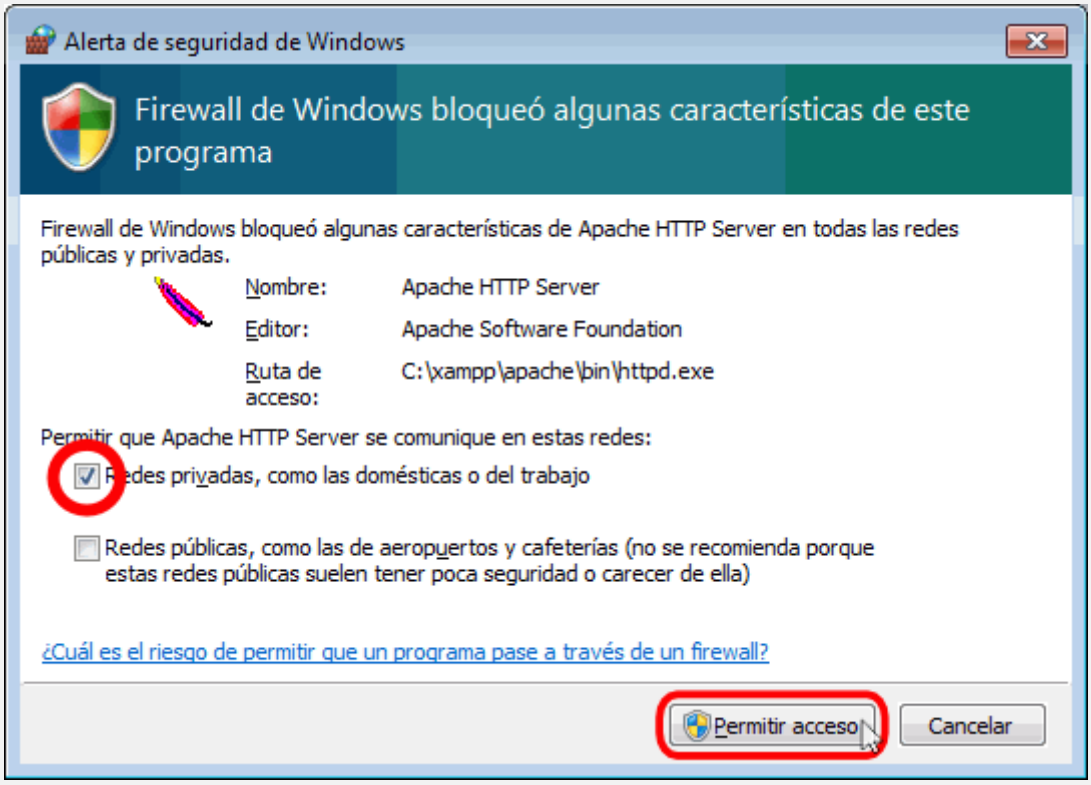


Figura 17. Firewall de Windows
Fuente: Autor

Si el arranque de Apache y MySQL tiene éxito, el panel de control mostrará el nombre del módulo con fondo verde, su identificador de proceso, los puertos abiertos (http y https), el botón "Start" se convertirá en el botón "Stop" y en la zona de notificación se verá el resultado de las operaciones realizadas (ver figura 18):

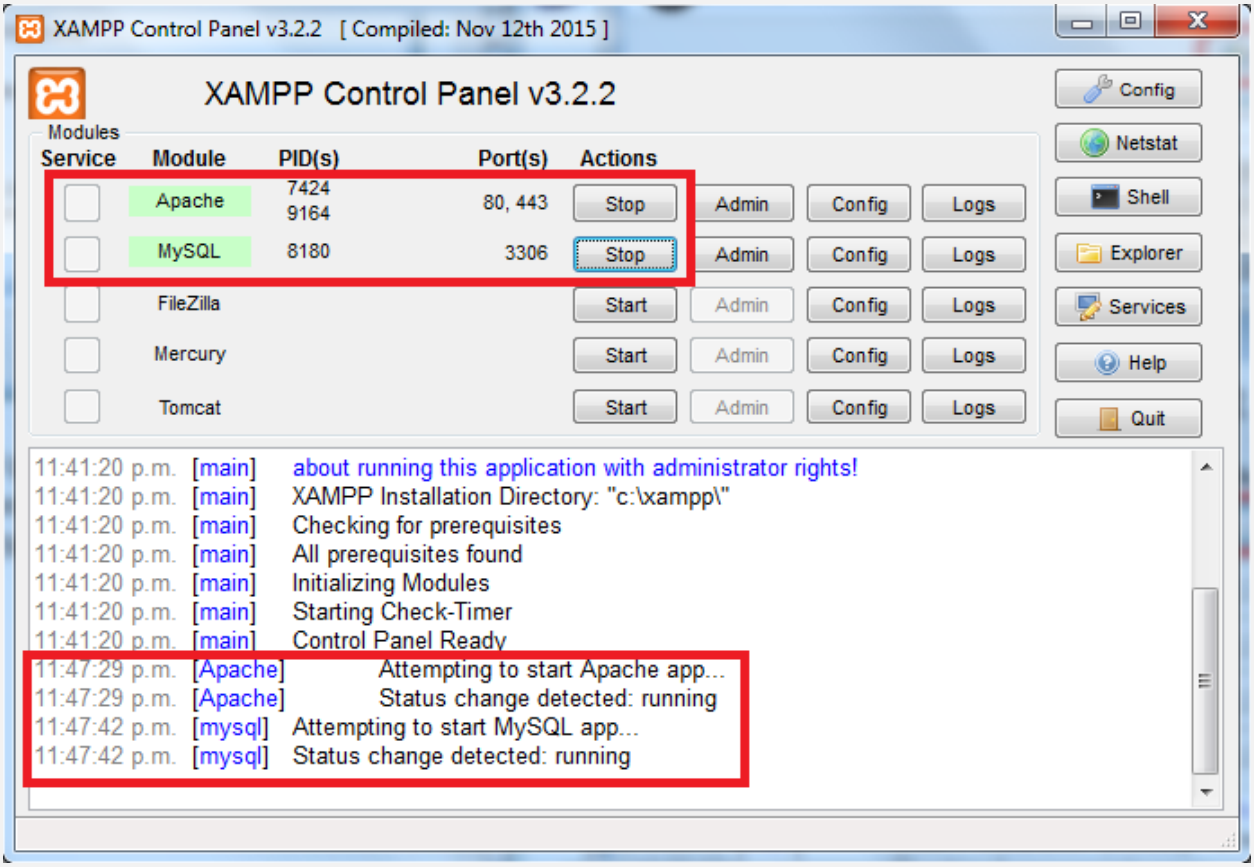


Figura 18. Activar servicios de XAMPP
Fuente: Autor

2.3 Crear Base de Datos

Una vez iniciados los servicios Apache y MySQL, nos dirigimos a uno de los exploradores de internet instalados y entramos al link http://localhost el cual nos dirigirá al panel de administración web de XAMPP.

En la parte superior derecha de la página ingresamos a la opción "**phpMyAdmin**" la cual nos abre la siguiente página (ver figura 19):

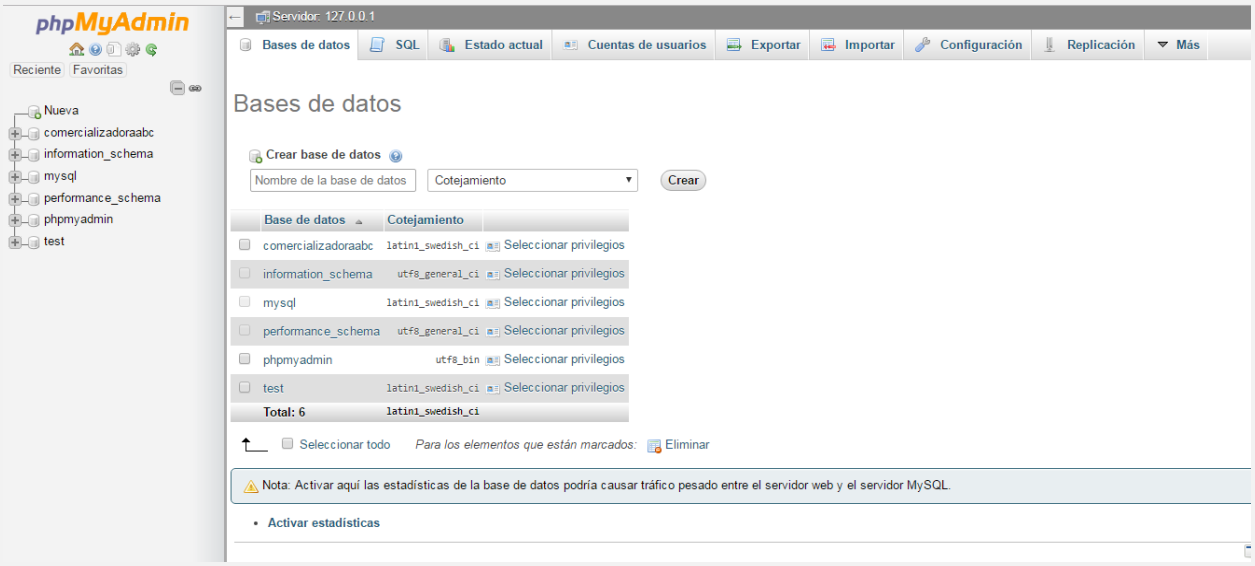


Figura 19. Panel de phpMyAdmin
Fuente: Autor

En el panel izquierdo damos clic en la opción "**Nueva**" para crear una nueva base de datos (ver figura 20):

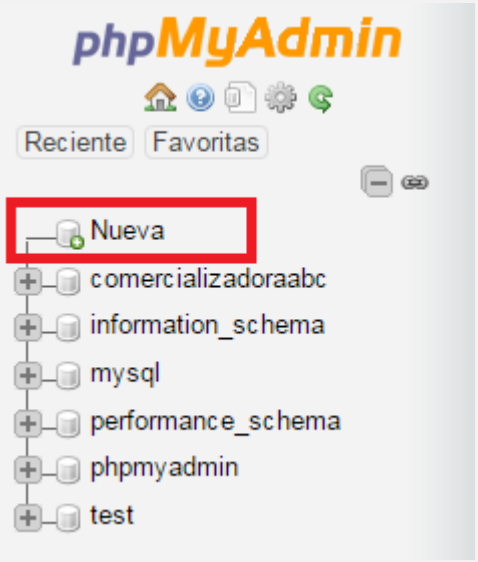


Figura 20. Crear nueva base de datos
Fuente: Autor

Asignar como nombre a la nueva base de datos “**labs**” y dar clic en “**Crear**” (ver figura 21):

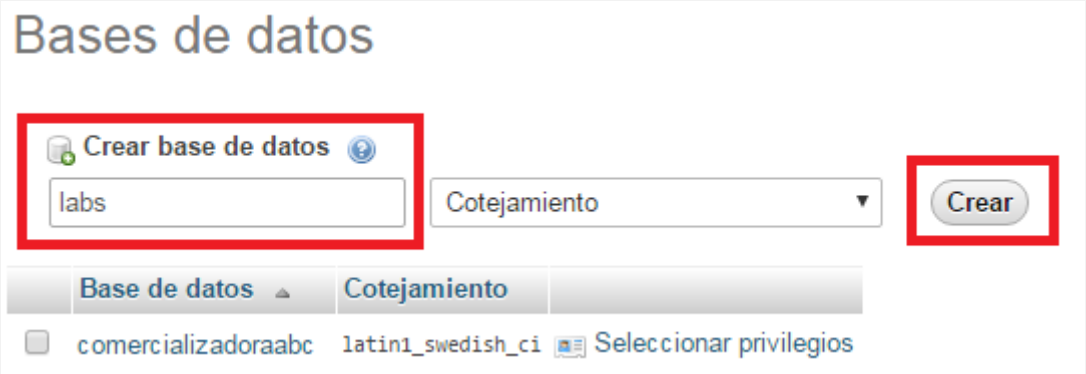


Figura 21. Crear base de datos
Fuente: Autor

Luego nos dirigimos al panel derecho donde damos clic en la opción “**Importar**” (ver figura 22):

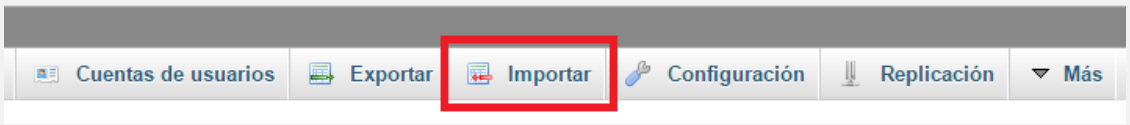


Figura 22. Importar datos
Fuente: Autor

A continuación, nos dirigimos a la opción “**Seleccionar archivo**” y seleccionaremos el archivo ubicado en la ruta “**APLICATIVO\BASE DE DATOS\labs.sql”** dentro del DVD del aplicativo (ver figura 23):

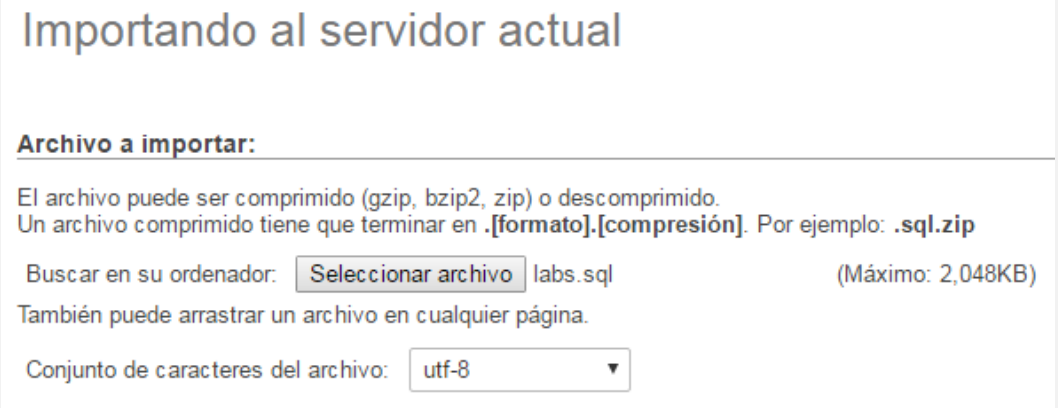


Figura 23. Menú de importación
Fuente: Autor

Y en la parte inferior de la página damos clic en "**Continuar**".

Si la operación fue exitosa, en el panel izquierdo se desplegarán todas las tablas asociadas creadas (ver figura 24):

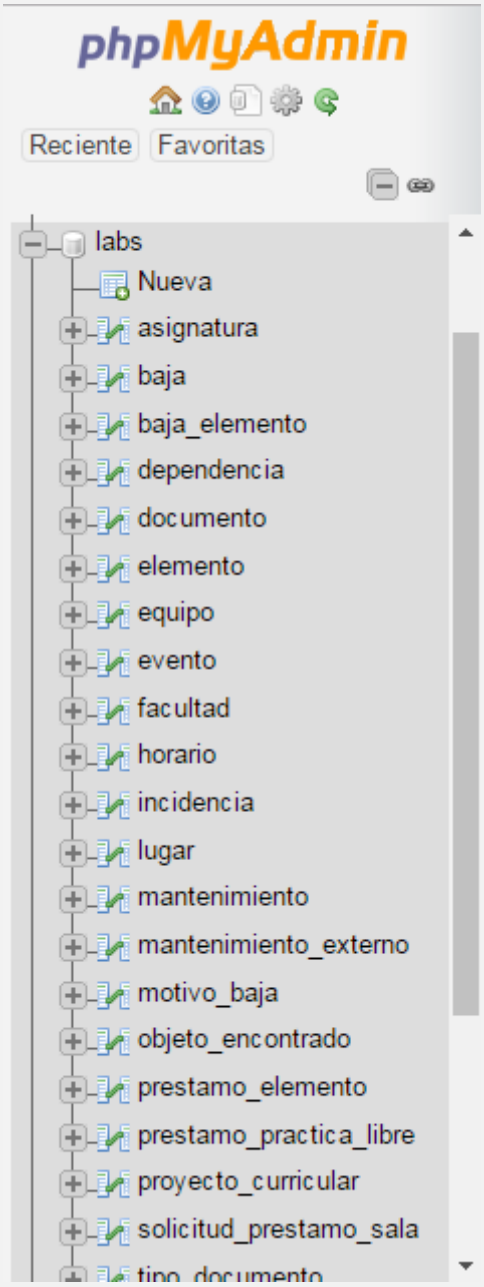


Figura 24. Lista de tablas
Fuente: Autor

2.4 Ingreso al aplicativo

Una vez cargada la base de datos, ingresar en el explorador la siguiente dirección: http://localhost/labs/web/app_dev.php/ la cual nos cargara el sistema de información web (ver figura 25):

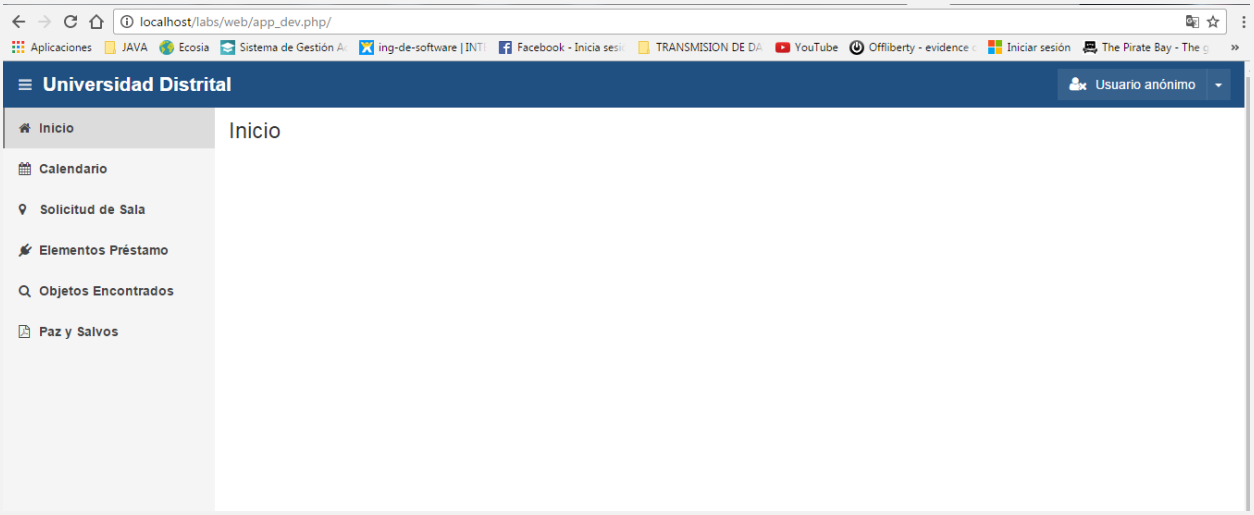


Figura 25. Página principal
Fuente: Autor

Nota: Es importante dirigirse a la ruta "**C:\xampp\htdocs\labs\app\config**" y abrir el archivo "**parameters.yml**" con block de notas para ingresar el puerto de conexión de MySQL (el que aparece en XAMPP), el nombre de usuario de la base de datos y la contraseña del usuario (ver figura 26)**.**

```
1  # This file is auto-generated during the composer install
2  parameters:
3      database_host: 127.0.0.1
4      database_port: 3307
5      database_name: labs
6      database_user: root
7      database_password: gladiador
8      mailer_transport: smtp
9      mailer_host: 127.0.0.1
10     mailer_user: null
11     mailer_password: null
12     secret: ThisTokenIsNotSoSecretChangeIt
13     recaptcha_site_key: 6LerVhMUAAAAAOjIMBLok_IDNboeKE0pqWsJBec9
14     recaptcha_secret: 6LerVhMUAAAAAE_Qko5maKK_bZ1eNoEwLm6EsUXY
15
```

Figura 26. Configuración del Sistema
Fuente: Autor

3. ESTRUCTURA DEL PROYECTO

Symfony es un proyecto open source publica pequeñas librerías PHP independientes llamadas componentes o bundles. Combinando varios de esos componentes, también ha publicado un framework para desarrollar aplicaciones web.

3.1 Uso de YAML en Symfony

Según la Wikipedia: YAML es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl, así como el formato para correos electrónicos especificado por el RFC 2822.

Los archivos de configuración de Symfony se pueden escribir en PHP, en XML o en YAML. En este proyecto se decidió usar YAML puesto que es el método más sencillo para generar la configuración principal de la aplicación.

A continuación, se listan los cinco conceptos básicos para entender la sintaxis de YAML:

- ✓ Nunca se debe incluir tabulación en un archivo YAML. Para indentar la información, se debe usar la barra espaciadora.

- ✓ La información se indica mediante pares `clave: valor`. Si la clave o el valor tienen espacios en blanco, se encierran con comillas simples o dobles indistintamente.

- ✓ La jerarquía de la información se establece escribiendo cuatro espacios en blanco por delante del par `clave: valor`.

- ✓ Los arrays normales se indican con los corchetes `[` y `]` y los arrays asociativos con las llaves `{` y `}`.

- ✓ Los comentarios se indican prefijando el carácter `#` por delante de cada una de sus líneas.

3.2 Estructura de directorios

Cualquier proyecto desarrollado con el framework Symfony está estructurado de la siguiente manera (ver figura 27):

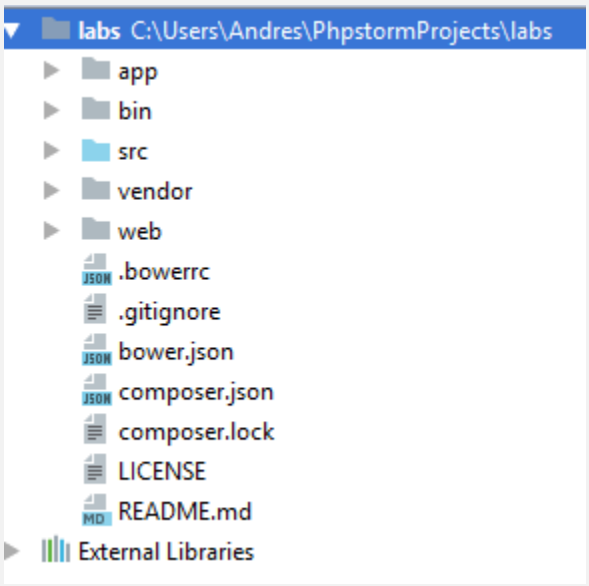


Figura 27. Estructura de directorios
Fuente: Autor

3.1.1 Directorio app/

Este directorio es el encargado de alojar los scripts PHP que realizan el proceso de carga del framework, y también todo lo que tiene que ver con la configuración general de la aplicación. Durante el desarrollo de una aplicación, normalmente el directorio app/ solo se utiliza para modificar la configuración (ver figura 28).

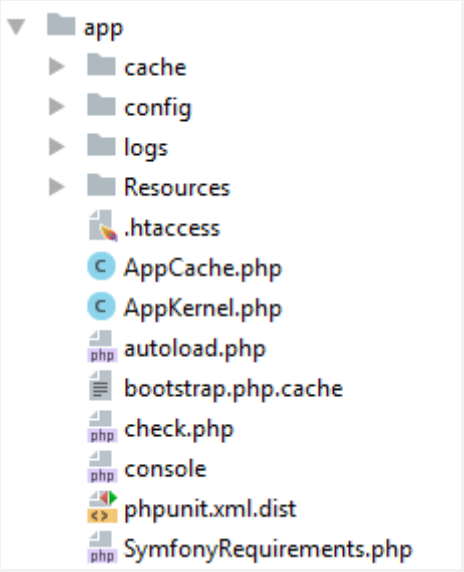


Figura 28. Estructura app/
Fuente: Autor

Directorio app/cache/: El directorio cache es encargado de guardar una copia simple del código PHP que ha sido compilado después de la primera ejecución de la aplicación, esto hace que una aplicación hecha con el framework Symfony nos provea una velocidad bastante sorprendente. En el entorno de desarrollo, Symfony2 es lo suficientemente inteligente como para vaciar la caché cuando se cambia un archivo. Pero en el entorno de producción, es nuestra responsabilidad borrar la caché cuando se actualiza el código o la configuración (ver figura 29).

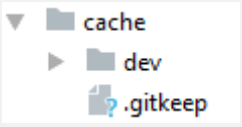


Figura 29. Estructura app/cache/
Fuente: Autor

Directorio app/config/: En este directorio se encuentran los archivos de configuración de la aplicación: **config.yml, parameters.yml, routing.yml, security.yml** y **services.yml**. El sistema de configuración de Symfony permite trabajar con distintos entornos de ejecución. Los más típicos son prod, para producción y dev, para desarrollo (ver figura 30).

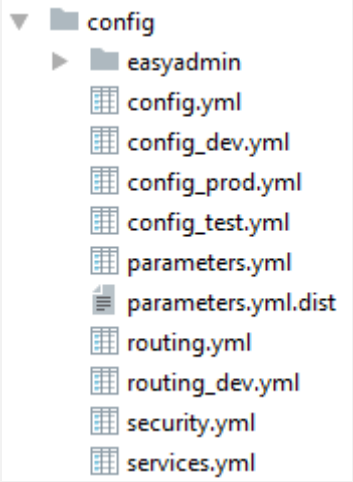


Figura 30. Estructura app/config/
Fuente: Autor

En el archivo **config.yml** se incluyen todas las configuraciones de los bundles que se usan en nuestro proyecto, en este mismo archivo se importan los otros archivos YAML, lo que hace a config.yml el archivo de configuración principal (ver figura 31).

```
imports:
    - { resource: parameters.yml }
    - { resource: security.yml }
    - { resource: services.yml }
    - { resource: easyadmin/ }
```

Figura 31. Importación de config.yml
Fuente: Autor

En el archivo **parameters.yml** como su nombre lo indica guardamos todos los parámetros específicos de nuestra aplicación, tales como: dirección del servidor de bases de datos, puerto del mismo, usuario, contraseña, nombre de la base de datos, entre otros (ver figura 32).

```
parameters:
    database_host: 127.0.0.1
    database_port: null
    database_name: symfony
    database_user: root
    database_password: null
    mailer_transport: smtp
    mailer_host: 127.0.0.1
    mailer_user: null
    mailer_password: null
    secret: ThisTokenIsNotSoSecretChangeIt
    recaptcha_site_key: recaptcha_site_key
    recaptcha_secret: recaptcha_secret
```

Figura 32. Parámetros de configuración
Fuente: Autor

Nuestro archivo **routing.yml** nos indica la configuración de las rutas de nuestra aplicación web, este archivo es muy importante puesto que dará acceso al usuario final a nuestra aplicación. **Nota:** En este caso nuestra aplicación mapea las rutas por medio de anotaciones directamente en nuestros controladores (ver figura 33).

```
app:
    resource: "@AppBundle/Controller/"
    type:     annotation

fos_js_routing:
    resource: "@FOSJsRoutingBundle/Resources/config/routing/routing.xml"

adesigns_calendar:
    resource: "@ADesignsCalendarBundle/Resources/config/routing.xml"

easy_admin_bundle:
    resource: "@EasyAdminBundle/Controller/"
    type:     annotation
    prefix:   /admin
```

Figura 33. Archivo routing.yml
Fuente: Autor

En el archivo **security.yml** configuramos todo lo referente a seguridad de nuestra aplicación, se incluyen las restricciones de acceso, las entidades que usaremos para el

sistema de login de nuestra aplicación, el tipo de encriptación, entre otras configuraciones.

El archivo **services.yml** nos brinda una funcionalidad bastante importante de Symfony y es poder definir como su nombre lo indica, servicios, que son clases especiales que se pueden acceder desde cualquier parte de la aplicación.

Por otra parte, en nuestra carpeta **easyadmin/** encontramos la configuración referente al CRUD de las entidades, esto mediante el uso del bundle de Symfony llamado EasyAdminBundle (ver figura 34).

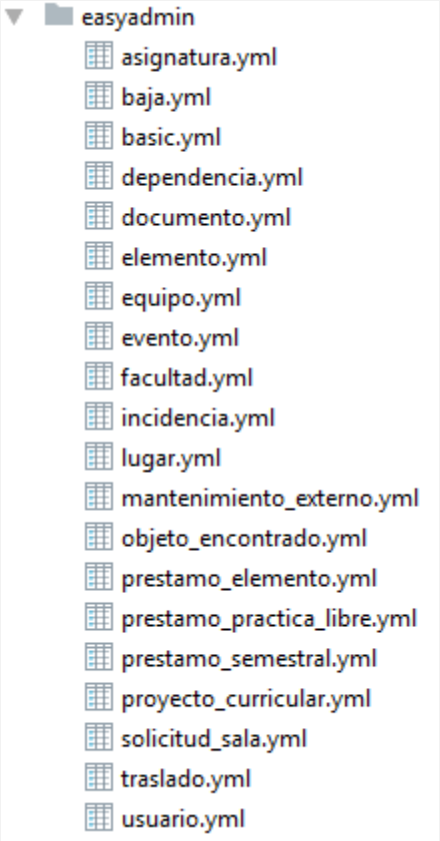


Figura 34. Archivos de configuración easyadmin
Fuente: Autor

El archivo de configuración principal de este bundle es **basic.yml** (ver figura 35).

```yaml
easy_admin:
    site_name: 'Universidad Distrital'

    formats:
        date:     'Y-m-d'
        time:     'H:i'
        datetime: 'Y-m-d H:i'

    list:
        actions:
            - { name: 'show', icon: 'eye', label: 'Ver' }
            - { name: 'edit', icon: 'edit', label: 'Editar' }
            - { name: 'delete', icon: 'remove', label: 'Borrar' }
```

Figura 35. Archivo basic.yml de EasyAdmin
Fuente: Autor

Cada uno de los otros archivos, **asignatura.yml**, **baja.yml**, etc.; nos proporcionan una configuración específica para cada entidad (ver figura 37).

```yaml
easy_admin:
    entities:
        Asignatura:
            class: AppBundle\Entity\Asignatura

            list:
                title: 'Asignaturas'
                sort: ['codigo', 'ASC']
                fields:
                    - 'codigo'
                    - 'nombre'

            show:
                fields:
                    - 'codigo'
                    - 'nombre'
```

Figura 36. Estructura configuración de entidades
Fuente: Autor

Directorio app/logs/: Cuando ocurre un error podemos encontrar los archivos de registro de peticiones en este directorio, permitiéndonos solucionar el problema de una manera más rápida (ver figura 37).

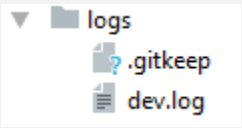


Figura 37. Estructura app/logs/
Fuente: Autor

Directorio app/Resources/: En este directorio se almacenan los recursos que se utilizan globalmente en el proyecto, tales como las vistas, los archivos de traducción, y archivos que se sobrescriben de otros bundles (ver figura 38).

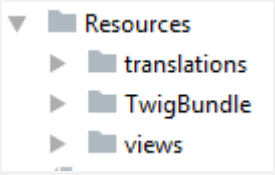


Figura 38. Estructura app/Resources/
Fuente: Autor

En el directorio **app/** también se incluyen dos archivos muy importantes para el framework, autoload.php y AppKernel.php.

En **autoload.php** se mapean los espacios de nombres contra los directorios en los que residirán las clases pertenecientes a dichos espacios de nombre.

En **AppKernel.php**, se declaran los bundles que se utilizarán en la aplicación.

Por último, en este directorio tambien encontramos la navaja suiza de Symfony2, la aplicación **app/console.** Es una aplicación que se ejecuta desde la línea de comandos de PHP y nos permite un control casi absoluto sobre la aplicación de Symfony, desde borrar el cache de la aplicación, actualizar la base de datos, hasta crear entidades para nuestra aplicación.

3.1.2 Directorio src/

Este directorio contiene todo el código fuente escrito en PHP de la aplicación, se incluye la lógica del negocio (controlador), los archivos que definen las entidades (modelo), archivos con la definición de los formularios para cada entidad, y

escuchadores de eventos. La estructura de subdirectorios no es obligatoria llevarla de esta manera, pero Symfony hace esta sugerencia para una mejor lectura del código (ver figura 39).

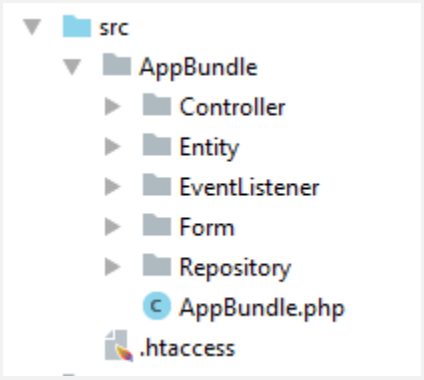


Figura 39. Estructura src/
Fuente: Autor

Directorio src/Controller/ encontramos los controladores de nuestra aplicación, que se definen como la lógica del negocio (ver figura 40):

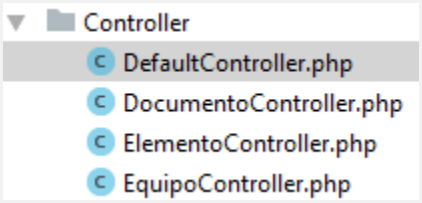


Figura 40. Estructura src/Controller/
Fuente: Autor

Cada controlador es básicamente una clase con funciones que nos permiten realizar acciones. Debido al uso de anotaciones nuestras acciones se pueden mapear a rutas específicas de la aplicación (ver figura 41):

```php
<?php

namespace AppBundle\Controller;

use ...

/**
 * Class DefaultController.
 */
class DefaultController extends Controller
{
```

```php
/**
 * Metodo que genera la pagina principal del sitio web.
 *
 * @Route("/", name="index")
 */
public function indexAction()
{
    return $this->render( view: 'index');
}
```

Figura 41. Controlador
Fuente: Autor

Directorio src/Entity/: En este directorio encontramos nuestras clases que definen las entidades a usar en nuestra aplicación, estas se mapean mediante el uso de **Doctrine ORM**, y también hace uso de anotaciones para su configuración (ver figura 42).

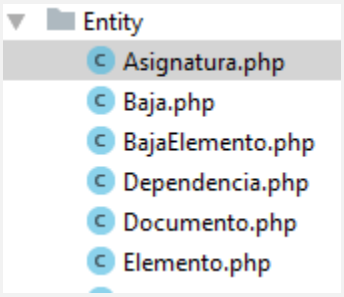


Figura 42. Estructura src/Entity/
Fuente: Autor

```
 * @ORM\Entity()
 * @ORM\Table(name="asignatura")
 * @DoctrineAssert\UniqueEntity("codigo")
 */
class Asignatura
```

Figura 43. Inicio entidad Asignatura
Fuente: Autor

Directorio src/EventListener/: En este directorio se incluyen clases que sirven como escuchadores de eventos, y que nos permiten manipular los eventos que se lanzan al ocurrir cierta condición, para poder usar estos eventos debemos definirlos en el archivo services.yml (ver figura 44).

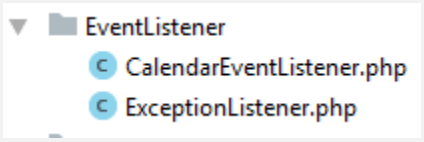


Figura 44. Estructura src/EventListener/
Fuente: Autor

Directorio src/Form/: En esta carpeta se definen los archivos que controlan los formularios, también los campos usados en los mismos (ver figura 45).

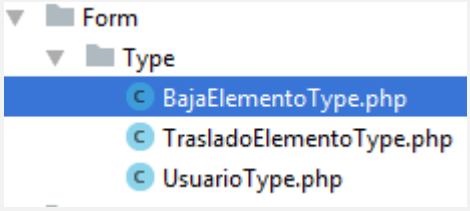


Figura 45. Estructura src/Form/
Fuente: Autor

Directorio src/Repository/: Es una carpeta usada para definir las consultas que se realizan por medio del ORM, esto nos facilita no mezclar código en los controladores (ver figura 46).



Figura 46. Estructura src/Repository/
Fuente: Autor

3.1.3 Directorio vendor/

En este directorio se guardan todas las librerías de terceros utilizadas en nuestra aplicación, tal como EasyAdminBundle (ver figura 47).

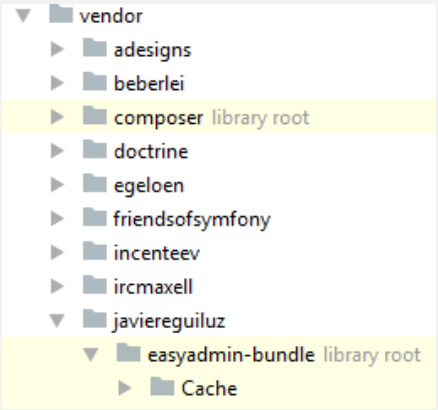


Figura 47. Estructura vendor/
Fuente: Autor

3.1.4 Directorio web/

En este directorio encontramos todos los archivos accesibles al público, contiene los archivos web (CSS, JavaScript e imágenes) y los controladores frontales de la aplicación (app.php y app_dev.php).

Cuando accedemos a nuestra aplicación en realidad accedemos a el archivo app.php si es el entorno de producción, o app_dev.php para el entorno de desarrollo (ver figura 48).

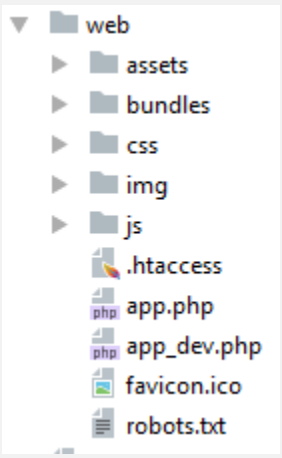


Figura 48. Estructura web/
Fuente: Autor

4. TECNOLOGIAS IMPORTANTES UTILIZADAS

4.1 Servidor

Apache: El servidor web que nos sirve para responder a las peticiones de los navegadores web. Se usa la versión 2.4.

MySQL: El gestor de bases de datos seleccionado por su facilidad y previo conocimiento sobre éste. Se usa la versión 5.7.

PHP: Es el lenguaje de programación. Se usa la versión 5.6.

Symfony: Es la base del proyecto, es un framework PHP. Se usa la versión 2.8.

Doctrine ORM: Es un mapeador objeto-relacional que nos sirve para abstraer la lógica de la base de datos y poder manejar las tablas como entidades en el framework. Se usa la versión 2.4.

mPDF: Es una librería PHP que nos permite simplificar la generación de PDF's usando como base código HTML. Se usa la versión 6.0.

EasyAdminBundle: Es un bundle de Symfony usado para simplificar el manejo de los CRUD de las entidades de nuestro proyecto.

4.2 Cliente

HTML5: El lenguaje de marcado por defecto de las aplicaciones web, se usa para darle la estructura a nuestra aplicación.

CSS3: Es un lenguaje que sirve para describir los estilos que llevan las aplicaciones web, se usa en nuestro proyecto para darle una buena presentación gráfica.

JavaScript: Es un lenguaje de programación del lado del cliente usado para mejorar la interacción del usuario con el sistema.

Bootstrap: Es un framework para el diseño de sitios web. Nos sirve de base para los estilos de la página.

JQuery: Es una librería JavaScript muy conocida que nos permite simplificar la creación de código del lado del navegador. Se usa la versión 3.1.

Moment.js: Es una librería JavaScript que nos permite simplificar la gestión de fechas en el navegador.

Noty: Otra librería JavaScript, usada para mostrar las alertas amigables al usuario.

Font Awesome: Es una librería que nos permite darle mejor aspecto a las fuentes de la web, incluye también un conjunto de iconos que son usados en la aplicación.

FullCalendar: Una librería JavaScript, se usa para mostrar el calendario de eventos en la web.

Highcharts: Es una librería JavaScript que nos sirve para mostrar los gráficos estadísticos a los administradores del sitio.