MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DEL AMBIENTE Y DEL SOFTWARE NECESARIO PARA EL PROYECTO DE PGR

Alpha Sport- Sistema de gestión de los recursos de Deportes de la Escuela

GUIA DE INSTALACIÓN Y CONFIGURACIÓN DEL SERVIDOR LAMP

Modulo con servicios para la creación y modificación de horarios para los estudiantes de la Escuela. Modulo de estudiantes en servidor LAMP, componente web

- Asignación de horario de inscripción aleatorio (Nuevos y Antiguos)
- Registro de estudiantes
  - En un día y hora especifico
  - Número maximo de inscritos
  - Lista de espera
  - Reasignación de cupos
- Creación y modificación de horarios
  - En un día y hora especifico

Configuración Servidor LAMP (Linux-Apache-Mysql-PHP)

El software requerido para instalar es el siguiente:

- Linux Mint 19/ Ubuntu 16/ Slackware
- Apache2: Servidor WEB
- Mysql / MariaDB: Motor de Base de datos
- PHP 7.0: Lenguaje de scripting del servidor
- phpmyadmin: Gestor de base de datos

Instalación de Apache:

sudo apt-get install apache2

Configuración:

Verificar puertos:

sudo ufw app info "Apache Full"

Si están cerrados:

sudo ufw allow in "Apache Full"

Instalación de Mysql:

sudo apt-get install mysql-server

Configuración:

SEGURIDAD ADICIONAL (seleccionar no)

sudo mysql_secure_installation

- password: 000
- eliminar usuarios anónimos: SI

Entrar a mysql así:

sudo mysql -u root -p deportes

Para ver tablas:

show databases;

Configuración global:

Aplicar configuraciones

sudo service apache2 restart

verificar archivo de index

cat /var/www/html/index.html

Agregar permisos:

sudo chown -R $USER:root /var/www

Instalación PHP

sudo apt-get install php7 libapache2-mod-php php-mysql

Configuración

cd /etc/php/7.0/apache2/
sudo xed php.ini

modificar las siguientes lineas:

short_open_tag = On
error_reporting=E_ALL & ~E_NOTICE
display_errors=On

Resumen de phpinfo, crear:

echo “<?php phpinfo(); ?>” > /var/www/html/info.php

ver: *http://localhost/info.php*

Instalación phpmyadmin:

sudo apt-get install phpmyadmin

Configuración:

usar: apache2
uso de dbconfig-ommon: SI
contraseña conexión de Mysql para phpmyadmin: 000

Configurar apache y phpmyadmin, editar apache2.conf

sudo xed /etc/apache2/apache2.conf

Agregar al final:

Include phpmyadmin file
Include /etc/phpmyadmin/apache.conf

Aplicar cambios:

sudo service apache2 restart

Probar phpmyadmin:

En: **localhost/phpmyadmin**

usuario: phpmyadmin
contraseña: 000

Agregar configuración para conexiones a BD con root:

sudo mysql -u root -p

UPDATE mysql.user SET authentication_string=password('admin') WHERE user='root';

ver modo de autenticación:

select user, plugin from mysql.user;

Modificar para pedir "por contraseña"

UPDATE mysql.user SET plugin='mysql_native_password' WHERE user = 'root';

Verificar inicio de sesión: user: root , password: admin

phpmyadmin

phpmyadmin

Configuración del dominio: Usar proyecto.com

editar hosts

sudo nano etc/hosts

Configuraciones para el optimo desempeño del servidor

Para **php**:

Agregar la linea:

sudo xed /etc/phpmyadmin/config.inc.php

$cfg['ExecTimeLimit'] = 6000;

Colocar los valores:

sudo xed /etc/php/7.0/apache2/php.ini

post_max_size = 750M

upload_max_filesize = 750M

max_execution_time = 5000

max_input_time = 5000

memory_limit = 1000M

short_open_tag = On

error_reporting=E_ALL & ~E_NOTICE

para ver errores:

display_errors=On

Para MySQL:

aumentar conexiones de mysql a minimo 300:

set GLOBAL max_connections = 300;

Aplicar cambios:

sudo service apache2 restart

sudo service mysql restart

sudo service php7.0-fpmrestart

Para remoto acceso a phpMyAdmin

Como el servidor no permite acceder a phpmyadmin desde internet, fue necesario incluir phpMyAdmin manualmente, para esto se creo public_html en la raiz y se hizo la isntalación manual.

mkdir public_html

wget -c https://files.phpmyadmin.net/phpMyAdmin/4.8.3/phpMyAdmin-4.8.3-all-languages.zip

unzip /phpMyAdmin-4.8.3-all-languages.zip

mv phpMyAdmin-4.8.3-all-languages phpmyadmin

Link: /deportes/phpmyadmin

Configuraciones adicionales en phpmyadmin instalado manualmente, agregar cambiar las lineas:

nano phpmyadmin/libraries/config.default.php

$cfg['blowfish_secret'] = 'a0sfo49nadf89fa3s8f789sf78asb7p1587balz';

Para probar conexión remota con MySQL

mysql -h estudiantes.is.escuelaing.edu.co -u deportes -p deportes

GUIA DE INSTALACIÓN Y CONFIGURACIÓN DE AMBIENTE .NET

Requisitos de software

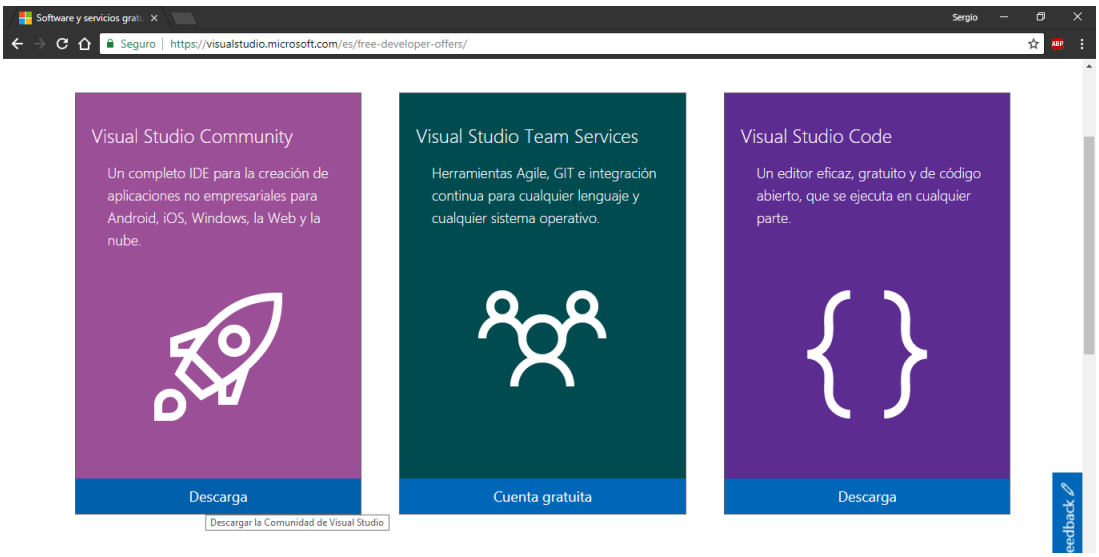
Para replicar el ambiente de trabajo necesario para este proyecto se necesita:

- Microsoft Windows 10 x64
- Microsoft Visual Studio 2017 o superior
- Paquetes de WPF y Windows Form para MSVisual Studio
- MYsql Connector para NET (Windows)
- Microsoft Project 2016 o superior
- Bonitasoft Community 7.6 o superior

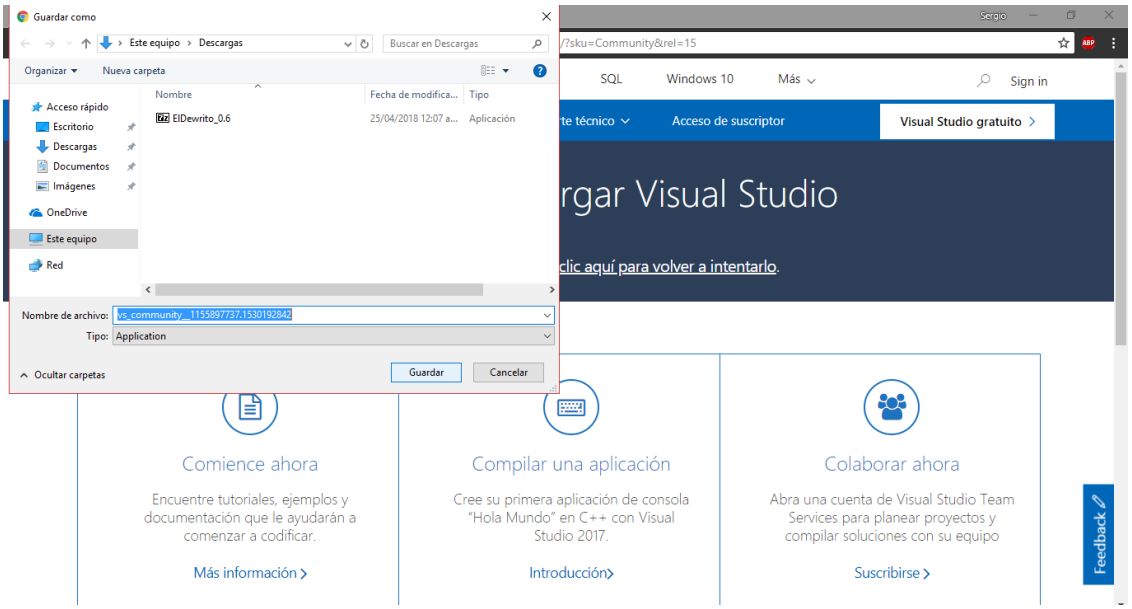
Instalación de Microsoft Visual Studio Community

Ir al enlace para descargar Visual Studio Community:
https://visualstudio.microsoft.com/es/thank-you-downloading-visual-studio/?sku=Community&rel=15

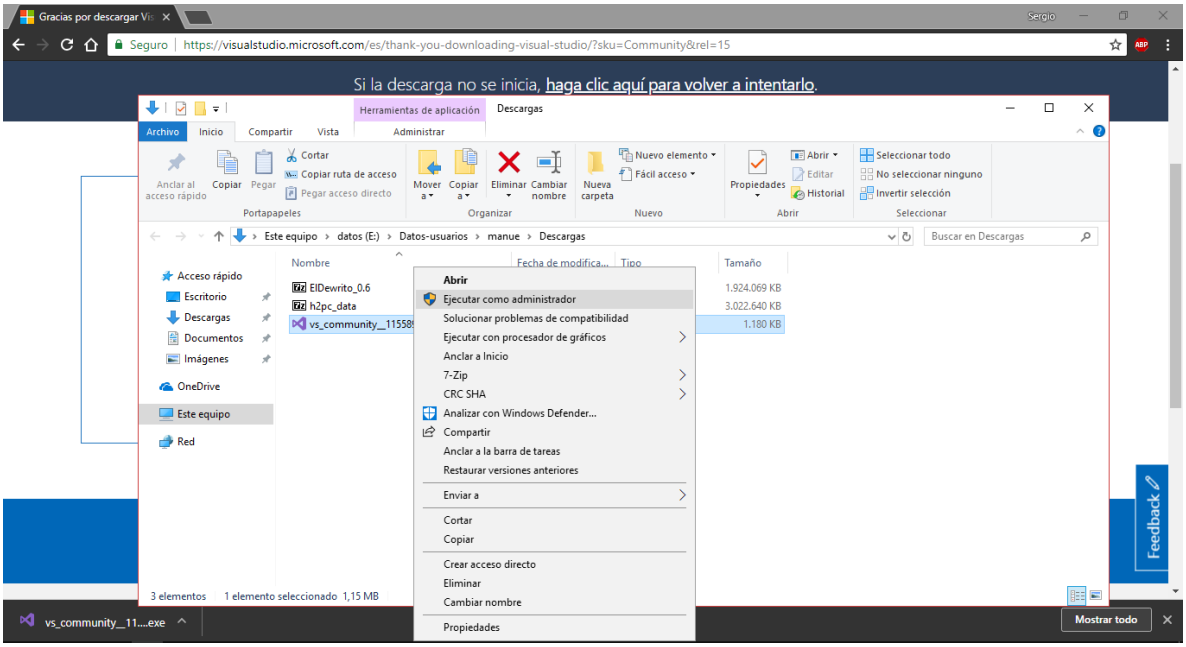
Seleccionar **Descargar**.



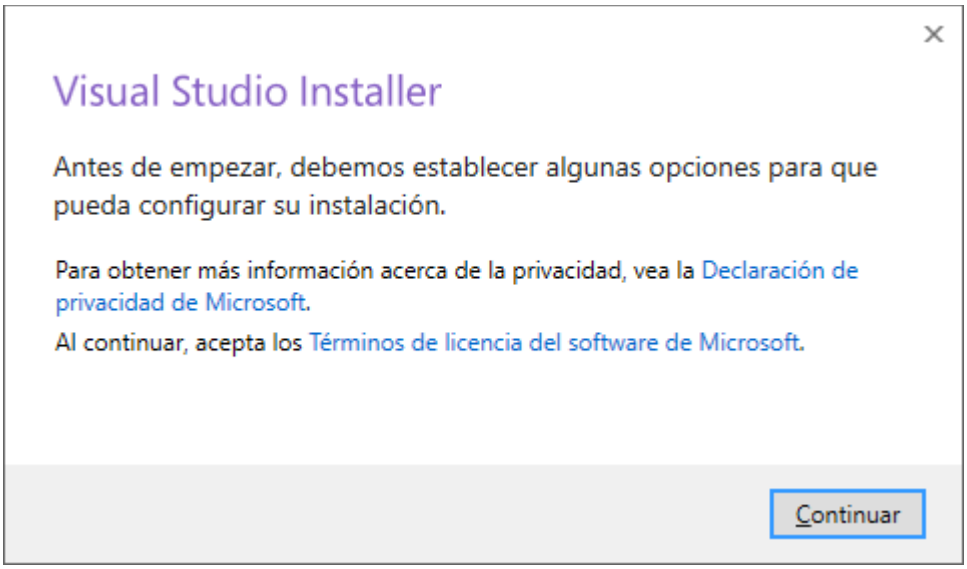
Guardar el archivo de instalación.



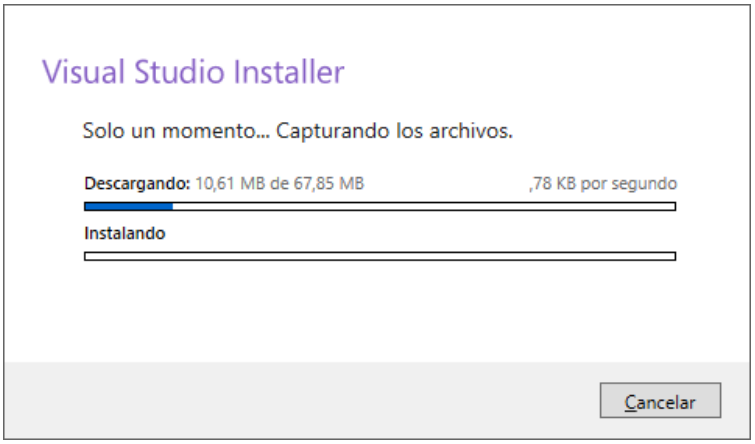
Ejecutar como **administrador**.

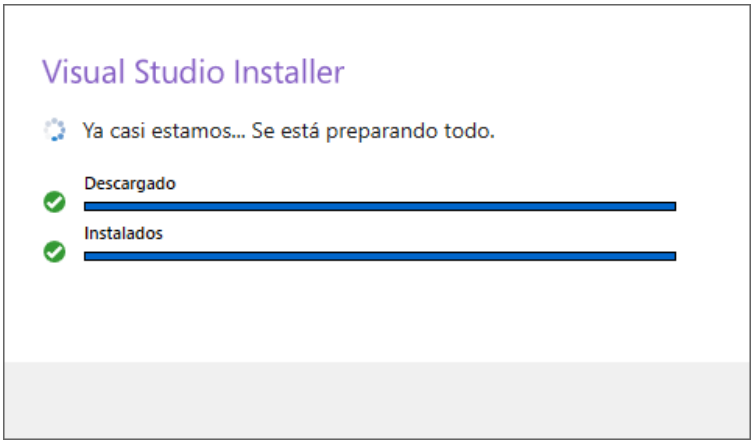


Empezar la instalación seleccionando **Continuar**.

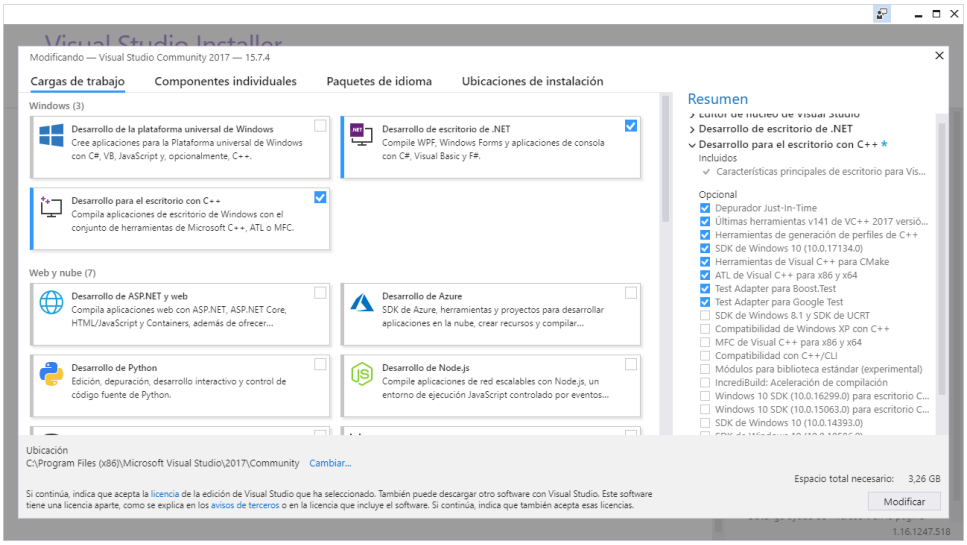


Esperar a que termine la descarga e instalación de Visual Studio Installer.

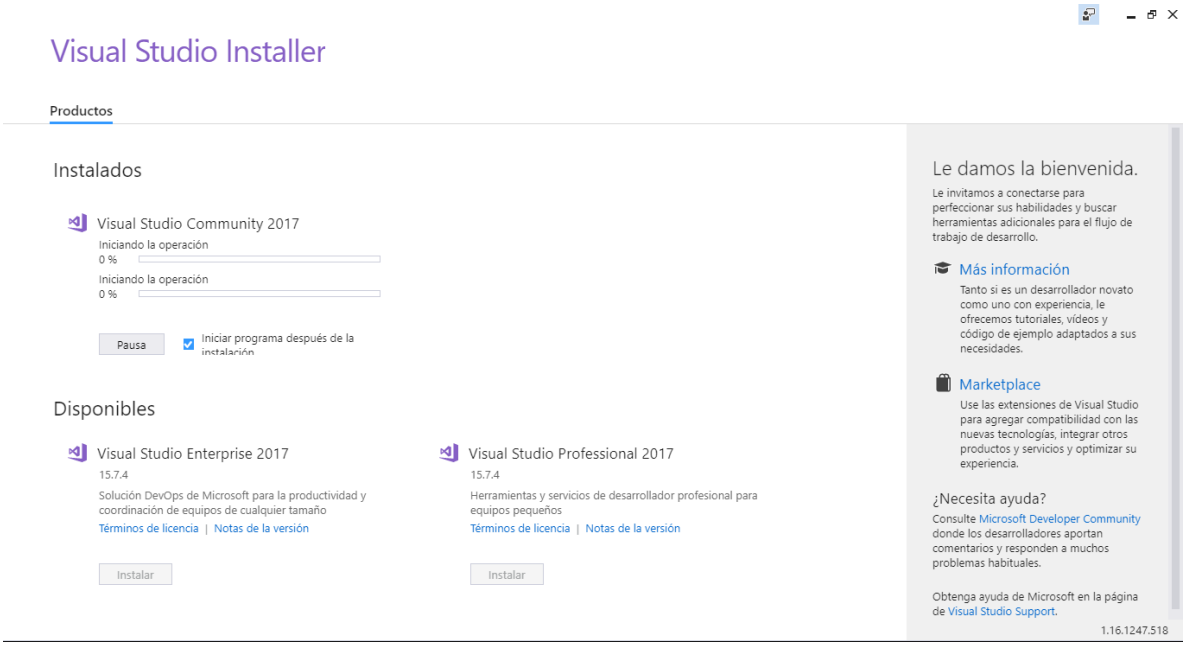




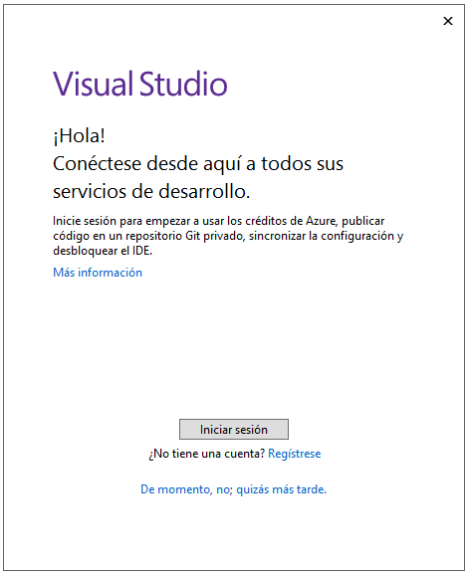
Se abrirá una nueva ventana para seleccionar los paquetes de software: **Seleccionar** Desarrollo de escritorio .NET:



Esperar a que termine la descarga.



Cuando haya terminado la descarga e instalación, se abrirá una ventana donde se debe autenticar con una cuenta Microsoft.

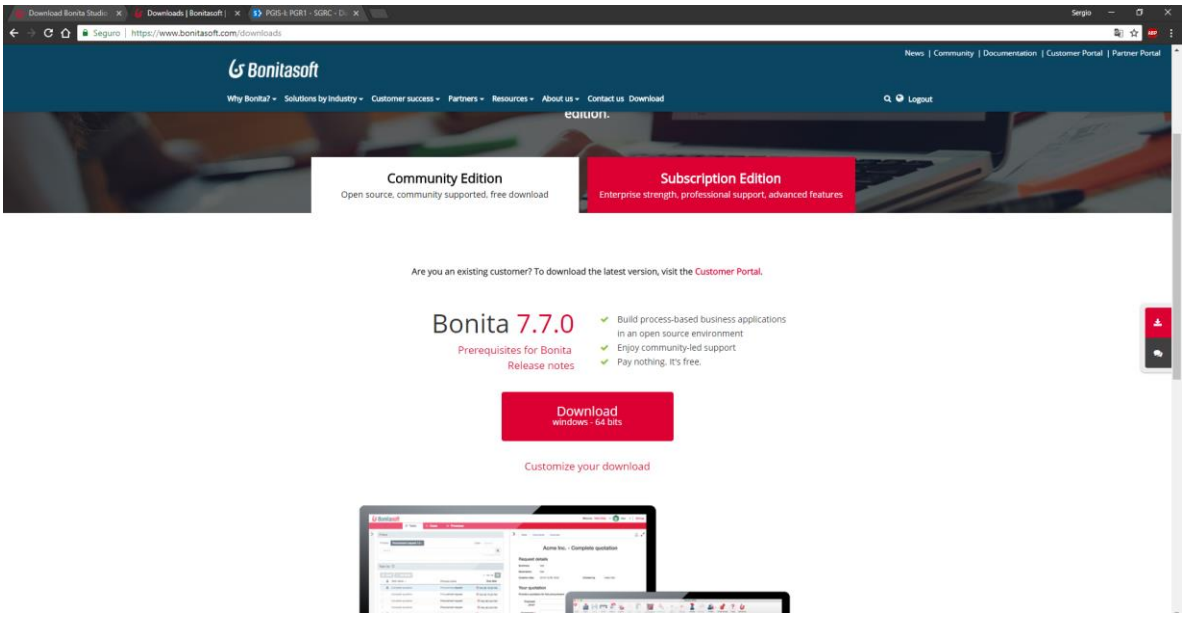


Instalación Bonita Soft Community

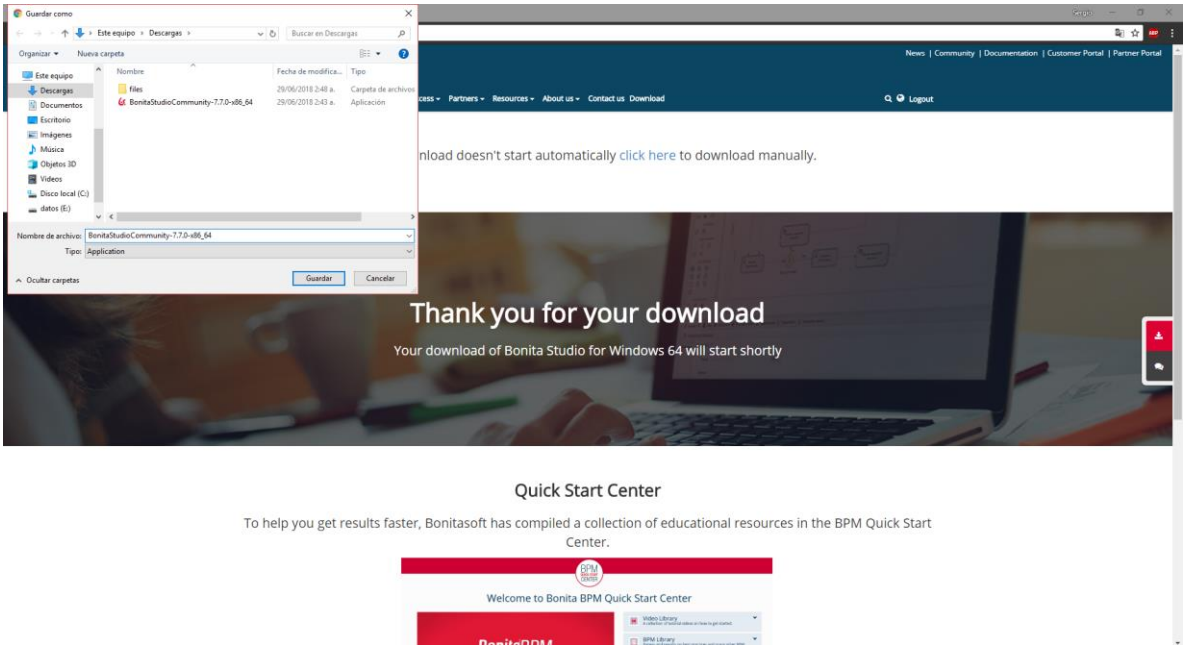
Ir a: https://www.bonitasoft.com/downloads

Registrarse y autenticarse en **bonitasoft.com.**

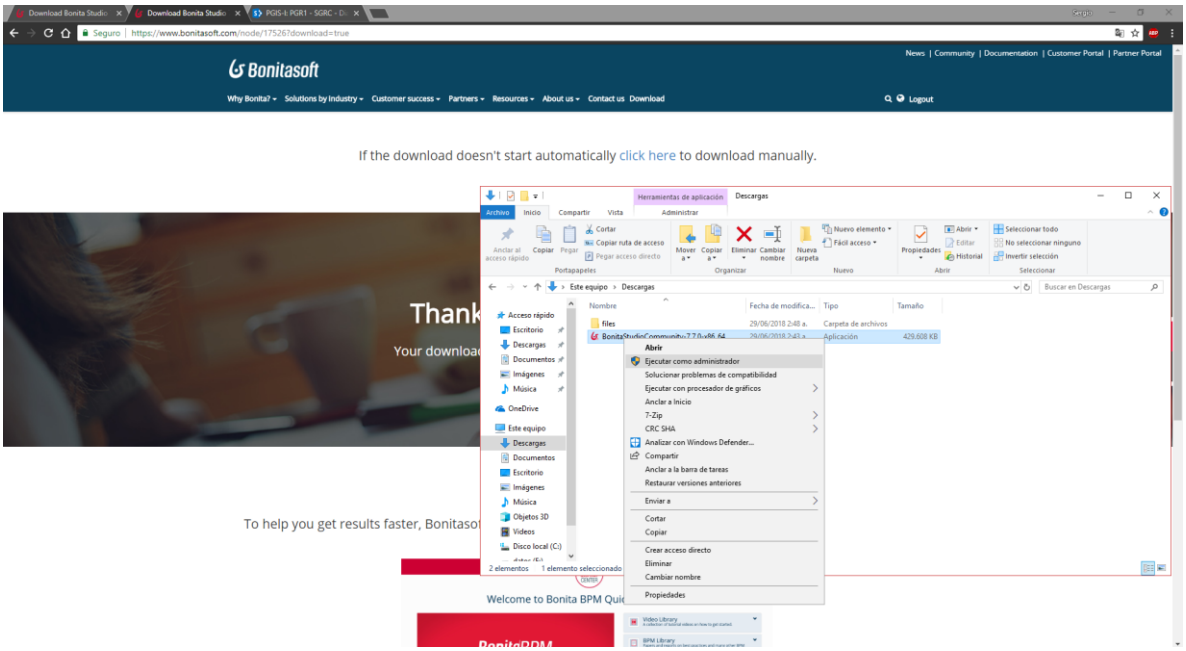
Luego ir a descargas y seleccionar **Descargar** Bonita Community Edition



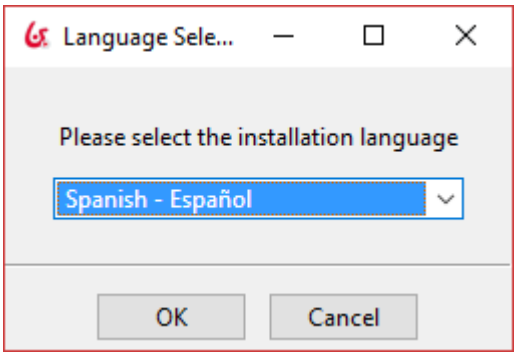
Seguido de esto, guarde el archivo de instalación:



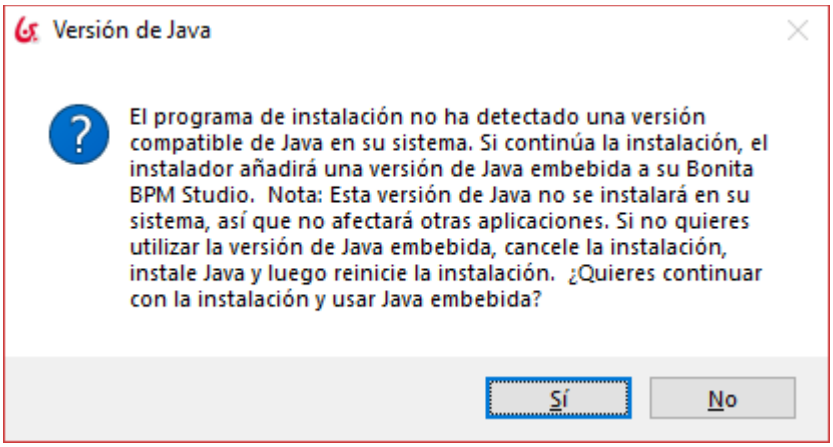
Ejecute como administrador el ejecutable.



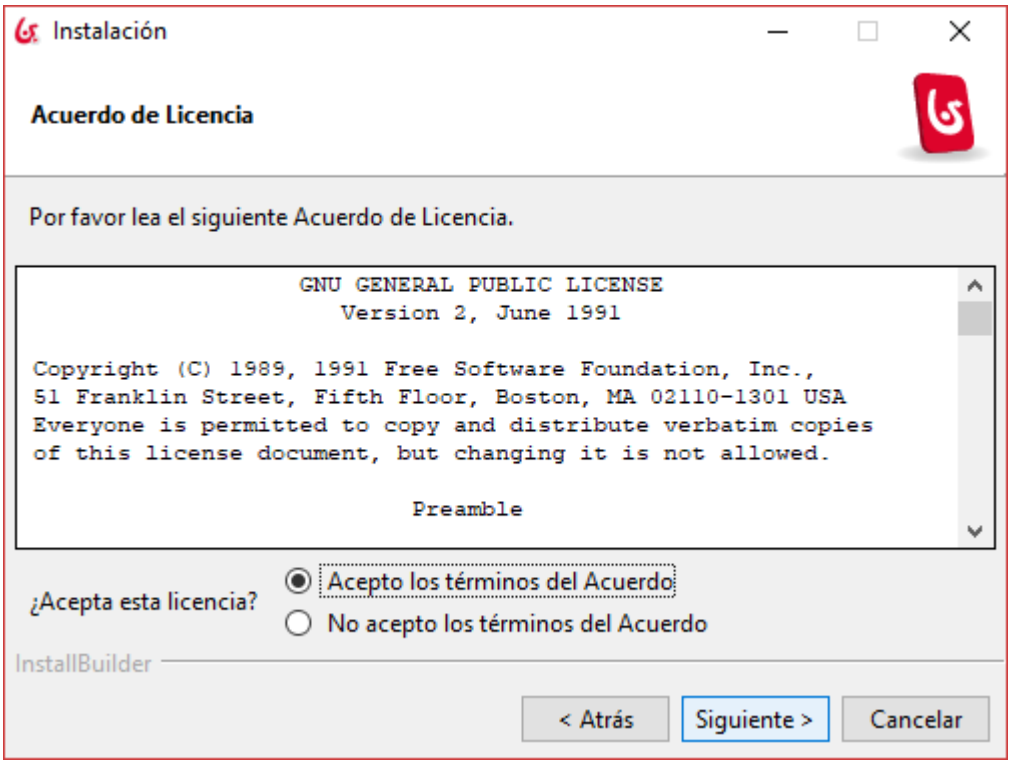
Escoja el idioma:



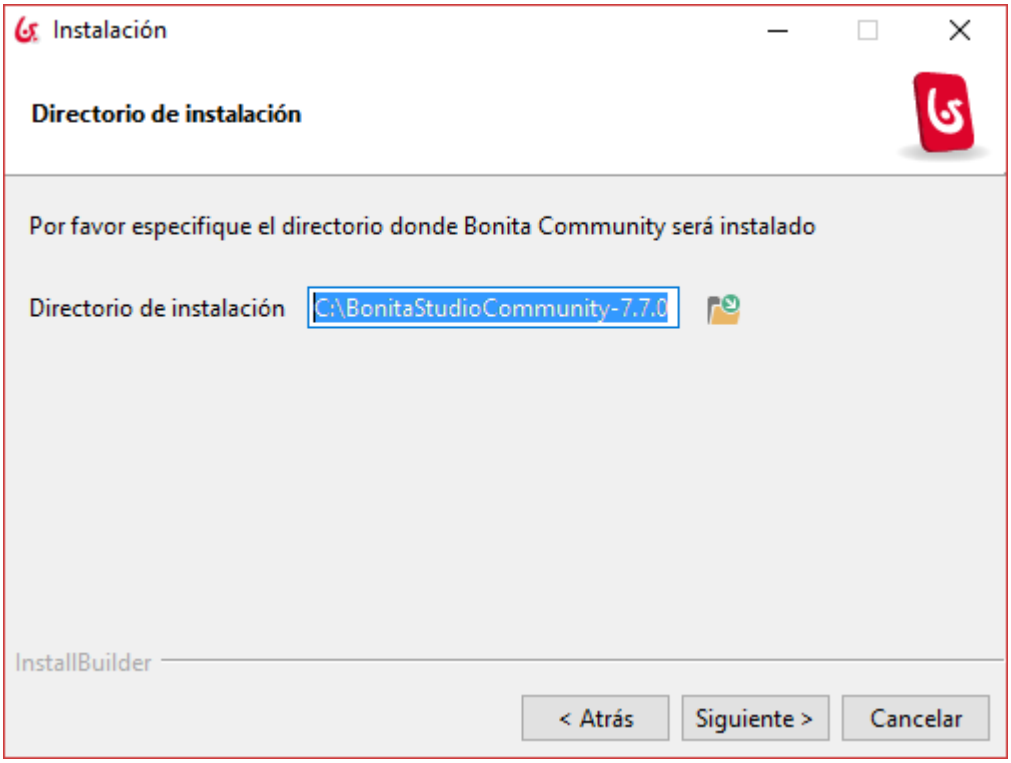
Si no se tiene Java instalado en el equipo, seleccionar **SI**.



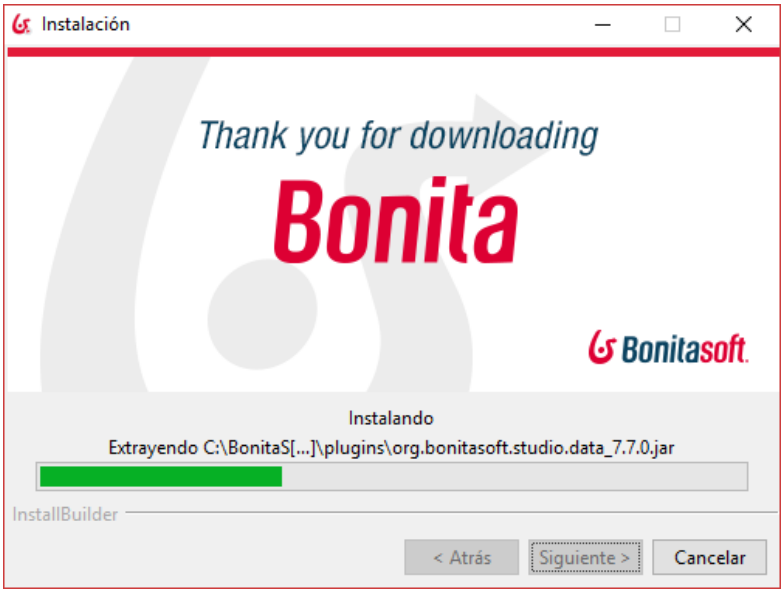
Continuar con la instalación, seleccionar siguiente, seguido de esto aprobar los acuerdos de uso y seleccionar siguiente.



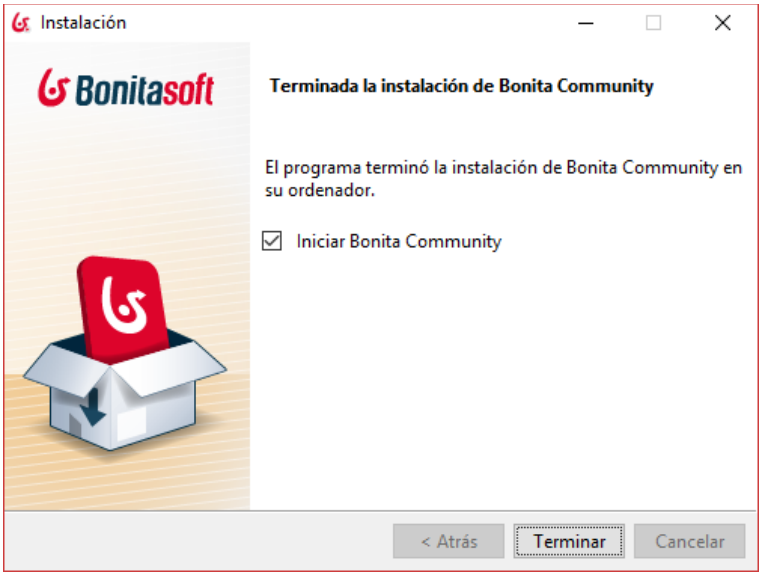
Escoger un directorio de trabajo, seleccionar siguiente:



Esperar a que termine la instalación



Seleccionar Terminar para salir del instalador.

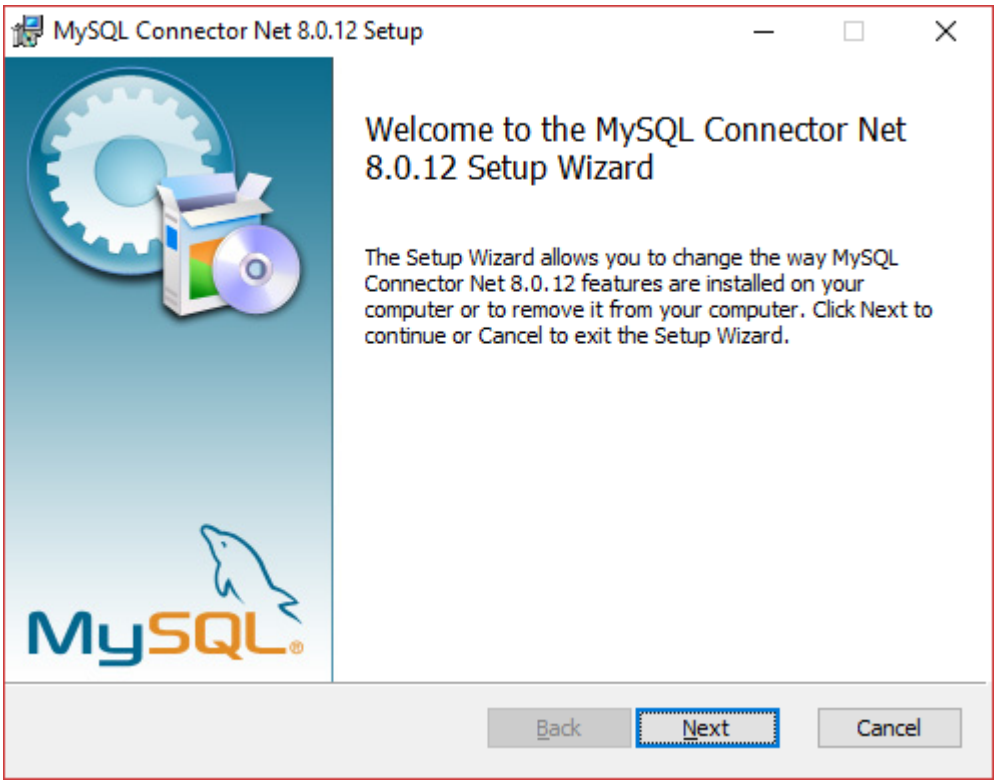


Instalación y Configuración de ambiente NET

Descargar Mysql Connect for NET de :

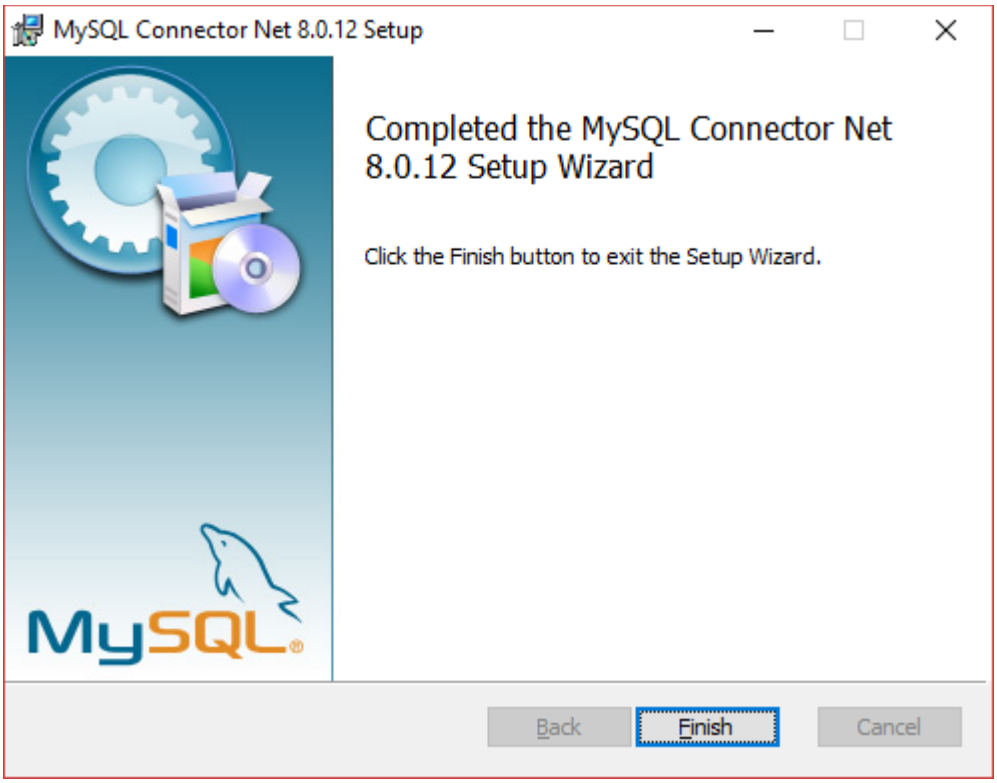
https://dev.mysql.com/downloads/connector/net/

Instalar el conector:



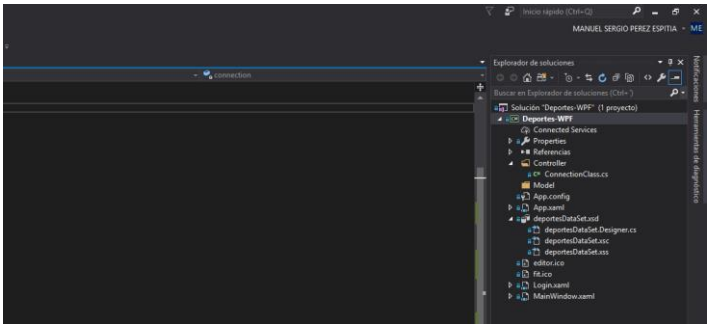
Seleccionar instalación habitual y realizar la instalación habitual como cualquier otro programa:

Al terminar, ser finaliza la instalación con **Finish**.

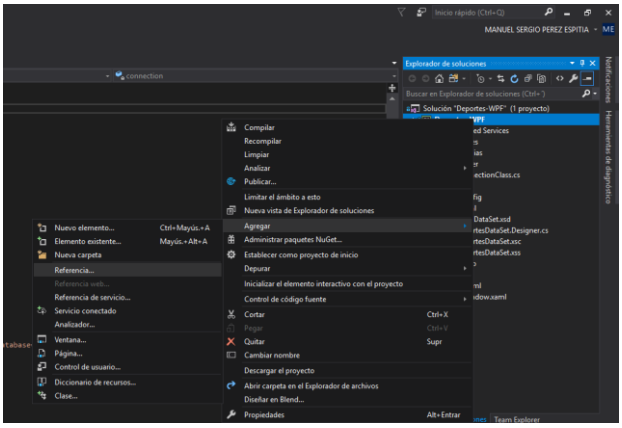


Para poderlo usar en .NET se debe agregar MS Visual Studio

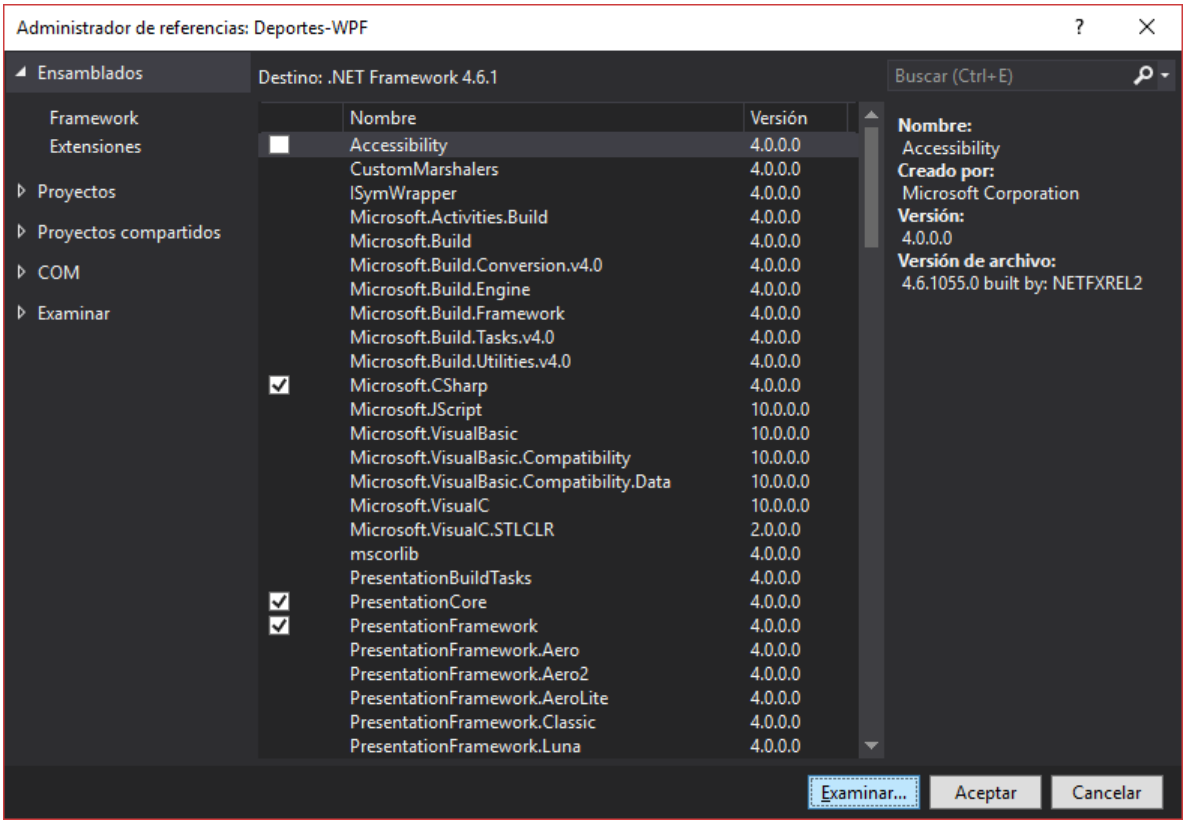
Ir al árbol del proyecto



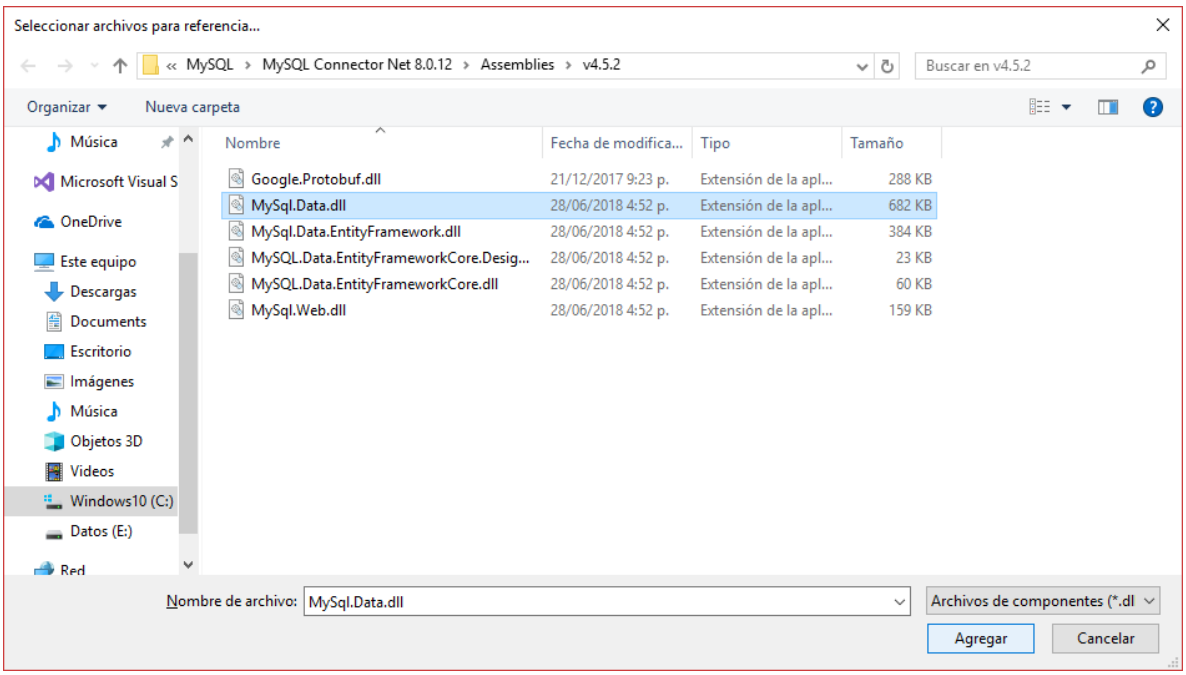
Encima de la solución abrir menú con clic derecho y entrar a “agregar referencia”



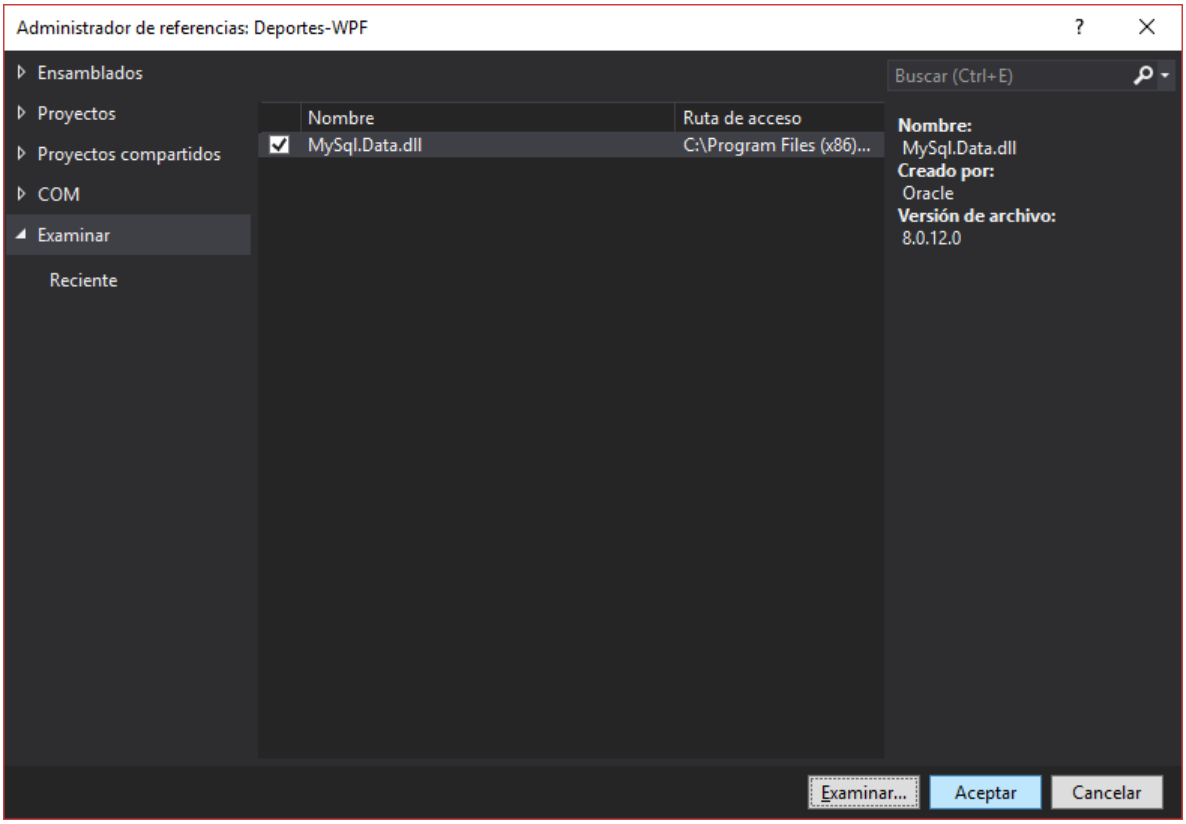
Dar en examinar y navegar hasta donde esta instalado Mysql connector



Abrir el archivo MySqlData.dll



Y dar en aceptar:

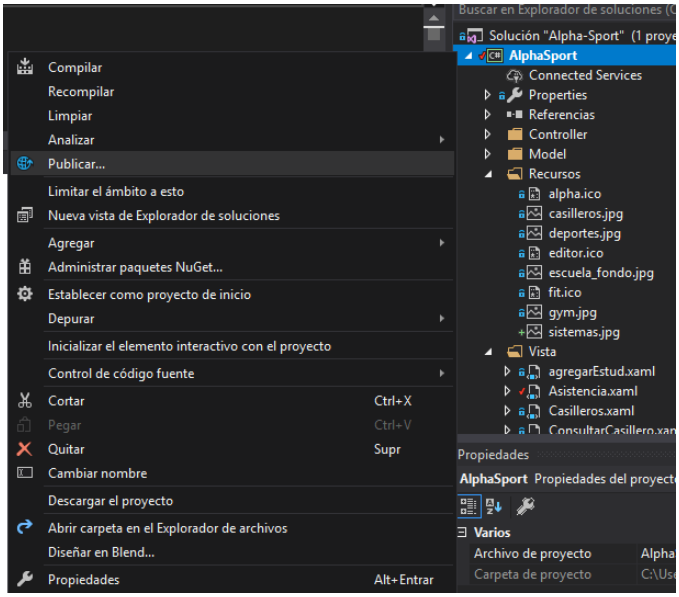


Para usar se importa la librería así:

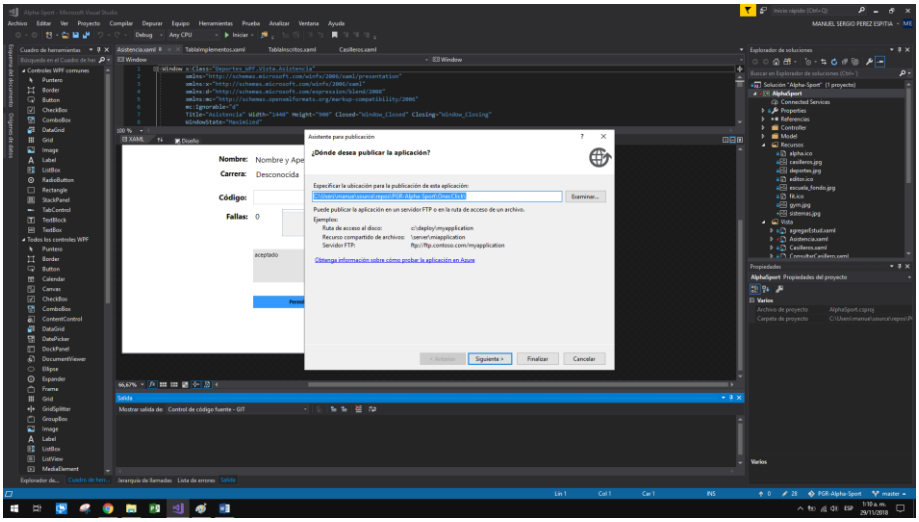
```
using MySql.Data.MySqlClient;
```

Para generar ejecutable y usar

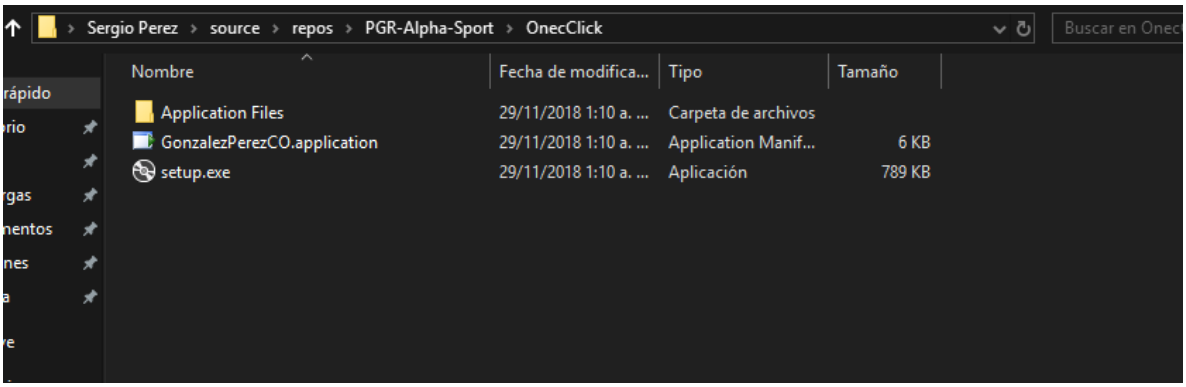
Proyecto > Nombre del proyecto > Publicar



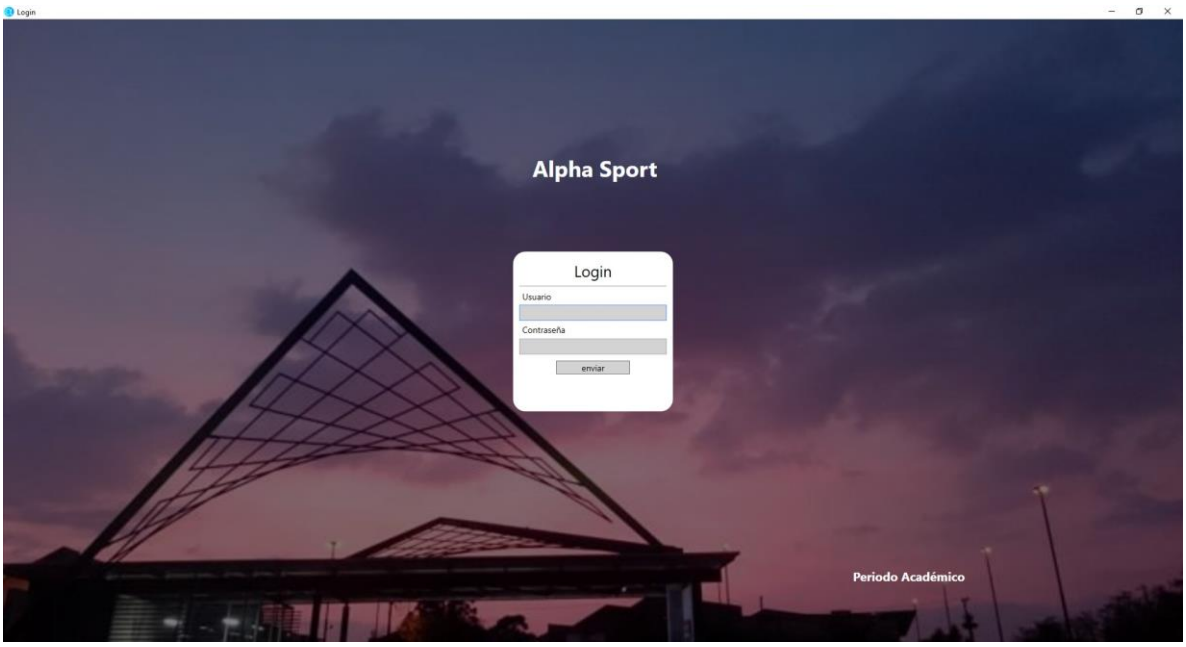
Se abrirá un cuadro para configurar los datos de la publicación donde para generar el ejecutable se continua con Finalizar.



Se abrirá en el explorador la carpeta que contiene los archivos de instalación.



Para ejecutar se copian ejecuta Setup.exe, no hace falta instalarlo. El único requisito es tener FrameworkNET 4.6 actualizado en el Windows.



GUIA DE INSTALACIÓN Y CONFIGURACIÓN PARA EL AMBIENTE DE LA WEBAPP

FROTNED

Este proyecto fue desarrollado con Create React App.

Tabla de Contenidos

Estructura de Carpetas

Los archivos en el proyecto tienen la siguiente estructura:

```
my-app/
  README.md
  node_modules/
  package.json
  public/
    index.html
    favicon.ico
  src/
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
```

Para el proyecto poder ser construido, **estos archivos deben existir con nombres exactos**:

- `public/index.html` es la plantilla de página;
- `src/index.js` es el punto de entrada de JavaScript.

Sólo los archivos dentro de `public` pueden ser usados desde `public/index.html`. Y los únicos archivos que son usados para construir el proyecto son los que están dentro de `src`.

Scripts Disponibles

En el directorio del proyecto, puede ejecutar el comando:

`npm start`

Esto corre la app en modo de desarrollo. Luego vaya a la dirección http://localhost:3000 para verlo en el navegador.

La página se recarga si se producen cambios y se guardan en el proyecto. También se pueden ver los errores en la consola.

`npm test`

Lanza el generador de pruebas en modo de vista interactivo.

`npm run build`

Crea la app lista para producción en la carpeta `build`.

Agrupa correctamente React en modo de producción y optimiza el build para obtener el mejor rendimiento.

El build es minificado y los nombres de archivos incluyen los hashes. Con esto la app esta lista para ser desplegada.

`npm run eject`

Esta es una operación de una sola vía, una vez se usa ya no se puede volver atrás

En caso de que uno no se sienta satisfecho con la herramienta de build y las decisiones de configuración, se puede usar `eject`en cualquier momento. Este comando remueve la única dependencia de construcción del proyecto.

En su lugar lo que hace es copiar todos los archivos de configuración y las dependencias transitivas (Webpack, Babel, ESLint, etc) justo adentro del proyecto para que de esta forma uno tenga full control sobre ellas. Tener en cuenta que los demás comandos van a seguir sirviendo pero una vez esta operación es ejecutada. Es uno mismo el que tiene que hacer la configuración.

Navegadores Soportados

Por defecto, el proyecto generado usa la última version de React.

Se puede referir a la documentacion de React para más información sobre navegadores soportados.

Depurando con el Editor

Esta característica es sólo soportada por el momento en Visual Studio Code y WebStorm.

Visual Studio Code y WebStorm soportan depuración de una sin configuración con Create React App. Esto permite a los desarrolladores a escribir y depurar el código de React sin salir del editor, y más importante también permite tener un flujo de trabajo de desarrollo continuo, donde el cambio de contexto es mínimo, dado que no hay necesidad de estar cambiando de herramientas constantemente.

Visual Studio Code

Hay que tener la última versión de VS Code y VS Code Chrome Debugger Extension instaladas.

Después, añadir el bloque de código siguiente al archivo `launch.json` y ponerlo adentro del folder `.vscode` en el directorio raíz de la app.

```
{
  "version": "0.2.0",
  "configurations": [{
    "name": "Chrome",
    "type": "chrome",
    "request": "launch",
    "url": "http://localhost:3000",
    "webRoot": "${workspaceRoot}/src",
    "sourceMapPathOverrides": {
      "webpack:///src/*": "${webRoot}/*"
    }
  }]
}
```

Corra la app usando el comnado `npm start`, y puede empezar a depurar en VS Code presionando F5 o clickeando el icono verde de depurar. Ahora se puede escribir código, poner puntos de quiebre, hacer cambios al código, y depurar el proyecto directamente desde el editor.

WebStorm

Hay que tener WebStorm y la extensión de Chrome de JetBrains IDE Support instalada.

En el menu Run de WebStorm seleccionar `Edit Configurations...`. Despúes clickear + y seleccionar `JavaScript Debug`. Pegar `http://localhost:3000` en el campo de de la URL y posteriormente guardar la configuración.

Corra la app usando el comnado `npm start`, luego presione ^D en macOS o F9 en Windows y Linux o clickee el icono verde de depurar para empezar la depuración en WebStorm.

Cambiando `<title>` en la página

Se puede encontrar el código HTML base en el folder `public` del proyecto generado. Se puede editar la etiqueta `<title>` para cambiar el título de "React App" a cualquier otra cosa.

Note que normalmente no se editarían los archivos del folder `public` muy seguido. Por ejemplo, añadir una hoja de estilos es hecho sin tocar el HTML.

Instalando una Dependencia

El proyecto generado incluye React y ReactDOM como dependencias. También un conjunto de scripts usados por Create React App dependencia de desarrollo. Se pueden instalar otras dependencias (por ejemplo, React Router) con `npm`:

```
npm install --save react-router
```

Alternativamente se puede usar `yarn`:

```
yarn add react-router
```

Esto funciona para cualquier paquete, no sólo `react-router`.

Agregar una hoja de estilos

Esta configuración de proyecto usa Webpack para manejar todos los archivos. Webpack ofrece una forma personalizada de "extender" el concepto de `import` más allá de JavaScript. Para expresar que un archivo JavaScript depende de un archivo de CSS, se necesita **importar el CSS desde el archivo JavaScript**:

`Button.css`

```
.Button {
  padding: 20px;
}
```

`Button.js`

```
import React, { Component } from 'react';
import './Button.css'; // Tell Webpack that Button.js uses these styles

class Button extends Component {
  render() {
    // You can use them as regular CSS styles
    return <div className="Button" />;
  }
}
```

Despliegue

`npm run build` crea un directorio `build` con un build de producción de la app. Se puede configurar el servidor HTTP de preferencia para que un visitante al sitio sea servido el archivo `index.html`, y las peticiones a rutas estáticas como `/static/js/main.<hash>.js` sean servidas con los contenidos del archivo `/static/js/main.<hash>.js`.

Servidor Estático

Para ambientes usando Node, la forma más fácil de manejar esto es instalar serve y dejarlo que maneje el resto:

```
npm install -g serve
serve -s build
```

Corriendo este comando se puede tener una lista completa de las opciones disponibles:

```
serve -h
```

Otras Soluciones

La selección de software del servidor tampoco es importante. Siendo que Create React App es completamente agnostico a la plataforma, no hay necesidad de usar Node explicitamente.

El folder `build` con archivos estáticos es el único producido creado por Create React App.

Sin embargo, esto puede no ser suficiente si se usa enrutamiento del lado del cliente.

Sirviendo Apps con Enrutamiento de lado del cliente

Si se usan enrutadores que usan la pushState history API de HTML5 por debajo (por ejmplo, React Router con `browserHistory`), varios servidores de archivos estáticos van a fallar. Por ejmplo, si se usa React Router con una ruta de `/todos/42`, el servidor de desarrollo va a responder a `localhost:3000/todos/42` apropiadamente, pero un servidor Express sirviendo un build de producción no lo va a hacer.

Si se está usando Apache HTTP Server, se necesita crear un archivo `.htaccess` en el folder `public` con el siguiente contenido:

```
Options -MultiViews
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^ index.html [QSA,L]
```

Este se va a copiar al folder `build` cuando se ejecute el comando `npm run build`.

Ahora las peticiones a `/todos/42` van a ser manejadas correctamente tanto en desarrollo como en producción.

En un build de producción, y un navegador que soporte service workers, el service worker va a manejar automaticamente todas las peticiones de navegación, como `/todos/42`, siriviendo una copia en cache del `index.html`. Este enrutamiento de navegación del service worker puede ser configurado o deshabilitado con eject y depués modificando las opciones navigateFallback y navigateFallbackWhitelist de la configuración del `SWPreachePlugin` .

Construyendo para Rutas Relativas

Por defecto, Create React App produce un build asumiendo que la app está alojado en la raíz del servidor. Para sobreescribir esto, especifique el homepage en el `package.json`, por ejemplo:

```
  "homepage": "http://mywebsite.com/relativepath",
```

Esto va a permitir a Create React App inferir correctamente el directorio raíz para usar en el archivo HTML generado.

Alternativas a Ejectar

Ejectar deja personalizar cualquier cosa, pero desde ese punto en adelante hay que mantener la configuración y los scripts por uno mismo. Esto puede ser desalentador si se tiene muchos proyectos similares. En dados casos en vez de ejectar es recomendable hacer un *fork* de`react-scripts` y cualquier otros paquetes que sean necesarios.

BACKEND

Esta aplicación funciona sobre un entorno LAMP, por lo que se asume que todos sus componentes (Apache, MariaDB, PHP) ya se encuentran instalados.

Instalación de Slim (Framework de PHP)

La forma más recomendada de instalar Slim es a través del administrador de dependencias de PHP *Composer*.

En caso de no tener Composer en el entorno de desarrollo, este se puede descargar usando los siguientes comandos en la terminal del servidor:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('SHA384', 'composer-setup.php') === '544e09ee996cdf
60ece3804abc52599c22b1f40f4323403c44d44fdfdd586475ca9813a858088ffbc1f233e
9b180f061') { echo 'Installer verified'; } else { echo 'Installer corrupt
'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Estos 4 comandos lo que hacen es descargar el instalador en el directorio actual, luego verifican el instalador SHA-384, lo ejecutan en el directorio y por último lo remueven.

Para instalar Slim se navega al directorio base del proyecto y se ejecuta este comando:

```
composer require slim/slim "^3.0"
```

Después basta con incluir el archivo autoload como requerimiento en el script de PHP de la siguiente forma:

```
<?php
require 'vendor/autoload.php';
```

Para hacer el despliegue de la aplicación en el servidor web, son necesarios 2 cambios en la configuración de Apache.
Asumiendo que no se tiene acceso a los archivos de configuración principal del servidor, se crea un archivo *.htaccess* el cual se añade en el mismo directorio del archivo *index.php*. En el archivo *.htaccess* se escriben las siguientes líneas:

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^ index.php [QSA,L]
```

Ahora para que estas reglas funcionen, hay que asegurarse de habilitar el módulo de Apache llamado *mod_rewrite* y que esté habilitada la opción *AllowOverride* en la configuración del servidor para que las directivas que fueron escritas en el archivo *.htaccess* puedan ser usadas.

```
AllowOverride All
```

Con estas configuraciones hechas, ya se puede hacer el despliegue de la aplicación al servidor web donde puede recibir y manejar peticiones HTTP hechas a los servicios.