**UNIVERSIDAD NACIONAL DE PIURA**

**FACULTAD DE INGENIERÍA INDUSTRIAL**

**ESCUELA PROFESIONAL INGENIERÍA INFORMÁTICA**

Logotipo

Descripción generada automáticamente

**MONOGRAFÍA SOBRE EL FRAMEWORK SYMFONY**

**PRESENTADO POR**

Castro Yovera, Jairo

Estrada Gutierrez, Anayelli

Madrid Reyes, Kevin

Pacheco Brophy, Luis

**DOCENTE ENCARGADO**

Saavedra Arango, Moisés

**CURSO**

Ingeniería de Software

**PIURA, PERÚ**

**2022**

# Índice

[Índice](#_lpwd2rtvth9p) **2**

[Introducción](#_uupnib5b5yw) **3**

[Objetivos](#_s3drtonken1y) **4**

[Objetivo General](#_pnatwi7n870v) 4

[Objetivos Específicos](#_v86j107zqrtt) 4

[Redacción](#_pyi9dea96h9k) **5**

[¿Qué es?](#_369ohx8xgym5) 5

[Partes](#_xlxwyb6garhd) 5

[Componentes](#_y9gavuhewdw8) 5

[Controladores](#_d0hl5tjcb6lg) 6

[Lenguajes de compilación](#_83ec0r4rdxc2) 7

[Servicios de aplicaciones y web](#_hb13v7ku0uvu) 7

[Comparación de versiones](#_vpgmmnbgi5y) 9

[Aplicación](#_p6ywksvm4twv) 13

[Diagrama de clases](#_s2guxj2kgdww) 13

[Diagrama de secuencia](#_qsn2ci0qc0) 14

[Fragmentos de código respectivo](#_1inc1yra9wfo) 14

[Conclusiones](#_t9akiub4bq2f) 21

[Bibliografía](#_sben6odf8rr) 22

# **Introducción**

Symfony es un marco de desarrollo de aplicaciones web de código abierto que se basa en el lenguaje de programación PHP. Fue lanzado en 2005 y desde entonces ha sido ampliamente utilizado para desarrollar aplicaciones web de gran escala.

Symfony se compone de varios componentes, cada uno de los cuales puede ser utilizado de forma independiente en cualquier proyecto PHP. Estos componentes incluyen: HttpFoundation, que proporciona una capa de abstracción para trabajar con peticiones y respuestas HTTP; Templating, que proporciona una forma de crear plantillas para la presentación de datos; y Routing, que permite definir las reglas para mapear las URL de una aplicación a las acciones de un controlador.

Symfony también proporciona un conjunto completo de herramientas para desarrollar aplicaciones web, incluyendo un sistema de gestión de dependencias, un sistema de configuración, un sistema de caché, un sistema de seguridad y una interfaz de línea de comandos (CLI) para realizar tareas comunes.

Otra ventaja de Symfony es su gran comunidad. Hay una gran cantidad de recursos y documentación disponibles en línea, así como una gran cantidad de paquetes y extensiones creados por la comunidad que pueden ser utilizados para agregar funcionalidades adicionales a una aplicación.

Symfony también es compatible con una variedad de bases de datos y ORM, incluyendo Doctrine, que proporciona una forma fácil de trabajar con bases de datos en una aplicación Symfony.

En resumen, Symfony es un marco de desarrollo de aplicaciones web de código abierto que proporciona una gran cantidad de herramientas y componentes para desarrollar aplicaciones web de gran escala. Su gran comunidad y compatibilidad con una variedad de bases de datos y ORM lo hacen una excelente opción para desarrolladores PHP.

# **Objetivos**

## **Objetivo General**

Buscar teoría y documentación acerca del framework Symfony, e implementar un programa o aplicación.

## **Objetivos Específicos**

* Buscar, seleccionar y documentar la teoría del framework Symfony.
* Comparar las diferentes versiones del framework Symfony, e identificar sus principales cambios y/o mejoras.
* Aplicar la teoría y práctica acerca del framework Symfony a un programa o aplicación.

# **Redacción**

## **¿Qué es?**

Symfony es un marco de desarrollo de aplicaciones web de código abierto que se basa en el lenguaje de programación PHP. Fue lanzado en 2005 y desde entonces ha sido ampliamente utilizado para desarrollar aplicaciones web de gran escala. Symfony se compone de varios componentes, cada uno de los cuales puede ser utilizado de forma independiente en cualquier proyecto PHP. Además, proporciona un conjunto completo de herramientas para desarrollar aplicaciones web, incluyendo un sistema de gestión de dependencias, un sistema de configuración, un sistema de caché, un sistema de seguridad y una interfaz de línea de comandos (CLI) para realizar tareas comunes.

## **Partes**

### Componentes

Symfony cuenta con varios componentes que se pueden utilizar de forma independiente en cualquier proyecto PHP. Algunos de los componentes más importantes incluyen:

* HttpFoundation: Proporciona una capa de abstracción para trabajar con peticiones y respuestas HTTP.
* Templating: Proporciona una forma de crear plantillas para la presentación de datos.
* Routing: Permite definir las reglas para mapear las URL de una aplicación a las acciones de un controlador.
* DependencyInjection: Proporciona un sistema de gestión de dependencias para inyectar servicios en las clases de una aplicación.
* Config: Proporciona un sistema de configuración para almacenar y acceder a la configuración de una aplicación.
* Debug: Proporciona herramientas para depurar y rastrear errores en una aplicación.
* Cache: Proporciona un sistema de caché para almacenar y recuperar datos de forma rápida.
* Security: Proporciona un sistema de seguridad para autenticar y autorizar a los usuarios de una aplicación.
* EventDispatcher: Proporciona un sistema de eventos para crear aplicaciones basadas en eventos.
* Validator: Proporciona un sistema de validación de datos para comprobar la validez de los datos de una aplicación.
* Translation: Proporciona un sistema de traducción para mostrar contenido en diferentes idiomas.
* Form: Proporciona un sistema de formularios para crear y procesar formularios en una aplicación.
* Serializer: Proporciona un sistema de serialización para convertir objetos en diferentes formatos, como JSON o XML.

### Controladores

En Symfony, un controlador es una clase que contiene acciones (métodos) que manejan las solicitudes HTTP de una aplicación. Los controladores son responsables de procesar las solicitudes, obtener los datos necesarios, ejecutar las tareas apropiadas y devolver una respuesta.

En Symfony, los controladores pueden ser creados de varias maneras, algunas de las cuales incluyen:

* Controladores de clase: Un controlador de clase es una clase PHP normal que contiene acciones (métodos) que manejan las solicitudes. Es posible crear un controlador de clase para cada acción de una aplicación, y estos controladores pueden ser utilizados para separar la lógica de una aplicación.
* Controladores de anotaciones: Es posible utilizar anotaciones para definir las acciones de un controlador. Con las anotaciones, se pueden especificar las rutas de acceso a las acciones, los métodos HTTP permitidos, etc.
* Controladores de servicios: Es posible crear un controlador como un servicio y registrarlo en el sistema de inyección de dependencias de Symfony. Esto permite inyectar servicios en el controlador y hacerlo más fácil de testear.
* Controladores de base de Symfony: Symfony proporciona algunos controladores de base como AbstractController, que proporciona algunos métodos útiles para trabajar con solicitudes y respuestas HTTP.

### Lenguajes de compilación

Symfony es un marco de desarrollo de aplicaciones web basado en PHP, por lo tanto, el lenguaje de programación principal para escribir código en Symfony es el PHP. Sin embargo, es posible utilizar otros lenguajes de programación en conjunto con Symfony, como JavaScript, TypeScript, CSS, y otros, pero estos son utilizados principalmente para el lado del cliente, para la parte de front-end de la aplicación.

Symfony no es un compilador, sino un marco de desarrollo, por lo que no requiere un lenguaje de programación específico para compilar el código. En su lugar, se ejecuta en un servidor web compatible con PHP, como Apache o Nginx, y utiliza el intérprete de PHP para ejecutar el código PHP.

En resumen, el lenguaje de programación principal para escribir código en Symfony es PHP, pero también se pueden utilizar otros lenguajes de programación para el lado del cliente. Sin embargo, Symfony no es un compilador, sino un marco de desarrollo, no requiere un lenguaje de programación específico para compilar el código.

### Servicios de aplicaciones y web

Symfony proporciona varios servicios para desarrollar aplicaciones y servicios web. Algunos de los servicios más importantes incluyen:

* Sistema de rutas: permite definir las reglas para mapear las URL de una aplicación a las acciones de un controlador.
* Controladores: son clases que contienen acciones (métodos) que manejan las solicitudes HTTP de una aplicación.
* Sistema de plantillas: proporciona una forma de crear plantillas para la presentación de datos.
* Sistema de validación: proporciona un sistema de validación de datos para comprobar la validez de los datos de una aplicación.
* Sistema de seguridad: proporciona un sistema de seguridad para autenticar y autorizar a los usuarios de una aplicación.
* Sistema de caché: proporciona un sistema de caché para almacenar y recuperar datos de forma rápida.
* Sistema de eventos: proporciona un sistema de eventos para crear aplicaciones basadas en eventos.
* Sistema de traducción: proporciona un sistema de traducción para mostrar contenido en diferentes idiomas.
* Sistema de formularios: proporciona un sistema de formularios para crear y procesar formularios en una aplicación.
* Sistema de serialización: proporciona un sistema de serialización para convertir objetos en diferentes formatos, como JSON o XML.
* Sistema de inyección de dependencias: permite inyectar servicios en las clases de una aplicación.
* Sistema de configuración: proporciona un sistema de configuración para almacenar y acceder a la configuración de una aplicación.
* Interfaz de línea de comandos (CLI): proporciona una interfaz de línea de comandos para realizar tareas comunes.

## **Comparación de versiones**

v2.0.x

* Estructura de directorios más organizada y modular, lo que facilita la creación y mantenimiento de aplicaciones.
* Sistema de enrutamiento más potente, que permite crear URLs amigables y personalizadas.
* Sistema de plantillas más flexible, que permite crear vistas más complejas.
* Sistema de seguridad mejorado, que permite proteger las aplicaciones contra ataques comunes.
* Mejoras en el rendimiento, lo que permite crear aplicaciones más rápidas y eficientes.
* La adición de una consola de línea de comandos, lo que facilita la ejecución de tareas comunes.
* Mejoras en el soporte para bases de datos, incluyendo un mejor soporte para ORM (Object-relational mapping) y soporte para bases de datos NoSQL.

v.3.0.x

* Simplificación del proceso de configuración y creación de aplicaciones.
* Mejoras en el rendimiento, con un enfoque especial en reducir el uso de memoria.
* Mejoras en el sistema de plantillas, con un enfoque especial en la compatibilidad con otras librerías de plantillas.
* Mejoras en el sistema de seguridad, con un enfoque especial en la prevención de ataques CSRF.
* Mejoras en el sistema de enrutamiento, con un enfoque especial en la compatibilidad con las últimas características de HTTP.
* Mejoras en el soporte para bases de datos, incluyendo mejoras en la compatibilidad con ORM y soporte para bases de datos MongoDB.
* La eliminación de algunas características obsoletas o poco utilizadas para mantener el código base limpio y fácil de mantener.

### 

v.4.0.x

* Una mayor flexibilidad en la configuración y creación de aplicaciones, gracias a la introducción de un enfoque "empiece desde lo mínimo" y la eliminación de algunos componentes opcionales.
* Mejoras en el rendimiento, con un enfoque especial en la eliminación de sobrecarga innecesaria y la mejora de la velocidad de ejecución.
* Mayor integración con otras herramientas y librerías populares, como la compatibilidad con PHP 7 y la integración con el paquete de desarrollo de front-end Webpack Encore.
* Nuevas características para ayudar a los desarrolladores a escribir código de forma segura y estable, como el sistema de tipado estricto en PHP 7.
* Mejoras en la facilidad de uso para los usuarios finales, mediante la mejora de la accesibilidad y la compatibilidad con dispositivos móviles.
* Mejoras en la gestión de la seguridad, mediante la mejora de la prevención de ataques CSRF. Mejoras en la compatibilidad con bases de datos, incluyendo el soporte de la versión 3 de MongoDB.

v.5.0.x

* Mayor flexibilidad en la configuración y creación de aplicaciones, gracias a la introducción de un enfoque "empiece desde lo mínimo" y la eliminación de algunos componentes opcionales.
* Mejoras en el rendimiento, con un enfoque especial en la eliminación de sobrecarga innecesaria y la mejora de la velocidad de ejecución.
* Mayor integración con otras herramientas y librerías populares, como la compatibilidad con PHP 8 y la integración con el paquete de desarrollo de front-end Webpack Encore.
* Nuevas características para ayudar a los desarrolladores a escribir código de forma segura y estable, como el sistema de tipado estricto en PHP 8.
* Mejoras en la facilidad de uso para los usuarios finales, mediante la mejora de la accesibilidad y la compatibilidad con dispositivos móviles.
* Mejoras en la gestión de la seguridad, mediante la mejora de la prevención de ataques CSRF.
* Mejoras en la compatibilidad con bases de datos, incluyendo el soporte de la versión 4 de MongoDB.
* Además de esto, se agregó una nueva forma de trabajar con las traducciones, introducción de los mensajes de depreciación, mejoras en la compatibilidad con los tests, y otros cambios menores.

v.6.0.x

* Type-hints: Symfony siempre se ha caracterizado por tener un gran soporte para nuevas versiones de PHP, y en este caso, han hecho un gran trabajo por lo que respecta a los type-hints de todos los componentes (puedes ver todos y cada uno de estos cambios agrupados en este PR).
* Nested validation rules: Una de las ventajas que nos ha traído esta última versión de PHP(8) es el soporte nativo de anotaciones, tal y como lo poseen otros lenguajes como Java o Ruby. Esto elimina prácticamente los docblocks en PHP y nos da una forma estándar de llevarlo a cabo.
* Enum Type: En lo que respecta a los formularios, contamos con un nuevo tipo, coincidiendo con la nueva versión de PHP 8.1, la cual incluye una nueva estructura de datos muy común en otros lenguajes, llamada enum. Con el EnumType, podemos crear nuestros enums y añadirlos al formulario, sin tener que mapearlos con un choice o similares como veníamos haciendo hasta ahora.
* Bash completion: Sin lugar a dudas, esto es un cambio que no supone ninguna mejora de cara al cliente, pero que a los desarrolladores nos facilita enormemente la vida, prueba de ello son los votos que le ha dado la gente en github.
* Mejoras en el Profiler: Por lo que respecta al Profiler, tenemos muchas mejoras que suponen un avance en el DX, al igual que el cambio anterior. Por una parte, tenemos el preview de emails sin tener que instalar nada adicional; o, en el caso de que estés usando docker compose, sin tener que usar un contenedor con una imagen para capturar el correo como mailhog.
* Han añadido soporte para bootstrap5 y tailwind en la renderización de los formularios, que ya puedes usar incluyendo el tema de forma global en la configuración de twig o de forma local en el template.
* Nuevas funciones en el componente String: trimPrefix y trimSuffix. Más integraciones con el componente notifier.
* Mejoras en la selección de idioma de forma automática a través del header Accept-language. Esto es una gran ventaja para delegar esta tarea al framework, en base a la petición que genera el navegador del usuario.
* Rendimiento de los Voters mejorado a través de la caché haciendo override del método supportsType, de la clase Voter. En caso de que quieras hacerlo a través de la interfaz, deberás implementar el resto de métodos.
* Nueva clase Path perteneciente al componente Filesystem, la cual puedes usar para normalizar el acceso a directorios y archivos, sin importar el sistema operativo desde el que se use. Ya puedes añadir parámetros en el objeto Translatable.
* Mensajes de ayuda en los forms traducibles. Handlers del messenger configurables a través de atributos, sin tener que añadirlo a la configuración. Consumo de mensajes del messenger en lote a través de la interfaz BatchHandlerInterface.
* Permitir el reseteo de los servicios tras consumir un mensaje.

### 

### Aplicación

### Diagrama de clases

| Gráfico 01 - Diagrama de clases de Punto de Venta |
| --- |
|  |
| Elaboración propia |

### 

### Diagrama de secuencia

| Diagrama de secuencia |
| --- |
|  |
| Elaboración propia |

### Fragmentos de código respectivo

Como gestor de paquetes estaremos empleando composer, de manera que para iniciar el proyecto deberemos instalarlo desde su web <https://getcomposer.org/> y además instalar el cli de symphony que lo obtendremos de su web <https://symfony.com/download/>

daremos inicio a nuestro proyecto con el siguiente comando:

| symfony new api |
| --- |

Esto nos generará una estructura simple sin grandes configuraciones para nosotros poder moldearlo a nuestras necesidades.

ahora instalaremos los siguientes paquetes que nos ayudarán a desarrollar la implementación:

| composer require annotations composer require logger composer require symfony/orm-pack composer require --dev symfony/maker-bundle |
| --- |

Hecho esto definiremos la conexión a la base de datos, en nuestro archivo .env colocaremos la siguiente cadena

| DATABASE\_URL="mysql://root:@127.0.0.1:3306/puntoventa?serverVersion=mariadb-10.5.8" |
| --- |

y ejecutaremos el siguiente comando para crearla:

| php bin/console doctrine:database:create |
| --- |

En nuestro gestor de base de datos podemos comprobarlo:

| Captura de gestor de base de datos con la base de datos “puntoventa” creada |
| --- |
|  |
| Elaboración propia |

Lo siguiente que tendremos que hacer será definir nuestras entidades las cuales se generan con el siguiente comando:

| php bin/console make:entity |
| --- |

Aquí definiremos el nombre y las propiedades con sus respectivos atributos. Nuestras entidades quedarían de la siguiente manera

| Entidad persona la cual será generada automáticamente en la base de datos |
| --- |
| <?php  namespace App\Entity;  use App\Repository\PersonaRepository; use Doctrine\ORM\Mapping as ORM;  #[ORM\Entity(repositoryClass: PersonaRepository::class)] class Persona {  #[ORM\Id]  #[ORM\GeneratedValue]  #[ORM\Column]  private ?int $id = null;   #[ORM\Column(length: 255)]  private ?string $dni = null;   #[ORM\Column(length: 255)]  private ?string $nombre = null;   #[ORM\Column(length: 255)]  private ?string $apellidos = null;   #[ORM\Column(length: 255)]  private ?string $fechaDelRegistro = null;   #[ORM\Column(nullable: true)]  private ?int $edad = null;   #[ORM\Column(nullable: true)]  private ?float $peso = null;   #[ORM\Column(nullable: true)]  private ?float $talla = null;   public function getId(): ?int  {  return $this->id;  }   public function getDni(): ?string  {  return $this->dni;  }   public function setDni(string $dni): self  {  $this->dni = $dni;   return $this;  }   public function getNombre(): ?string  {  return $this->nombre;  }   public function setNombre(string $nombre): self  {  $this->nombre = $nombre;   return $this;  }   public function getApellidos(): ?string  {  return $this->apellidos;  }   public function setApellidos(string $apellidos): self  {  $this->apellidos = $apellidos;   return $this;  }   public function getFechaDelRegistro(): ?string  {  return $this->fechaDelRegistro;  }   public function setFechaDelRegistro(string $fechaDelRegistro): self  {  $this->fechaDelRegistro = $fechaDelRegistro;   return $this;  }   public function getEdad(): ?int  {  return $this->edad;  }   public function setEdad(?int $edad): self  {  $this->edad = $edad;   return $this;  }   public function getPeso(): ?float  {  return $this->peso;  }   public function setPeso(?float $peso): self  {  $this->peso = $peso;   return $this;  }   public function getTalla(): ?float  {  return $this->talla;  }   public function setTalla(?float $talla): self  {  $this->talla = $talla;   return $this;  } } |

Con todas las entidades creadas procederemos a la elaboración de nuestros controladores

Para nuestros controladores todos seguirán la siguiente estructura:

| <?php  namespace App\Controller\Api;  use App\Entity\Persona; use App\Repository\PersonaRepository; use FOS\RestBundle\Controller\AbstractFOSRestController; use FOS\RestBundle\Controller\Annotations as Rest; use Symfony\Component\HttpFoundation\Request; use Doctrine\Persistence\ManagerRegistry; use Symfony\Component\HttpFoundation\JsonResponse;  class PersonaController extends AbstractFOSRestController {  /\*\*  \* @Rest\Get(path="/personas")  \* @Rest\View(serializerGroups={"persona"}, serializerEnableMaxDepthChecks=true)  \*/  public function getPersonas(PersonaRepository $repository)  {  return $repository->findAll();  }  /\*\*  \* @Rest\Get(path="/persona/{id}")  \* @Rest\View(serializerGroups={"persona"}, serializerEnableMaxDepthChecks=true)  \*/  public function getPersona(PersonaRepository $repository, $id)  {  return $repository->find($id);  }  /\*\*  \* @Rest\Post(path="/persona/")  \* @Rest\View(serializerGroups={"persona"}, serializerEnableMaxDepthChecks=true)  \*/  public function createPersona(PersonaRepository $repository, Request $request)  {  $parameters = json\_decode($request->getContent(), true);  $persona = $repository->findBy(['dni' => $parameters['dni']]);  if (!!$persona) {  return new JsonResponse([  'success' => false,  'code' => 400,  'message' => 'La persona con DNI: ' . $parameters['dni'] . ', Ya existe.',  ], 400);;  }   $persona = new Persona();  $persona->setDni(isset($parameters['dni']) ? $parameters['dni'] : '');  $persona->setNombre(isset($parameters['nombre']) ? $parameters['nombre'] : '');  $persona->setApellidos(isset($parameters['apellidos']) ? $parameters['apellidos'] : '');  $persona->setFechaDelRegistro(date\_format(date\_create(), 'Y-m-d H:i:s'));  $persona->setEdad(isset($parameters['edad']) ? $parameters['edad'] : null);  $persona->setPeso(isset($parameters['peso']) ? $parameters['peso'] : null);  $persona->setTalla(isset($parameters['talla']) ? $parameters['talla'] : null);   if (  empty($persona->getDni()) ||  empty($persona->getNombre()) ||  empty($persona->getApellidos())  ) {  $error =  (empty($persona->getDni()) ? 'El Campo de dni está vacío, ' : '') .  (empty($persona->getNombre()) ? 'El Campo de nombre está vacío, ' : '') .  (empty($persona->getApellidos()) ? 'El Campo de apellido está vacío' : '');  return new JsonResponse([  'success' => false,  'code' => 400,  'message' => $error,  ], 400);;;  }   return $repository->save($persona, true);  }  /\*\*  \* @Rest\Post(path="/persona/{id}")  \* @Rest\View(serializerGroups={"persona"}, serializerEnableMaxDepthChecks=true)  \*/  public function updatePersona(PersonaRepository $repository, Request $request, $id)  {  $parameters = json\_decode($request->getContent(), true);  $persona = $repository->find($id);  $persona->setDni(isset($parameters['dni']) ? $parameters['dni'] : $persona->getDni());  $persona->setNombre(isset($parameters['nombre']) ? $parameters['nombre'] : $persona->getNombre());  $persona->setApellidos(isset($parameters['apellidos']) ? $parameters['apellidos'] : $persona->getApellidos());  $persona->setEdad(isset($parameters['edad']) ? $parameters['edad'] : $persona->getEdad());  $persona->setPeso(isset($parameters['peso']) ? $parameters['peso'] : $persona->getPeso());  $persona->setTalla(isset($parameters['talla']) ? $parameters['talla'] : $persona->getTalla());   return $repository->save($persona, true);  }  /\*\*  \* @Rest\Delete(path="/persona/{id}")  \* @Rest\View(serializerGroups={"persona"}, serializerEnableMaxDepthChecks=true)  \*/  public function RemovePersona(PersonaRepository $repository, Request $request, $id)  {  $persona = $repository->find($id);  return $repository->remove($persona, true);  } } |
| --- |

Donde se puede acceder a los datos de la base de datos mediante la entidad, y trabajando con los métodos que se generaron en el paso anterior

Para los controladores se emplean repositorios los cuales trabajan con una libreria llamada doctrine se instalaron con los siguientes comandos:

| composer require symfony/orm-pack composer require --dev symfony/maker-bundle |
| --- |

Para hacer uso de las anotaciones que permitieron diferenciar los tipos de metodos http se empleo la libreria Doctrine/annotations, la cual se obtiene con el siguiente comando:

| composer require doctrine/annotations |
| --- |

# **A continuación se mostrará una muestra de las rutas manejadas con sus respectivos outputs.**

# **Ruta Persona**

# 

# 

# 

# 

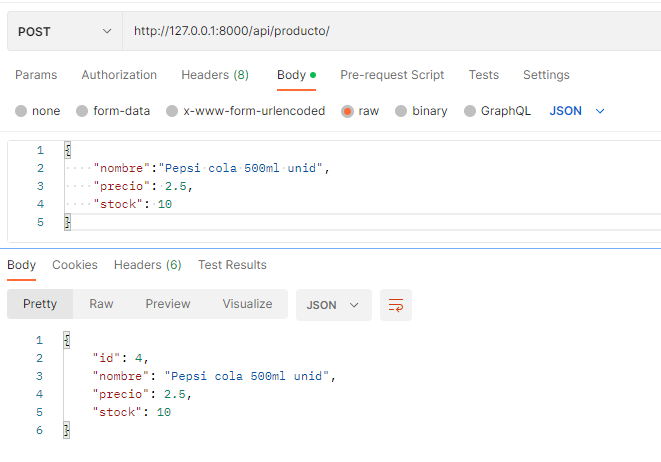
# **Ruta Cliente**

# 

# 

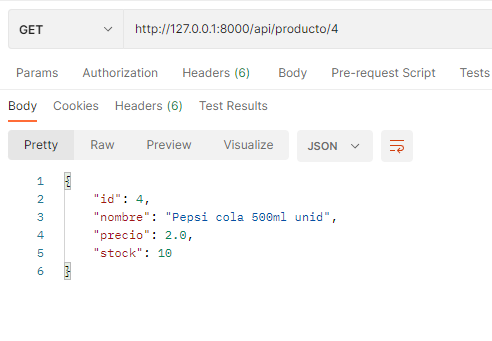
# 

#### Ruta Producto

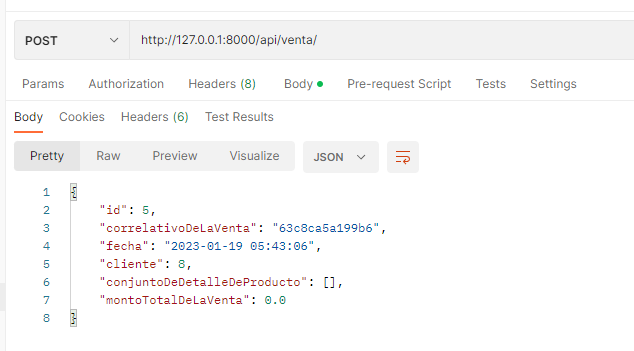


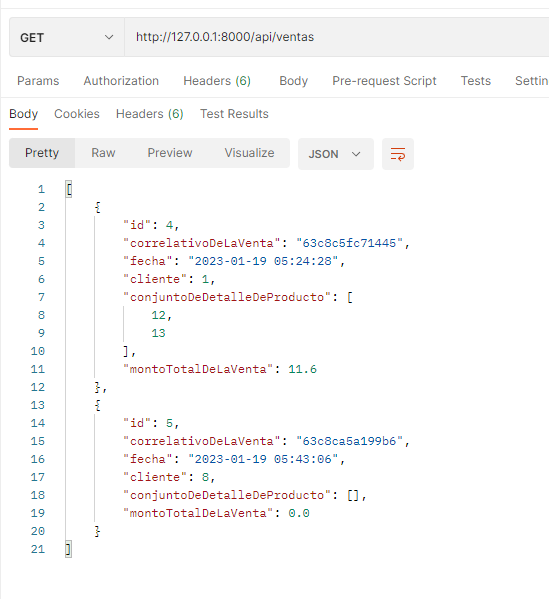
# 

# 



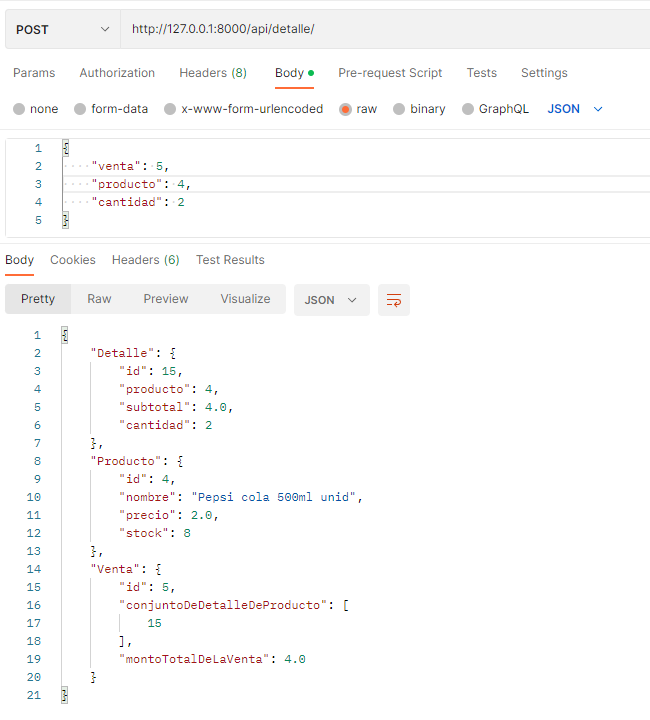
# **Ruta venta**

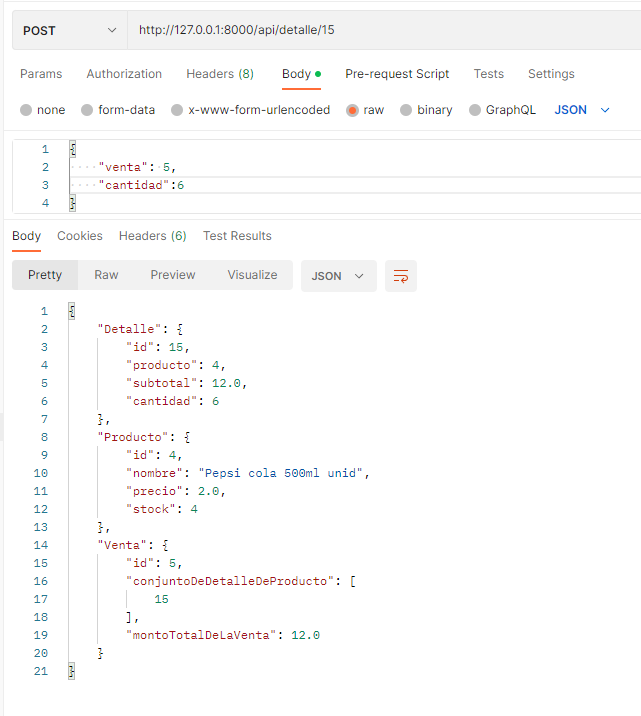


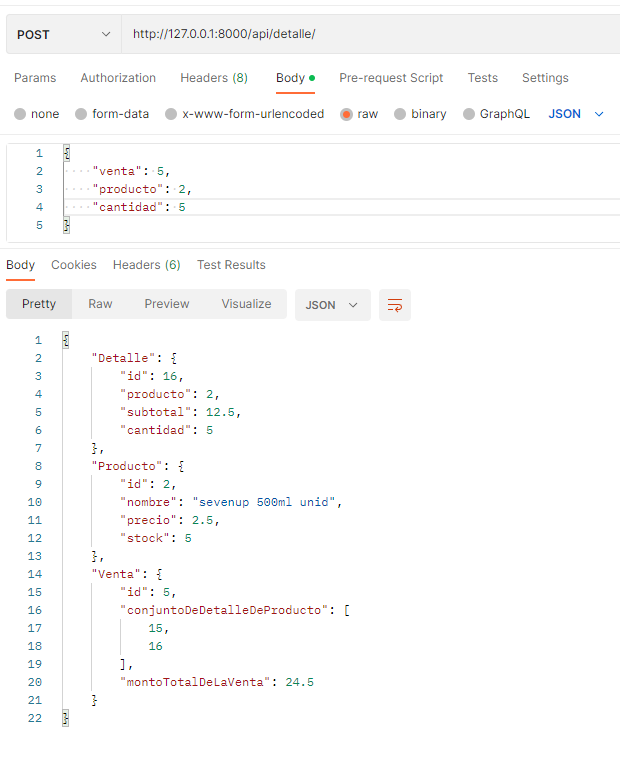


# 

**Ruta Detalle**







# 

# 

# 

# 

# 

# **Conclusiones**

En conclusión, la implementación de Symfony en una aplicación de punto de venta puede ofrecer varias ventajas, como la modularidad, escalabilidad, madurez y facilidad de integración con otras herramientas. Sin embargo, es importante tener en cuenta que el éxito de la implementación también dependerá del equipo de desarrollo y su habilidad para utilizar el marco de manera efectiva. Es recomendable evaluar las necesidades específicas de la aplicación de punto de venta y comparar con otras opciones de marcos antes de tomar una decisión.

La modularidad, escalabilidad, madurez y facilidad de Symfony, permite la reducción del tamaño y la complejidad de la aplicación, al poder elegir solo los módulos necesarios para el proyecto, se puede evitar el uso de componentes innecesarios, lo que reduce el tamaño y la complejidad de la aplicación. Esto también facilita el mantenimiento y el desarrollo futuro de la aplicación.

Permite mayor flexibilidad, para que los desarrolladores cambien o reemplacen los módulos individuales de la aplicación sin afectar a otros componentes, lo que brinda mayor flexibilidad al equipo de desarrollo; mejor escalabilidad al dividir la aplicación en módulos independientes, es más fácil escalar cada componente por separado según sea necesario, en lugar de tener que escalar toda la aplicación de manera simultánea; mejor mantenibilidad, permite que los equipos de desarrollo trabajen en diferentes módulos de manera independiente, lo que ayuda a evitar conflictos y problemas de integración entre componentes.

# 

# 

# **Bibliografía**

*Databases and the Doctrine ORM (Symfony Docs)*. (n.d.). Symfony.com. Retrieved January 16, 2023, from <https://symfony.com/doc/current/doctrine.html>

Mahiques, J. (2021, December 30). *Mejoras y novedades en Symfony 6*. Acceseo. <https://www.acceseo.com/mejoras-y-novedades-symfony-6.html>

*What Is Symfony? (Definition, How It Works, Benefits) | Built In*. (n.d.). Builtin.com. Retrieved January 16, 2023, from <https://builtin.com/software-engineering-perspectives/symfony>