



# CHECKSTOCK

**Autor:** Casandra Marín Angulo

**I.E.S. Francisco Romero Vargas** (Jerez de la  
Frontera)

Desarrollo de Aplicaciones Web

**Curso:** 2022/2024

---

# Índice

Diseño Y Planificación Del Proyecto.....	2
1. Introducción.....	2
2. Finalidad.....	2
3. Objetivos.....	3
4. Medios necesarios.....	4
1. Hardware:.....	4
2. Software:.....	4
5. Planificación.....	5
Realización del Proyecto.....	7
1. Trabajos realizados.....	7
1.1. Investigación (6h).....	7
1.1.1. Competencia.....	7
1.1.2. Aplicaciones o bibliotecas.....	9
1.2. Desarrollo de la idea (14h).....	9
1.3 Instalación(4h).....	11
1.3.1 comandos de instalación:.....	11
1.3.2 comandos usuales:.....	13
1.4 Desarrollo(73h).....	14
1.5 Manual de usuario(7h).....	17
1.6 Documentación y Powerpoint (20h).....	17
2. Problemas encontrados.....	18
3. Modificaciones sobre el proyecto.....	19
4. Posibles mejoras al proyecto.....	19
5. Conclusión.....	20
Bibliografía.....	21

# Diseño Y Planificación Del Proyecto

## 1. Introducción

Muchas empresas del sector comercial necesitan tener un control del stock de los productos que tienen en tienda, muchas de ellas son grandes empresas que también necesitan almacenar productos para no tener que realizar tantos pedidos de envío.

Es por ello, que este documento trata de desarrollar una aplicación de almacenaje para las empresas. Esta es la documentación para el proyecto final de grado superior de desarrollo de aplicaciones web, en él se desarrollarán los diferentes apartados de la creación de dicho proyecto, donde se explicara cual es su finalidad, qué herramientas se han usado o cuales han sido las dificultades que ha conllevado el proyecto.

## 2. Finalidad

El principal propósito de este proyecto es proporcionar a los usuarios una plataforma fácil y sencilla de manejar, que sea intuitiva, útil y que facilite sobre todo el trabajo en las empresas. Algunos de los objetivos específicos que se buscan lograr incluyen:

1. **Optimizar el control de inventario:** Busco ofrecer una solución a las empresas que les permita controlar las existencias que se encuentran en la empresa y las que llegan nuevas a la empresa, con ello el manejo del almacén en tiempo real, y el control de exceso de productos existentes, buscando ser útil tanto para pequeñas como grandes empresas.
2. **Agilizar la gestión de inventario:** Este objetivo se enfoca específicamente en simplificar y acelerar el proceso de gestión de inventario. Para lograrlo, se requiere una interfaz intuitiva y herramientas que permitan realizar operaciones como la verificación de existencias y la actualización del stock de manera rápida y eficiente.
3. **Mejora de la eficiencia operativa:** Al automatizar y simplificar la gestión de inventario, Check Stock contribuye a mejorar la eficiencia operativa de las empresas. Esto se traduce en ahorro de tiempo y

recursos, así como en la reducción de errores en la gestión de existencias.

4. **Anticiparse a la demanda:** Se busca minimizar las pérdidas económicas por la falta de productos en las tiendas. Al manejar la información completa del inventario, la empresa podrá comprobar la cantidad de stock disponible y garantizar la disponibilidad de productos para satisfacer las necesidades de los clientes.
5. **Facilitar el trabajo manual:** Reducir al máximo la repetición de acciones por parte de los trabajadores, otorgándoles espacio para centrarse en otra tarea y ahorrando posibles errores de mala gestión por parte de los mismos.

Para resumir, su finalidad es ofrecer a las empresas una solución completa para manejar de forma eficiente el inventario, ahorrando tiempo y trabajo para los encargados del mismo y ayudar a la gestión del stock disponible, reducir pérdidas por falta de stock y mejorar la satisfacción del cliente mediante la anticipación de la demanda.

### 3. Objetivos

Anteriormente, se ha mostrado que la aplicación tiene varias finalidades, si queremos cumplir con las finalidades, es importante marcarnos unos objetivos, estos son:

1. **Seguimiento de inventario:** Controlar el stock de los artículos en tiempo real, comprobando si se encuentran en almacén, se encuentran en tienda, vienen de camino o sin stock.
2. **Interfaz de usuario sencilla y clara:** Mostrar una interfaz sencilla e intuitiva que ayude a las empresas a realizar las acciones pertinentes en cada caso, para ello se necesitará de un menú de navegación para separar las acciones, botón para agregar más artículos y opciones de búsqueda de un producto para comprobar las existencias.
3. **Automatizar los procesos:** Implementar acciones para avisar de las necesidades de un producto si se encuentra o no disponible, comprobación de peticiones de más artículos, simplificar la repetición de comprobación de artículos así como de posible choque con productos que aún no tienen ubicación.
4. **Unificación de información:** Gestionar de forma fluida las 3 partes de la información, realizando los cambios según las decisiones de la

empresa así como del posterior borrado de los artículos que ya deben haber ingresado en los dos apartados de almacén y tienda.

5. **Gestión de usuario:** Al ser un manejo importante de la información de la empresa, se debe tener constancia de quien realiza los cambios, así como la restricción del uso de la aplicación de los usuarios ajenos a ese proceso.

## 4. Medios necesarios

### 1. Hardware:

- o **Ordenador o Portátil:** Es el mínimo que necesitaremos para desarrollar la API, en mi caso usaré un portátil con bastante RAM y Procesador.

### 2. Software:

- o **Entorno de Desarrollo:** Se usará principalmente Visual Studio Code con sus extensiones para HTML, CSS, JavaScript, PHP y Symfony.
- o **Sistema Operativo:** Se usará principalmente Windows 10 para desarrollar la API, aún no está decidido si usare Ubuntu para el servidor y el montaje mediante docker.
- o **Acceso a Internet:** Se necesitara internet para buscar información necesaria y dudas.
- o **Servidor web:** Usaré un servidor web para ejecutar la aplicación Symfony durante el desarrollo y la prueba. Este será xampp con Apache, este es compatible con PHP.
- o **Bibliotecas y Dependencias:** Instalación de las bibliotecas y dependencias necesarias para realizar el proyecto en symfony. Symfony necesita de ellas para añadir clases, bases de datos, etc...
- o **Base de Datos:** Para almacenar la información se necesitará de una base de datos, en principio MySql por su facilidad con xampp. Esto será útil para guardar datos de usuarios, ubicaciones, así como de los productos.

- o **Herramientas de control de versiones:** Utilizar Github para compartir mi proyecto así como tener un control del mismo.
- o **Recursos externos:** Documentación en formato de CSV para añadir la información que se genera de los productos nuevos.

**Licencias:** En mi caso yo ya tengo la de Microsoft office del 2013 para usar Excel, las demás son de código abierto o son gratuitas.

## 5. Planificación

Es importante hacer una planificación del trabajo por lo que mi intención es trabajar todos los días 2h el proyecto sin contar festivos ni fines de semana, para no quedarme sin tiempo, para ello creo que se deberán seguir las siguientes etapas:

### Etapas 1: Investigación y Diseño (1 semana)

- o Día 1: Investigación sobre otros gestores de inventario, plantilla del documento.
- o Día 2-5: Realizar Diseño de la API, Bocetos y wireframe, selección de colores.

### Etapas 2: Desarrollo de la información (4 semanas)

- o Día 6-8: Creación del proyecto en symfony, crear base de datos, seleccionar las clases que se necesitan crear.
- o Día 9-11: Selección de apartados de la aplicación a desarrollar y cómo quiero que se realicen.
- o Día 12-22: Desarrollo de la aplicación, unificación con la base de datos y las funciones necesarias de las páginas.
- o Día 23-26: Pruebas unitarias y funcionales de todas las funcionalidades.

### Etapas 3: Unificar la información (2 semanas)

- o Día 26-30: Integración y unificación de la información con la base de datos

- o Día 31-35: Implementación de la gestión de usuarios (autenticación, restricción de acceso).

#### **Etapas 4: Pruebas y Ajustes (1 semana)**

- o Día 36: Corrección de errores y ajustes.

#### **Etapas 5: Documentación y Entrega (2 días)**

- o Día 37: Documentación del código, instrucciones de instalación y uso.
- o Día 38: Preparación para la entrega y última revisión.

**Horas totales que se planifican para el proyecto: 84 horas.**

# Realización del Proyecto

## 1. Trabajos realizados

### 1.1. Investigación (6h)

#### 1.1.1. Competencia

He investigado varias apps de gestor de almacenes, cada uno de ellas cumplen unas características u otras:

- **Catinfog:** Gestor de tienda, controla el stock del comercio y las ventas, está dirigido a comercios más pequeños, sus colores son gris y blanco con detalles en color azul.

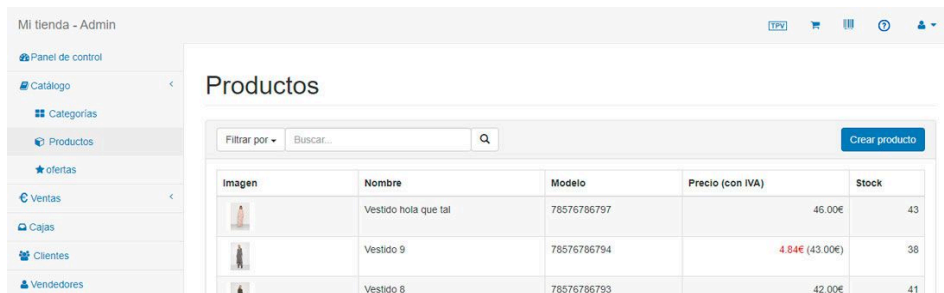



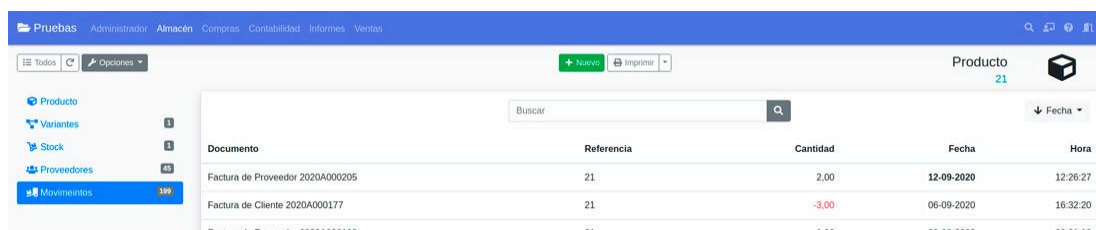


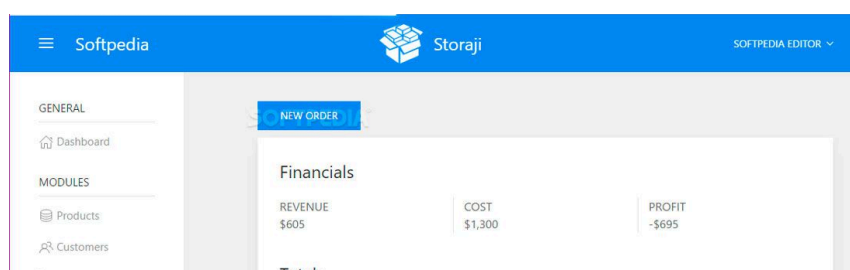
Imagen	Nombre	Modelo	Precio (con IVA)	Stock
	Vestido hola que tal	78576786797	46.00€	43
	Vestido 9	78576786794	4.84€ (43.00€)	38
	Vestido 8	78576786793	42.00€	41

- **Facturascripts:** Permite tener varios almacenes y movimiento de la mercancía entre ellos, también dirigido a crear facturas y albaranes. Sus colores son morado azulado, gris, azul y blanco.



Documento	Referencia	Cantidad	Fecha	Hora
Factura de Proveedor 2020A000205	21	2.00	12-09-2020	12:26:27
Factura de Cliente 2020A000177	21	-3.00	06-09-2020	16:32:20
Factura de Proveedor 2020A000163	21	1.00	20-09-2020	20:21:13

- **Storaji:** Hace seguimiento de productos, vendedores y clientes. También incorpora una forma de controlar las entradas y salidas de almacén. En este momento se encuentra estancado ya que no se actualiza. Sus colores son azul blanco y gris.



REVENUE	COST	PROFIT
\$605	\$1,300	-\$695
Totals		

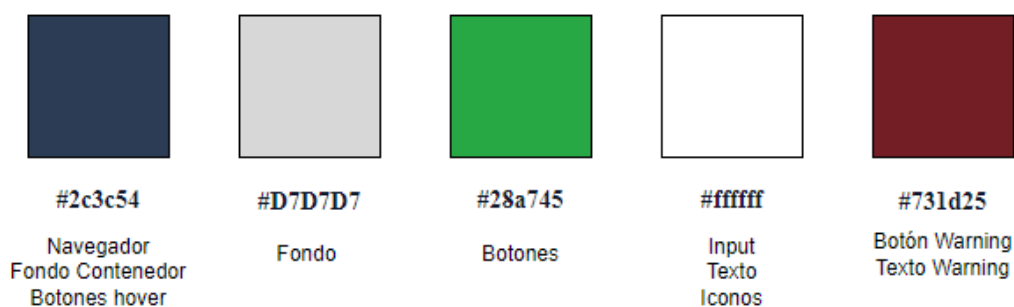


Todos tienen en común los colores, azul, blanco y gris, estos destacan porque transmiten:

- **Azul:** Honestidad, armonía, seguridad, confianza, calma, inteligencia, también es asociado a la profesionalidad y la franqueza.
- **Blanco:** pureza, limpieza, simplicidad, frescura y tranquilidad, se utiliza como fondo de la web o para conseguir contraste con otros colores.
- **Gris:** paz y tranquilidad, mayormente para dar contraste con el blanco, este se suele utilizar en el sector tecnológico asociado a la limpieza y neutralidad.

Gracias a esta investigación y a mi experiencia en el sector comercial se que el uso de esos colores son en especial para que sean fáciles de ver para los usuarios del programa, tienen un diseño sencillo y claro para que no se pierda el tiempo en buscar algo innecesario, por eso se escogen colores poco llamativos y que no sean molestos a no ser que se usen para advertencias o para confirmaciones.

Por ello he escogido los siguientes colores:



Escogí el azul más oscuro para distinguirme de los demás, es un color elegante y legible para el texto.

Como el texto y los input mayormente son blancos elegí un tono gris para darle color al fondo y resaltar los contenedores y el texto.

El verde es un color que demuestra que algo es correcto o esta bien, por eso lo uso en ocasiones para realizar envíos en botones o si se ha actualizado correctamente.

El rojo o naranja se utiliza para urgencias o advertencias, es por eso que lo uso para advertir al usuario de que debe hacer una comprobación.

### 1.1.2. Aplicaciones o bibliotecas

El objetivo del proyecto es gestionar la información de artículos recibidos por la empresa y su ubicación en tienda. Para lograr esto, es necesario leer un archivo externo, como un CSV o Excel, que contenga esta información.

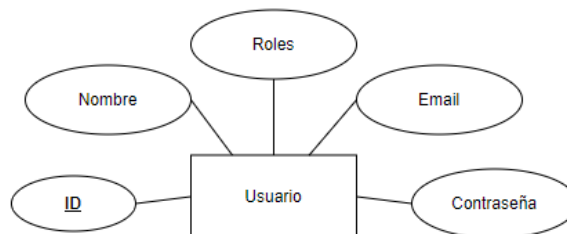
Symfony ya contiene lo necesario para que se puedan leer archivos CSV, para eso utiliza una clase llamada UploadedFile, esta crea una instancia del archivo introducido por formulario. Para leer los archivos introducidos, este tiene el método "getPathname()" que devuelve la ruta completa del archivo subido en el sistema de archivos temporal del servidor.

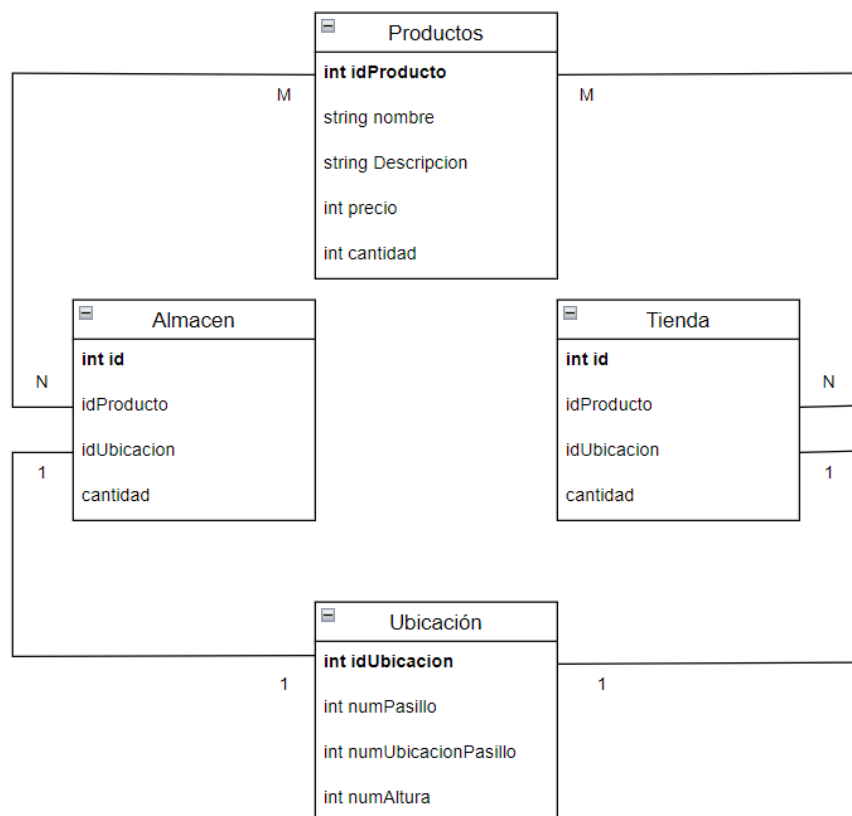
## 1.2. Desarrollo de la idea (14h)

Una vez ya puesto los objetivos y haber estudiado otras competencias y saber que necesito para ello, decidí realizar lo que se necesita para la creación del proyecto.

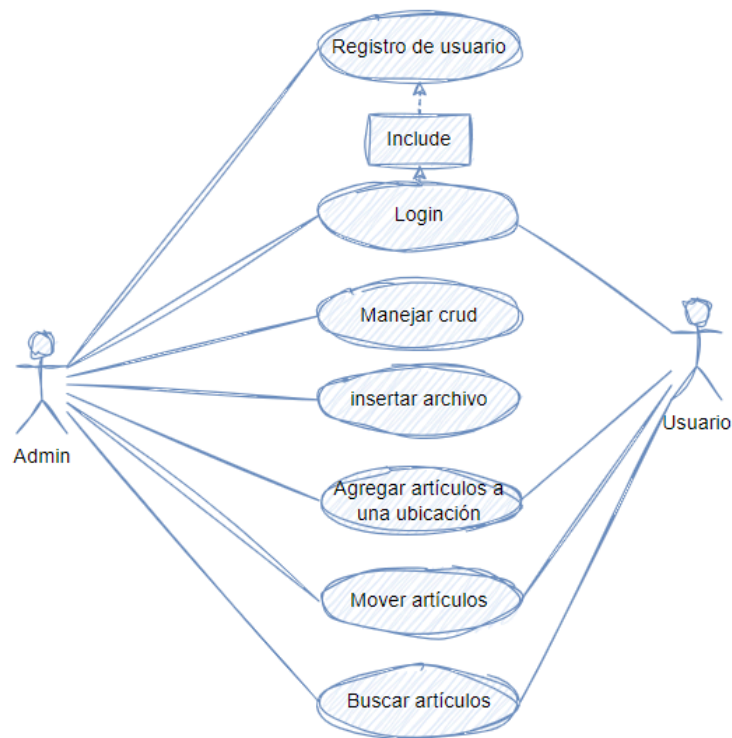
Al ser un proyecto para una empresa decidí realizarlo con symfony, ya que va dirigido más a guardar la información en la base de datos y el manejo de la información y control de la autenticación del usuario.

Para visualizar mejor cómo se organizarán y relacionarán los datos en la base de datos, he creado un diagrama entidad-relación (ERD) y un diagrama de clases. Este diagrama muestra las entidades principales, sus atributos y las relaciones entre ellas:





Además al ser para una empresa es importante definir a qué se puede acceder en la base de datos y quien:



Una vez claro esto me puse con la instalación de la paquetería de symfony.

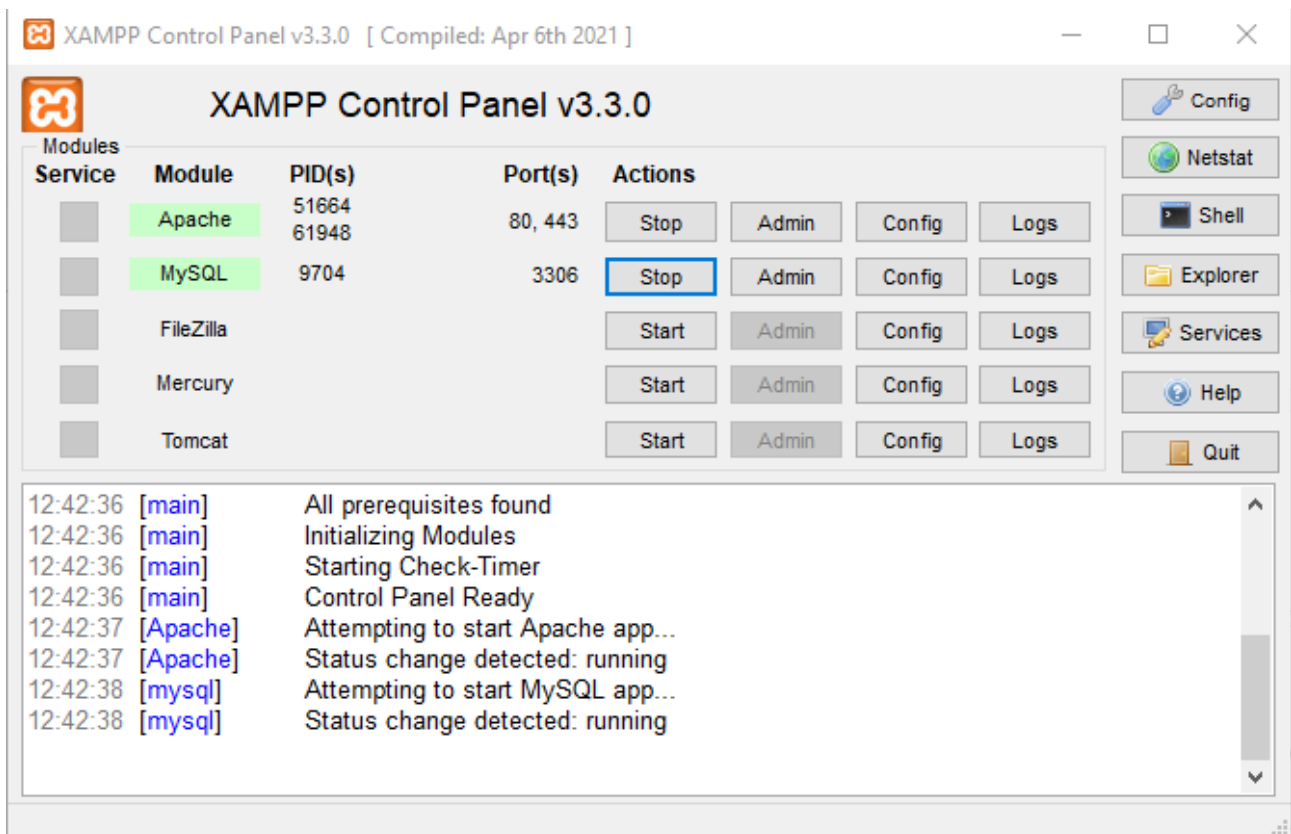
### 1.3 Instalación(4h)

Para poder usar symfony es necesario el instalar una serie de paquetes para su funcionamiento, los paquetes instalados son:

#### 1.3.1 comandos de instalación:

- `composer create-project symfony/skeleton checkStock;`
  - Este comando es para generar el proyecto, donde se crean las carpetas y subcarpetas con la información base para el proyecto.
- `composer require templates:`
  - Los templates son las plantillas de diseño que se necesitan para mostrar la información.
- `composer require symfony/debug-pack.`
- `composer require symfony/orm-pack.`
- `composer require maker-bundle.`

- Todos estos paquetes son necesarios para usar paquetes como doctrine.
- En el archivo .env hay que añadir la línea
  - `DATABASE_URL="mysql://root@127.0.0.1:3306/checkstock"`, 3306 es el puerto que le da XAMPP a mysql :



- `symfony console doctrine:database:create`
  - Crear la base de datos que se ha especificado en el archivo .env.
- `composer require admin`
- `symfony console make:admin:dashboard`
- `symfony console make:crud`
  - Estos tres sirven para la instalación del crud para el administrador.
- `composer require symfony/security-bundle`
  - Paquete para manejar la seguridad del proyecto, con este paquete se incluye security.yaml.

- `make:user`
  - Crea la entidad con su repositorio y actualiza el yaml.
- `php bin/console make:registration-form`
  - para crear el controller de registro
- `php bin/console make:security:form-login`
  - crea el controller y el formulario de login

### 1.3.2 comandos usuales:

Se usan una serie de comandos ya sea para crear, entidades, formularios o para iniciar sesión:

- `symfony console make:entity`
  - Para crear entidades.
- `symfony console make:migration`
  - Hace las migraciones
- `symfony console doctrine:migration:migrate`
  - envía las migraciones creadas anteriormente
- `composer require symfony/form`
  - Genera un nuevo formulario
- `symfony server:start` y `symfony server:stop`
  - para iniciar y cerrar el servidor.

## 1.4 Desarrollo(73h)

Una vez instalado lo anterior, se debe de desarrollar la aplicación, cree las entidades con los casos anteriores, Producto, Usuario, Tienda, Almacén y ubicación, cuando hablo de tienda y almacén me refiero a las ubicaciones que se encuentran en almacén y en tienda:

```
<?php

namespace App\Entity;

use App\Repository\ProductoRepository;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: ProductoRepository::class)]
62 references | 1 implementation
class Producto
{
    #[ORM\Id]
    #[ORM\GeneratedValue]

    #[ORM\Column]
    3 references
    private ?int $id = null;

    #[ORM\Column(length: 255)]
    4 references
    private ?string $nombre = null;

    #[ORM\Column(length: 255)]
    4 references
    private ?string $descripcion = null;

    #[ORM\Column]
    2 references
    private ?int $cantidad = null;
```

Crear los controller fue el siguiente paso, estos son los controladores que hacen las funciones de la app y la envían a una plantilla.

Este es el apartado donde me lleve más tiempo del proyecto, ya que es la parte más importante del mismo. Una de sus funciones se encarga de las páginas que no existen:

```

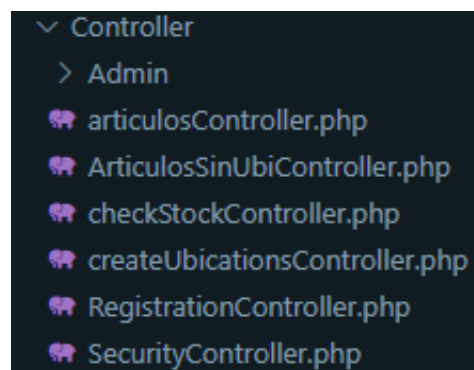
/**
 * Renders a custom page for a given slug.
 *
 * @Route('/{slug}', name: 'comodin', requirements: ['slug' => '^((?!checkStock|firstTime).)*$'])
 */
#[Route('/{slug}', name: 'comodin', requirements: ['slug' => '^((?!checkStock|firstTime).)*$'])]

4 references | 0 overrides
public function comodin(string $slug = null): Response
{
    if ($slug) {
        $textono = 'La página web ' . u(str_replace('-', ' ', $slug)) . ' no existe';
    } else {
        $textono = 'Esta página web no existe ';
    }

    return $this->render('base.html.twig', ['textono' => $textono,]);
}

```

Este proyecto contiene 6 controller aunque dentro de ellos existen más de una función, a parte se encuentran los controller del CRUD, que sirven para que el admin pueda hacer los cambios desde el mismo. ArticulosController contiene 3 funciones:



```

class articulosController extends AbstractController
{
    /**...
    #[Route('/artTienda', name: 'busquedaTienda')]
    8 references | 0 overrides
    public function buscarTienda(Request $request, EntityManagerInterface $entityManager): Response...
    }

    #[Route('/artAlmacen', name: 'busquedaalmacen')]
    8 references | 0 overrides
    public function buscarAlmacen(Request $request, EntityManagerInterface $entityManager): Response...
    }

    #[Route('/moveArt', name: 'moveart')]
    8 references | 0 overrides
    public function moverArticulos(Request $request, EntityManagerInterface $entityManager): Response...
    }
}

```

Estas funciones dependen muchas de formularios, estos formularios se crean de base con una entidad si utilizas el comando “composer require symfony/form”, sin embargo hay que hacerle cambios para recoger la



información que se necesita. Para este proyecto he necesitado 8 formularios que he tenido que modificar, uno de los que más problemas me ha dado es :

```
use Symfony\Component\Form\Extension\Core\Type\NumberType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Form\Extension\Core\Type\ChoiceType;

7 references | 0 implementations
class InsertProductType extends AbstractType
{
    0 references | 0 overrides
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $ubicacionesTiendaChoices = [];
        foreach ($options['ubicacionTienda'] as $ubicacionTienda) {
            $ubicacion = $ubicacionTienda->getUbicacion();
            $ubicacionesTiendaChoices['Pasillo ' . $ubicacion->getNumPasillo() . ' - Ubicación ' . $ubicacion->getNumUbicacionPasillo() . ' - Altura ' . $ubicacion->getNumAlturaPasillo()] = $ubicacion->getId();
        }

        $ubicacionesAlmacenChoices = [];
        foreach ($options['ubicacionAlmacen'] as $ubicacionAlmacen) {
            $ubicacion = $ubicacionAlmacen->getUbicacion();
            $ubicacionesAlmacenChoices['Pasillo ' . $ubicacion->getNumPasillo() . ' - Ubicación ' . $ubicacion->getNumUbicacionPasillo() . ' - Altura ' . $ubicacion->getNumAlturaPasillo()] = $ubicacion->getId();
        }

        $producto = $options['productoNombre'];
        $productoId = null;
        if ($producto != null) {
            $productoId = $producto->getId();
        }

        $builder
            ->add('productoNombre', TextType::class, [
                'data' => $productoId,
                'label' => ' ',
                'disabled' => true,
            ])

            ->add('ubicacionesTienda', ChoiceType::class, [
                'choices' => $ubicacionesTiendaChoices,
                'label' => ' ',
                'placeholder' => 'Ubicación',
            ])

            ->add('cantidadTienda', NumberType::class, [
                'required' => false,
                'label' => ' ',
            ])

            ->add('ubicacionesAlmacen', ChoiceType::class, [
                'choices' => $ubicacionesAlmacenChoices,
                'label' => ' ',
                'placeholder' => 'Ubicación',
            ])
    }
}
```

Los controller envían los formularios y la información para generar la página a un template, esta es la plantilla donde se genera la imagen que va a tener la web:

```
{% extends 'base.html.twig' %}

{% block body %}
<link href="/css/checkstock.css" rel="stylesheet">

<div class="contenedorbase">
<div class="fondo">
<div></div>
</div>
{% if is_granted('ROLE_ADMIN') %}
<div class="formulario">
<h1>¿Han llegado más productos? Introduce el listado:</h1>
{{ form_start(form) }}
<div class="file-upload-wrapper">
<label for="insert_document_archivo" class="file-upload-label" id="upload-label">
<i class="fa fa-upload"></i> Subir archivo
</label>
<input id="file-name-display" type="text" readonly class="file-name-display" placeholder="No se ha seleccionado ningún archivo">
{{ form_widget(form.archivo, { 'attr': { 'id': 'insert_document_archivo', 'class': 'file-upload-input' } }) }}
</div>
<button type="submit" class="submit-button" id="submit-button"><i class="fa fa-paper-plane"></i> Subir archivo</button>
{{ form_end(form) }}
{% if mensaje %}
<p class="mensaje">{{ mensaje }}</p>
{% endif %}
</div>
{% endif %}
{% if mensajeArt != "" %}
<div class="warning">
<div class="icon-advertice">
<i class="fa fa-exclamation-triangle"></i>
</div>
<p class="mensajeArt">{{ mensajeArt }}</p>
<a href="/moveArt" class="bordeAnimado"> <i class="fa fa-exclamation-triangle"></i> Mover artículos <i class="fa fa-exclamation-triangle"></i></a>
</div>
{% else %}
<div class="actualice">
<p class="mensajeArt">No hay artículos con poco Stock, mira si hay artículos sin ubicación: </p>
<a href="/nuevos-articulos"> <i class="fa fa-truck"></i> Artículos sin ubicación</a>
</div>
{% endif %}
</div>

{% if registroExitoso %}
<!-- Modal -->
<div class="modal fade" id="registroExitosoModal" tabindex="-1" role="dialog" aria-labelledby="registroExitosoModalLabel" aria-hidden="true">
<div class="modal-dialog modal-dialog-centered" role="document">
<div class="modal-content">
<div class="modal-body text-center">
<h5 class="modal-title" id="registroExitosoModalLabel">{{ mensaje }}</h5>
<p>¡Se redireccionará a los artículos que aún no tienen ubicación!</p>
</div>
</div>
</div>
</div>
{% endif %}

{% endblock %}
{% block javascripts %}
{{ parent() }}
<script>
document.addEventListener('DOMContentLoaded', function() {
```

Todos suelen extender de la plantilla base, está suele contener el header porque es lo que recogen todas las páginas. Estas plantillas contienen los enlaces de css y javascript.

## 1.5 Manual de usuario(7h)

Desarrollé por último el Manual de Usuario donde se explica cómo se utiliza la app y los pasos que deben hacer tanto el administrador como el usuario.

## 1.6 Documentación y Powerpoint (20h)

Es este mismo documento, en él se detalla todo el proyecto y el powerpoint es el que se dará en la presentación.

## 2. Problemas encontrados

- **Problemas al instalar el CRUD:** Tuve problemas con la instalación del crud en el apartado de comandos, tuve que realizar la instalación dos veces.
- **Añadir un archivo:** Tuve varios problemas en este caso, en internet encontré que debía instalar una serie de paquetes para que me funcionara, pero me indicaba que estaban desactualizados y no funcionaban, intenté varias veces la instalación pero me daba error, busqué en otras páginas pude encontrar la manera de cómo leer un CSV en vez de un Excel.
- **Formularios:** Tuve bastantes problemas aquí, la mayoría de los formularios eran sencillos pero no podía utilizar el mismo formulario para añadir un producto y que al volver a darle la información me salieran las opciones del mismo, para agregar opciones concretas tuve que crear apartados dentro de la función que me generara la información concreta.

```
$productosSinUbicacion = [];  
  
foreach ($productos as $producto) {  
    $productosTienda = $tiendaRepository->findOneBy(['producto' => $producto]);  
    $productosAlmacen = $almacenRepository->findOneBy(['producto' => $producto]);  
  
    $cantidadTotalTienda = $productosTienda ? $productosTienda->getCantidadProducto() : 0;  
    $cantidadTotalAlmacen = $productosAlmacen ? $productosAlmacen->getCantidadProducto() : 0;  
  
    $ubicacionEnTienda = $productosTienda ? $productosTienda->getUbicacion() : null;  
    $ubicacionEnAlmacen = $productosAlmacen ? $productosAlmacen->getUbicacion() : null;  
  
    $cantidadTotalUbicaciones = $cantidadTotalTienda + $cantidadTotalAlmacen;  
    $cantidadTotal = $producto->getCantidad() - $cantidadTotalUbicaciones;  
  
    if ($cantidadTotalUbicaciones < $producto->getCantidad()) {  
        $productosSinUbicacion[] = [  
            'producto' => $producto,  
            'ubicacionEnTienda' => $ubicacionEnTienda,  
            'ubicacionEnAlmacen' => $ubicacionEnAlmacen,  
            'cantidadSinUbicacion' => $cantidadTotal,  
            'cantidadTotalTienda' => $cantidadTotalTienda,  
            'cantidadTotalAlmacen' => $cantidadTotalAlmacen,  
        ];  
    }  
}  
  
$subtiendas = $tiendaRepository->findBy(['producto' => null]);  
$subalmacenes = $almacenRepository->findBy(['producto' => null]);  
  
if (!$producto) {  
    throw $this->createNotFoundException('El producto no existe.');
```

- **Funciones:** Los problemas en las funciones eran de ensayo y error, donde comprobaba que hiciera bien algo.
- **Templates:** Los errores que me daban los templates también eran por los formularios, la forma de lectura del formulario y como recibir esa información, en muchos de ellos se encuentra el javascript añadido para que no se visualice el formulario original o coger la id del usuario.

### 3. Modificaciones sobre el proyecto

- **Seguimiento de inventario:** Me propuse el control en tiempo real, sin embargo no he llegado a añadir el control de los artículos que ya se han pedido y vienen de camino .
- **Automatizar los procesos:** Como no he añadido la función de pedir artículos, no se puede comprobar si se ha hecho la petición de los mismos.
- **Unificación de información:** En un principio mi intención era agregar una tabla donde se ingresaran los productos de camión, sin embargo es innecesario porque ya tenía productos, solo debía agregar nuevos artículos o en cuyo caso sumar la cantidad, lo que no se encuentre en tienda o en almacén serán los artículos que se han ingresado .
- **Gestión de usuario:** La gestión de quien puede entrar está realizada por el administrador por lo que no se pueden registrar usuarios a no ser que los cree él, sin embargo no he asociado que usuario hace los cambios a la app.

### 4. Posibles mejoras al proyecto

Añadir los planteamientos faltantes anteriores, es decir:

- Agregar un apartado de pedido de stock y las cantidades pedidas donde el administrador se encargue de hacer la petición y el usuario pueda dar sugerencia de la misma y ver si se han realizado esos pedidos.
- Controlar las cantidades totales en las posiciones en tienda o almacén y controlar cuantos artículos caben en las baldas, la aplicación no tiene un tope sobre los artículos que se agregan en un lado o en otro, sin embargo en la vida real cada artículo tiene un espacio, es por ello que se debe añadir un apartado donde indique cantidad máxima en cada uno de esos apartados.

- Dar la opción de colocar el producto en otra ubicación que no sea la que se ha añadido inicialmente, en principio se iba a desarrollar de esta manera pero para controlar si ya existía y agregarlo directamente a esta ubicación fue eliminada.
- Controlar que usuario ha hecho el último cambio en tienda o en almacén.

## 5. Conclusión

Esta aplicación web es altamente útil para empresas de todos los tamaños, tanto grandes como pequeñas.

Su interfaz intuitiva asegura que cualquier usuario pueda manejarla sin dificultad. Además, la aplicación está diseñada para crecer y mejorar continuamente, adaptándose a las necesidades específicas de cada empresa.

La seguridad es una prioridad, con restricciones de acceso que previenen el acceso indebido de usuarios externos y minimizan los errores por parte de los empleados.

La facilidad para introducir archivos permite a las empresas despreocuparse de realizar inventarios manuales o ingresar artículos uno a uno desde el albarán.

Este proyecto me ha permitido profundizar en el uso de Symfony, descubriendo tanto sus ventajas como las dificultades asociadas a su implementación. Elegí este proyecto debido a los desafíos que enfrenté en la asignatura de bases de datos, ya que este proyecto se centra en el manejo de información. Aunque no utiliza exactamente el mismo tipo de consultas, ha sido una excelente oportunidad para aplicar y expandir mis conocimientos en esta área.

## Bibliografía

- <https://catinfog.com/programas-almacen/>
- <https://www.rumpelstinski.es/actualidad/psicolog%C3%ADa-del-color-en-dise%C3%B1o-web>
- <https://symfony.com/doc/current/security.html><https://symfony.com/doc/current/security.html>
- <https://symfonycasts.com/screencast/symfony-uploads/upload-in-form>
- <https://www.php.net/>
- <https://chatgpt.com/>
- <https://stackoverflow.com/>
- <https://app.diagrams.net/>
- <https://www.w3schools.com/>
- <https://fontawesome.com/icons>
- <https://getbootstrap.com/docs/>