FRAMEWORKS

- **Symfony** como framework de PHP
- **FosRestBundle.** Definición de controladores para las API REST.
- **NelmioAPIDoc.** Documentación de la API
- **MongoOdbDoctrine.** Framework para conectarse al modelo de datos de nosql
- **ElasticaBundle.** Framework para persistir/buscar los modelos de mongodb.
- **Monolog.** Log de la api.

CONTROLLERS

- **GetOrderController.** Permite dada una id de compra obtener los valores.
- **GetOrderStatusController.** Permite dada una id de compra obtener los valores.
- **PostNewOrderController.** Crear una nueva orden de compra.
- **UpdateOrderStatusController.** Actualizo un status de orden de compra.
- **ExceptionController.** Cuando se produce o se provoca una APIException/HttpException, este será el controlador que recoja la respuesta.

DOCUMENT

- **Order.** Mapping de objeto de mongoDB que representa una nueva compra.
- **OrderStatus.** Mapping de objeto de MongoDB que representa el status de una compra.

SERVICES

- **UpdateOrderStatusListenerService.** Servicio que se encarga de capturar los eventos de actualización de ordenes de pedido.
- **OrderRepositoryService.** Servicio que crea un nuevo documento nosql de nueva de compra.
- **OrderStatusRepositoryService.** Servicio que crea un nuevo documento nosql de actualización de status de compra.
- **SearcherOrderService.** Servicio que nos permite hacer búsquedas usando elastic de un pedido por id.
- **OrderService.** Servicio de entrada de los controladores, donde en función de lo que precise el controller, hacemos unas cosas u otras.
- **ResponseViewService.** Servicio que monta las vistas en caso de error o success para tener la lógica de éstas centralizado en un mismo servicio.

UTILS

- **ApiException.** Excepción custom que se usará en determinados puntos de la aplicación.
- **Utilities.** Clase estática donde se alojaran una serie de operaciones.

CONSIDERACIONES

- No se ha hecho un control específico de skus de productos. Se ha dado por supuesto que todo sku que se envia, es de un producto existente. Como añadido a la aplicación estaría bien tener un listado en json con todos los productos y un servicio que dada una nueva petición de nueva compra, el OrderService compruebe si el sku existe en un servicio aparte.
- He considerado oportuno añadir el Nelmio con swagger para la documentación de la api rest.
- En mi repositorio de github tengo otra api rest con autentificación con JWT (https://github.com/xgs1986/api-rest.git) . - En esta aplicación no he añadido autentificación pero puede verse en la otra api su implementación.
- He considerado oportuno no hacer un update del status de compra. He preferido crear uno nuevo cada vez que llegue una petición para tener un histórico de los diferentes estados. Además he añadido un nuevo campo fecha, informando de la fecha del estado.

- El listener de estado es simple. Únicamente se hace un log de esto. Aquí se podrían hacer muchas cosas, como enviar un mail al usuario que ha hecho el pedido informando de la actualización del estado. Se crearía un servicio que envie mails que se encargue de esta faena.
- He utilizado inyección de dependencias de los servicios. He separado el servicio de searcher, del servicio de indexación por si se decidierá utilizar otro cliente para la búsqueda. Además, he creado el servicio de creación de vistas para que si se desea modificar el output, tenerlo todo centralizado.

PREGUNTAS

a- En nuestro trabajo usamos Jenkins. Nos encargamos de hacer las releases y posteriormente, el equipo de sistemas lo sube a producción.

b- Al igual que el punto anterior, usamos Jenkins. Todos los productos tienen asociado un proyecto de test que una vez se sube al trunk el proyecto, se lanza la tarea de tests asociada y posteriormente, en Jenkins vemos las ejecuciones de los tests.

c- Se ha usado monolog para introducir logs en el fichero /var/logs/dev.log. Periodicamente se puede filtrar este fichero para ver si hay errores o notice interesantes.

d- Esta pregunta no tengo mucho conocimiento. A nivel autodidacta he mirado la herramienta Docker que facilita toda esta faena pero poco más puedo decir ya que como he dicho en el apartado a, sistemas se encarga de la subida a producción de los diferentes rpms que vamos generando.