Proyecto

Diseña e implementa una aplicación web

NOTA: El contenido de este enunciado puede actualizarse y completarse a medida que avanza el curso. Cualquier actualización será notificada en los foros de la asignatura en el aula virtual.

Objetivo

Que el alumno implemente una aplicación web de la forma más parecida a como lo haría a nivel profesional. Algunas de las características de la práctica que simulan un entorno real son las siguientes:

- La práctica se desarrollará en un equipo formado por 4 o 5 alumnos. De esa forma el equipo será similar al que se puede encontrar en un desarrollo web profesional.
- Se utilizarán herramientas profesionales, tanto para el desarrollo en sí como para compartir el código entre los miembros del equipo (Eclipse, Visual Studio Code, GitHub, Docker...)
- La temática de la web podrá ser elegida libremente por los alumnos de cada equipo. De esta forma, los alumnos estarán más motivados y podrán incluir en la aplicación aquellas funcionalidades que deseen (dentro de unos límites que se definirán más adelante en este enunciado)

Temática de la aplicación web

La temática de la aplicación web que hay que diseñar e implementar será elegida libremente por los miembros de cada equipo. Cada equipo puede decidir qué funcionalidades ofrece al usuario, el aspecto gráfico, el esquema de navegación, etc.

A modo de ejemplo, se presentan algunos tipos de aplicaciones web que se podrían implementar, pero el equipo podrá elegir cualquier temática (aunque no esté aquí listada):

- Web de compra/venta de objetos usados (listado, proceso de compra, registro de compras, comunicación entre comprador y vendedor).
- Web de selección/contratación de cuidadores de niños por horas (listado, valoraciones/reputación, filtrado por disponibilidad, características…).
- Web de un gimnasio con seguimiento de entrenamiento (clases colectivas, horarios, registro, información pública…).

- Web de una liga de fútbol / torneo de pádel (equipos, jugadores, partidos, clasificación, calendario…).
- Web de gestión docente de una universidad (horarios, fechas de exámenes, asignación de profesores a asignaturas, fichas de asignaturas, alumnos..).
- Web de un ayuntamiento (noticias, votaciones, registrarse en actividades, calendario de actividades…).
- Web para diseñar viajes turísticos (lugares de interés, planificación, mapas, horarios…).
- Web de una academia de formación (cursos ofertados, activos, reserva, matrícula…).
- Web de venta online (productos, carrito de la compra, stock, pedido, análisis de ventas…).
- Web de reserva de aulas de informática en la universidad (registro de software por aula, calendario, conflictos, gestión manual…).
- Web para formación (tipo Moodle) (asignaturas/cursos por alumno, material trabajo, entrega de trabajos, foro..).
- Web para seguimiento y evaluación de una práctica por fases: (equipos, notas del equipo y de cada integrante, lista de items de corrección, análisis de datos, fechas…).

Es posible copiar alguna web que esté disponible en Internet, por ejemplo una web de noticias como menéame, un gestor de blogs como tumblr, la página web de consulta de horarios de la universidad, etc. En caso de que copie una web existente, se podrán copiar también su diseño gráfico, logotipos, iconos, etc.

Metodología de desarrollo

El proyecto deberá realizarse siguiendo la siguiente metodología de desarrollo.

Desarrollo colaborativo

La aplicación web se desarrollará usando un repositorio de la plataforma GitHub[1]. El código fuente de la aplicación tendrá la licencia Apache 2. El repositorio será creado por el profesor de la asignatura y se le darán permisos de edición a todos los miembros del mismo.

Durante el desarrollo de la aplicación se irá subiendo el código al repositorio a medida que se vaya desarrollando. Es importante que se sigan las buenas prácticas y los commits no sean muy grandes. Es decir, no se considerará adecuado que una aplicación se implemente con un commit que añada decenas de ficheros nuevos. El repositorio no sólo se utilizará para entregar la aplicación, si no que debe usarse para desarrollar la aplicación. Idealmente deberá realizarse un commit del menor tamaño posible que deje la aplicación en un estado estable (que compile). Como mucho, habrá un commit por funcionalidad.

Los mensajes de commit también tienen que ser adecuados y describir correctamente el objetivo del commit. Se recomienda que cada mensaje tenga en su primera línea un resumen del commit de no más de 50 caracteres y opcionalmente una descripción más extensa a partir de la tercera línea

1 https://github.com

(dejando una línea en blanco entre el resumen y la descripción). Existen muchas páginas web con recomendaciones sobre cómo hacer buenos commits en git[2] [3].

Para gestionar el repositorio de github se puede usar cualquier cliente de git.

Los miembros del equipo de desarrollo pueden coordinarse entre sí como consideren conveniente, aunque es recomendable que utilicen un tablero Trello[4] o el servicio de GitHub Projects[5] para gestionar el estado de las tareas y su responsable.

Desarrollo por fases

El proyecto será desarrollado en diferentes fases. De esa forma, los alumnos realizarán un trabajo continuo a lo largo del curso y el profesor podrá hacer un mejor seguimiento del mismo.

Las fases en las que se divide el desarrollo de la aplicación web son:

- **Fase 0:** Formación del equipo y definición de las funcionalidades de la web.
- **Fase 1:** Maquetación de páginas web con HTML y CSS.
- **Fase 2:** Web con HTML generado en servidor y AJAX.
- **Fase 3:** API REST a la aplicación web, despliegue con docker y despliegue remoto.
- **Fase 4:** Web con arquitectura SPA.

El contenido de cada una de estas fases se describe en detalle más adelante en el enunciado del proyecto.

El código de cada una de las fases deberá estar en el repositorio de código en las siguientes fechas antes de que comience la clase:

- **Fase 0:** 05 Febrero 2023
- **Fase 1:** 12 Febrero 2023
- **Fase 2:** 04 Marzo 2023
- **Fase 3:** 02 Abril 2023
- **Fase 4:** 29 Abril 2023

Cuando el código del repositorio esté listo para la entrega de una fase, será necesario hacer un tag en el repositorio con el nombre de la fase que se vaya a entregar. Por ejemplo, al entregar la Fase 2 se deberá crear un tag llamado “fase2”, que será el que será evaluado. Los tags se pueden hacer desde la interfaz web de GitHub al crear una “release”.

IMPORTANTE: No se evaluará el código de la rama main. Si no se ha creado el tag se considera que la fase no se ha entregado.

2 https://github.com/trein/dev-best-practices/wiki/Git-Commit-Best-Practices
3 https://github.com/erlang/otp/wiki/writing-good-commit-messages
4 https://trello.com
5 https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects

Documentación

La documentación del proyecto se incluirá en el fichero README.md en la raíz del repositorio de código. Este fichero se editará usando adecuadamente el formato Markdown (títulos, negritas/cursivas, tablas, texto en formato código, etc.). Se debe verificar que el documento README.md se visualiza correctamente desde la web de GitHub.

En la descripción de cada una de las fases se indica de forma detallada el contenido de la documentación solicitada.

Evaluación

Cada una de las fases serán evaluadas mediante defensa presencial en horario de clase. Todos los alumnos del equipo deben estar presentes en la defensa al tratarse de una actividad de evaluación, aunque tengan dispensa académica.

Presentación de la fase 0 (no evaluable)

La fase 0 no lleva asociada calificación porque consiste únicamente en definir la funcionalidad de la web y la creación del equipo.

En la presentación de esta fase el profesor verificará que la funcionalidad de la web definida por los alumnos cumple con los requisitos exigidos en el enunciado. Si se identifican carencias serán indicadas a los alumnos para que las solventen antes de comenzar con la siguiente fase.

Presentación de la fase 1 (no evaluable)

La fase 1 no lleva asociada calificación porque se asume que los alumnos ya han adquirido los conocimientos necesarios para desarrollar esta fase en la asignatura “Fundamentos de la web” de 2º curso.

En la presentación de esta fase el profesor verificará que el diseño de las pantallas y la navegación entre ellas cumple con los requisitos exigidos en el enunciado. Si se identifican carencias serán indicadas a los alumnos para que las solventen antes de comenzar con la siguiente fase.

Evaluación de las fases 2, 3 y 4 (prácticas 1, 2 y 3)

La evaluación de las fases 2, 3 y 4 del proyecto llevará asociada una calificación que corresponderá con las notas de la práctica 1, 2 y 3 respectivamente.

La evaluación de cada fase se realizará en dos partes:

- **Parte 1 - Cumplimentar formulario de auto-evaluación:** Un único miembro del equipo cumplimentará un cuestionario de auto-evaluación sobre el estado de la práctica (sólo se completará un cuestionario por cada equipo). Este cuestionario se podrá actualizar todas las veces que sea necesario hasta el momento límite de presentación de la fase. En él se realizan preguntas detalladas sobre el estado de las funcionalidades, la calidad del código y el estado de la documentación. Asociado a cada uno de los elementos del cuestionario se indica la

penalización que tiene para la calificación no completar ese elemento.

IMPORTANTE: Si se detecta cualquier intento de completar el cuestionario con información no veraz el profesor calificará la práctica como **suspensa** hasta la convocatoria extraordinaria.

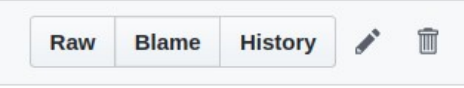
- **Parte 2 - Evaluación por el profesor:** El profesor evaluará cada fase de la siguiente forma:
  - **Demostración:** El profesor pedirá que se le haga una demostración de la aplicación usándola como si fuera un usuario. Los miembros del equipo realizarán un recorrido por la aplicación mostrando sus funcionalidades más importantes con los diferentes tipos de usuarios: anónimo, registrado y administrador.

    IMPORTANTE: La demostración deberá estar preparada de antemano, de forma que los alumnos la puedan realizar de forma rápida. Por ejemplo, si hay que subir una imagen, el alumno que haga la demostración deberá tener una imagen adecuada preparada en su disco.
  - **Revisión de la auto-evaluación:** El profesor revisará cada una de las preguntas del cuestionario de de auto-evaluación durante la defensa. Para cada pregunta, pedirá que los miembros del equipo realicen una demostración del funcionamiento de la aplicación web o pedirá que le muestren la documentación o el código fuente. Por ejemplo, si un diagrama se indica como "completado", se podrá pedir al equipo que muestre ese diagrama durante la defensa.

Participación de los miembros del equipo

Todos los miembros del equipo tienen que participar activamente en el desarrollo de cada una de las fases de la práctica. Para poder justificar su participación, cada uno de los miembros del equipo debe incluir en el README.md:

- Un párrafo describiendo de forma textual las tareas realizadas en esa fase.
- Listado de los 5 commits más significativos durante la fase. Cada uno de esos commits deberá ser un link que lleve a este commit en la interfaz web de GitHub.
- Listado de los 5 ficheros en los que más haya participado el miembro. Se podrá demostrar su participación al usar la opción "blame" de GitHub al visualizar ese fichero.



Si el profesor detecta una baja participación de un alumno concreto, podrá reducirse la nota de ese alumno y podría llegar a **suspender** dicha fase.

Aunque el trabajo es colaborativo y es importante el reparto de tareas, todos los miembros del equipo deberán conocer el código completo de la aplicación (aunque algunas partes hayan sido desarrolladas por otro compañero).

Todos los miembros del equipo deberán estar presentes en la defensa porque el profesor podrá preguntar individualmente a cualquier alumno si lo considera necesario. La ausencia no justificada

llevará aparejada una penalización de **2 puntos** en la calificación o el **suspenso** de la práctica.

Calidad del código fuente

Se exigirá una mínima calidad del código fuente. Al menos deberán tenerse en cuenta los siguientes aspectos:

- El código deberá estar formateado correctamente y usar la misma reglas de estilo en todos los ficheros (se recomienda configurar el entorno de desarrollo para formatear al guardar y así evitar problemas de este tipo).
- El código y los comentarios del mismo deberán estar escritos completamente en inglés. Sólo podrán existir términos en castellano en el código cuando vayan a mostrarse en el interfaz de usuario. La existencias de texto en castellano en el código llevará aparejada una penalización en la calificación.
- Las variables, parámetros, atributos, clases e interfaces deberán tener nombres adecuados que describan su objetivo.
- Se evitará el código duplicado. Cuando dos fragmentos de código sean similares, se utilizarán las técnicas adecuadas para fomentar la reutilización (herencia, composición, subprogramación…)
- Los métodos no serán muy largos y no tendrán mucha complejidad ciclomática.
- El código deberá ser razonablemente eficiente. Por ejemplo, no se considerará válido hacer una consulta a la base de datos para obtener una lista de elementos y luego filtrar esa lista en memoria usando código Java.
- Si se quiere mostrar texto de depuración en la consola del proceso Java, se usará la librería de logs en vez de usar System.out.println().

Si se incumple alguna de estos buenas prácticas la calificación podrá verse penalizada hasta en **2 puntos** dependiendo de su grado.

Calificación

Al evaluar cada una de las fases se tendrán en cuenta los siguientes aspectos:

- Las fases 2, 3 y 4 tendrán una calificación entre 0 y 10.
- Si la fase no cumple con los mínimos exigidos (falta de funcionalidad, incompleta) o no funciona correctamente (tiene errores) se considerará suspensa. El equipo que suspenda una entrega podrá realizar las entregas de las sucesivas fases (subsanando los errores de la entrega suspensa) y recuperar la fase suspensa en la convocatoria extraordinaria.
- Si el profesor detecta carencias menores durante la evaluación de fase podrá poner una nota condicionada a que esos aspectos sean subsanados en un plazo de 1 semana. Si no son subsanados, la nota final para esa fase sería inferior a la nota indicada.
- La nota de cada alumno del equipo es independiente. Un alumno puede obtener más o menos nota que el resto de sus compañeros. La nota individual de cada alumno se determina en base a:

- El conocimiento de la aplicación durante la defensa, ya que todos los alumnos tienen que conocer el código de toda la aplicación aunque ellos no hayan desarrollado esa parte.
- La contribución de cada miembro en esa fase. Se puede llegar al extremo de suspender la práctica a un miembro del equipo si no ha colaborado de forma mínima en la fase.

Convocatoria extraordinaria

Aquellas fases aprobadas en la convocatoria ordinaria se guardan para la convocatoria extraordinaria. Es decir, en la convocatoria extraordinaria los alumnos sólo tendrán que realizar aquellas fases suspensas o no entregadas.

La nota final de la práctica se calculará con los mismos porcentajes que en la convocatoria ordinaria.

En la convocatoria extraordinaria si no se cumplen con las funcionalidades o la aplicación tiene fallos, la práctica (y por tanto la asignatura) se considerará suspensa.

Modificación de miembros del equipo

Aunque se debería evitar en la medida de lo posible, los equipos pueden tener modificaciones en sus miembros debido a las siguientes causas:

- **División del equipo en dos debido a conflictos entre los miembros:** En ese caso, el equipo se puede dividir en 2 y cada nuevo equipo podrá seguir el desarrollo de la práctica partiendo del código en el momento de la división. Hay que evitar esta situación por el incremento de trabajo que supondría para los dos nuevos equipos.
- **Alumno abandona un equipo por conflictos:** Se podrá unir a otro equipo siempre y cuando tenga menos de 4 alumnos y tenga aprobadas las mismas fases que el alumno que quiere incorporarse. Es decir, un alumno no se puede ir a un equipo que tenga aprobadas las fases 2 y 3 si él no tiene aprobada la fase 2 en su equipo original.
- **Alumno deja la asignatura:** Se considera que el alumno ya no forma parte del equipo.

Fase 0: Formación del equipo y definición de las funcionalidades de la web

En la fase 0 se deben realizar las siguientes tareas:

- Formar el equipo de desarrollo.
- Definir las funcionalidades de la aplicación web.

Esta fase no es evaluable.

Formación del equipo

Los alumnos registrarán su equipo en el siguiente documento compartido antes del 30 de Enero a las 14:00.

https://urjc-my.sharepoint.com/:x:/g/personal/micael_gallego_urjc_es/EScLoFvvjG9Bs3QkfNJtOIoBUYZrzQ-48A_jHkZWZLVPeA?e=hOz6dv

Los equipos serán de 4 o 5 integrantes. Aquellos alumnos que no hayan podido formar equipo en la fecha prevista serán asignados a un equipo por el profesor.

Aquellos equipos de 4 alumnos podrán incorporar un nuevo alumno no asignado si el profesor lo considera oportuno.

Requisitos de la aplicación web

Para que la complejidad sea similar para todos los equipos, la aplicación web se deberán diseñar de forma que cumpla con los siguientes requisitos:

- **Entidades:** Una entidad representa un concepto que la aplicación web guarda en la base de datos. Son las clases de dominio, las tablas de la base de datos. La aplicación web deberá gestionar 4 entidades. Una de ellas será la entidad Usuario (para guardar la información de los usuarios que acceden a la web). Las otras 3 entidades dependen de la temática de la web. Es importante que las entidades estén relacionadas entre sí. Por ejemplo:
  - **Web de gestión de torneos o ligas:** Entidades usuario, equipo, torneo y partido. La relación se tiene porque un equipo juega en partidos que forman parte de un torneo.
  - **Web de gestión de cursos:** Entidades usuario, curso, material, mensaje. La relación se tiene porque un alumno está en un curso, que tiene materiales y mensajes en el foro.
  - **Web de cuidado de niños:** Entidades usuario (progenitor o cuidador), solicitud, jornada de cuidado. La relación se tiene porque una jornada de trabajo ha sido realizada a un progenitor por un cuidador. Además, el progenitor puede poner una solicitud, que se puede convertir en una jornada realizada por un cuidador.
  - **Web de gestión de exámenes:** usuario (profesor o alumno), examen, asignatura, realización de examen. Las entidades están relacionadas porque un profesor imparte una asignatura con alumnos. Esos alumnos tendrán que realizar el examen de la asignatura.
- **Tipos de usuarios:** La aplicación web deberá considerar tres tipos de usuarios:
  - **Usuario anónimo:** Aquél usuario que visita la web y no introduce ningún tipo de credenciales para consultar contenido y realizar búsquedas. Salvo que esté justificado por el tipo de aplicación, este usuario sólo podrá consultar información de la web, pero no podrá crearla ni modificarla.
  - **Usuario registrado:** Aquél usuario que tiene que usar sus credenciales para acceder a la web. Este usuario tendrá datos personalizados como su nombre, una imagen, el histórico de acciones realizadas en la web (mensajes en foros, compras, etc..). La web deberá permitir el registro de nuevos usuarios.
  - **Usuario administrador:** Aquél usuario que tiene control total sobre la información de la web. Por ejemplo el alta de productos, la creación de los campeonatos, registro de información o actividades, etc. La web tendrá un único usuario administrador con una contraseña especificada en un fichero de configuración (cifrada).
- **Permisos de los usuarios:** La web tiene que estar diseñada para que los usuarios registrados

puedan ser dueños de ciertos datos. Por ejemplo, si se trata de una tienda, los pedidos previos que ha realizado. Si es una web de contratación de cuidadores, el histórico de cuidados. Si es una web de un gimnasio, los comentarios que pone a cada clase. Esta funcionalidad es importante porque la web debe implementar los mecanismos de seguridad adecuados para que sólo el usuario que ha creado un elemento (su dueño) pueda borrarlo o editarlo.

- **Imágenes:** La web tiene que permitir la subida de imágenes desde el navegador web. Por ejemplo como avatar de los usuarios, fotos de productos, etc.
- **Gráficos:** Se deberán usar gráficos (*charts*) para mostrar algún tipo de información en la web. Por ejemplo:
  - **Web de gestión de torneos o ligas:** Gráfica de líneas con la puntuación/clasificación de los equipos a lo largo del tiempo.
  - **Web de gestión de cursos:** Número de alumnos registrados en cada uno de los cursos.
  - **Web de cuidado de niños:** Gráfica de líneas con las jornadas realizadas por mes para un cuidador.
  - **Web de gestión de exámenes:** Gráfica de barras con el número de alumnos por asignatura.
- **Tecnología complementaria:** Se deberá hacer uso de alguna funcionalidad que se implemente con alguna tecnología o librería que no se haya visto en el material de la asignatura. Por ejemplo:
  - Envío de correos a los usuarios.
  - Generación de PDFs. (facturas, entradas, etc.)
  - Uso de websockets para implementar edición o avisos en tiempo real (chat, pizarra compartida, etc.)
  - Uso de mapas de GoogleMaps / OpenStreetMap para posicionar elementos de la aplicación web (restaurantes, pedidos, etc.)
  - Uso de una API REST de algún servicio externo (Información meteorológica, libros, películas, etc...).
- **Algoritmo o consulta avanzada:** La aplicación web deberá ofrecer alguna funcionalidad que requiera la implementación de un algoritmo o un tratamiento avanzado sobre los datos que gestiona. No basta con que las entidades se puedan crear, modificar, actualizar y borrar. Por ejemplo:
  - Si se opta por una web que permita hacer el seguimiento de una liga de fútbol, la clasificación se deberá calcular de forma automática a medida que se vayan registrando los resultados los partidos.
  - Si se opta por una web de contratación de cuidadores para niños, se implementará un sistema de valoraciones, de forma que la búsqueda tenga en cuenta las valoraciones (mejor valorados primero), o una búsqueda basada en la distancia a la que puede ir el cuidador, etc.
  - Si es una página web de venta de productos, se deberá implementar un sistema de ofertas personalizadas en base a los productos que haya comprado previamente el usuario (por ejemplo mostrar como recomendado un producto de la categoría que más haya comprado el usuario en el pasado).

- Si se diseña una página de gestión de exámenes, se deberán poder analizar las posibles restricciones que existan (que un mismo alumno no tenga dos exámenes el mismo día, que un profesor no tenga exámenes en el mismo momento, etc).

Documentación

Se creará un fichero README.md usando el formato Markdown en la raíz del repositorio de GitHub. Este fichero se visualizará al entrar en la web de GitHub de ese repositorio y tendrá que estar correctamente formateado usando secciones, subsecciones, tablas, código fuente como texto en formato monoespaciado, imágenes, etc.

La documentación se podrá escribir en castellano, aunque se recomienza que se haga en inglés.

El contenido del fichero README.md deberá contener la siguiente información:

- Nombre de la aplicación web.
- Integrantes del equipo de desarrollo: Nombre, Apellidos, correo oficial de la universidad y cuenta en GitHub.
- Si se utiliza trello o cualquier otra herramienta para la coordinación del equipo, deberá ser pública y se incluirá el link de la misma.
- Una sección describiendo cada uno de los aspectos principales de la aplicación web:
  - **Entidades:** Es necesario indicar las entidades principales que gestionará la aplicación. No es necesario definir sus atributos, aunque si las relaciones que existan entre ellas.
  - **Permisos de los usuarios:** Describir someramente los permisos de cada uno de los tipos de usuario. Es importante indicar de qué entidades es dueño el usuario.
  - **Imágenes:** Indicar qué entidades tendrán asociadas uno o varias imágenes por cada objeto/registro.
  - **Gráficos:** Qué información se mostrará usando gráficos. De qué tipo serán los gráficos (líneas, barras, tarta, etc).
  - **Tecnología complementaria:** Qué tecnología complementaria se empleará.
  - **Algoritmo o consulta avanzada:** Indicar cuál será el algoritmo o la consulta avanzada que se implementará.

Fase 1: Maquetación de páginas con HTML y CSS

En la fase 1 se debe definir el esquema de navegación de la web y la maquetación (HTML y CSS) de las páginas más importantes de la web. Se deben incluir las páginas de los administradores (que tienen permisos para editar toda la información de la web).

Para realizar el diseño, se utilizarán datos de ejemplo. Por ejemplo, si la aplicación web es una tienda virtual, para diseñar la página principal, se utilizarán productos de ejemplo con sus imágenes, descripciones, etc. Esta información de ejemplo puede copiarse de cualquier aplicación web de temática similar.

Para el diseño web se utilizará un framework CSS (Bootstrap[6] usando una plantilla gratuita o comercial o Material Design[7]). El objetivo es que la aplicación web tenga un aspecto profesional.

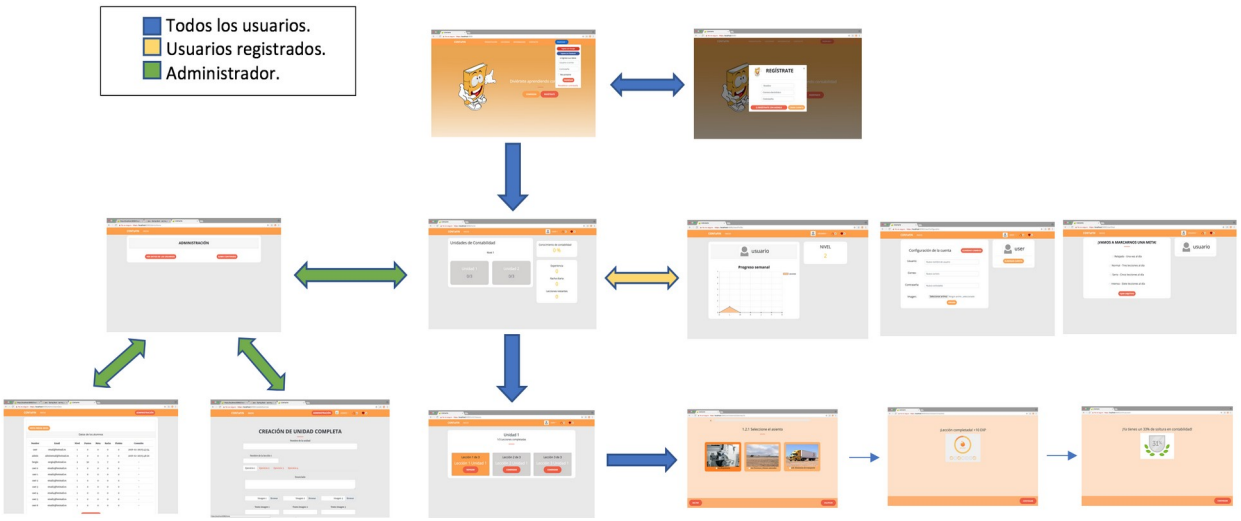
En los casos en los que sea posible, algunos enlaces de las pantallas permitirán simular la navegación real por la página. Por ejemplo, si la aplicación es una tienda, cuando se pulse el enlace para ver la información del producto en detalle, se debería navegar a una página de detalle de un producto (no tiene por qué ser el producto concreto). No obstante, en esta fase los formularios no funcionarán.

Esta fase no es evaluable.

Documentación

Se añadirá a la información de la fase 0 en el README del repositorio la siguiente documentación:

- **Capturas de las pantallas:** Se incluirán capturas de pantalla de cada una de las páginas principales que hayan sido maquetadas. Se acompañarán con una breve descripción en cada una de ellas (un párrafo como mucho).
- **Diagrama de navegación**: Para mostrar la navegación se creará un diagrama en el que se indicará desde qué página se puede navegar hasta otras páginas. Para ello, las páginas del diagrama pueden ser capturas de pantalla en miniatura de las maquetaciones que se hayan realizado. A continuación se muestra un ejemplo de diagrama de navegación:



6 https://getbootstrap.com/
7 https://github.com/material-components/material-components-web/

Fase 2: Web con HTML generado en servidor y AJAX (Práctica 1)

En la fase 2 se debe implementar la aplicación web completamente funcional.

Funcionalidades de la aplicación web

La aplicación deberá tener las siguientes características:

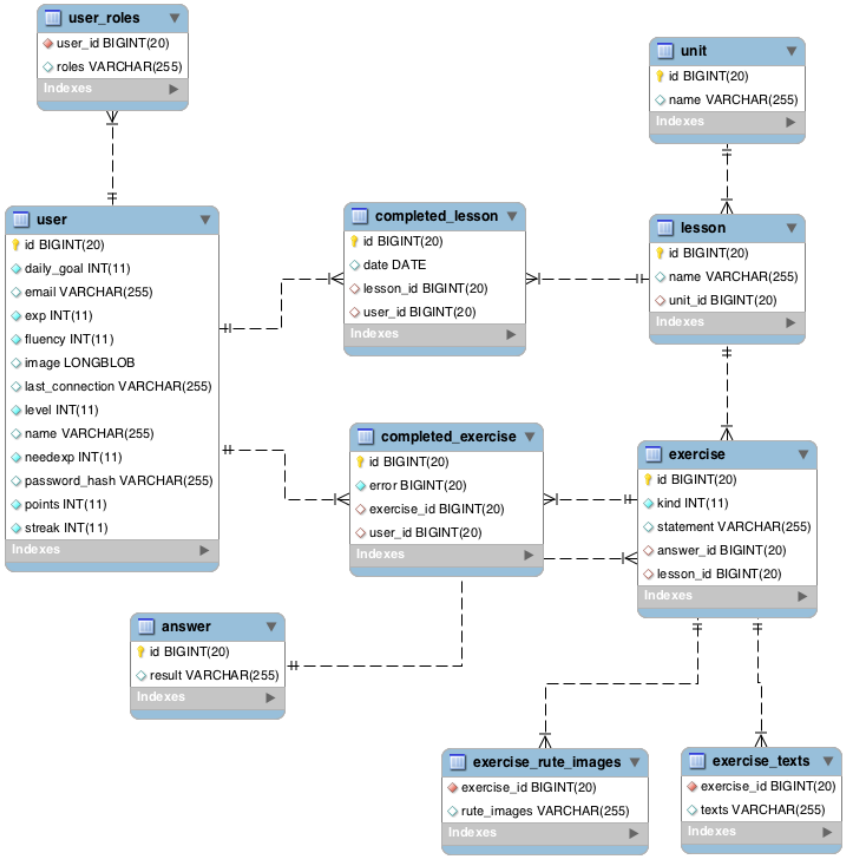
- **Aplicación completa:** La aplicación web se implementará con SpringBoot y base de datos MySQL. Es muy importante que la funcionalidad de la aplicación web sea completa y no queden cosas sin implementar. Es preferible tener menos funcionalidades de las inicialmente previstas a tener funcionalidades a medio implementar. Por ejemplo, no tiene sentido tener correctamente implementado el acceso a la base de datos y tener sin completar la mitad de las páginas. Es preferible tener funcionalidades implementadas completamente (aunque sean menos de las inicialmente previstas) que tener funcionalidades a medias e incompletas. Obviamente, el profesor valorará si el recorte en funcionalidades es razonable o la fase tiene que recuperarse.
- **Datos de ejemplo:** Se cargarán datos de ejemplo en la base de datos al ejecutar la aplicación. Estos datos de ejemplo estarán especificados en el código Java. Los datos e imágenes deberán ser representativos del tipo de aplicación web que se esté desarrollando: libros, restaurantes, productos, etc. Esta información se puede obtener de otras webs.
- **Páginas de error:** Cuando se intente acceder a una URL inexistente o se produzca un error en el servidor se deberá generar una página de error que tenga el mismo estilo gráfico que el resto de páginas de la aplicación.
- **Paginación:** Todas las páginas que potencialmente vayan a mostrar más de 10 elementos deberán mostrar únicamente los 10 primeros y se permitará al usuario cargar más. Para la carga de más elementos, se mostrará un botón o enlace de "Más resultados". Al pulsar este botón, un código JavaScript realizará una llamada AJAX al servidor y cargará los siguientes 10 elementos, que se añadirán en la página actual a los elementos ya existentes. Pulsar el botón o enlace de "Más resultados" no debe recargar la página completa en el navegador web. El servidor web devolverá el HTML que será incrustado usando JavaScript en la página web previamente cargada. Durante el tiempo que dure la carga de elementos, en la página deberá aparecer una animación de espera (spinner). Se puede ver un ejemplo de este funcionamiento en la página de Google Images.
- **Usuarios:** La aplicación gestionará los usuarios usando Spring Security. Si un usuario intenta acceder a una URL sobre la que no tiene permisos, se debería mostrar un error de acceso. Por ejemplo, si un usuario intenta editar un registro del que no es dueño, se deberá mostrar un error. El administrador tendrá una contraseña especificada en el fichero de propiedades (cifrada con BCrypt). Existirán dos usuarios registrados precargados en la base de datos. Se deberá ofrecer un mecanismo para que un usuario pueda registrarse en la aplicación web con un formulario de registro.
- **Código fuente:** Todo el código fuente del proyecto SpringBoot se guardará en una carpeta "backend" del proyecto. Es decir, el fichero pom.xml estará dentro de la carpeta "backend", no en la raíz del repositorio.

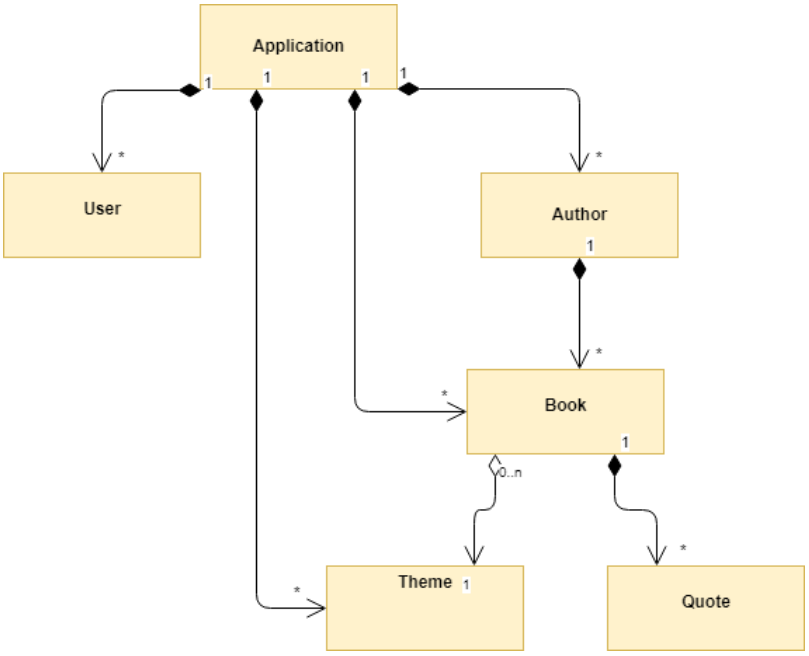
- **Seguridad:** La aplicación web deberá servirse por https en el puerto 8443.
- **Imágenes en base de datos:** Para facilitar el despliegue de la aplicación en entornos restringidos, las imágenes se guardarán en la base de datos en vez de en el sistema de ficheros.

Documentación

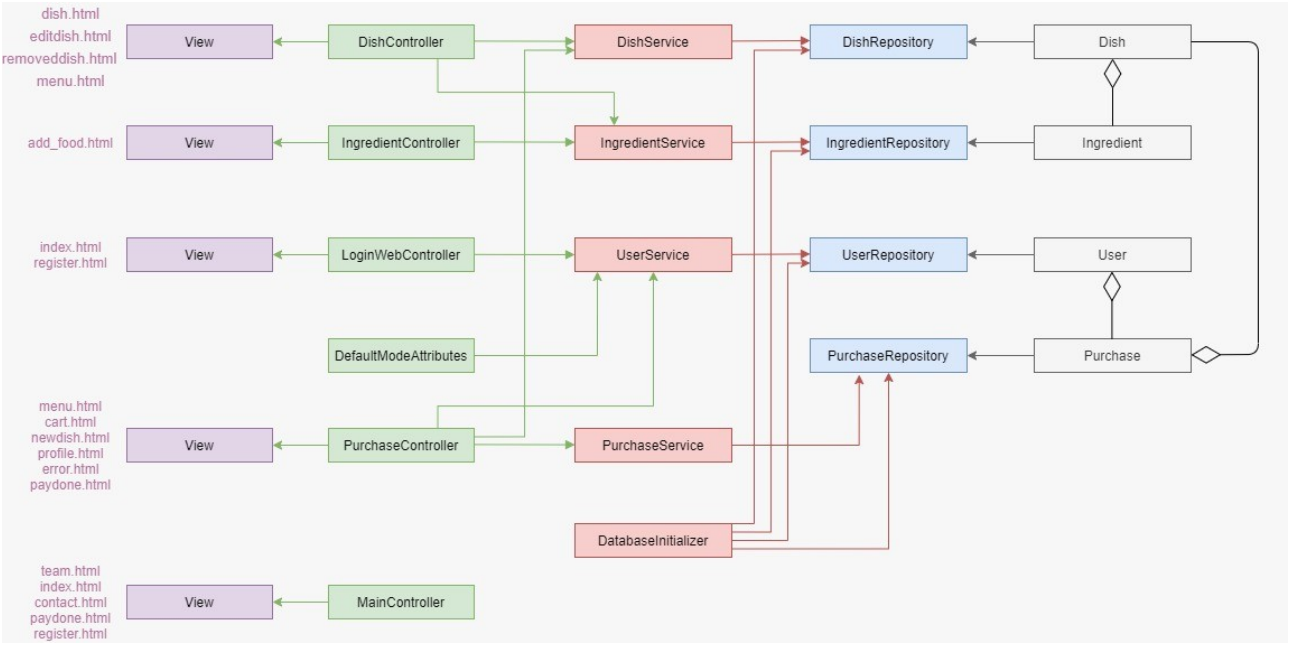
En el README se deberá incluir la siguiente información:

- **Navegación:** Se actualizarán las capturas de pantalla de las páginas principales de la aplicación que se hicieron en la fase 1. En caso de que haya cambiado la navegación, se deberá actualizar también el diagrama de navegación.
- **Instrucciones de ejecución:** Se indicará qué pasos deben seguirse para poder descargar el código del repositorio, construir y ejecutar la aplicación. También se debe especificar cuales son los requisitos (versión de Java, versión de MySQL, maven, etc.). Las instrucciones se especificarán preferentemente como comandos a ejecutar por la línea de comandos. En caso de que no sea posible, se indicará cómo instalar/configurar aplicaciones de forma interactiva.
- **Diagrama con las entidades de la base de datos:** Se incluirá un diagrama con las entidades de la base de datos, sus campos y las relaciones entre ellas. Se podrá incluir un diagrama entidad-relación de la base de datos o un diagrama UML de las clases Java. A continuación se muestran algunos ejemplos de diagramas:





- **Diagrama de clases y templates:** Se creará un diagrama de clases de la aplicación. No se incluirán ni atributos ni métodos en las clases. Se mostrarán las relaciones entre las clases (asociación, composición y herencia) y se diferenciará claramente qué clases son @Controller, @Service, Repository, clases del dominio (entidades) u otro tipo de clases. Para ello se puede usar un código de colores, una distribución de las clases por partes u otro mecanismo. En este diagrama también se incluirán los ficheros que contienen los templates y se indicará con qué @Controller se relacionan. A continuación se muestra un diagrama de ejemplo que usa los colores para indicar el tipo de elemento.



- **Participación de miembros:** Se debe incluir una sección en el README en la que cada miembro del equipo indique su participación en esa fase con:
  - Descripción textual de las tareas realizadas en la fase.
  - Listado de los 5 commits más significativos durante la fase (enlazados a dichos commits en GitHub).
  - Listado de los 5 ficheros en los que más haya participado (enlazados a dichos ficheros en GitHub).

Defensa

Los miembros del equipo deberán tener preparada una demostración de las funcionalidades de la aplicación. Por ejemplo, deberán diseñar un guión con los pasos a seguir y deberán tener preparados datos de ejemplo (imágenes, otros datos, etc.). En esa demostración se mostrará el comportamiento con diferentes tipos de usuarios: no autenticado, autenticado y administrador. Para facilitar la demostración, uno de los integrantes del equipo ya tendrá la aplicación lista para ser usada en su ordenador. Puede utilizar varios navegadores o el modo de incógnito para simular varios usuarios.

Fase 3: Incorporación de una API REST a la aplicación web, despliegue con Docker y despliegue remoto (Práctica 2)

En la fase 3 se deberá implementar una API REST para la aplicación, se deberá empaquetar y distribuir usando la tecnología Docker y finalmente hacer un despliegue en una maquina remota.

Requisitos de la API REST

La implementación de la API REST se tendrá que realizar teniendo en cuenta las siguientes cuestiones:

- La aplicación web ofrecerá a la misma vez la interfaz web y la API REST. Esto permite que la web siga disponible y que clientes móviles u otros servicios puedan acceder a la funcionalidad proporcionada por la web.
- Todas las URLs de la API REST comenzarán con “/api” para diferenciarse de las rutas con las que se accede a páginas de la web.
- La API REST debe diseñarse teniendo en cuenta las buenas prácticas:
  - Los métodos http GET, PUT, POST, DELETE deben usarse de forma adecuada para las operaciones de consulta, modificación, creación y borrado.
  - Las URLs deben identificar los recursos. Cada tipo de recurso se identificará en inglés y en plural.
  - Los códigos de estado de respuesta deben usarse de forma adecuada.
  - Las operaciones de creación deberán devolver la header “Location” cuyo valor es la

URL con la que se puede obtener la representación del recurso recién creado.

- Los operaciones de la API REST que permitan filtrar o buscar elementos deberán tener los criterios de filtrado o búsqueda como parámeotros de la URL (después del ?).

- Toda funcionalidad que se pueda realizar desde el interfaz web debe poder realizarse desde la API REST. Es decir, si se implementara una aplicación SPA o una aplicación móvil podría realizar cualquier operación disponible en la web.
- Los listados que estén paginados en la aplicación web también lo estarán en la API REST. Los parámetros usados para configurar la paginación serán los mismos que los usados en la aplicación web (?page=1).
- Para evitar la duplicación de código y para que la lógica de negocio no esté acoplada a los controladores, se deberá implementar un @Service que será usado tanto por el @Controller y por el @RestController. Ese @Service será el que utilice los repositorios. Un controlador nunca podrá acceder a un repositorio directamente.
- Se deberá incluir en la raíz del repositorio de código una colección de peticiones Postman (api.postman_collection.json) con ejemplos de peticiones a todos los endpoints de la API REST que deberá ser completamente funcional con los datos de ejemplo de la aplicación. De esta forma, será muy sencillo verificar el comportamiento de la API REST desde Postman.
- La API REST deberá estar documentada con OpenAPI generado mediante SpringDoc.

Empaquetado con Docker

El empaquetado con Docker deberá implementarse con las siguientes características:

- **Aplicación web y API REST:**
  - Todas las funcionalidades de la web deberán estar disponibles cuando se ejecute en un contenedor Docker. Hay que prestar especial cuidado en la subida de imágenes, ya que deberá implementarse correctamente para que funcione en este entorno.
  - La aplicación deberá estar disponible en el puerto 8443 por https.
  - Para facilitar la implementación, se desactivará la protección contra CSRF en la API REST (en la web seguirá estando disponible).
- **Docker-compose**
  - Se usará docker compose para ejecutar la aplicación completa formada por el contenedor de la base de datos y el contenedor de la aplicación web.
  - Se usará el contenedor de MySQL estándar de DockerHub. No hay que crear una nueva imagen de base de datos.
  - Se creará un nuevo contenedor que contendrá la aplicación SpringBoot (aplicación web y API REST). Este contenedor se debe publicar en DockerHub en una cuenta creada por alguno de los miembros del equipo (puede ser su cuenta personal).
  - Para que la aplicación web se inicie correctamente una vez que la base de datos ha arrancado y es accesible se implementará algún mecanismo de espera o reintentos. Por

ejemplo usando la política `restart:always`[8].

- El fichero docker-compose.yml configurará el nombre del esquema de la base de datos, la ruta de la base de datos y la password de root de la base de datos en el contenedor de la web y en el de la BBDD. Para ello, usará la capacidad de especificar variables de entorno. Conviene revisar la documentación sobre las variables de entorno de la imagen Docker de MySQL y las variables de entorno SPRING_DATASOURCE_URL, SPRING_DATASOURCE_USERNAME y SPRING_DATASOURCE_PASSWORD que permiten configurar la base de datos en una aplicación implementada con SpringBoot.

- **Código fuente:** En el repositorio de GitHub se deberá crear una carpeta /docker que tendrá los siguientes ficheros:
  - **Dockerfile:** Fichero utilizado para crear la imagen docker de la web.
  - **create_image.sh/.bat/ps1:** Script de bash, bat o power shell que construirá la imagen docker de la aplicación desde el código fuente y la publicará en Docker Hub. El script se ejecutará una vez clonado el repositorio de código. El único requisito para que el script funcione correctamente debería ser que el sistema tenga instalado docker. No se deberá usar el JDK del sistema. Es decir, el script deberá compilar la aplicación utilizando la imagen Docker con compilación multi-stage[9]. Hay que prestar especial atención a las buenas prácticas para crear el contenedor de forma que tenga exclusivamente el software necesario para ejecutar la aplicación.
  - **docker_compose.yml:** Fichero que al ejecutarse con el comando "docker-compose up" se descargue la imagen de la aplicación y de la base de datos de DockerHub y ejecute la aplicación en el puerto 8443 local de forma que sea accesible en https://localhost:8443/.

Despliegue en la infraestructura de la universidad

La aplicación web deberá desplegarse en la infraestructura de la universidad utilizando la misma imagen Docker empleada en el docker-compose.

Para al inicio de la Fase 3 se proporcionará a cada grupo una máquina virtual con un sistema operativo Linux. Dicha máquina virtual estará desplegada en un cluster de la infraestructura de la universidad.

En la máquina virtual se deberá instalar Docker y Docker Compose[10].

La defensa de la práctica se realizará con la aplicación web desplegada en la maquina virtual.

8 https://docs.docker.com/compose/compose-file/compose-file-v3/#restart
9 https://docs.docker.com/build/building/multi-stage/
10 https://docs.docker.com/engine/install/

Documentación

El README se deberá ampliar con:

- **Documentación de la API REST:** La documentación de la API REST se alojará en la carpeta /api-docs de la raíz del repositorio y estará formada por la especificación OpenAPI (en un fichero api-docs.yaml) y por un fichero HTML generado a partir de esa documentación (api-docs.html).

  El proyecto de ejemplo de la fase 3[11] contiene la configuración de Maven (en el pom.xml) que permite generar ambos ficheros de forma automática cuando se construye el proyecto con el comando:

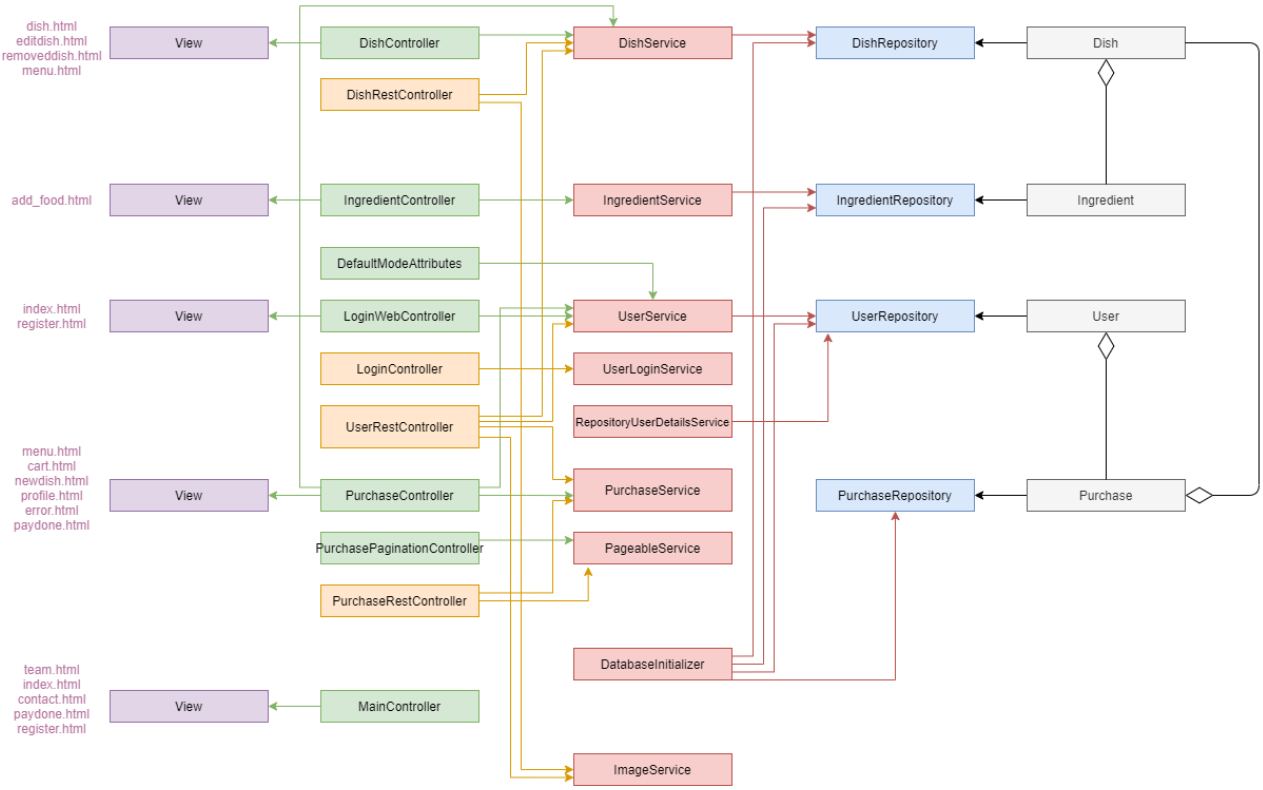
  ```
  $ mvn verify
  ```

  Se debe definir la documentación de forma adecuada en el código Java con SpringDoc de forma que las descripciones del documento HTML sean comprensibles.

  En el README se debe poner un link a la especificación OpenAPI (fichero /api-docs/api-docs.yaml). También se debe poner un link a la documentación en HTML, pero para que se visualice correctamente en el browser se deberá generar el link con el servicio https://raw.githack.com. Este servicio permite acceder a ficheros HTML de GitHub que se visualizan correctamente en un navegador web.

- **Actualización de diagrama de clases:** Se actualizará el diagrama de clases y templates incluyendo las nuevas clases @RestController. Se prestará especial atención a que los @Service compartidos entre los @Controller y los @RestController aparezcan correctamente reflejados en el diagrama. A continuación se muestra un ejemplo de este tipo de diagrama:

11 https://github.com/codeurjc/daw/tree/main/proyectos_de_ejemplo/sample-project-phase3



- **Instrucciones de ejecución de la aplicación dockerizada:** Instrucciones de ejecución usando el docker-compose.yml. Se deberán indicar los requisitos para que se pueda ejecutar, el comando necesario y cómo acceder a la página cuando esté lista para ser usada.
- **Documentación para construcción de la imagen docker:** Se indicará el requisito necesario para construir la imagen dockerizada y cómo ejecutar el script necesario para construir y publicar la imagen Docker.
- **Documentación para desplegar en la maquina virtual:** Se deberán describir los requisitos y comandos para poder desplegar la aplicación en la maquina virtual.
- **La URL de la aplicación desplegada en la maquina virtual:** Se deberá incluir en el README la URL de la máquina virtual donde se ha alojado la aplicación. También se deberán incluir las credenciales de los usuarios de ejemplo (incluyendo el administrador).
- **Participación de miembros:** Se debe incluir una sección en el README en la que cada miembro del equipo indique su participación en esa fase con:
  - Descripción textual de las tareas realizadas en la fase.
  - Listado de los 5 commits más significativos durante la fase (enlazados a dichos commits en GitHub).
  - Listado de los 5 ficheros en los que más haya participado (enlazados a dichos ficheros en GitHub).

Defensa

La aplicación deberá estar correctamente desplegada en la maquina virtual al empezar la defensa.

Para la demostración de la API REST se deberá tener cargada la colección de peticiones Postman que serán ejecutadas a petición del profesor. Las peticiones deberán estar parametrizadas de forma que se puedan ejecutar contra el despliegue en local o contra el despliegue en la maquina virtual.

Para probar la funcionalidad del empaquetado docker, se deberá arrancar la aplicación usando "docker compose up". Los contenedores deberán estar ya descargados en la máquina para no esperar el tiempo de descarga durante la defensa.

Fase 4: Implementación de la web con arquitectura SPA (Práctica 3)

En la fase 4 se debe implementar la aplicación web como cliente SPA con Angular.

Funcionalidades de la aplicación SPA

Esta fase se implementará teniendo en cuenta los siguientes aspectos:

- Este cliente debe tener exactamente la misma funcionalidad que la web implementada en la fase 2. Aunque no es necesario que la interfaz gráfica sea exactamente igual mientras se puedan realizar las mismas operaciones. No obstante, se tendrá que seguir manteniendo el mismo estilo general.
- La aplicación web se podrá usar desde el interfaz original (implementado en la fase 2) o desde el cliente SPA de esta fase. Si se editan datos desde la interfaz original, serán visibles desde la aplicación Angular y viceversa.
- La web con arquitectura tradicional (MVC) será accesible desde la URL principal (https://localhost:8443/) y la web SPA desde la URL (https://localhost:8443/new/).
- Las pantallas se implementarán usando las librerías gráficas ng-bootstrap[12] o angular-material[13] dependiendo del estilo gráfico de la web de la fase 2.
- Todo el código JavaScript implementado en la fase 2 tendrá que volver a implementarse con TypeScript dentro del proyecto Angular. En concreto, se deberá implementar el mismo mecanismo de paginación de la fase 2 basado en AJAX, pero usando la API REST y mostrando las páginas con Angular.
- Se deberán seguir las buenas prácticas de diseño Angular, separando la lógica de gestión del interfaz en componentes y la lógica de conexión con el backend en servicios.
- Publicación:
  - La aplicación Angular se publicará como otro recurso estático más de la aplicación web SpringBoot. Es decir, el resultado de construir la aplicación Angular con `ng build` que está en la carpeta `dist` deberá copiarse a la carpeta `src/main/resources/public/new`. De esa forma, se podrá acceder a la aplicación cuando se acceda a la ruta

12 https://ng-bootstrap.github.io/
13 https://material.angular.io/

https://localhost:8443/new/.

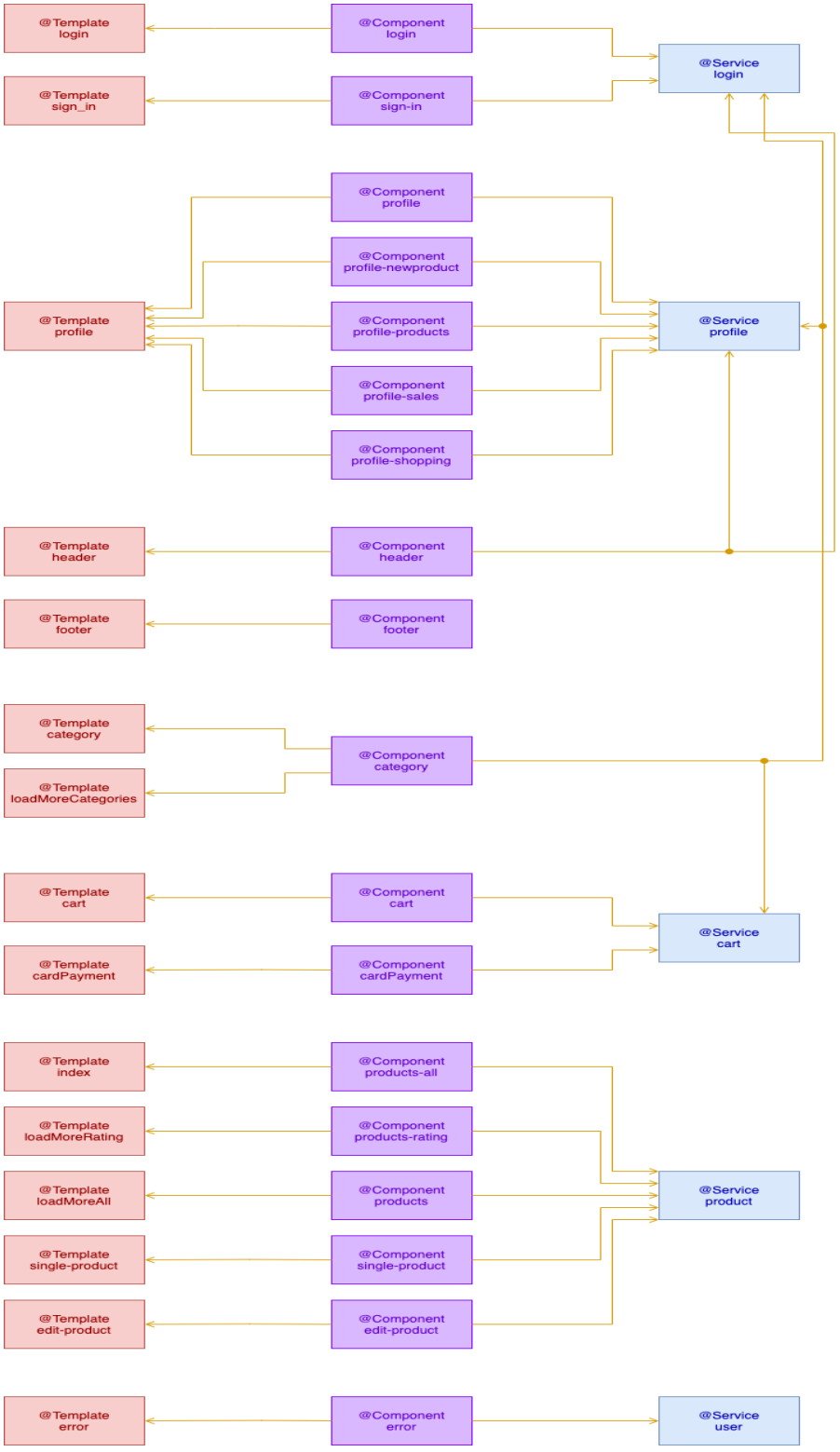
- Como la construcción de la aplicación deberá estar completamente dockerizado, deberá ampliarse el script `create_image` de la fase 3 para que construya también la aplicación Angular. Para ello, debe utilizarse la imagen oficial de Node.js como otro paso más del Docker Multi-stage en la que se ejecutará el comando "npm install" antes del "ng build".
- Se deberá desplegar la aplicación actualizada en la maquina virtual de la universidad. Pudiendo acceder a la web MVC tradicional desde (https://IP_MAQUINA:8443/) y a la web SPA desde (https://IP-MAQUINA:8443/new/).

- **Código fuente:** Todo el código fuente del proyecto Angular se guardará en una carpeta "frontend" del repositorio. Es decir, el fichero angular.cli estará dentro de la carpeta frontend, no en la raíz del repositorio.

Documentación

El README se deberá ampliar incluyendo la información sobre el cliente SPA:

- **Preparación del entorno de desarrollo:** Se añadirá a las instrucciones cómo instalar y configurar el entorno de desarrollo para poder compilar y ejecutar la aplicación SPA con Angular.
- **Diagrama de clases y templates de la SPA:** Reflejará las clases y templates del código Angular. El diagrama deberá diferenciar claramente qué elementos son componentes y cómo se relacionan entre sí (relaciones padre-hijo). También se deberán incluir los servicios y el resto de clases auxiliares. A continuación se muestra un diagrama de este tipo:


@Template login
@Component login
@Service login
@Template sign_in
@Component sign-in
@Component profile
@Component profile-newproduct
@Template profile
@Component profile-products
@Service profile
@Component profile-sales
@Component profile-shopping
@Template header
@Component header
@Template footer
@Component footer
@Template category
@Component category
@Template loadMoreCategories
@Template cart
@Component cart
@Service cart
@Template cardPayment
@Component cardPayment
@Template index
@Component products-all
@Template loadMoreRating
@Component products-rating
@Template loadMoreAll
@Component products
@Service product
@Template single-product
@Component single-product
@Template edit-product
@Component edit-product
@Template error
@Component error
@Service user

- **Participación de miembros:** Se debe incluir una sección en el README en la que cada miembro del equipo indique su participación en esa fase con:
  - Descripción textual de las tareas realizadas en la fase.
  - Listado de los 5 commits más significativos durante la fase (enlazados a dichos commits en GitHub).
  - Listado de los 5 ficheros en los que más haya participado (enlazados a dichos ficheros en GitHub).
- **Vídeo:** En esta fase además se deberá crear un vídeo, subir ese vídeo a YouTube y enlazar el vídeo en el inicio del README. Este vídeo tendrá una duración de entre 2 y 3 minutos y mostrará las principales funcionalidades de la aplicación web usada desde la interfaz Angular. Uno de los alumnos deberá grabarse la voz describiendo qué está haciendo según vaya interactuando con la aplicación web. El vídeo deberá mostrar, como mínimo, los aspectos más relevantes de la aplicación web:
  - Acceso de un usuario anónimo (sin hacer login).
  - Hacer login con un usuario existente y mostrar las funcionalidades adicionales.
  - Si es necesario se usarán varios usuarios (usando varios navegadores web) para que se observe cómo las acciones de un usuario afectan a otro (mensajes, puntos, comentarios, solicitudes de amistad, etc.)
  - Modificación de los datos de una entidad de la aplicación web (cita, autor, libro, entrega, etc...)
  - Subida de imágenes.
  - Login como administrador y mostrar las funcionalidades propias del administrador.