01-2023

code
URJC

Desarrollo de Aplicaciones Web

Colecciones de pruebas de evaluación

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Francisco Gortázar
Correo: francisco.gortazar@urjc.es
Twitter: @fgortazar

Michel Maes
michel.maes@urjc.es

Óscar Soto
oscar.soto@urjc.es

code
URJC

©2023

Micael Gallego, Francisco Gortázar, Michel Maes, Óscar Soto

Algunos derechos reservados

Este documento se distribuye bajo la licencia
"Atribución-CompartirIgual 4.0 Internacional"
de Creative Comons Disponible en
https://creativecommons.org/licenses/by-sa/4.0/deed.es

Ejercicio 1. Programación (6p)

Se desea implementar una aplicación web SPA para consultar la información de los alumnos de la universidad con la siguiente funcionalidad:

- Al entrar en la aplicación se mostrará un título “Alumnos URJC” y un listado con el nombre de cada alumno en un enlace.
- Cuando el usuario pinche en el nombre de un alumno, la aplicación navegará a una nueva página y mostrará una ficha con los datos de ese usuario (Nombre, Dirección y correo electrónico). El título deberá seguir estando visible.
- Habrá un botón “volver” que al ser pulsado volverá a la página principal.
- Se asume que la base de datos a la que se conecta la aplicación ya tiene información de alumnos. La aplicación no tiene la capacidad de dar de alta alumnos ni modificarlos.

Se pide:

- **A) Implementar el backend de la aplicación web con Spring MVC, SpringData y JPA. (2p)**
  - No es necesario escribir el fichero pom.xml, se puede asumir que tiene todas las dependencias correspondientes y está conectada a una base de datos externa correctamente configurada.
  - No es necesario escribir la clase Application.
  - Es necesario escribir TODOS los demás ficheros de la aplicación.
  - No es necesario incluir los imports en los ficheros Java.
  - No es necesario implementar los getter y setter. Se pueden dejar indicados o se puede usar Lombok.
  - Se valorará especialmente que la API REST esté diseñada de forma correcta (formato de las URLs, uso de los códigos de estado, etc.)
- **B) Implementar el frontend usando Angular (2p)**
  - Se puede asumir que se dispone de un proyecto Angular con todos los ficheros necesarios correctamente configurados (index.html, angular.cli, package.json, tsconfig.json, app.module.ts, etc.)
  - Hay que implementar todos los componentes necesarios y un servicio para las peticiones al backend. Hay que implementar el fichero para configurar las rutas de los componentes.
  - Se usará HTML plano en el template de los componentes.
  - No es necesario incluir los imports en los ficheros TypeScript.
  - Se asumirá que el proxy está correctamente configurado y se pueden usar URLs relativas para acceder a la API REST del backend.
- **C) Empaquetar la aplicación en un contenedor Docker (2p)**
  - Escribe un script (en Bash, Bat o PowerShell) que al ejecutarse construya la imagen daw/alumnos:1.0.0 con la aplicación (tanto backend como frontend) empaquetada. Se deberán implementar los ficheros Dockerfile que sean necesarios.
  - Se asumirá que el código Spring está alojado en una carpeta “backend” y el código Angular está alojado en una carpeta “frontend” y el script y el/los Dockerfile están en la carpeta raíz.
  - Se asumirá que en el sistema donde se ejecute el script o se construya el Dockerfile sólo tiene Docker instalado. No dispone de Node.js ni Maven.
  - Se asumirá que en DockerHub existen las imágenes maven:3.6.0, node:14.0.0 y openjdk-jre:11.0.0

Ejercicio 2. Programación (6p)

Se desea implementar una aplicación web SPA para consultar los libros con la siguiente funcionalidad:

- Al entrar en la aplicación aparecerá un formulario con un campo de texto para insertar un criterio de búsqueda y un botón para realizar la búsqueda.
- Al pulsar el botón se realizará una petición al servidor web solicitando los libros que tengan en el título el texto introducido por el usuario en el campo de texto.
- Desde que el usuario pulsa el botón hasta que se muestran los resultados se mostrará una imagen de un spinner de carga (consideramos que está disponible en ruta /assests/spinner.gif relativa a la URL en la que se carga la app).
- Cuando llegan los resultados del servidor, se muestran debajo del formulario.
- Si no hay ningún resultado, se deberá mostrar un mensaje “No hay resultados”.
- Cada libro se representa con la siguiente información: Título y precio.
- Aquellos libros que estén en oferta deberán mostrarse con el texto en color verde.
- Se asume que hay libros ya almacenados en una base de datos correctamente configurada en el pom.xml.

Se pide:

- **A) Implementar el backend de la aplicación web con Spring MVC, SpringData y JPA. (2p)**
  - No es necesario escribir el fichero pom.xml, se puede asumir que tiene todas las dependencias correspondientes y una base de datos adecuada y correctamente configurada.
  - No es necesario escribir la clase Application.
  - Es necesario escribir TODOS los demás ficheros de la aplicación.
  - No es necesario incluir los imports en los ficheros Java.
  - No es necesario implementar los getter y setter. Se puede usar Lombok si se considera.
  - Se valorará especialmente que la API REST esté diseñada de forma correcta (formato de las URLs, uso de los códigos de estado, etc.)
- **B) Implementar el frontend usando Angular (2p)**
  - Se puede asumir que se dispone de un proyecto Angular con todos los ficheros necesarios correctamente configurados (index.html, angular.cli, package.json, tsconfig.json, etc.)
  - Hay que implementar un componente y un servicio para las peticiones al backend. Concretamente, hay que implementar los ficheros: app.component.ts, app.component.css, app.component.html y books.service.ts.
  - Se usará HTML plano en el template.
  - No es necesario incluir los imports en los ficheros TypeScript.
  - Se asumirá que el proxy está correctamente configurado y se pueden usar URLs relativas para acceder a la API REST del backend.
- **C) Empaquetar la aplicación en un contenedor Docker (2p)**
  - Escribe un script (en Bash, Bat o PowerShell) que al ejecutarse construya la imagen daw/libros:1.0.0 con la aplicación (tanto backend como frontend) empaquetada. Se deberán implementar los ficheros Dockerfile que sean necesarios.
  - Se asumirá que el código Spring está alojado en una carpeta “backend” y el código Angular está alojado en una carpeta “frontend” y el script y el/los Dockerfile están en la carpeta raíz.
  - Se asumirá que en el sistema donde se ejecute el script o se construya el Dockerfile sólo tiene Docker instalado. No dispone de Node.js ni Maven.
  - Se asumirá que en DockerHub existen las imágenes maven:3.6.0, node:14.0.0 y openjdk-jre:11.0.0

Ejercicio Práctico: Aplicación web de reserva de sala (6p)

Se desea implementar una aplicación web SPA para reservar salas con la siguiente funcionalidad:

- Al entrar en la aplicación aparecerá una lista de franjas horarias en las que se puede reservar una sala. Para facilitar la implementación, sólo se podrá reservar la sala en horas enteras desde las 9:00 hasta las 19:00 de un único día (el día actual).
- Cada franja horaria podrá estar en dos estados:
  - **Libre:** Al lado de la hora pondrá la palabra “Libre” en la web y a la derecha aparecerá el botón “Reservar”. Al pulsar en reservar, aparecerá un cuadro de texto justo al lado de la franja en el que el usuario podrá poner su nombre. Además aparecerá un botón “Guardar” para dar de alta la reserva en el backend.
  - **Reservada:** Al lado de la hora pondrá el nombre de la persona que tiene la reserva y a la derecha aparecerá un botón con el nombre “Anular”. Al pulsar el botón, la reserva mostrará un cuadro de diálogo de confirmación (confirm) que avisará al usuario si realmente desea borrar la reserva. Si acepta, se borrará la reserva.
- Cualquiera de las operaciones de gestión de reservas (Creación y borrado) será enviada al backend usando una API REST.
- Las reservas estarán almacenadas en una base de datos.

Se pide:

- **A) Implementar el backend de la aplicación web con Spring, SpringData y JPA. (2p)**
  - No es necesario escribir el fichero pom.xml, se puede asumir que tiene todas las dependencias correspondientes y una base de datos adecuada y correctamente configurada.
  - No es necesario escribir la clase Application.
  - Es necesario escribir TODOS los demás ficheros de la aplicación.
  - No es necesario incluir los imports en los ficheros Java.
  - No es necesario implementar los getter y setter. Se puede usar Lombok si se considera.
  - Se valorará especialmente que la API REST esté diseñada de forma correcta (formato de las URLs, uso de los códigos de estado, etc.)
- **B) Implementar el frontend usando Angular (2p)**
  - Se puede asumir que se dispone de un proyecto Angular con todos los ficheros necesarios correctamente configurados (index.html, angular.cli, package.json, tsconfig.json, etc.)
  - Hay que implementar un componente y un servicio para las peticiones al backend. Concretamente, hay que implementar los ficheros: app.component.ts, app.component.css, app.component.html y reservations.service.ts.
  - Se usará HTML plano en el template.
  - No es necesario incluir los imports en los ficheros TypeScript.
  - Se asumirá que el proxy está correctamente configurado y se pueden usar URLs relativas para acceder a la API REST del backend.
- **C) Empaquetar la aplicación en un contenedor Docker (2p)**
  - Escribe un script (en Bash, Bat o PowerShell) que al ejecutarse construya la imagen daw/libros:1.0.0 con la aplicación (tanto backend como frontend) empaquetada. Se deberán implementar los ficheros Dockerfile que sean necesarios.
  - Se asumirá que el código Spring está alojado en una carpeta “backend” y el código Angular está alojado en una carpeta “frontend” y el script y el/los Dockerfile están en la carpeta raíz.
  - Se asumirá que en el sistema donde se ejecute el script o se construya el Dockerfile sólo tiene Docker instalado. No dispone de Node.js ni Maven.
  - Se asumirá que en DockerHub existen las imágenes maven:3.6.0, node:14.0.0 y openjdk-jre:11.0.0

Universidad Rey Juan Carlos

Desarrollo de Aplicaciones Web
3º Grado en Ingeniería del Software

Examen 17 de Mayo de 2022

El examen tiene una duración de 1:30 horas.

Ejercicio de Programación (6p)

Se desea implementar una aplicación web para anunciar los conciertos de una sala. Esta aplicación web deberá tener las siguientes características:

- Funcionalidades:
  - Al entrar en la aplicación se mostrará un título “Conciertos MusiURJC”, un formulario para dar de alta un nuevo concierto y la lista de conciertos existentes.
  - Los datos de un conciertos son: Tïtulo, fecha y sala (A, B o C).
  - Cuando el usuario complete los datos de un concierto y le de al botón de “Crear”, se enviarán los datos del concierto al servidor, que los guardará en la base de datos y se volverá a recargar la página de nuevo. Ese nuevo concierto se mostrará en la lista de conciertos si se ha grabado correctamente en la base de datos. Opcionalmente se puede poner una página informando de que el concierto se ha guardado correctamente antes de mostrar la página principal.
  - Los conciertos no se pueden borrar.
  - Cuando un usuario quiere crear un concierto, el servidor verificará que no hay ningún otro concierto en esa misma sala en esa misma fecha. En caso de error de validación, se le mostrará un error al usuario.
- Cuestiones de implementación:
  - Se deberá implementar la web con arquitectura web tradicional (generando los HTML en el servidor) y también como una aplicación SPA. Es decir, como hemos hecho en la práctica.
  - La aplicación web tradicional estará en la raíz de la URL (http://localhost/) y la aplicación SPA estará en (http://localhost/next/).
  - No es necesario usar ningún tipo de estilo CSS ni librería de componentes.
  - La validación de fecha y sala se implementará en el servidor. Para ello se usará código Java (no se usarán restricciones de la base de datos). Se valorará que no haya código repetido en el servidor.
  - La forma de mostrar el error al usuario puede ser diferente en la arquitectura web tradicional y en SPA. Por ejemplo, en arquitectura web tradicional puede ser una nueva página de error que muestre el mensaje “Error al crear el concierto” y en SPA puede ser una alerta de JavaScript (`alert("Error al crear el concierto");`).

Se pide:

- **A) Implementar la aplicación web con Spring MVC, SpringData y JPA. (2.5p)**
  - No es necesario escribir el fichero pom.xml, se puede asumir que tiene todas las dependencias correspondientes y está conectada a una base de datos externa correctamente configurada.
  - No es necesario escribir la clase Application.
  - Es necesario escribir TODOS los demás ficheros de la aplicación.
  - No es necesario incluir los imports en los ficheros Java.
  - No es necesario implementar los getter y setter. Se pueden dejar indicados con un comentario.
  - En caso de que se necesite definir una API REST, se valorará especialmente que esté diseñada de forma correcta (formato de las URLs, uso de los códigos de estado, etc.)
  - Se valorará que no haya código duplicado.
- **B) Implementar el frontend usando Angular (2p)**
  - Se puede asumir que se dispone de un proyecto Angular con todos los ficheros necesarios (index.html, angular.cli, package.json, tsconfig.json, app.module.ts, etc.).

- Hay que escribir todos los ficheros necesarios para implementar los componentes, servicios y configuración de rutas (en caso de que sean necesarias).
- No es necesario incluir los imports en los ficheros TypeScript.
- Se usará HTML plano en el template de los componentes (sin ninguna librería de componentes como ng-bootstrap o angular-material).
- Se asumirá que el proxy está correctamente configurado y se pueden usar URLs relativas para acceder a la API REST del backend.

- **C) Empaquetar la aplicación en un contenedor Docker (1.5p)**
  - Escribe un script (en Bash, Bat o PowerShell) que al ejecutarse construya la imagen daw/alumnos:1.0.0 con la aplicación (tanto backend como frontend) empaquetada. Se deberán implementar los ficheros Dockerfile que sean necesarios.
  - Se asumirá que el código Spring está alojado en una carpeta “backend” y el código Angular está alojado en una carpeta “frontend” y el script y el/los Dockerfile están en la carpeta raíz.
  - Se asumirá que en el sistema donde se ejecute el script sólo tiene Docker instalado. No dispone de Node.js ni Maven.
  - Se asumirá que en DockerHub existen las imágenes maven:3.6.0, node:14.0.0 y openjdk-jre:11.0.0

Universidad Rey Juan Carlos

Desarrollo de Aplicaciones Web
3º Grado en Ingeniería del Software

Examen 5 de Julio de 2022

El examen tiene una duración de 1:30 horas.

Ejercicio de Programación (6p)

Se desea implementar una aplicación web para gestionar una lista de tareas pendientes de hacer. Esta aplicación web deberá tener las siguientes características:

- Funcionalidades:
  - Al entrar en la aplicación se mostrará el título “Tareas”, un formulario para dar de alta una nueva tarea y la lista de tareas que se han dado de alta previamente.
  - Una tarea es simplemente su descripción. Por ejemplo: “Comprar pan”, “Estudiar DAW”, “Preparar vacaciones”, etc.
  - Cuando el usuario pone la descripción de la tarea y le de al botón de “Crear”, se enviarán la descripción al servidor, que la guardará en la base de datos.
  - Al guardar una nueva tarea en la base de datos, la lista de tareas se debe actualizar para incluir a la nueva tarea.
  - Las tareas se guardarán en la base de datos como mayúsculas, independientemente de cómo se hayan escrito en el formulario de la web. Esta transformación a mayúsculas se realizará en el servidor (por seguridad).
  - En la lista de tareas, al lado de cada tarea aparecerá un botón “Borrar”. Al pulsar ese botón, la tarea será borrada de la base de datos y se actualizará la lista de tareas para que se elimine también de la lista.
- Cuestiones de implementación:
  - Se deberá implementar la web con arquitectura web tradicional (generando los HTML en el servidor) y también como una aplicación SPA. Es decir, como hemos hecho en la práctica.
  - La aplicación web tradicional estará en la raíz de la URL (http://localhost/) y la aplicación SPA estará en (http://localhost/new/).
  - No es necesario usar ningún tipo de estilo CSS ni librería de componentes.
  - La conversión a mayúsculas de la descripción de la tarea deberá realizarse en un único lugar del código, independientemente de si se trata de la web MVC o de la API REST que permite implementar la web SPA.

Se pide:

- **A) Implementar la aplicación web con Spring MVC, SpringData y JPA. (2.5p)**
  - No es necesario escribir el fichero pom.xml, se puede asumir que tiene todas las dependencias correspondientes y está conectada a una base de datos externa correctamente configurada.
  - No es necesario escribir la clase Application.
  - Es necesario escribir TODOS los demás ficheros de la aplicación.
  - No es necesario incluir los imports en los ficheros Java.
  - No es necesario implementar los getter y setter. Se pueden dejar indicados con un comentario.
  - En caso de que se necesite definir una API REST, se valorará especialmente que esté diseñada de forma correcta (formato de las URLs, uso de los códigos de estado, etc.)
  - Se valorará que no haya código duplicado.
- **B) Implementar el frontend usando Angular (2p)**
  - Se puede asumir que se dispone de un proyecto Angular con todos los ficheros necesarios (index.html, angular.cli, package.json, tsconfig.json, app.module.ts, etc.).
  - Hay que escribir todos los ficheros necesarios para implementar los componentes, servicios y configuración de rutas (en caso de que sean necesarias).
  - No es necesario incluir los imports en los ficheros TypeScript.

- Se usará HTML plano en el template de los componentes (sin ninguna librería de componentes como ng-bootstrap o angular-material).
- Se asumirá que el proxy de Angular está correctamente configurado y se pueden usar URLs relativas para acceder a la API REST del backend.

- **C) Empaquetar la aplicación en un contenedor Docker (1.5p)**
  - Escribe un fichero **Dockerfile multistage** que construya la imagen daw/tareas:1.0.0 con la aplicación (tanto backend como frontend) empaquetada.
  - Se asumirá que el código Spring está alojado en una carpeta “backend” y el código Angular está alojado en una carpeta “frontend” y el Dockerfile está en la carpeta raíz.
  - Se asumirá que en el sistema donde se ejecute el script sólo tiene Docker instalado. No dispone de Node.js ni Maven.
  - Se asumirá que en DockerHub existen las imágenes maven:3.6.0, node:14.0.0 y openjdk-jre:11.0.0