Pablo Segura Velasco

AlbumService				
Método	**Test**	**Prueba**	**Resultado esperado**	**Resultado obtenido**
getAlbum<AlbumServiceImpl>	whenGetAlbum_success()	Comprobar si se muestra el album buscado por id	Listado de Albums	Listado de Albums
	whenGetAlbumNotFound_throwNotFoundException()	Comprobar si se devuelve la excepción NotFound al no encontrar el album buscado por id	NotFoundException	NotFoundException
deleteAlbum<AlbumServiceImpl>	whenDeleteAlbum_success()	Comprobar si el album buscado se borra correctamente	Album Borrado	Album Borrado
addAlbum<AlbumServiceImpl>	whenAddAlbum_success()	Comprobar si se agrega un nuevo album correctamente	Album	Album
getAllAlbums<AlbumServiceImpl>	whenGetAllAlbums_success()	Comprobar si se devuelve correctamente el listado de albums	PagedResponse<AlbumResponse>	PagedResponse<AlbumResponse>
updateAlbum<AlbumServiceImpl>	whenUpdateAlbum_success()	Comprobar si se actualizan los datos de un album ya existente	Httpstatus.OK	Httpstatus.OK
	updateAlbumUnauthorized_blogApiException()	Comprobar que no puedas actualizar el album si no es de tu propiedad	BlogApiException	BlogApiException
AlbumRepository				
Método	**Test**	**Prueba**	**Resultado esperado**	**Resultado obtenido**
findByCreatedBy<AlbumRepository>	whenfindByCreatedBy_success()	Comprobar que se encuentra por el createdby	Resultado != 0	Resultado != 0
AlbumController				
Método	**Test**	**Prueba**	**Resultado esperado**	**Resultado obtenido**
getAlbum<AlbumController>	whenGetAlbum_success()	Comprobar si se muestra el album buscado por id	Listado de Albums	Listado de Albums
deleteAlbum<AlbumController>	whenDeleteAlbum_success()	Comprobar si el album buscado se borra correctamente	Album Borrado	Album Borrado
	whenDeleteAlbumUnauthorized_blogapiException()	Comprobar si el album buscado se borra correctamente	Album Borrado	Album Borrado
addAlbum<AlbumController>	whenAddAlbum_success()	Comprobar si se agrega un nuevo album correctamente	Album	Album
getAllAlbums<AlbumController>	whenGetAlbums_success()	Comprobar si se devuelve correctamente el listado de albums	PagedResponse<AlbumResponse>	PagedResponse<AlbumResponse>
updateAlbum<AlbumController>	whenUpdateAlbum_success()	Comprobar si se actualizan los datos de un album ya existente	Httpstatus.OK	Httpstatus.OK
	updateAlbumUnauthorized_blogApiException()	Comprobar que no puedas actualizar el album si no es de tu propiedad	BlogApiException	BlogApiException
getAllPhotosByAlbums<AlbumController>	whenGetAllPhotosByAlbums_success()	Comprobar si se devuelve correctamente el listado de las fotos por los albums	Photos	Photos
UserController				
Método	**Test**	**Prueba**	**Resultado esperado**	**Resultado obtenido**
getUserAlbums<UserController>	whenGetUserAlbums_success()	Comprobar si se muestran los albums de un ususario	isOK	isOK
addUser<UserController>	whenAddUser_success()	Comprobar si se agrega un nuevo usuario correctamente	.isCreated	.isCreated
giveAdmin<UserController>	whenGiveAdmin_success()	Comprobar si se le dan permisos de admin a un usuario	isOK	isOK
UserRepository				
Método	**Test**	**Prueba**	**Resultado esperado**	**Resultado obtenido**
existByEmail<UserRepository>	whenFoundUser_existsByEmail_success()	Comprobar si el usuario existe buscandolo por email	True	True
getUserUser<UserRepository>	whenGetUser_success()	Comprobar si el usuario esta logueado como un user principal	user = userPrincipal	user = userPrincipal
	whenGetUserNotFound_ThrowExceptionNotFound	Devolver un not found si el usuario no se encuentra logueado	NotFoundException	NotFoundException

Manuel Expósito

Método a probar	Prueba	Resultado esperado	Resultado obtenido	Nombre TEST
getComment <CommentServiceImpl>	Encuentra un comentario existente en un post	El comentario de un post concreto	El comentario de un post concreto	***getComment_success***
	Excepción ResourceNotFound al no encontrar post o comentario relacionados entre sí	ResourceNotFoundException.class	ResourceNotFoundException.class	***getCommentNotFound_resourceNotFoundException***
	Excepción BlogApiNotFound al no encontrar ningún comentario similar en el post indicado	BlogApiException(Not found)	BlogApiException(Not found)	***getCommentWrongPost_blogApiException***
updateComment <CommentServiceImpl>	Actualiza un comentario PROPIO escrito previamente (USUARIO)	El comentario actualizado	El comentario actualizado	***updateComment_success***
	Actualiza un comentario escrito previamente (ADMIN)	El comentario actualizado	El comentario actualizado	
	Status UNATHORIZED si un usuario que no sea dueño del comentario intenta editarlo	BlogApiException (Unathorized)	BlogApiException(Unathorized)	***updateCommentUnathorized_blogApiException***
addComment <CommentServiceImpl>	Nuevo comentario añadido en un post determinado	El nuevo comentario	El nuevo comentario	***addComment_success()***
deleteComment <CommentServiceImpl>	Elimina el comentario PROPIO del post (USUARIO)	Comentario eliminado del post	Comentario eliminado del post	***deleteComment_success***
	Elimina el comentario de otro usuario (ADMIN)	Comentario eliminado del post	Comentario eliminado del post	
findByPostId <CommentRepository>	Encontrar una lista paginada de comentarios de un post buscado por ID	La lista paginada de comentarios	La lista paginada de comentarios	***findByPostId_success***
getComment <CommentController>	Mostrar un solo comentario	Mostrar un solo comentario devolviendo 200	Mostrar un solo comentario devolviendo 200	***getComment_success***
getAllComments <CommentController>	Recibir correctamente una respuesta paginada de comentarios	Todos los comentarios paginados	Todos los comentarios paginados	***getAllComments_success***
addComment <CommentController>	Añadir correctamente un comentario a un post cuyo id se pasa por parámetro.	Una respuesta 200 con el comentario creado.	Una respuesta 200 con el comentario creado.	***addComment_success***
updateComment <CommentController>	Actualizar un comentario PROPIO existente	Una respuesta 200 con el comentario actualizado	Una respuesta 200 con el comentario actualizado	***updateComment_success***
deleteComment <CommentController>	Borrar un comentario de un post pasado por parámetro	Una respuesta 200 con el comentario borrado	Una respuesta 200 con el comentario borrado	***deleteComment_returns200_success***
	Pasarle un comentario que no pueda encontrar	Una respuesta 400 Bad Request	Una respuesta 400 Bad Request	***deleteComment_returns400_badRequest***
findByUsernameOrEmail <UserRepository>	Encontrar a un usuario por su nombre (si solo se le pasa eso), email (si solo recibe el email), o ambos, si ambos son pasados por parámetro	El usuario cuyo username o email es uno de los pasados por parámetro	El usuario cuyo username o email es uno de los pasados por parámetro	***findByUsernameOrEmail_success***
getUserByName <UserRepository>	Debe devolver un usuario cuyo nombre sea igual al pasado por parámetro	El usuario cuyo nombre sea el pasado por parámetro	El usuario cuyo nombre sea el pasado por parámetro	***getUserByName_success***
	Lanzar una excepción ResourceNotFoundException si no encuentra el nombre del usuario pasado por parámetro	ResourceNotFoundException.class	ResourceNotFoundException.class	***getUserByName_ThrowExceptionNotFound***
takeAdmin <UserController>	Quitar los privilegios de admin a un ADMIN	Respuesta 200 y privilegios de admin eliminados del usuario	Respuesta 200 y privilegios de admin eliminados del usuario	***takeAdmin_success***
	Intentar quitar los privilegios de un admin con privilegios de USER	Respuesta 401 FORBIDDEN	Respuesta 401 FORBIDDEN	***takeAdmin_forbidden***
setAddress <UserController>	Cambiar la dirección de un usuario	Respuesta 200 con la direccion actualizada	Respuesta 200 con la direccion actualizada	***setAddress_success***

Daniel	Clase	Método	Prueba	Resultado esperado	Resultado obtenido
	categoryService	getCategory()	-Comprobar si devuelve la categoria correcta	Devuelve la categoria	Devuelve la categoria
	categoryService	getCategory()	-Comprobar si devuelve la excepción.	Devuelve la excepción	Devuelve la excepción
	categoryService	getAllCategories()	Comprueba que trae todas las categorias paginadas	Devuelve todas las categorias paginadas	Devuelve todas las categorias paginadas
	categoryService	addCategory()	-Comprobar que guarda la categoria	Guarda la categoria y la devuelve	Guarda la categoria y la devuelve
	categoryService	updateCategory()	-Comprobar que encuentra la categoria por id	Encuentra la categoria	Encuentra la categoria
	categoryService	updateCategory()	Comprobar la excepción	Salta la excepción	Salta la excepción
	categoryService	updateCategory()	Comprobar que funciona correctamente con un usuario creador de la categoria	Devuelve el tamaño del array de post (en este caso 1)	Devuelve el tamaño del array de post (en este caso 1)
	categoryService	updateCategory()	Comprobar que funciona correctamente con un usuario administrador	Devuelve el tamaño del array de post (en este caso 1)	Devuelve el tamaño del array de post (en este caso 1)
	categoryService	updateCategory()	Comprueba que salta la excepción por usuario no autorizado	Devuelve a excepción	Devuelve a excepción
	categoryService	deleteCategory()	Comprueba que se borra la categoria con un usuario dueño de la categoria	Borra la categoria	Borra la categoria
	categoryService	deleteCategory()	Comprueba que se borra la categoria con un administrador	Borra la categoria	Borra la categoria
	categoryService	deleteCategory()	Comprueba que salta la excepción por usuario no autorizado	Borra la categoria	Borra la categoria
	photoRepository	findByAlbumId()	Comprueba que trae todas las fotos de un álbum	Devuelve una lista de fotos paginada	Devuelve una lista de fotos
	categoryController	getAllCategories()	Comprueba que se realiza la petición get y trae todas las categorias	Devuelve una lista con todas las categorias	Devuelve una lista con todas las categorias
	categoryController	addCategory()	Comprueba que se realiza la petición post para agregar una nueva categoria con un rol de usuario	Guarda la categoria	Guarda la categoria
	categoryController	addCategory()	Comprueba que salta la excepción por usuario no autorizado	Salta la excepción	Salta la excepción
	categoryController	putCategory()	Comprueba que se modifica una categoria con un rol de usuario	Modifica la categoria	Modifica la categoria
	categoryController	putCategory()	Comprueba que salta la excepción por usuario no autorizado	Salta la excepción	Salta la excepción
	categoryController	deleteCategory()	Comprueba que se elimina la categoria	Borra la categoria	Borra la categoria
	categoryController	deleteCategory()	Comprueba que salta la excepción por usuario no autorizado	Salta la excepción	Salta la excepción
	userController	getCurrentUser()	Comprueba que trae un usuario con rol de usuario	Devuelve un usuario	Devuelve un usuario
	userController	getCurrentUser()	Comprueba que salta la excepción por usuario no autorizado	Devuelve una respuesta con el estado Forbidden	Devuelve una respuesta con el estado Forbidden
	userController	checkUsernameAvailability()	Comprueba si el nombre de usuario ya está en uso	Devuelve true	Devuelve true