Plan de pruebas proyecto final

Tipo de prueba	Descripción	Pre-requisitos	Pasos a ejecutar	Resultado esperado	Resultado obtenido
Unitaria	Obtener colección de categorías	CategoryRepository debe poder ser mockeado y deben existir datos.	1. Instanciar Mock de Repositorio y Modelo. 2. Simular respuesta de makeModel->latest->get. 3. Ejecutar CategoryService->getRecords().	Retorna una instancia de EloquentCollection con la cantidad correcta de registros simulados.	Exitoso
Unitaria	Crear una categoría exitosamente	CategoryDTO válido.	1. Crear CategoryDTO con datos de prueba. 2. Mockear create en el repositorio. 3. Ejecutar CategoryService->create($dto).	El método retorna una instancia de Category con el ID y nombre asignados.	Exitoso
Unitaria	Buscar categoría por ID	Debe existir el registro con ese id	1. Mockear findOrFail en el modelo. 2. Ejecutar CategoryService->find(10).	Retorna el modelo Category con ID 10 y los datos correctos.	Exitoso
Unitaria	Actualizar categoría	DB Transaction simulada. Debe existir el registro con ese id	1. Mockear DB Transaction. 2. Mockear findOrFail, update y fresh en el modelo. 3. Ejecutar CategoryService->update(10, $dto).	El modelo ejecuta la actualización, refresca los datos y retorna la instancia actualizada.	Exitoso

Tipo de prueba	Descripción	Pre-requisitos	Pasos a ejecutar	Resultado esperado	Resultado obtenido
Unitaria	Eliminar categoría	DB Transaction simulada. Debe existir el registro con ese id	1. Mockear findOrFail y delete en el modelo. 2. Ejecutar CategoryService->delete(5).	El método retorna true indicando que la eliminación fue exitosa.	Exitoso
Unitaria	Obtener colección de productos con relaciones	ProductRepository mockeable y deben existir datos.	1. Mockear Builder de Eloquent (latest, with('category'), get). 2. Ejecutar ProductService->getRecords().	Retorna una ResourceCollection con productos cargando la relación de categoría.	Exitoso
Unitaria	Crear un producto exitosamente	ProductDTO válido con category_id. Debe existir la categoría a relacionar	1. Configurar DTO con datos completos (precio, stock). 2. Mockear create en repositorio. 3. Ejecutar ProductService->create($dto).	Retorna instancia de Product con los datos persistidos (mockeados).	Exitoso
Unitaria	Buscar producto por ID	Debe existir el registro con ese id	1. Mockear findOrFail en modelo. 2. Ejecutar ProductService->find(10).	Retorna objeto Product correcto (ej: "Axe Spray").	Exitoso
Unitaria	Actualizar producto	DB Transaction simulada. Debe existir el registro con ese id.	1. Mockear transacción y métodos de actualización (update, fresh). 2. Ejecutar ProductService->update(5, $dto).	El modelo de producto se actualiza y se retorna la nueva instancia.	Exitoso
Unitaria	Eliminar producto	Debe existir el registro con ese id.	1. Mockear búsqueda y método delete. 2. Ejecutar ProductService->delete(7).	Retorna true tras eliminar el recurso.	Exitoso

Tipo de prueba	Descripción	Pre-requisitos	Pasos a ejecutar	Resultado esperado	Resultado obtenido
Integración	Crear categoría vía API (POST)	Base de datos de prueba. Cuerpo de la petición acorde al DTO.	1. Enviar petición POST a /api/categories con payload JSON. 2. Verificar código HTTP. 3. Verificar base de datos.	Código 200. JSON contiene el nombre creado. El registro existe en la tabla categories.	Exitoso
Integración	Listar categorías vía API (GET)	Datos pre-cargados (Factories).	1. Crear 2 categorías con Factory. 2. Enviar petición GET a /api/categories.	Código 200. El JSON de respuesta contiene exactamente 2 elementos en data.	Exitoso
Integración	Ver detalle de categoría (GET)	Categoría existente.	1. Enviar petición GET a /api/categories/{id}.	Código 200. El JSON contiene el ID y datos de la categoría solicitada.	Exitoso
Integración	Actualizar categoría vía API (PUT)	Categoría existente.	1. Enviar petición PUT a /api/categories/{id} con nuevo nombre. 2. Verificar DB.	Código 200. Respuesta con nombre actualizado. DB refleja el cambio.	Exitoso
Integración	Eliminar categoría vía API (DELETE)	Categoría existente.	1. Enviar petición DELETE a /api/categories/{id}. 2. Verificar DB.	Código 200. El registro ya no existe en la tabla categories.	Exitoso
Integración	Crear producto vía API (POST)	Debe existir una categoría (Foreign Key).	1. Crear Categoría. 2. Enviar POST a /api/products con category_id, precio, stock, etc.	Código 200. JSON confirma creación. Registro presente en tabla products.	Exitoso
Integración	Listar productos vía API (GET)	Datos pre-cargados.	1. Crear productos con Factory.	Código 200. JSON retorna la cantidad correcta de productos.	Exitoso

Tipo de prueba	Descripción	Pre-requisitos	Pasos a ejecutar	Resultado esperado	Resultado obtenido
			2. Enviar GET a /api/products.		
Integración	Ver detalle de producto (GET)	Producto existente.	1. Enviar GET a /api/products/{id}.	Código 200. Datos del producto coinciden con los creados.	Exitoso
Integración	Actualizar producto vía API (PUT)	Producto existente.	1. Enviar PUT a /api/products/{id} manteniendo precio/stock pero cambiando nombre. 2. Verificar DB.	Código 200. JSON actualizado. DB refleja cambios.	Exitoso
Integración	Eliminar producto vía API (DELETE)	Producto existente.	1. Enviar DELETE a /api/products/{id}.	Código 200. El registro desaparece de la tabla products.	Exitoso
Sistema	Flujo completo CRUD: Categorías y Productos	Servidor web levantado. Base de datos accesible. Navegador (Chromium/Webkit).	1. Gestión Categoría: - Ir a /view/categories. - Crear nueva categoría "Bebidas". - Verificar que aparece en la tabla. 2. Gestión Producto: - Ir a /view/products. - Crear producto "Coca Cola" asignado a "Bebidas". - Llenar precio (5500), stock (25) y guardar. - Verificar que aparece en tabla con precio correcto. 3. Eliminación: - Localizar fila de "Coca Cola" y dar clic en Eliminar. - Aceptar alerta nativa del navegador. - Verificar que desaparece de la tabla. - Ir a /view/categories. - Localizar fila de "Bebidas" y dar clic en Eliminar.	1. La categoría "Bebidas" es visible en la tabla. 2. El producto "Coca Cola" se muestra con precio $5500. 3. Tras eliminar, el producto ya no es visible en la tabla. 4. Tras eliminar, la categoría ya no es visible en la tabla.	Exitoso

Tipo de prueba	Descripción	Pre-requisitos	Pasos a ejecutar	Resultado esperado	Resultado obtenido
			- Aceptar alerta nativa del navegador. - Verificar que desaparece de la tabla.		