

Testing (Utest)- Entrega 06

grupo 2 - #abrancupos

Integrantes:

- *Salazar Carvajal Omar Alejandro*
- *Pajaro Sanchez Nicolas*
- *Diaz Acosta Nathalia Milena*
- *Camargo Barrera Samuel Francisco*

En el presente trabajo se realizaron test de integración y la razón de esto es que el enfoque principal fue testing de integración usando TestCase, que crea una base de datos de prueba aislada para cada suite, es así que, si bien debería haber alguna función que probar a partir de test unitarios, se ve necesario usar test de integración por estas conexiones con la base de datos (manejo de CRUD principalmente) migrada de Django, mockeando esta base para realizar los tests, procurando darle unos parámetros iniciales (algún usuario paciente que se requiera para ejecutar la búsqueda por ejemplo).

- **Test 01: Crear Usuario de Paciente**

Valida la creación de un paciente verificando que la información suministrada por los campos obligatorios cumpla con los requisitos de formato previamente establecidos.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ *Nombre del paciente diligenciado correctamente*
 - ✓ *Apellidos del paciente diligenciados correctamente*
 - ✓ *id único creado correctamente*
- **Casos Límites y Ejecutables:**
 - *Caso Normal: paciente creado con datos correctos.*
 - *Caso Límite Válido: paciente con nombres cortos y contraseña mínima permitida.*
 - *Caso Límite Inválido: contraseña demasiado corta.*

- **Test 02: Crear Usuario de Médico**

Valida la creación de un médico verificando que la información suministrada por los campos obligatorios cumpla con los requisitos de formato previamente establecidos y la validez en el campo de especialidad.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ *Nombres y apellidos almacenados correctamente*
 - ✓ *Especialidad almacenada correctamente*
 - ✓ *Médico registrado correctamente en la base de datos*
- **Casos Límites y Ejecutables:**
 - *Caso Normal: médico creado con todos los campos válidos.*
 - *Caso Límite Válido: especialidad con mínimo de caracteres permitidos.*
 - *Caso Límite Inválido: especialidad demasiado corta.*

- **Test 03: Crear Usuario de Administrador**

Valida la creación de un superusuario(Administrador) asociado a Django verificando que la información suministrada por los campos obligatorios cumpla con los requisitos de formato previamente establecidos.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ Asociación correcta entre el usuario Administrador y el usuario de Django
 - ✓ Nombre de usuario y contraseña correctamente almacenados
- **Casos Límites y Ejecutables:**
 - Caso Normal: superusuario creado correctamente.
 - Caso Límite Válido: contraseña de longitud máxima permitida.
 - Caso Límite Inválido: contraseña demasiado larga.

- **Test 04: Agendar/Crear Citas Médicas**

Verifica que una cita médica pueda ser registrada correctamente, teniendo en cuenta todas las condiciones necesarias.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ Fecha, hora y especialidad correctamente registrados
 - ✓ Cita correctamente asociada al paciente
 - ✓ Cita correctamente asociada al médico
- **Casos Límites y Ejecutables:**
 - Caso Normal: cita con hora y fecha estándar.
 - Caso Límite Válido: hora límite (23:59).
 - Caso Límite Inválido: formato de hora incorrecto.

- **Test 05: Cancelar Cita Médica**

Comprueba si el estado de una cita puede cambiarse a “cancelada”.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ Estado de la cita modificado correctamente a “cancelada”
 - ✓ Modificación reflejada correctamente en la base de datos
- **Casos Límites y Ejecutables:**
 - Caso Normal: cita cancelada correctamente.
 - Caso Límite Válido: volver a establecer estado como “Cancelada”.
 - Caso Límite Inválido: estado con valor no permitido

- **Test 06: Aplazar Cita Médica**

Valida que una cita pueda ser reprogramada a una nueva fecha y hora válida.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ Fecha modificada correctamente
 - ✓ Hora modificada correctamente
- **Casos Límites y Ejecutables:**
 - Caso Normal: reprogramación con nueva fecha y hora válidas.
 - Caso Límite Válido: hora límite (23:59).
 - Caso Límite Inválido: hora en formato incorrecto.

- **Test 07: Confirmar Cita Médica**

Verifica que el estado de la cita pueda ser modificado a “confirmada” correctamente.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ Estado de la cita modificado correctamente a “confirmada”
- **Casos Límites y Ejecutables:**
 - Caso Normal: confirmación exitosa.
 - Caso Límite Válido: confirmación repetida.
 - Caso Límite Inválido: estado con valor no permitido

- **Test 08: Crear receta**

Se comprueba que cuando se realicen cambios en el historial médico del paciente, el sistema actualice automáticamente e inmediatamente los cambios.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ Historial creado correctamente en caso de no existir ya
 - ✓ Incremento de los registros del historial médico en caso de la creación de una receta nueva
- **Casos Límites y Ejecutables:**
 - Caso Normal: receta creada y reflejada en historial.
 - Caso Límite Válido: receta con campos mínimos válidos.
 - Caso Límite Inválido: receta con diagnóstico demasiado corto.

- **Test 09: Actualización del Historial Médico al Agregar Receta**

Se comprueba que cuando se formulen nuevas recetas al paciente el historial médico de este se actualice por medio del sistema automáticamente e inmediatamente .

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ Historial actualizado correctamente
 - ✓ Número total de recetas en el historial reflejado correctamente
- **Casos Límites y Ejecutables:**
 - Caso Normal: agregar receta y reflejarse en historial.
 - Caso Límite Válido: agregar varias recetas simultáneamente.
 - Caso Límite Inválido: receta con campos no válidos.

- **Test 10: Eliminación de Receta Médica y actualización del Historial**

Valida que, al eliminar una receta médica, el sistema actualice el historial removiendo la receta correspondiente.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ Receta eliminada correctamente
 - ✓ Historial actualizado suprimiendo la receta eliminada
- **Casos Límites y Ejecutables:**
 - Caso Normal: eliminar la receta existente.
 - Caso Límite Válido: no aplica.
 - Caso Límite Inválido: intentar eliminar receta inexistente

- **Test 11: Eliminar Historial Médico**

Comprueba que un historial médico pueda eliminarse correctamente sin afectar el resto de información del paciente.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ *Historial eliminado de forma independiente*
 - ✓ *Paciente permanece registrado*
 - ✓ *No se afectan otras estructuras del sistema*
- **Casos Límites y Ejecutables:**
 - *Caso Normal: eliminar historial existente.*
 - *Caso Límite Válido: verificar que el paciente siga registrado.*
 - *Caso Límite Inválido: intentar eliminar historial inexistente*

- **Test 12: Eliminar Paciente Junto con su Información**

Se garantiza que al eliminar un paciente, su historial médico y toda su información asociada también sea eliminado en cascada.

- **Herramienta Utilizada:** Django.test.TestCase
- **Validación:**
 - ✓ *Paciente eliminado correctamente*
 - ✓ *Historial médico eliminado automáticamente*
 - ✓ *Recetas asociadas eliminadas en cascada*
- **Casos Límites y Ejecutables:**
 - *Caso Normal: paciente con historial eliminado.*
 - *Caso Límite Válido: paciente sin historial.*
 - *Caso Límite Inválido: eliminar paciente inexistente*