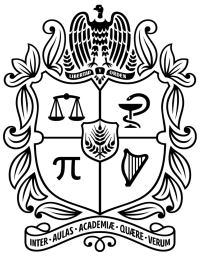
Informe Testing
Ingeniería de software 1



UNIVERSIDAD
NACIONAL
DE COLOMBIA

Autores:
John Jairo Paez Albino
Nicolas Arturo Jaramillo Garrido
Juan Diego Velásquez Pinzón
Juan David Suarez Coronado

Universidad Nacional de Colombia - sede Bogotá
Facultad de Ingeniería
Departamento de Sistemas e Industrial
Ingeniería de Software 1 (2016701)
27 de febrero 2025

✨pruebas John Jairo Paez Albino

Se utilizó TestCase de Django para ejecutar pruebas automatizadas en una base de datos temporal sin afectar los datos reales. Estas pruebas verifican la correcta creación de instancias, la integridad de los datos y las relaciones entre ellos, asegurando el buen funcionamiento de los modelos de la aplicación.

las pruebas realizadas son:

Tipo de Prueba	**Componente Probado**	**Descripción de la Prueba**	**Herramienta/ Framework Usado**
Prueba unitaria	**Modelo Doctor**	Verifica que se pueda crear un doctor con atributos correctos y que se almacenen correctamente en la base de datos.	Django TestCase
Prueba unitaria	**Modelo Department**	Comprueba que se pueda crear un departamento con un nombre y descripción, asegurando que los valores sean los esperados.	Django TestCase
Prueba unitaria	**Modelo DoctorAvailability**	Testea que un doctor pueda tener disponibilidad asignada con fechas y horas correctas, asegurando que los datos sean válidos.	Django TestCase
Prueba unitaria	**Modelo MedicalNote**	Valida que se pueda registrar una nota médica asociada a un doctor, con el contenido y la fecha adecuados.	Django TestCase

📌**Código test:**

```python
from django.test import TestCase
from django.utils import timezone
from .models import Doctor, Department, DoctorAvailability, MedicalNote

class DoctorModelTest(TestCase):
    def setUp(self):
        self.doctor = Doctor.objects.create(
            first_name="John",
            last_name="Doe",
            qualification="MD",
            contact_number="1234567890",
            email="johndoe@example.com",
            address="123 Main Street",
            biography="Experienced general practitioner.",
            is_on_vacation=False
        )

    def test_doctor_creation(self):
        self.assertEqual(self.doctor.first_name, "John")
        self.assertEqual(self.doctor.last_name, "Doe")
        self.assertEqual(self.doctor.email, "johndoe@example.com")
        self.assertFalse(self.doctor.is_on_vacation)
```

```python
class DepartmentModelTest(TestCase):
    def test_department_creation(self):
        department = Department.objects.create(
            name="Cardiology",
            description="Heart-related treatments"
        )
        self.assertEqual(department.name, "Cardiology")
        self.assertEqual(department.description, "Heart-related treatments")
```

```python
class DoctorAvailabilityModelTest(TestCase):
    def setUp(self):
        self.doctor = Doctor.objects.create(
            first_name="Alice",
            last_name="Smith",
            qualification="MBBS",
            contact_number="0987654321",
            email="alicesmith@example.com",
            address="456 Side Street",
            biography="Specialist in family medicine.",
            is_on_vacation=True
        )

    def test_doctor_availability_creation(self):
        availability = DoctorAvailability.objects.create(
            doctor=self.doctor,
            start_date=timezone.now().date(),
            end_date=timezone.now().date(),
            start_time=timezone.now().time(),
            end_time=timezone.now().time()
        )
        self.assertEqual(availability.doctor, self.doctor)
        self.assertTrue(isinstance(availability, DoctorAvailability))
```

```python
class MedicalNoteModelTest(TestCase):
    def setUp(self):
        self.doctor = Doctor.objects.create(
            first_name="Bob",
            last_name="Marley",
            qualification="MD",
            contact_number="5555555555",
            email="bobmarley@example.com",
            address="789 Beach Avenue",
            biography="Expert in alternative medicine.",
            is_on_vacation=False
        )

    def test_medical_note_creation(self):
        note = MedicalNote.objects.create(
            doctor=self.doctor,
            note="Patient requires further evaluation.",
            date=timezone.now().date()
        )
        self.assertEqual(note.doctor, self.doctor)
        self.assertEqual(note.note, "Patient requires further evaluation.")
```

📌 **resultados**

Salida de la ejecución de pruebas:

```
(.venv) PS C:\Users\estap\OneDrive\Escritorio\clon\IngenieriaSoftware1\Proyecto\proyecto_v1> python manage.py test doctors
Found 4 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
....
----------------------------------------------------------------------
Ran 4 tests in 0.006s

OK
Destroying test database for alias 'default'...
```

📌 **Interpretación:**
Los tests pasaron correctamente, lo que indica que los modelos están funcionando como se espera.

✨ Pruebas Juan Diego Velásquez Pinzón: Patiens

Tipo de Prueba	**Componente Probado**	**Herramienta/ Framework Usado**
Prueba unitaria	Modelo Patient - Verifica la creación y almacenamiento de pacientes con datos válidos en la base funcione correctamente.	Django TestCase
Prueba unitaria	Modelo Insurance- Comprueba la creación y asociación de seguros médicos a pacientes.	Django TestCase
Prueba unitaria	Modelo Medical_Record - Valida el funcionamiento del registro de diagnósticos, tratamientos y fechas de seguimiento de los pacientes.	Django TestCase

📌Código test:

C: > Users > Juanchiis > Documents > UTrabajos > 6° Semestre > Ingesoft > IngenieriaSoftware1-main > Proyecto > proyecto_v1 >

```python
from django.test import TestCase
from datetime import date
from .models import Patient, Insurance, MedicalRecord

class PatientModelsTest(TestCase):

    def setUp(self):
        # Datos iniciales

        self.patient = Patient.objects.create(
            first_name="Juan",
            last_name="Velásquez",
            date_of_birth=date(1990, 5, 20),
            contact_number="3204567890",
            email="juan.velasquez@example.com",
            address="Calle 45 #10-23, Bogotá",
            medical_history="Paciente con antecedentes de hipertensión."
        )

        self.insurance = Insurance.objects.create(
            patient=self.patient,
            provider="Seguro Salud Plus",
            policy_number="ABC123456",
            expiration_date=date(2026, 12, 31)
        )

        self.medical_record = MedicalRecord.objects.create(
            patient=self.patient,
            date=date(2025, 2, 25),
            diagnosis="Gripe común",
            treatment="Reposo e hidratación",
            follow_up_date=date(2025, 3, 5)
        )

```

```python

    def test_patient(self):
        # Verifica que el paciente se creó correctamente
        patient = Patient.objects.get(email="juan.velasquez@example.com")
        self.assertEqual(patient.first_name, "Juan")
        self.assertEqual(patient.last_name, "Velásquez")
        self.assertEqual(patient.medical_history, "Paciente con antecedentes de hipertensión.")

    def test_insurance(self):
        # Verifica que el seguro se registró correctamente
        insurance = Insurance.objects.get(patient=self.patient)
        self.assertEqual(insurance.provider, "Seguro Salud Plus")
        self.assertEqual(insurance.policy_number, "ABC123456")
        self.assertEqual(insurance.expiration_date, date(2026, 12, 31))

    def test_medical_record(self):
        # Verifica que el historial médico se guarda correctamente
        record = MedicalRecord.objects.get(patient=self.patient)
        self.assertEqual(record.diagnosis, "Gripe común")
        self.assertEqual(record.treatment, "Reposo e hidratación")
        self.assertEqual(record.follow_up_date, date(2025, 3, 5))

```

```python
from django.test import TestCase
from datetime import date
from .models import Patient, Insurance, MedicalRecord

class PatientModelsTest(TestCase):

    def setUp(self): ···

    def test_patient(self): ···

    def test_insurance(self): ···

    def test_medical_record(self): ···

```

📌 **resultados:**

```
(.venv) PS C:\Users\Juanchiis\Documents\UTrabajos\6° Semestre\Ingesoft\IngenieriaSoftware1-main\Proyecto\proyecto_v1> python manage.py test
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
----------------------------------------------------------------------
Ran 3 tests in 0.005s

OK
Destroying test database for alias 'default'...
```

📌 **Interpretación:**

No hubo errores al ejecutar los tests, esto nos indica que las funcionalidades probadas funcionan de manera adecuada trabajando cooperativamente y sin desaciertos con la base de datos.

✨Pruebas Juan David Suárez Coronado:

Tipo de Prueba	**Componente Probado**	**Herramienta/ Framework Usado**
Prueba unitaria	Modelo Doctor - Verifica la creación y almacenamiento de odontólogos.	Django TestCase
Prueba unitaria	Modelo Department - Comprueba la creación de departamentos con datos válidos.	Django TestCase

Prueba unitaria	Modelo DoctorAvailability - Valida la correcta asignación de disponibilidad horaria de un odontólogo.	Django TestCase
Prueba unitaria	Modelo MedicalNote - Verifica el registro de notas médicas con información detallada.	Django TestCase

📌**Código test:**

doctors > tests.py > ...

```python
from datetime import date, time
from django.test import TestCase
from .models import Doctor, Department, DoctorAvailability, MedicalNote

class OdontologyModelsTest(TestCase):

    def setUp(self):
        #Configura los datos iniciales para las pruebas
        self.department = Department.objects.create(
            name="Odontología General",
            description="Departamento encargado de la salud bucal general."
        )

        self.doctor = Doctor.objects.create(
            first_name="Camila",
            last_name="Martínez",
            qualification="Especialista en Endodoncia",
            contact_number="3204567890",
            email="camila.martinez@odontoclinic.com",
            address="Calle 45 #10-23, Bogotá",
            biography="Doctora con más de 10 años de experiencia en tratamientos de conducto.",
            is_on_vacation=False
        )

        self.availability = DoctorAvailability.objects.create(
            doctor=self.doctor,
            start_date=date(2025, 3, 1),
            end_date=date(2025, 3, 31),
            start_time=time(8, 0),
            end_time=time(17, 0)
        )

        self.medical_note = MedicalNote.objects.create(
            doctor=self.doctor,
            note="Paciente presenta caries avanzada en molar superior derecho, se recomienda endodoncia.",
            date=date(2025, 2, 25)
        )
```

```
38
39      def test_doctor_creation(self):
40          #Verifica que el doctor se creó correctamente
41          doctor = Doctor.objects.get(email="camila.martinez@odontoclinic.com")
42          self.assertEqual(doctor.first_name, "Camila")
43          self.assertEqual(doctor.qualification, "Especialista en Endodoncia")
44
45      def test_department_creation(self):
46          #Verifica que el departamento de odontología fue registrado
47          department = Department.objects.get(name="Odontología General")
48          self.assertEqual(department.description, "Departamento encargado de la salud bucal general.")
49
50      def test_doctor_availability(self):
51          #Verifica la disponibilidad del doctor
52          availability = DoctorAvailability.objects.get(doctor=self.doctor)
53          self.assertEqual(availability.start_date, date(2025, 3, 1))
54          self.assertEqual(availability.end_time, time(17, 0))
55
56      def test_medical_note(self):
57          #Verifica que la nota médica se guarda correctamente
58          note = MedicalNote.objects.get(doctor=self.doctor)
59          self.assertIn("caries avanzada", note.note)
60          self.assertEqual(note.date, date(2025, 2, 25))
61
```

```
1   from django.test import TestCase
2   from datetime import date, time
3   from .models import Doctor, Department, DoctorAvailability, MedicalNote
4
5   class OdontologyModelsTest(TestCase):
6
7       def setUp(self):···
38
39      def test_doctor_creation(self):···
44
45      def test_department_creation(self):···
49
50      def test_doctor_availability(self):···
55
56      def test_medical_note(self):···
61
```

📌 Resultados:

```
(.venv) PS C:\Users\juand\Desktop\git hub\IngenieriaSoftware1\Proyecto\proyecto_v1> python manage.py test
Found 4 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
....
----------------------------------------------------------------------
Ran 4 tests in 0.009s

OK
Destroying test database for alias 'default'...
```

📌 Interpretación:

Los resultados de las pruebas muestran que todo funciona como se esperaba sin errores ni fallos, lo que significa que los modelos de la aplicación odontológica están bien diseñados y que la información se está guardando y relacionando correctamente en la base de datos.

La aplicación es capaz de manejar la creación de odontólogos, departamentos, horarios de disponibilidad y notas médicas sin problemas, ya que los datos ingresados se almacenan de forma precisa y las relaciones entre los distintos elementos funcionan bien.

✨ Pruebas Nicolás Arturo Jaramillo Garrido: Bookings

Tipo de Prueba	Componente Probado	Herramienta/Framework Usado
Prueba unitaria	Modelo Appointment - Verifica la creación y almacenamiento de citas.	Django TestCase
Prueba unitaria	Modelo MedicalNote - Verifica el registro de notas médicas con información detallada.	Django TestCase

📌**Código test:**

```python
from django.test import TestCase
from datetime import date, time
from doctors.models import Doctor
from patients.models import Patient
from .models import Appointment, MedicalNote


class AppointmentModelsTest(TestCase):

    def setUp(self):
        # Configura los datos iniciales para las pruebas
        self.patient = Patient.objects.create(
            first_name="Juan",
            last_name="Pérez",
            date_of_birth=date(1990, 5, 15),
            contact_number="3101234567",
            email="juan.perez@example.com",
            address="Carrera 12 #34-56, Bogotá"
        )

        self.doctor = Doctor.objects.create(
            first_name="Laura",
            last_name="Gómez",
            qualification="Médico General",
            contact_number="3209876543",
            email="laura.gomez@clinic.com",
            address="Avenida Siempre Viva 742",
            biography="Médico general con amplia experiencia en consultas de atención primaria.",
            is_on_vacation=False
        )

        self.appointment = Appointment.objects.create(
            patient=self.patient,
            doctor=self.doctor,
            appointment_date=date(2025, 4, 10),
            appointment_time=time(10, 30),
            notes="Consulta de seguimiento para control de presión arterial.",
            status="Pendiente"
        )

        self.medical_note = MedicalNote.objects.create(
            appointment=self.appointment,
            note="Paciente presenta presión arterial dentro de valores normales. Se recomienda continuar con la medicación.",
            date=date(2025, 4, 10)
        )

    def test_appointment_creation(self):
        # Verifica que la cita se creó correctamente
        appointment = Appointment.objects.get(patient=self.patient)
        self.assertEqual(appointment.doctor, self.doctor)
        self.assertEqual(appointment.status, "Pendiente")

    def test_medical_note_creation(self):
        # Verifica que la nota médica se guarda correctamente
        note = MedicalNote.objects.get(appointment=self.appointment)
        self.assertIn("presión arterial dentro de valores normales", note.note)
        self.assertEqual(note.date, date(2025, 4, 10))

```

📌 **resultados:**

```
(.venv) PS C:\Users\artur\Downloads\IngenieriaSoftware1-main (1)\IngenieriaSoftware1-main\Proyecto\proyecto_v1> python manage.py test
Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
----------------------------------------------------------------------
Ran 2 tests in 0.003s

OK
Destroying test database for alias 'default'...
(.venv) PS C:\Users\artur\Downloads\IngenieriaSoftware1-main (1)\IngenieriaSoftware1-main\Proyecto\proyecto_v1>
```

📌 **Interpretación:**
Los resultados de las pruebas resultaron satisfactorios, evidenciando que funciona correctamente, sin errores ni fallos. Esto confirma que la información sobre citas médicas se almacena y relaciona adecuadamente en la base de datos.

✨ Lecciones aprendidas y dificultades:

Testear nuestra aplicación odontológica fue una experiencia que nos permitió entender mejor la importancia de garantizar que el sistema funcione correctamente antes de su implementación, ya que al principio pensamos que las pruebas serían un simple paso técnico, pero nos dimos cuenta de que detectar y prevenir errores puede ser más complejo de lo que imaginábamos.

Durante el proceso nos enfrentamos a pequeños errores en la configuración y en la lógica de algunos modelos, lo que nos hizo darnos cuenta de que escribir código no es suficiente porque debemos asegurarnos de que realmente haga lo que se espera en diferentes escenarios, lo que nos ayudó a reforzar la idea de que las pruebas no solo buscan verificar que algo funcione sino también anticiparse a posibles fallos antes de que lleguen a los usuarios.