Informe 3.1: Caso Práctico Frameworks para Pruebas Automáticas

Asignatura: Programación Avanzada

NOMBRE:

Erika Achig

Wagner Cuenca

Kevin Quimuña

Departamento: DCCO

Universidad: Universidad Fuerzas Armadas - ESPE

Fecha de entrega: 21 de febrero del 2025

SANGOLQUI – ECUADOR

Índice

Contenido

1. Objetivos

   3.1 General

   - Desarrollar una aplicación que tenga back end y front end que gestione una clínica para el desarrollo de pruebas automáticas.

   3.2 Específicos

   - Desarrollar pruebas automáticas como lo son las pruebas de estrés para ver el rendimiento de la aplicación.
   - Aplicar pruebas de integración para verificar cómo interactúan los módulos de la aplicación.
   - Realizar pruebas para la creación, modificación y eliminación de registros.

2. Resumen

   Este proyecto se basó en pruebas en un sistema de citas médicas, que lo principal es adquirir conocimientos prácticos sobre pruebas de estrés, recuperación, y buenas prácticas.

   Aplicamos Apache JMeter para evaluar el rendimiento bajo carga, mediante la simulación de 500 solicitudes concurrentes. Además, se realizan pruebas de recuperación mediante copias de seguridad y restauración de la base de datos, y pruebas de integración que aseguran para los pacientes en este caso.

   Con los resultados obtenidos la integridad de los datos nos muestra la importancia de realizar pruebas para la atención médica y aplicarla en buen funcionamiento.

3. Introducción

- Se debe desarrollar un programa con una API que gestione una clínica y su base de datos. Las tablas existentes incluyen citas, consultorios, pacientes y médicos; todas estas forman parte de la lógica de negocio y la gestión de la clínica.

  Se implementarán diversas pruebas a la API, entre las cuales se encuentran las pruebas de estrés, que simularán 500 solicitudes concurrentes durante un lapso de 5 minutos, y las pruebas de integración, donde se diseñarán pruebas para verificar la relación entre las tablas.

  Se utilizará el lenguaje SQL para verificar el funcionamiento de la base de datos. Para la prueba de estrés, se emplea la herramienta JMeter, mientras que para las pruebas de integración se utilizarán consultas SQL para las tablas.

4. Implementación

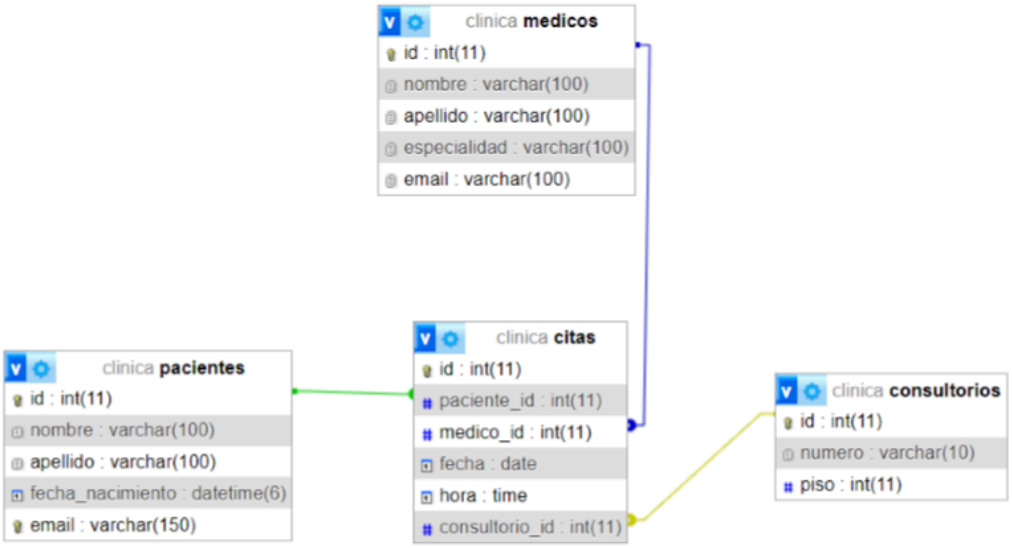
   4.1 Conceptos base

- Spring Boot: es un framework de Java que simplifica la creación de aplicaciones backend y microservicios.

- Apache JMeter es una herramienta de código abierto utilizada principalmente para realizar pruebas de rendimiento y pruebas de carga en aplicaciones web y otros servicios.

- SQL (Structured Query Language) es un lenguaje de programación utilizado para gestionar y manipular bases de datos relacionales permite realizar operaciones como consultar, insertar, actualizar y eliminar datos, así como administrar la estructura de la base de datos.

   4.2 Diagramas UML


clinica medicos
id : int(11)
nombre : varchar(100)
apellido : varchar(100)
especialidad : varchar(100)
email : varchar(100)
clinica pacientes
id : int(11)
nombre : varchar(100)
apellido : varchar(100)
fecha_nacimiento : datetime(6)
email : varchar(150)
clinica citas
id : int(11)
paciente_id : int(11)
medico_id : int(11)
fecha : date
hora : time
consultorio_id : int(11)
clinica consultorios
id : int(11)
numero : varchar(10)
piso : int(11)

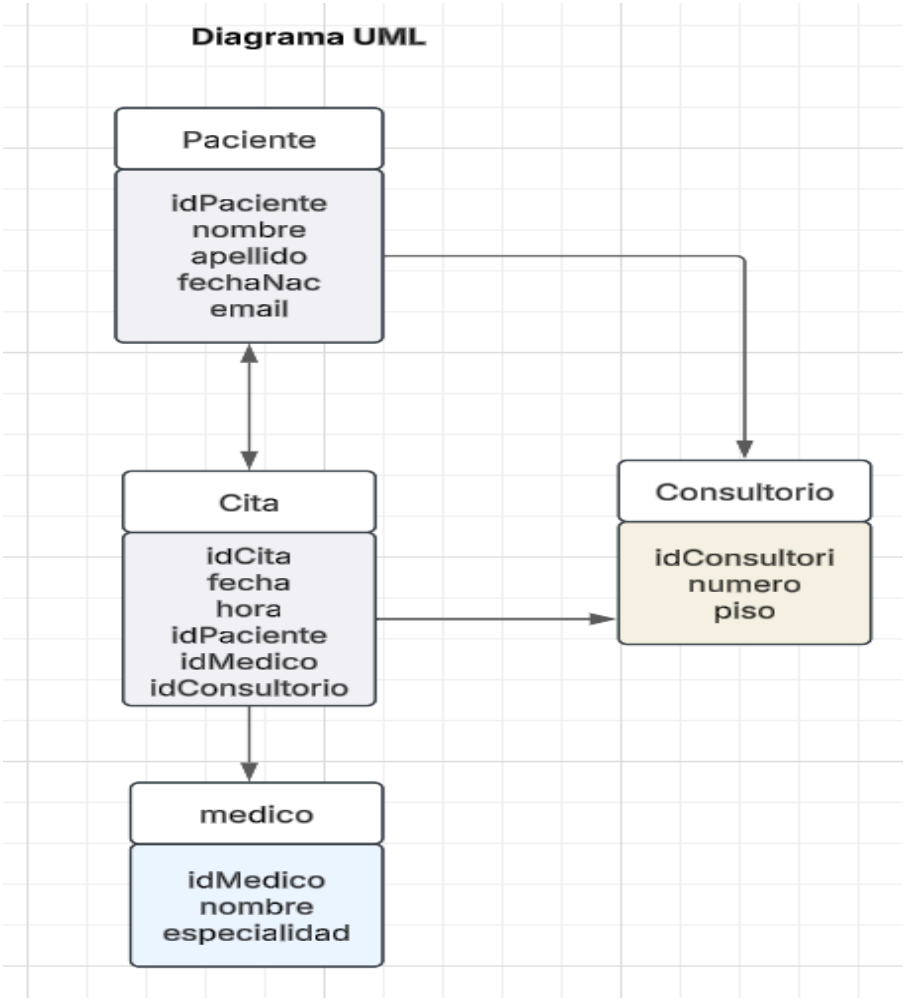


Diagrama UML
Paciente
idPaciente
nombre
apellido
fechaNac
email
Cita
idCita
fecha
hora
idPaciente
idMedico
idConsultorio
Consultorio
idConsultori
numero
piso
medico
idMedico
nombre
especialidad

5. Requisitos Funcionales

Id. Requerimiento	REQ-001
Nombre	Gestión de Pacientes
Registrar pacientes	Información personal (nombre, apellido, fecha de nacimiento, email).
Actualizar y eliminar	Datos de pacientes.
Consultar	Los detalles de los pacientes registrados.

Id. Requerimiento	REQ-002
Nombre	Gestión de Médicos
Registrar médicos	Datos de médicos.
Consultar	Información de los médicos

Id. Requerimiento	REQ-003
Nombre	Gestión de Citas
Agendar citas médica	Para pacientes con un médico y fecha.
Modificar o cancelar	Citas agendadas.
Consultar	Citas programadas para un paciente o médico

Id. Requerimiento	REQ-004
Nombre	Gestión de Consultorios
Registrar y gestionar consultorios	Número de piso
Modificar o cancelar	Citas agendadas.

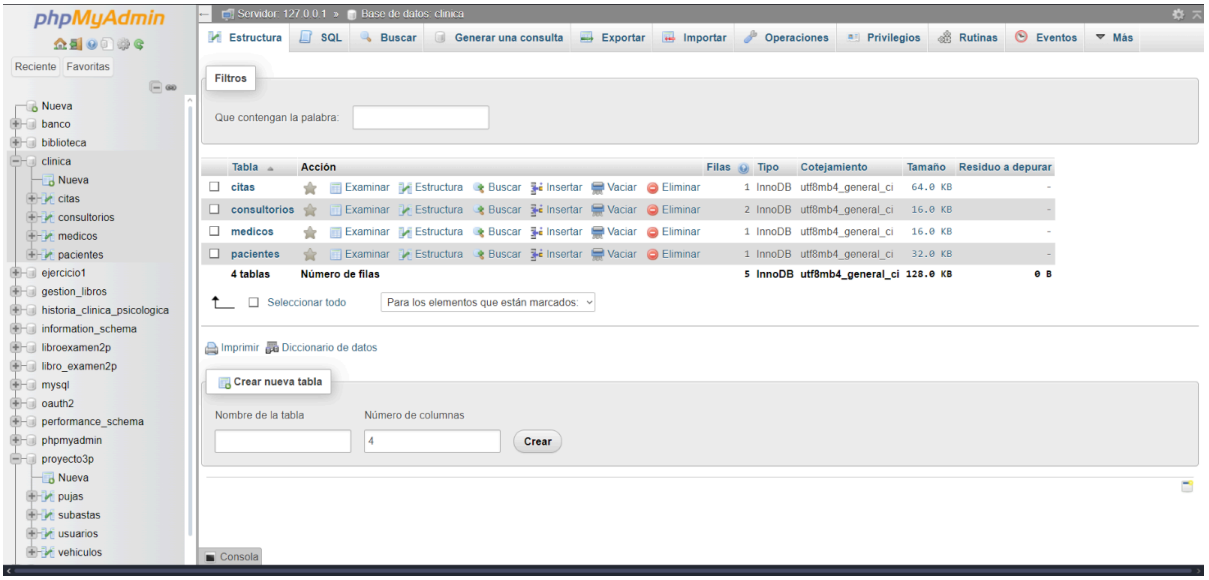
Asignar consultorios	Citas médicas.

Tabla

6.

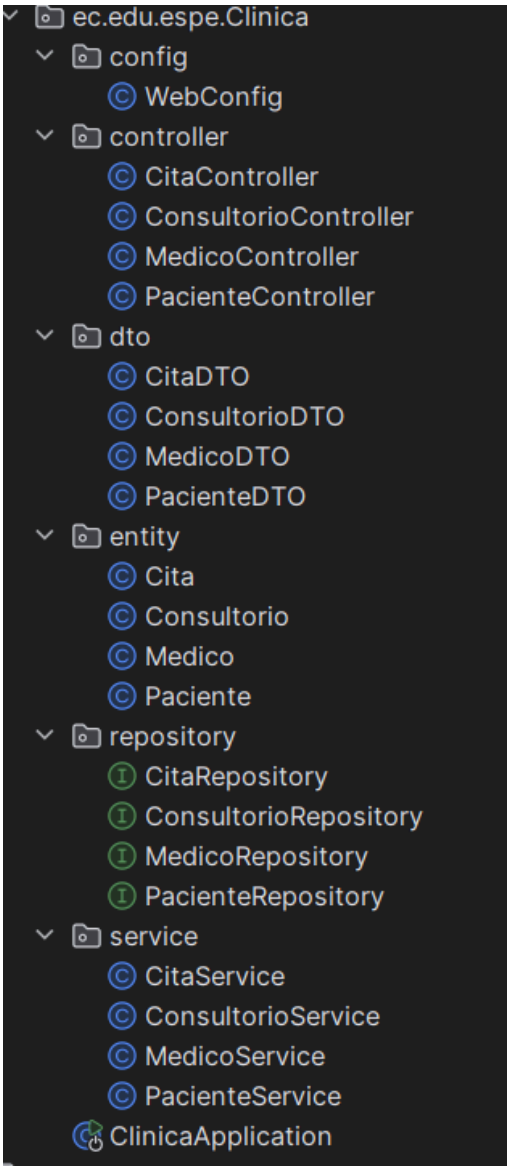
Configuración del Entorno de Desarrollo:

a. Configurar las herramientas de desarrollo: IntelliJ IDEA para backend (Spring Boot) y Visual Studio Code para frontend (React).
b. Instalar y configurar phpMyAdmin para la gestión de la base de datos MySQL.



Desarrollo del Backend:

c. Implementar los microservicios con Spring Boot.



d. Crear las APIs RESTful para la comunicación entre el frontend y el backend.

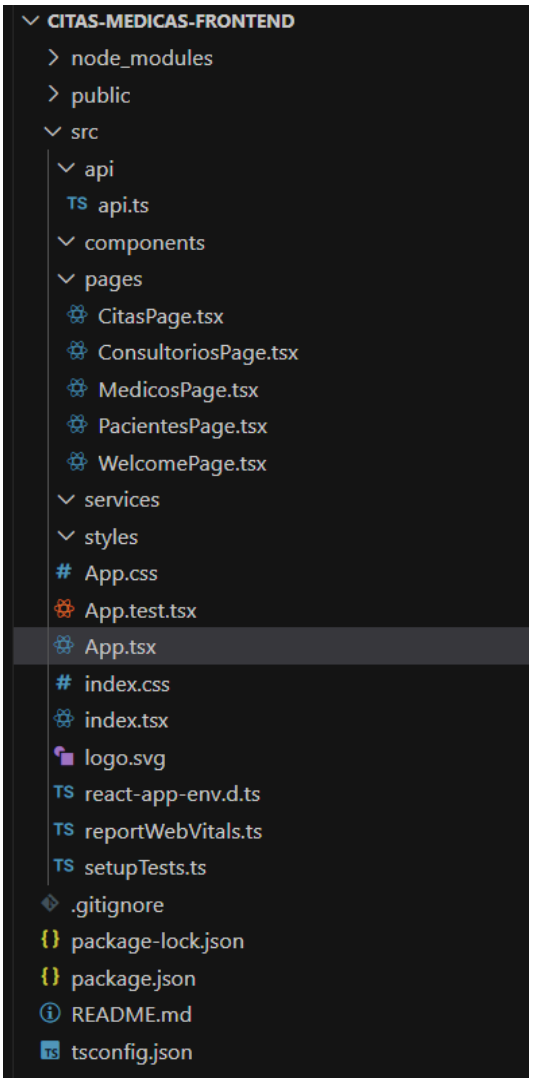
```java
@RestController
@RequestMapping("/api/citas")
public class CitaController {

    @Autowired
    private CitaService citaService;

    @Autowired
    private PacienteRepository pacienteRepository;

    @Autowired
    private MedicoRepository medicoRepository;

    @Autowired
    private ConsultorioRepository consultorioRepository;

    @GetMapping("/pacientes")
    public List<Paciente> listarPacientes() {
        return pacienteRepository.findAll();
    }

    @GetMapping("/medicos")
    public List<Medico> listarMedicos() {
        return medicoRepository.findAll();
    }

    @GetMapping("/consultorios")
    public List<Consultorio> listarConsultorios() {
        return consultorioRepository.findAll();
```

e. Configurar la conexión a la base de datos y realizar las operaciones CRUD.

```
spring.application.name=Clinica


spring.datasource.url=jdbc:mysql://localhost:3306/clinica
spring.datasource.username=root
spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

server.port=9090
```

Desarrollo del Frontend:

f. Implementar las vistas y la lógica de la interfaz de usuario.



g. Conectar el frontend con las APIs del backend para obtener y enviar datos.

```
const CitasPage: React.FC = () => {
  useEffect(() => {
    obtenerConsultorios();
  }, []);

  const obtenerCitas = () => {
    axios.get("http://localhost:9090/api/citas")
      .then(response => setCitas(response.data))
      .catch(error => console.error("Error al obtener citas", error));
  };

  const obtenerPacientes = () => {
    axios.get("http://localhost:9090/api/citas/pacientes")
      .then(response => setPacientes(response.data))
      .catch(error => console.error("Error al obtener pacientes", error));
  };

  const obtenerMedicos = () => {
    axios.get("http://localhost:9090/api/citas/medicos")
      .then(response => setMedicos(response.data))
      .catch(error => console.error("Error al obtener médicos", error));
  };

  const obtenerConsultorios = () => {
    axios.get("http://localhost:9090/api/consultorios")
      .then(response => setConsultorios(response.data))
      .catch(error => console.error("Error al obtener consultorios", error));
  };
```

Paso 1: Descargar dependencias

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core -->
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.24.2</version>
</dependency>
```

Paso 2: Instalar dependencias

```
{
  "dependencies": {
    "react": "^17.0.2",
    "react-dom": "^17.0.2"
  },
  "devDependencies": {
    "typescript": "^4.1.2",
    "eslint": "^7.14.0"
  }
}
```

Paso 3: Lógica para obtener datos

```
const CitasPage: React.FC = () => {  Show usages
  const [citas : Cita[] , setCitas : React.Dispatch<React.SetStateA... ] = useState<Cita[]>( initialState: []);
  const [pacientes : Paciente[] , setPacientes : React.Dispatch<React.SetStateA... ] = useState<Paciente[]>( initialState: []);
  const [medicos : Medico[] , setMedicos : React.Dispatch<React.SetStateA... ] = useState<Medico[]>( initialState: []);
  const [consultorios : Consultorio[] , setConsultorios : React.Dispatch<React.SetStateA... ] = useState<Consultorio[]>( initialState: []);
  const [modalMode : "crear" | "editar" | "borrar" ... , setModalMode : React.Dispatch<React.SetStateA... ] = useState<"crear" | "editar" | "borrar" | null>( initialState: null);
  const [selectedCita : Cita | null , setSelectedCita : React.Dispatch<React.SetStateA... ] = useState<Cita | null>( initialState: null);
  const [nuevaCita : Cita , setNuevaCita : React.Dispatch<React.SetStateA... ] = useState<Cita>( initialState: {
```

7. Evaluación

1.1 Pruebas unitarias

Tabla 1 Pruebas de estrés

Nota: se crea la prueba de estrés usando la herramienta JMeter donde como fue lo solicitado de instertan 500 usuarios por un lapso de 5 minutos.

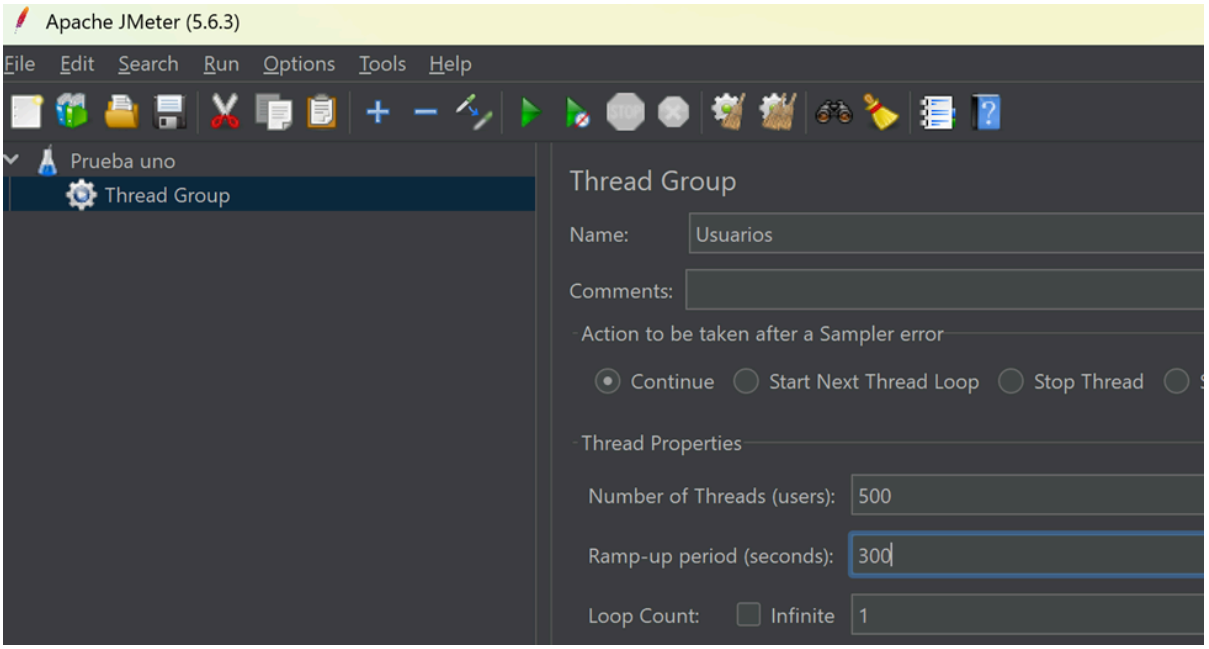


Fig 1: Creación de pruebas

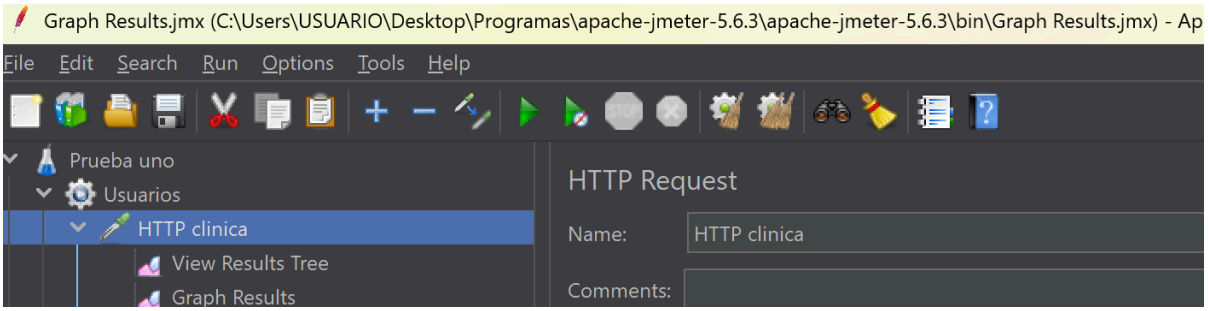


Fig 2: Creación sampler

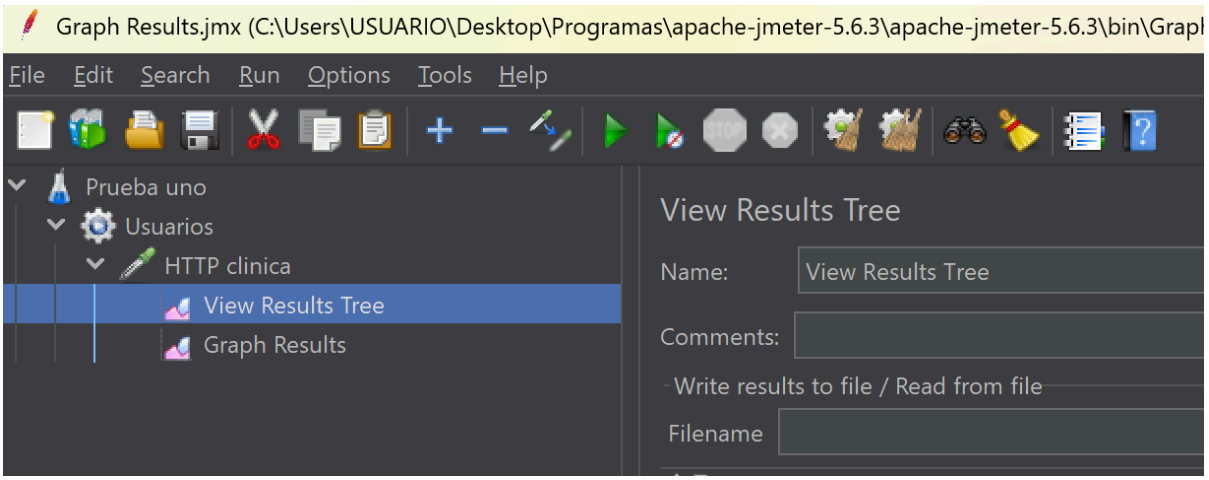


Fig 3: Creación Vistas

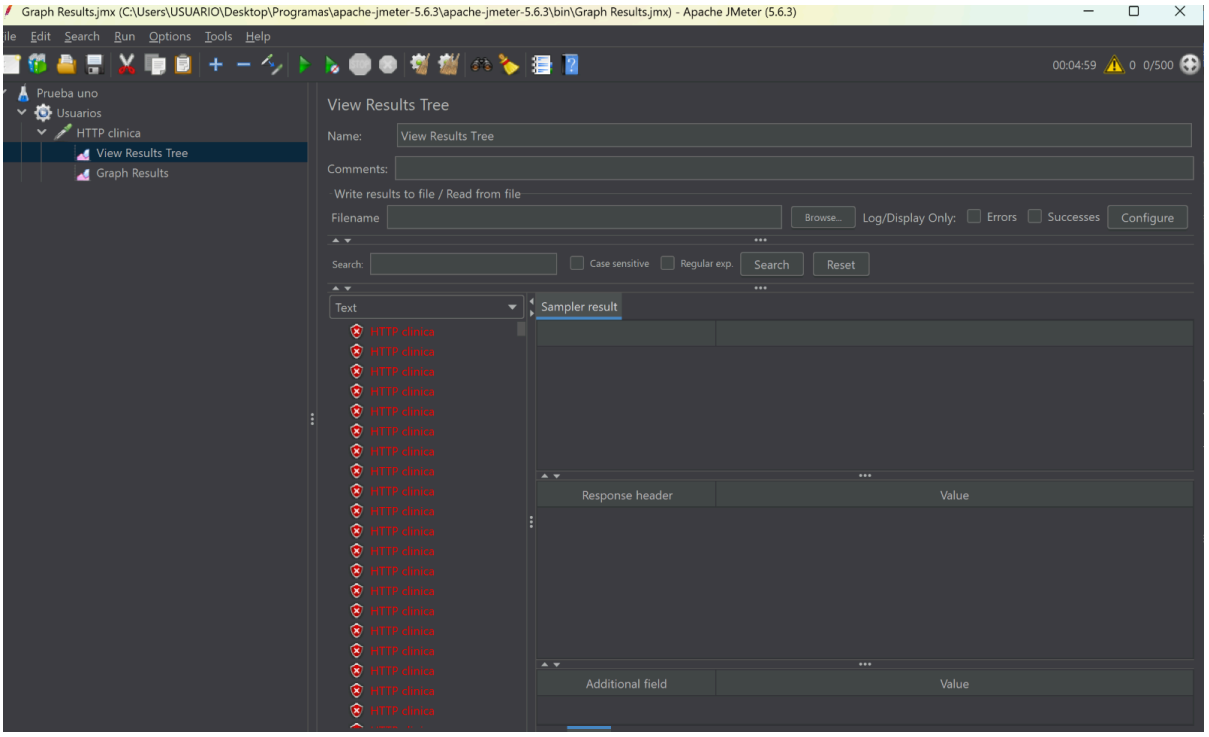


Fig4: Resultado prueba vista result tree.

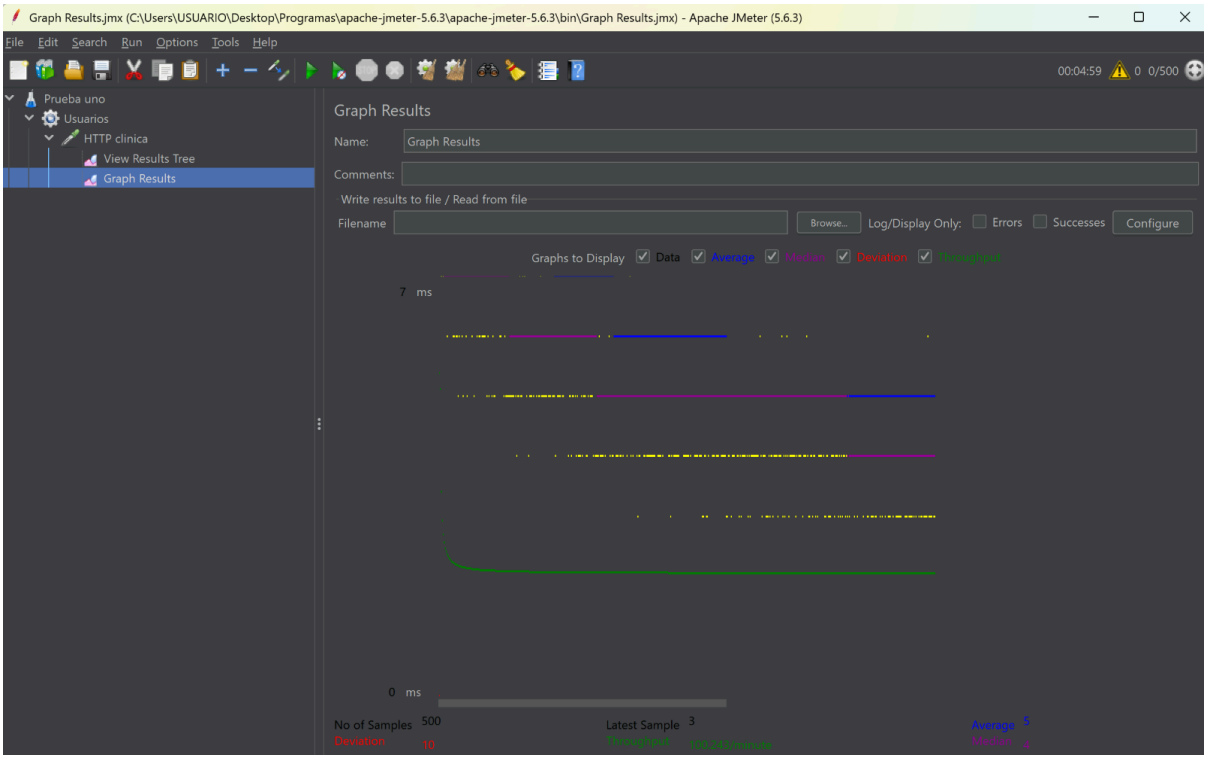


Fig 5: Vista gráfica del resultado de 500 solicitudes por 5 minutos

2.1 Pruebas de recuperación

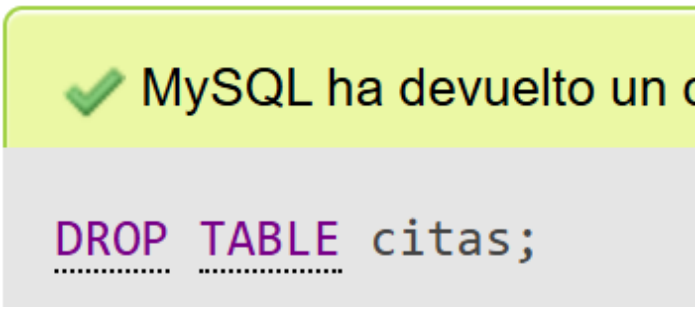


Fig 6: Eliminamos una de las tablas de nuestra base

```
S C:\WINDOWS\system32> mysqldump -u root -p clinica > backup_clinica.sql
nter password:
S C:\WINDOWS\system32>
```

Fig 7: En el powershell realizamos la copia de seguridad de la base

```
PS C:\WINDOWS\system32> cat backup_clinica.sql
-- MySQL dump 10.13  Distrib 8.3.0, for Win64 (x86_64)
--
-- Host: localhost    Database: clinica
-- ------------------------------------------------------
-- Server version       8.3.0

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `consultorios`
--

DROP TABLE IF EXISTS `consultorios`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `consultorios` (
  `id` int NOT NULL AUTO_INCREMENT,
  `numero` varchar(10) NOT NULL,
  `piso` int NOT NULL,
```

Fig 8: Verificación de la copia de seguridad

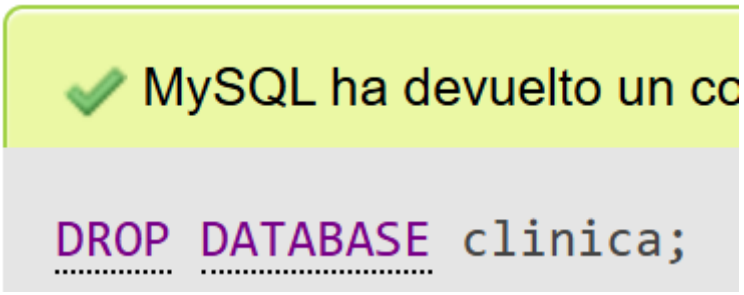


Fig 9: Eliminación de la base de datos dañada

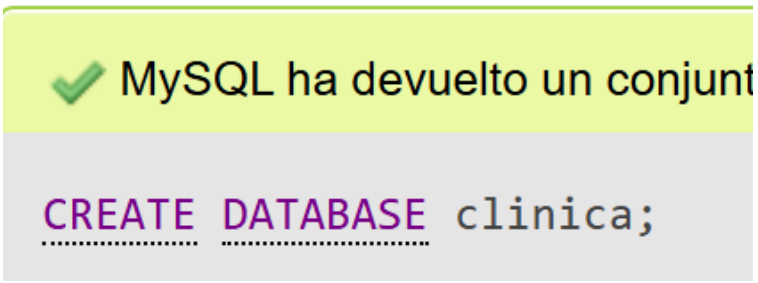


Fig 10: Creamos una nueva base.

```
PS C:\WINDOWS\system32> Get-Content .\backup_clinica.sql | mysql -u root -p clinica
>>
Enter password:
```

Fig 11: Restauramos la copia de seguridad

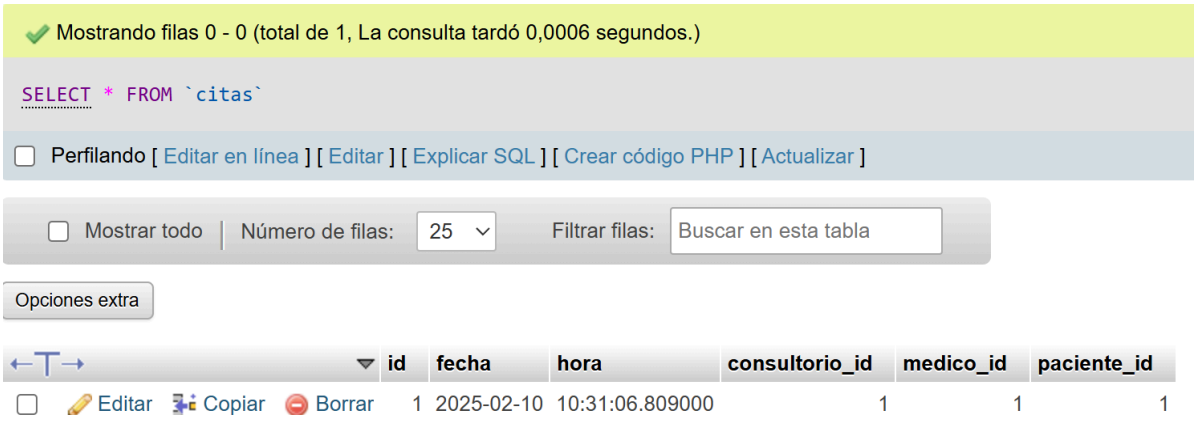


Fig 12: Verificamos la restauración de la base y sus tablas

2.3 Pruebas de integración:

```java
@DataJpaTest
public class ClinicaIntegrationTest {

    @PersistenceContext  30 usages
    private EntityManager entityManager;

    @Test
    @Transactional
    @Rollback(true)
    public void testCrearConsultorio() {
        // Crear un consultorio
        Consultorio consultorio = new Consultorio();
        consultorio.setNumero("A-101");
        consultorio.setPiso(1);

        // Persistir y recuperar para verificar
        entityManager.persist(consultorio);
        entityManager.flush();
```

Fig 13: Creamos un nuevo consultorio.

```java
@Rollback(true)
public void testCrearMedico() {
    // Crear un médico
    Medico medico = new Medico();
    medico.setNombre("Carlos");
    medico.setApellido("Rodríguez");
    medico.setEspecialidad("Cardiología");

    // Persistir y recuperar para verificar
    entityManager.persist(medico);
    entityManager.flush();

    Medico recuperado = entityManager.find(Medico.class, medi

    // Verificar
    assertNotNull(recuperado);
    assertEquals( expected: "Carlos", recuperado.getNombre());
    assertEquals( expected: "Cardiología", recuperado.getEspeci
```

Fig 14: Creación nuevo médico

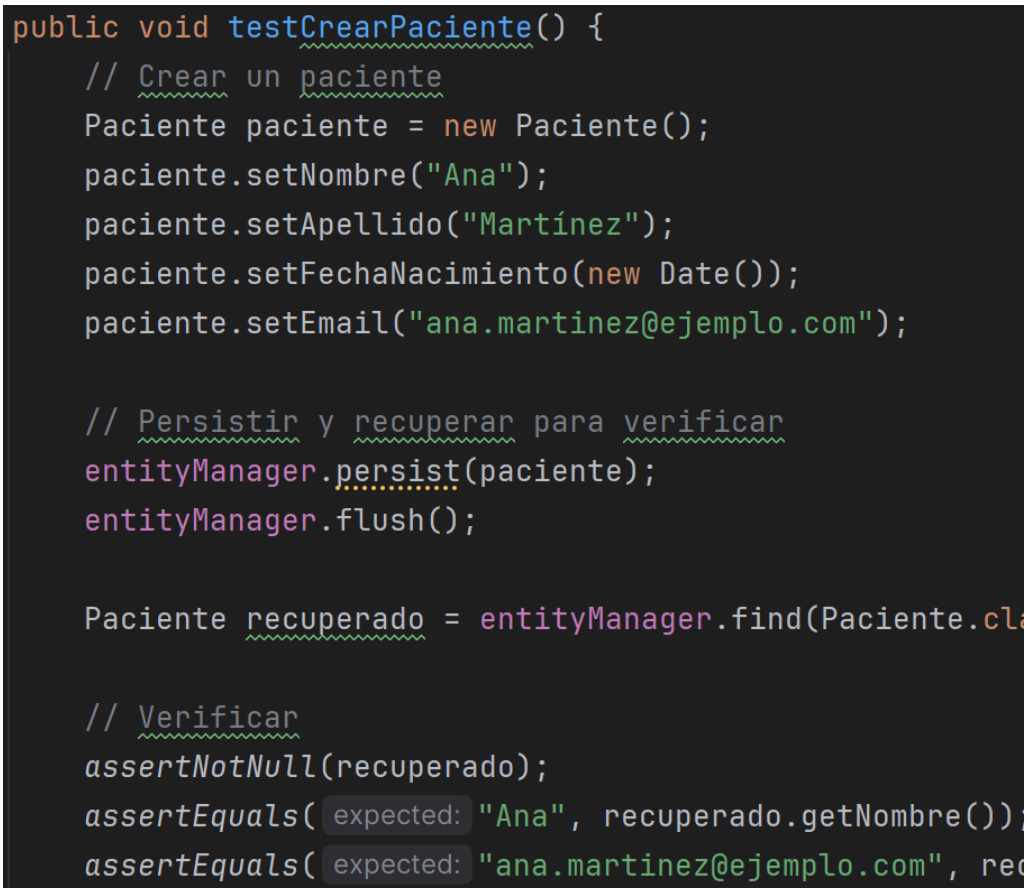


Fig 15: Creamos un nuevo paciente.

```java
// Crear cita
Cita cita = new Cita();
cita.setFecha(LocalDate.now());
cita.setHora(LocalTime.of( hour: 10,  minute: 30));
cita.setConsultorio(consultorio);
cita.setMedico(medico);
cita.setPaciente(paciente);

// Persistir y recuperar
entityManager.persist(cita);
entityManager.flush();

Cita recuperada = entityManager.find(Cita.class, cit

// Verificar
assertNotNull(recuperada);
assertEquals(LocalTime.of( hour: 10,  minute: 30), recup
assertEquals( expected: "Laura", recuperada.getMedico()
```

Fig 16: Creamos una nueva cita

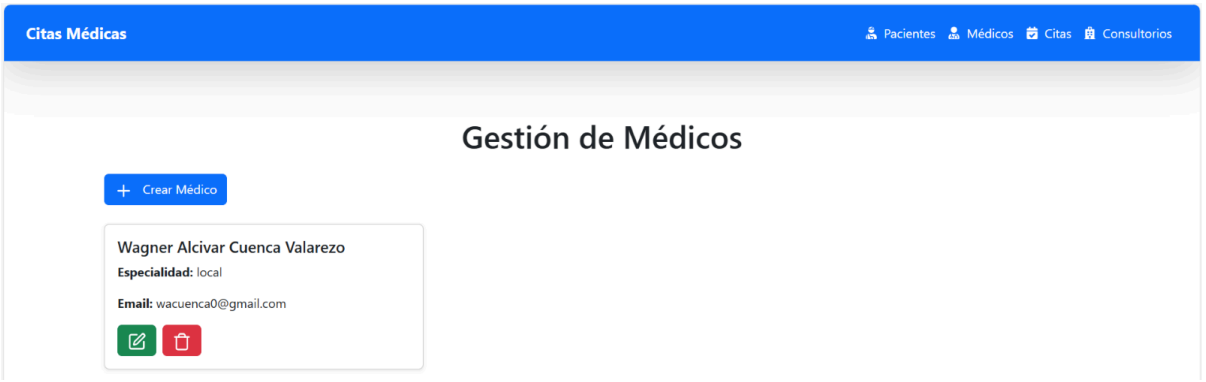
```
PS C:\Users\USUARIO\Desktop\Sexto Semestre\PROGRAMACION AVANZADA\3 PARCIAL\Clinica v> mvn test
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.170 s
[INFO] Finished at: 2025-02-21T01:26:01-05:00
[INFO] ------------------------------------------------------------------------
```

Fig 17: Ejecutamos el test usando JUnit

Interfaz gráfica:


Citas Médicas
Pacientes
Médicos
Citas
Consultorios
¡Bienvenido a Citas Médicas!
Gestiona tus citas médicas de manera fácil y rápida. Accede a la sección que más necesites.
Ir a Pacientes
Ir a Médicos
Ir a Citas
Ir a Consultorios


Citas Médicas
Pacientes
Médicos
Citas
Consultorios
Citas Médicas
Nueva Cita
ID
Paciente
Médico
Fecha
Hora
Consultorio
Acciones
1
Wagner Alcivar
Wagner Alcivar
2025-02-04
21:10:10
1 - Piso 10
Editar
Borrar


Citas Médicas
Pacientes
Médicos
Citas
Consultorios
Gestión de Médicos
Crear Médico
Wagner Alcivar Cuenca Valarezo
Especialidad: local
Email: wacuenca0@gmail.com

9. Conclusiones y trabajos futuros

Este proyecto nos sirvió para implementar un sistemas de que nos va garantizar buenas bases para crea estructura de automatización en la parte de medicina

Incluimos un Apache para simular pruebas de estrés así como para inicializar las bases de datos y migraciones.

10. Referencias

[1]. Dependencias: https://mvnrepository.com/

Repositorio:
https://github.com/KevinQuimuna/FRAMEWROKS-PRUEBAS-AUTOM-TICAS.git