



# INGENIERÍA DEL SOFTWARE III

Primera Iteración ( Implementación )

Documento de Implementación

**Profesor de prácticas asignado:** Miguel Lastra Leidinger

**Grupo de prácticas:** M 1.2 – Miércoles 12- 14h

**Firman este documento:**     **(SubGrupo1)**

F. Javier Briones Rodriguez (Encargado) – 74728484N

Alejandro Mesa Rodriguez – 75908151D

José Peso Buendia – 75158910Q

Fco Manuel Abril Barrilado – 75562375S

03/04/11

**Nombre Documento:** Documento\_Implementacion

**Fase de Iteración:** Implementación      **Nº Iteración:** Primera Iteración

**Número Grupo:** M 1.2      **Versión :** 1.0

## CONTROL DE VERSIONES

<b>Identificador de documento:</b>	<b>Versión</b>	<b>Fecha</b>	<b>Resumen de cambios</b> (documentación de cambios al final del documento, ejem: documento de cambios c1)
DocuImple.pdf	1.0	03/04/11	Versión Inicial

## Índice de contenido

1.Diagrama de Clases de Diseño Optimizado.....	2
2.Implementación Relacional.....	3
3.Filtros Adoptados en la Interfaz.....	7
5.Documento de control de cambios.....	8

**Nombre Documento:** Documento\_Implementacion  
**Fase de Iteración:** Implementación **Nº Iteración:** Primera Iteración  
**Número Grupo:** M 1.2 **Versión :** 1.0

Las clases “Paciente” y “Personal” hacen uso de operaciones get y set para acceder a sus atributos, pero son innecesarias pues pueden usar éstas mismas operaciones que su clase “Usuario” generalizada les ofrece.

```

classDiagram
    class Usuario {
        -DNI: String
        -Contraseña: String
        -Nombre: String
        -Apellidos: String
        -Direccion: String
        -Email: String
        -Telefono: String
        -FechaNac: Date
        -LugarNac: String
        -Foto: Image
        -Tipo: String
        +getDni(): String
        +getContraseña(): String
        +getNombre(): String
        +getApellidos(): String
        +getDireccion(): String
        +getEmail(): String
        +getTelefono(): String
        +getFechaNac(): Date
        +getLugarNac(): String
        +getFoto(): Image
        +getTipo(): String
        +setDni(Dni: String): void
        +setContraseña(Contraseña: String): void
        +setNombre(Nombre: String): void
        +setApellidos(Apellidos: String): void
        +setDireccion(Direccion: String): void
        +setEmail(Email: String): void
        +setTelefono(telefono: String): void
        +setFechaNac(fechaNac: String): void
        +setLugarNac(LugarNac: String): void
        +setFoto(foto: String): void
        +setTipo(Tipo: String): void
    }

    class GestorPacientes {
        +altaPaciente(Dni: String, Nombre: String, Apellidos: String, Direccion: String, Email: String, Telefono: String, FechaNacimiento: Date, LugarNacimiento: String, Fotografia: Image, TipoUsuario: String): String
        +generarContraseña(): String
    }

    class Paciente {
        +Estado: Boolean
        +Fecha: Date
        +DNI: String
        +DNI: String
        +DNI: String
        +getDniPaciente(): String
        +getDniMedico(): String
        +getFecha(): Date
        +getEstado(): String
        +setEstado(Estado: Boolean): void
    }

    class Cita {
        +Estado: Boolean
        +Fecha: Date
        +DNI: String
        +DNI: String
        +DNI: String
        +getDniPaciente(): String
        +getDniMedico(): String
        +getFecha(): Date
        +getEstado(): String
        +setEstado(Estado: Boolean): void
    }

    class GestorCitas {
        +alta_cita(Dni: String): String
        +selfFecha(fecha: Date): String
        +alta_citaonline(): String
        +cancelarCitaOnline(): String
        +cancelarCita(Dni: String): String
        +mostrarErrorCita(): String
        +ConsultarCita(Dni: String): String
        +MostrarErrorPaciente(): String
        +ConsultarCitaOnline(): String
        +ModificarCita(Dni: String): Boolean
        +ModificarCitaOnline(): Boolean
        +EstadisticasCitas(fecha_inicio: Date, fecha_fin: Date): Collection
    }

    class GestorUsuarios {
        +validacionUsuario(Dni: String, Pass: String): Boolean
        +modificarDatosPersonalesAdmin(Nombre: String, Apellidos: String, Dni: String, Direccion: String, Email: String, Contraseña: String, Telefono: String, FechaNac: Date, LugarNac: String, Foto: Image, Tipo: String): String
        +modificarDatosPersonales(Nombre: String, Apellidos: String, Dni: String, Direccion: String, Email: String, Contraseña: String, Telefono: String, FechaNac: Date, LugarNac: String, Foto: Image, Tipo: String): String
        +consultarDatosPersonalesAdmin(Dni: String): String
        +consultarDatosPersonales(): String
    }

    class GestorPersonal {
        +altaPersonal(Dni: String, Nombre: String, Apellidos: String, Direccion: String, Email: String, Telefono: String, FechaNacimiento: String, LugarNacimiento: String, Fotografia: Image, TipoUsuario: String): String
        +generarContraseña(): String
        +existePersonal(Dni: String): Boolean
    }

    class Personal {
        +Tipo: String
        +getTipoPersonal(): String
    }

    class Analista
    class Medico
    class Administrativo
    class Radiologo
    class Farmaceutico

    class Turno {
        -DNI: String
        -FechaIni: Date
        -FechaEnd: Date
        -Tipo: String
        +getFechaInicio(): Date
        +getTipo(): String
        +getFechaFin(): Date
    }

    class GestorTurnos {
        +altaTurno(tipo: String): String
        +bajaTurno(tipo: String): String
        +asignarTurnoPersonal(Dni: String, fechaInicio: Date, fechaFin: Date, tipo: String): String
        +modificarTurnoPersonal(Dni: String, fechaInicio: Date, fechaFin: Date, tipo: String): String
        +consultarTurnoPersonal(Dni: String): String
        +EstadisticasPersonas(fecha: Date): Arraylist
    }

    class TurnoBD {
        +existeTurno(tipo: String): Boolean
        +altaTurno(tipo: String): Boolean
        +puedeBorrarTurno(tipo: String): Boolean
        +borraTurno(Dni: String): Boolean
        +asignarTurno(Dni: String, fechaInicio: Date, fechaFin: Date, tipo: String): Boolean
        +modificaTurno(Dni: String, fechaInicio: Date, fechaFin: Date, tipo: String): Boolean
        +obtenerTurno(Dni: String): Turno
        +ConsultarTurnos(fecha: Date): Arraylist
    }

    class CitasBD {
        +existeCita(Dni: String): Boolean
        +almacenarCita(cita: Cita): Boolean
        +cancelarCita(Dni: String): Boolean
        +mostrarErrorCita(): String
        +obtenerCita(Dni: String): String
        +EstadisticasCitas(fecha_inicio: Date, fecha_fin: Date): Arraylist
    }

    class UsuarioBD {
        +validacion(Dni: String, Pass: String): Boolean
        +obtenerUsuario(Dni: String): Usuario
        +almacenarUsuario(Usuario: Usuario): Boolean
        +existeUsuario(Dni: String): Boolean
    }

    class PacBD {
        +existePaciente(Dni: String): Boolean
        +almacenarPaciente(Paciente: Paciente): Boolean
        +obtenerPaciente(Dni: String): Paciente
    }

    class JDBC {
        +Rs: ResultSet
        +consultaSelect(consulta: String): ResultSet
        +consultaUpdate(consulta: String): Void
    }

    Usuario "0..*" -- "0..*" GestorUsuarios
    Usuario "0..*" -- "0..*" GestorPersonal
    Usuario "0..*" -- "0..*" Turno
    Usuario "0..*" -- "0..*" TurnoBD
    Usuario "0..*" -- "0..*" CitasBD
    Usuario "0..*" -- "0..*" UsuarioBD
    Usuario "0..*" -- "0..*" PacBD
    Usuario "0..*" -- "0..*" JDBC

    GestorPacientes -- Paciente
    GestorCitas -- Cita
    GestorTurnos -- Turno
    GestorTurnos -- TurnoBD
    GestorTurnos -- CitasBD
    TurnoBD -- CitasBD
    TurnoBD -- PacBD
    CitasBD -- PacBD
    PacBD -- JDBC
    JDBC -- Usuario
  
```

03/04/11

**Nombre Documento:** Documento\_Implementacion

**Fase de Iteración:** Implementación      **Nº Iteración:** Primera Iteración

**Número Grupo:** M 1.2

**Versión :** 1.0

## Implementación Relacional

Del diagrama de Clases de Diseño obtenemos las siguientes tablas relacionales:

```
CREATE TABLE `Usuarios`(  
  `Dni` VARCHAR(9) NOT NULL,  
  `Nombre` VARCHAR(200) NOT NULL,  
  `Apellidos` VARCHAR(200) NOT NULL,  
  `Contrasena` VARCHAR(20) NOT NULL,  
  `Direccion` VARCHAR(200) NOT NULL,  
  `Email` VARCHAR(200),  
  `Telefono` VARCHAR(20),  
  `FechaNacimiento` DATE,  
  `LugarNacimiento` VARCHAR(200),  
  `Fotografia` VARCHAR(200),  
  `TipoUsuario` VARCHAR(20) NOT NULL,  
  PRIMARY KEY (`Dni`)  
);
```

```
CREATE TABLE `Pacientes`(  
  `Dni` VARCHAR(9) NOT NULL,  
  PRIMARY KEY (`Dni`),  
  FOREIGN KEY (`Dni`) REFERENCES `Usuarios`(`Dni`) ON DELETE CASCADE  
);
```

```
CREATE TABLE `Personal`(  
  `Dni` VARCHAR(9) NOT NULL,  
  `TipoPersonal` VARCHAR(20) NOT NULL,
```

03/04/11

**Nombre Documento:** Documento\_Implementacion

**Fase de Iteración:** Implementación **Nº Iteración:** Primera Iteración

**Número Grupo:** M 1.2 **Versión :** 1.0

```
PRIMARY KEY (`Dni`),  
FOREIGN KEY (`Dni`) REFERENCES `Usuarios`(`Dni`) ON DELETE CASCADE  
);
```

```
CREATE TABLE `Administrativos`(  
`Dni` VARCHAR(9) NOT NULL,  
PRIMARY KEY (`Dni`),  
FOREIGN KEY (`Dni`) REFERENCES `Personal`(`Dni`) ON DELETE CASCADE  
);
```

```
CREATE TABLE `Medicos`(  
`Dni` VARCHAR(9) NOT NULL,  
PRIMARY KEY (`Dni`),  
FOREIGN KEY (`Dni`) REFERENCES `Personal`(`Dni`) ON DELETE CASCADE  
);
```

```
CREATE TABLE `Analistas`(  
`Dni` VARCHAR(9) NOT NULL,  
PRIMARY KEY (`Dni`),  
FOREIGN KEY (`Dni`) REFERENCES `Personal`(`Dni`) ON DELETE CASCADE  
);
```

```
CREATE TABLE `Radiologos`(  
`Dni` VARCHAR(9) NOT NULL,  
PRIMARY KEY (`Dni`),  
FOREIGN KEY (`Dni`) REFERENCES `Personal`(`Dni`) ON DELETE CASCADE  
);
```

03/04/11

**Nombre Documento:** Documento\_Implementacion

**Fase de Iteración:** Implementación **Nº Iteración:** Primera Iteración

**Número Grupo:** M 1.2

**Versión :** 1.0

```
CREATE TABLE `Farmaceuticos`(  
  `Dni` VARCHAR(9) NOT NULL,  
  PRIMARY KEY (`Dni`),  
  FOREIGN KEY (`Dni`) REFERENCES `Personal`(`Dni`) ON DELETE CASCADE  
);
```

```
CREATE TABLE `Turnos`(  
  `Dni` VARCHAR(9) NOT NULL,  
  `Tipo` VARCHAR(20) NOT NULL,  
  `FechaInicio` DATE,  
  `FechaFin` DATE,  
  PRIMARY KEY (`Dni`)  
);
```

```
CREATE TABLE `Citas`(  
  `DniPaciente` VARCHAR(9) NOT NULL,  
  `DniMedico` VARCHAR(9) NOT NULL,  
  `Fecha` DATE,  
  `Estado` BOOLEAN,  
  `DniAdministrativoCita` VARCHAR(9) NOT NULL,  
  PRIMARY KEY (`DniPaciente`, `DniMedico`),  
  FOREIGN KEY (`DniPaciente`) REFERENCES `Pacientes`(`Dni`),  
  FOREIGN KEY (`DniMedico`) REFERENCES `Medicos`(`Dni`),  
  FOREIGN KEY (`DniAdministrativoCita`) REFERENCES `Administrativos`(`Dni`)  
);
```

03/04/11

**Nombre Documento:** Documento\_Implementacion

**Fase de Iteración:** Implementación      **Nº Iteración:** Primera Iteración

**Número Grupo:** M 1.2

**Versión :** 1.0

```
CREATE TABLE `PersonalTrabajaEnTurno`(  
  `DniPersonal` VARCHAR(9) NOT NULL,  
  `DniTurno` VARCHAR(9) NOT NULL,  
  PRIMARY KEY (`DniPersonal`, `DniTurno`),  
  FOREIGN KEY (`DniPersonal`) REFERENCES `Personal`(`Dni`),  
  FOREIGN KEY (`DniTurno`) REFERENCES `Turnos`(`Dni`)  
);
```

```
CREATE TABLE `AdministrativoAsignaTurno`(  
  `DniAdministrativo` VARCHAR(9) NOT NULL,  
  `DniTurno` VARCHAR(9) NOT NULL,  
  PRIMARY KEY (`DniAdministrativo`, `DniTurno`),  
  FOREIGN KEY (`DniAdministrativo`) REFERENCES `Administrativos`(`Dni`),  
  FOREIGN KEY (`DniTurno`) REFERENCES `Turnos`(`Dni`)  
);
```

03/04/11

**Nombre Documento:** Documento\_Implementacion

**Fase de Iteración:** Implementación      **Nº Iteración:** Primera Iteración

**Número Grupo:** M 1.2

**Versión :** 1.0

## **Filtros Adoptados desde la Interfaz**

\*Para todos los campos introducidos:

- No se permite la introducción de cadenas vacías

\*Los nombres ,apellidos y Lugar de Nacimiento,

- No se puedan introducir números
- Que la longitud máxima sea de 50 caracteres, (aunque la base de datos permite 200 caracteres)

\*Los Dnis : las 8 primeras pulsaciones sólo permite la pulsación de números, y la 9ª una letra.

\*El teléfono: Solo permite la pulsación de números.

\*Las Fechas :

- Han de ser válidas (ejem.31 de febrero-->Error)
- Las fechas de nacimiento no pueden ser mayores que la fecha actual del Sistema.

\*En el alta de Personal, ha de estar obligatoriamente pulsado un tipo de Personal.

\*En Gestionar Turno, la fecha de Inicio no puede ser mayo que la fecha de Fin y debe de estar seleccionado uno de los turno de Trabajo obligatoriamente.