

UNIVERSIDAD DE LA SIERRA SUR

DOCUMENTACIÓN DEL
SOFTWARE

Integrantes del equipo:

Mario Hexai Valencia Reyes

Alexi Daniel Ramírez Ruiz

Fazziel Pérez Hernández

Grupo: 506 A

Semestre: Quinto

Materia: Ingeniería de Software

Maestro: M.T.C.A. Rolando Pedro Gabriel

Colaboradores del proyecto

Nombre del colaborador	Cargo	Perfil (experiencia)	Fotografía
M.T.C.A Rolando Pedro Gabriel	Usuario Final	Ingeniería de software	
M.C Lirio Ruiz Guerra	BDA	Base de datos (PostgreSQL, MySQL).	
M.T.E Everardo de Jesús Pacheco Antonio	Cliente	Programación en Java	
Mario Hecxai Valencia Reyes	Project Manager	Programador en Java, C, lenguaje ensamblador, JavaScript, HTML, CSS. Base de datos manejada PostgreSQL, MySQL.	
Jazziel Pérez Hernández	Programador Senior	Programador en Java, C, lenguaje ensamblador, JavaScript, HTML, CSS. Base de datos manejada PostgreSQL, MySQL.	
Alexi Daniel Ramírez Ruiz	Programador Jr	Programador en Java, C, lenguaje ensamblador, JavaScript, HTML, CSS. Base de datos manejada PostgreSQL, MySQL.	

Indice de contenido

1. Introducción	8
2. Especificación de requisitos	9
2.1. Ámbito del proyecto	9
2.2. Factibilidad	10
2.3. Viabilidad	11
2.3.1. Viabilidad técnica	11
2.3.2. Viabilidad económica	13
2.4. Requisitos específicos	14
2.4.1. Requisitos funcionales	14
2.4.2. Requisitos no funcionales	18
2.5. Mapa de navegación	19
2.5.1. Requerimientos de interfaz	20
3. Análisis	29
3.1. Diagramas UML	29
3.1.1. Diagramas de estructura	29
3.1.2. Diagramas de comportamientos	57
4. Diseño	118
4.1. Introducción	118
4.2. Descripción del sistema	119
4.3. Arquitectura del sistema	120
4.3.1. Físico	120
4.3.2. Lógico	122
4.4. Estándar de codificación	125
4.5. Diseño de datos	130
4.5.1. MC	130
4.5.2. ML	131
4.5.3. MF	133
4.5.4. Diccionario de datos	134

5. Conclusión	136
5.1. Definiciones, acrónimos, abreviaturas	137
5.2. Anexo 2	138

Índice de figuras

2.1.	RF01	14
2.2.	RF02	15
2.3.	RF03	15
2.4.	RF04	16
2.5.	RF05	16
2.6.	RF06	17
2.7.	RF07	17
2.8.	RNF01	18
2.9.	RNF02	18
2.10.	RNF03	18
2.11.	Mapa de navegación	19
2.12.	Login	20
2.13.	Pantalla inicio	21
2.14.	Consejos de salud	22
2.15.	Herramientas(Calculadora de calorías)	23
2.16.	Herramientas(Calcular tu IMC)	24
2.17.	Datos de registro de pacientes	25
2.18.	Evaluación del paciente	26
2.19.	Evaluación del paciente 2	27
2.20.	Biblioteca de alimentos	28
3.1.	Diagrama nivel 0	29
3.2.	Diagrama nivel 1	30
3.3.	Diagrama de clases	31
3.4.	Diagrama de objetos	39
3.5.	Diagrama de componenentes	41
3.6.	Diagrama de actividades del login	45
3.7.	Diagrama de actividades del Registro paciente	47
3.8.	Diagrama de actividades de Herramientas	49
3.9.	Diagrama de actividades de Diagnóstico	51
3.10.	Diagrama de actividades de Consejos de salud	53

3.11. Diagrama de actividades de Biblioteca	55
3.12. Caso de uso login	57
3.13. Caso de uso pantalla principal	59
3.14. Caso de uso registro paciente	61
3.15. Caso de uso diagnóstico	64
3.16. Caso de uso consejos de salud	66
3.17. Caso de uso cálculo de IMC	68
3.18. Caso de uso cálculo de calorías	70
3.19. Caso de uso biblioteca de alimentos	72
3.20. Documentación login	75
3.21. Documentación pantalla principal	77
3.22. Documentación registro paciente	78
3.23. Documentación diagnóstico	79
3.24. Documentación consejos de salud	97
3.25. Documentación cálculo de IMC	98
3.26. Documentación cálculo de calorías	99
3.27. Documentación biblioteca de alimentos	100
3.28. Diagrama de estados: login	101
3.29. Diagrama de estados: registro paciente	102
3.30. Diagrama de estados: herramientas	103
3.31. Diagrama de estados: diagnóstico	104
3.32. Diagrama de estados: consejos salud	105
3.33. Diagrama de estados: consejos salud	106
3.34. Diagrama de estados: biblioteca de alimentos	107
3.35. Diagrama de secuencias: login	108
3.36. Diagrama de secuencias: Pantalla principal	109
3.37. Diagrama de secuencias: Registro paciente	110
3.38. Diagrama de secuencias: Herramientas	111
3.39. Diagrama de secuencias: Calculo de calorias	112
3.40. Diagrama de secuencias: diagnósticos	113
3.41. Diagrama de secuencias: consejo salud	114
3.42. Diagrama de secuencias: biblioteca	115
3.43. Diagrama de colaboración	116
3.44. Diagrama de colaboración	117
4.1. computadora portátil	120
4.2. teléfono móvil	121
4.3. Java	122
4.4. Netbeans	123
4.5. Netbeans	124
4.6. Paquetes	125

4.7. Nombre de las interfaces	126
4.8. Nombre de las clases	126
4.9. Métodos	126
4.10. Variables	127
4.11. Variables for	127
4.12. Constantes	128
4.13. Comentarios	128
4.14. Declaraciones	128
4.15. Sentencias	129
4.16. Sentencias	129
4.17. Propiedades	129
4.18. Esquema relacional	131
4.19. Diagrama entidad relación	132
4.20. Diccionario de la tabla consultorio	134
4.21. Diccionario de la tabla pacientes	134
4.22. Diccionario de la tabla médico	135
4.23. Diccionario de la tabla visita médica	135
5.1. Exposición	139
5.2. Noche de desvelo	140
5.3. Durmiendo	141
5.4. Cenando	142

Capítulo 1

Introducción

El software es un conjunto de instrucciones lógicas que le permite al usuario interactuar con los sistemas informáticos a través de una interfaz, es lo que comúnmente se conoce como las aplicaciones de un ordenador. A través de este modelo implementado en los servicios informáticos se pueden encontrar un sinfín de alternativas que cumplan con un específico. El desarrollo de software parte de la obligación de contribuir, facilitar y provocar un impacto que cumpla con las necesidades de usuarios o clientes específicos. Dentro del desarrollo de programas informáticos se encuentran las malas técnicas e implementaciones en el inicio del desarrollo. Gran parte de desarrolladores comenten errores en la fase inicial por la falta de planificación y estructuración de un proceso constructivo apegado a las necesidades de un cliente. La construcción de un software tiene un proceso donde se definen las tareas y actividades que se van a realizar para alcanzar un objetivo. La ingeniería de software es una rama que deriva de la ciencia y, por lo tanto, esta contiene estudios como métodos, técnicas y un arreglo de herramientas que permiten a los profesionales el desarrollo y mantenimiento de software de calidad. Los productos de software cumplen con un proceso donde son definidos por quien va a hacer qué, cómo y cuándo. El arduo trabajo en el desarrollo de sistemas es alcanzado por las etapas de mantenimiento como fase final del desarrollo de software.

Capítulo 2

Especificación de requisitos

2.1. Ámbito del proyecto

Hoy en día los trastornos de conducta alimentaria han impactado en una gran parte de la población. Las personas en la etapa de adolescencia han sido los más afectados y siendo más común en las mujeres. Esta enfermedad ha surgido principalmente en la aceptación social muy ligada a la apariencia física, debido a esto las personas se auto imponen hábitos que pueden conllevar desbordes de todo tipo. Con base a la necesidad que se ha tomado para resolver de una forma más organizable el tratamiento y el diagnóstico de los pacientes con Anorexia, se desarrollará un Software que ayude a los profesionales en la salud nutricional. Esto con el objetivo de proporcionar de manera organizada métodos que cumplan con las fases del padecimiento del paciente. Usualmente hoy en día existen aplicaciones de este tipo, pero no cumplen con requisitos más completos para un sistema con mejor alcance para los usuarios de los trastornos alimenticios. El nombre de la aplicación que se encuentra en la industria del software es TCApp, es una aplicación tipo móvil, donde los pacientes comen comida de un plato que está en una balanza conectada a su teléfono inteligente. La pérdida de peso del plato es registrada por la báscula y, a través de la aplicación crea una curva de ingestión de alimentos, duración de la comida y velocidad de esta. Esta aplicación contiene también una puntuación en la pantalla y se le pide que califique su sensación de saciedad. A diferencia de la aplicación que se desarrolla, esta es de tipo escritorio, no requiere conexión a internet y tiene más herramientas que pueden ayudar a diagnosticar el trastorno alimenticio y su posible tratamiento nutrimental.

2.2. Factibilidad

1 El proyecto cuenta con los recursos necesarios para llevar a cabo el proceso del desarrollo informático previamente planteado. La Universidad de la Sierra Sur cuenta con un centro de tecnologías de la información en la cual se tiene computadoras que cumplen con las características a nivel hardware y software, así también servicio de internet, asesoría de docentes altamente calificados en sus ramas. En cuanto nuestro conocimiento como estudiantes manejamos lenguajes como; JAVA, C, JavaScript, ensamblador, Html y CSS, los cuales nos serán útiles para elaboración y resolución del problema.

2.3. Viavilidad

2.3.1. Viavilidad técnica

Recursos del Hardware

- 1.- Ordenador de sobremesa. Ordenador montado por piezas (clónico) con arquitectura 386.
- 2.- Ordenador portátil (notebook). Ordenador de marca Lenovo, procesador Intel(R) Core (TM) i7-8550U CPU @ 1.80GHz 1.99 GHz, RAM instalada de 8.00 GB.
- 3.- Teléfono móvil celular tipo smartphone, con acceso WiFi y GPS. teclado alfanumérico completo.

Recursos del Software

1. Sistema operativo que uso habitualmente en los ordenadores: GNU Linux, distribuciones Ubuntu y openSUSE.
2. Aplicaciones informáticas que empleo bajo sistema operativo GNU Linux:

- Navegadores internet: Mozilla Firefox, Google Chrome y Brave.
- Cliente correo electrónico POP: Evolution
- Base de datos: Postgre SQL
- Entorno de desarrollo Apache NetBeans IDE 12.3
- Libreoffice: editor de textos, hojas de cálculo.
- Google Drive
- Tratamiento de imágenes: Gimp, ImageMagick
- Gestor y lector de libros electrónicos: Calibre
- YouTube, Spotify.
- Antivirus: En GNU Linux no se precisa antivirus
- Sistema de búsqueda e instalación de software: Centro de software de Ubuntu.
- Plataforma platzi
- Aplicación Web: Overleaf, Editor de LaTeX online.
- Aplicación Web: GitHub
- Aplicación Web: Balsamiq Studios

3. Aplicaciones informáticas habituales bajo sistema operativo Windows:

- Base de datos: Postgre SQL
- Entorno de desarrollo Apache NetBeans IDE 12.3
- Navegadores internet: Mozilla Firefox, Google Chrome y Opera.
- Ofimática (editor de textos, hoja de cálculo, base de datos, presentaciones): Libreoffice
- Ofimática en línea (online): Google Drive
- Gestor de imágenes, operaciones básicas: FastStone
- Antivirus: Avira
- YouTube, Spotify
- Plataforma platzi
- Aplicación Web: Overleaf, Editor de LaTeX online.
- Aplicación Web: GitHub
- Aplicación Web: Balsamiq Studios

2.3.2. Viabilidad económica

Costos de Personal

La propuesta no estima que se deba realizar un gasto adicional en costos de personal.

Costos de desarrollo

El sistema a desarrollar estará a cargo de los alumnos de la Universidad de la Sierra Sur como proceso de evaluación ordinaria, con el objetivo de desarrollar un sistema con un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de los programas informáticos

Costos de hardware

Teniendo en cuenta que los desarrolladores poseen el equipo necesario, los costos asociados a hardware son nulos.

Costos de software

TTeniendo en cuenta que los equipos poseen el software necesario y apropiado sin la necesidad de recurrir a gastos extras o adquisición de alguna licencia,podemos afirmar que los costos asociados a software son nulos.

2.4. Requisitos específicos

2.4.1. Requisitos funcionales

Código de requerimiento	RF01
Nombre	Alta o baja de pacientes
Propósito	Dentro del sistema se permitirá dar alta y/o baja según la situación del paciente.
Descripción	En una interfaz donde se almacenen los pacientes, se podrá abrir una nueva ventana la cual te permitirá agregar nuevos pacientes, así mismo en la lista de pacientes habrá opciones de baja.
Entrada	Formulario de alta de pacientes
Salida	-Mensaje de éxito de alta.
Prioridad	Alta

Figura 2.1: RF01

Código de requerimiento	RF02
Nombre	Entrada al sistema
Propósito	El personal encargado tendrá acceso exclusivo al sistema.
Descripción	En una interfaz de inicio, se agregará el nombre y contraseña del profesional encargado de pacientes.
Entrada	Formulario de autenticación.
Salida	-Carga de pantalla de inicio.
Prioridad	Alta

Figura 2.2: RF02

Código de requerimiento	RF03
Nombre	Modificación de la información del paciente.
Propósito	Hacer correcciones o actualizar la información del paciente.
Descripción	Dentro de una serie de preguntas almacenadas, serán contestadas en secuencia por los pacientes.
Entrada	Formulario de perfil
Salida	Mensaje de actualización en el sistema.
Prioridad	Alto

Figura 2.3: RF03

Código de requerimiento	RF04
Nombre	Facilidad de uso (Manual de usuario)
Propósito	Que el usuario tenga la facilidad de navegar en la aplicación y cuente con un manual de interpretación del sistema.
Descripción	El usuario será capaz de utilizar las funciones completas del sistema.
Entrada	Aplicación
Salida	El uso correcto de la aplicación
Prioridad	Alta

Figura 2.4: RF04

Código de requerimiento	RF05
Nombre	Respuestas ante fallos
Propósito	Que el usuario conozca sus errores en el sistema.
Descripción	Cuando el sistema presente un fallo no se demorará mucho tiempo en restaurar los datos del sistema y volver ponerlos en marcha.
Entrada	Pausa del sistema
Salida	Continuación del sistema
Prioridad	Medio

Figura 2.5: RF05

Código de requerimiento	RF06
Nombre	Historial clínico
Propósito	Que el usuario profesional en nutrición conozca las situaciones actuales del paciente.
Descripción	Se generará historiales de en forma de reporte por parte del profesional en nutrición.
Entrada	Acceso al historial
Salida	Datos generales del paciente
Prioridad	Alta

Figura 2.6: RF06

Código de requerimiento	RF07
Nombre	Integridad de la información
Propósito	Que la información sea verídica
Descripción	Se generará historiales de en forma de reporte por parte del profesional en nutrición.
Entrada	Acceso al historial
Salida	Datos generales del paciente
Prioridad	Alta

Figura 2.7: RF07

2.4.2. Requisitos no funcionales

Código de requerimiento	RNF01
Nombre	Privacidad de la información
Descripción	La información almacenada de los pacientes será confidencial y de uso exclusivo para sus tratamientos.
Prioridad	Alta

Figura 2.8: RNF01

Código de requerimiento	RNF02
Nombre	Almacenamiento definido por software
Descripción	Permite que los recursos de almacenamiento formen parte de una arquitectura de datos.
Prioridad	Media

Figura 2.9: RNF02

Código de requerimiento	RNF03
Nombre	Software ejecutable multiplataforma
Descripción	El software o programa funciona en varios sistemas operativos.
Descripción	Alta

Figura 2.10: RNF03

2.5. Mapa de navegación



Figura 2.11: Mapa de navegación

2.5.1. Requerimientos de interfaz

La interfaz inicio de sesión requiere la entrada de datos como el nombre y la contraseña que correspondan al rol, posteriormente hay una validación de datos para entrar a la pantalla principal.

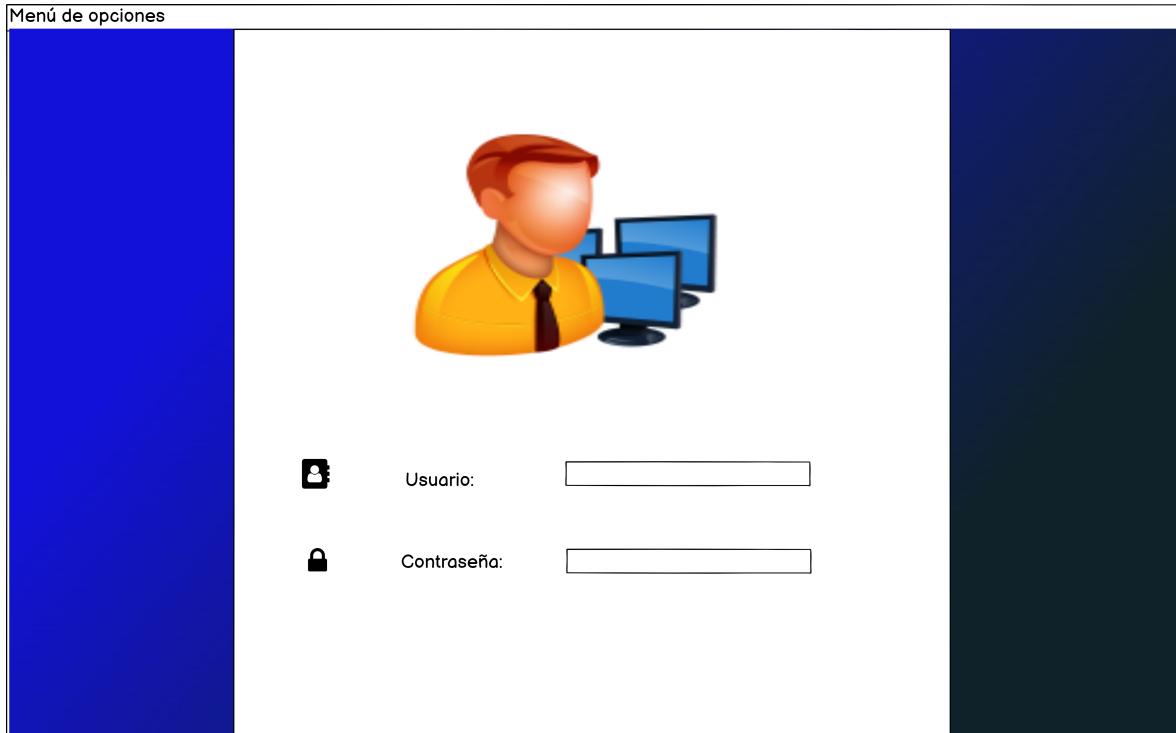


Figura 2.12: Login

La interfaz "pantalla principal" permite navegar sobre un panel de navegación sobre los diferentes módulos que la aplicación contiene, además, nos muestra algunas imágenes referentes a la salud nutricional e información relevante.

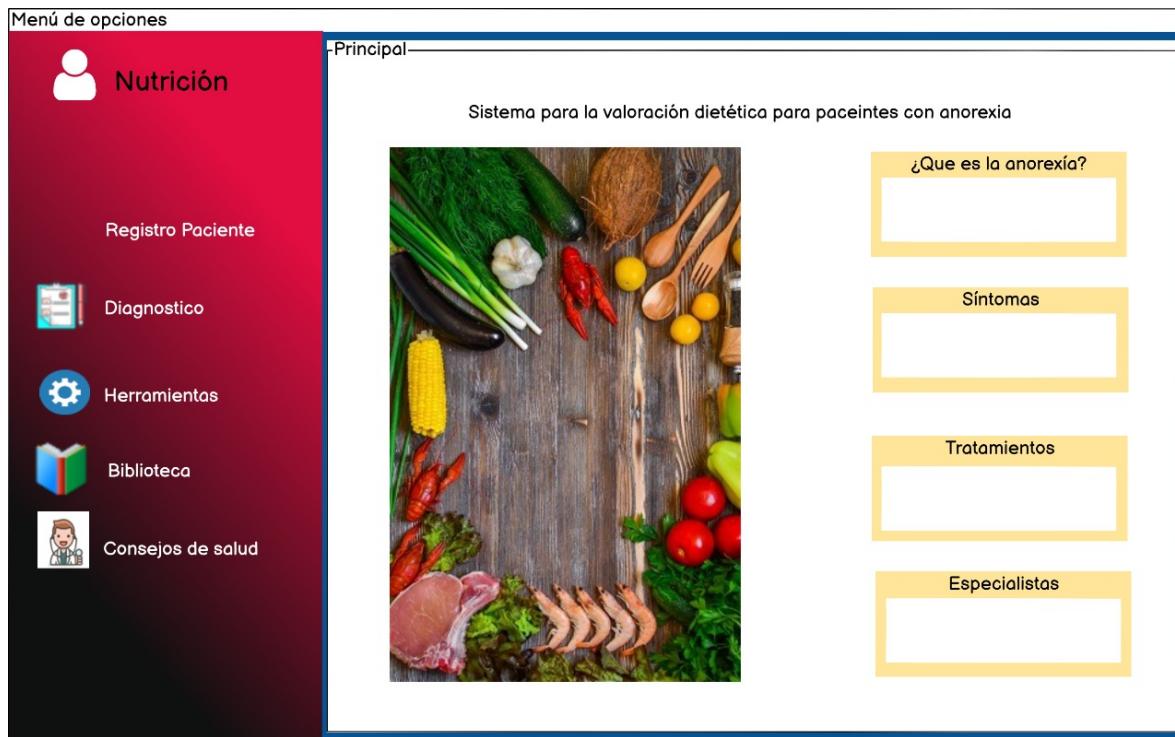


Figura 2.13: Pantalla inicio

La interfaz "consejos de salud" permite conocer algunos datos referentes a la salud nutricional y que se debe de hacer en caso de un padecimiento

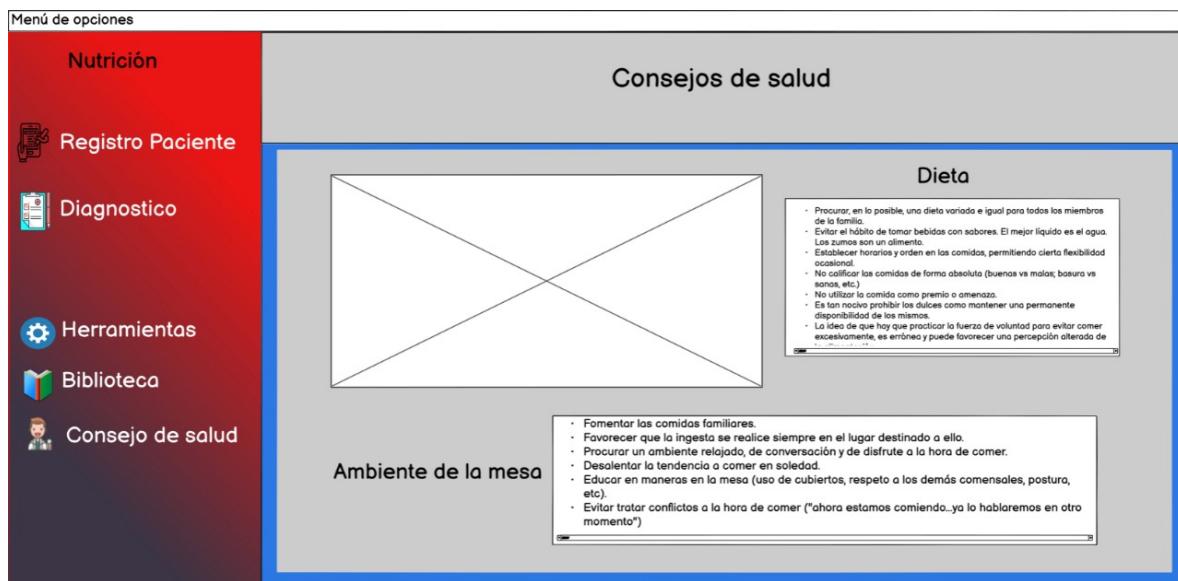


Figura 2.14: Consejos de salud

La interfaz "calculadora de calorías" permite la entrada de datos personales del paciente para hacer cálculos sobre las calorías necesarias sobre el consumo del paciente, posteriormente mostrará un resultado numérico.

The screenshot shows a software application window titled 'Herramientas' (Tools). On the left, there is a vertical menu bar titled 'Menú de opciones' (Options Menu) with the following items: Nutrición (Nutrition), Registro Paciente (Patient Registration), Diagnóstico (Diagnosis), Herramientas (Tools), Biblioteca (Library), and Consejo de salud (Health Advice). The 'Herramientas' item is currently selected and highlighted in blue. The main workspace is titled 'Calculadora de calorías' (Calorie Calculator). It contains several input fields: 'Edad:' (Age) with a dropdown menu 'Elige su edad', 'Altura:' (Height) with a text input field, 'Sexo:' (Sex) with a dropdown menu 'Elige su sexo', and 'Peso:' (Weight) with a text input field. To the right of these fields is a section titled 'Actividades físicas' (Physical Activities) with a list of five options: 1 Poco o ningún ejercicio (Very little or no exercise), 2 Ejercicio ligero (1-3 días a la semana) (Light exercise (1-3 days a week)), 3 Ejercicio moderado (3-5 días a la semana) (Moderate exercise (3-5 days a week)), 4 Ejercicio fuerte (6-7 días a la semana) (Intense exercise (6-7 days a week)), and 5 Ejercicio muy fuerte (dos veces al día, entrenamientos muy duros) (Very intense exercise (twice a day, very hard training)). Below this list are five radio buttons numbered 1 through 5, corresponding to the activity levels. At the bottom right of the calculator area is a yellow button labeled 'Calcular calorías' (Calculate calories).

Figura 2.15: Herramientas(Calculadora de calorías)

La interfaz calculadora de IMC"permite la entrada de datos personales del paciente para hacer cálculos sobre el IMC correspondiente al paciente, posteriormente mostrará un resultado numérico

Menú de opciones

Nutrición

Herramientas

Registro Paciente

Diagnóstico

Herramientas

Biblioteca

Consejo de salud

Calcula tu IMC | Calculadora de calorías

Sexo:

Peso: Kgs

Edad:

Altura: cms

Figura 2.16: Herramientas(Calcular tu IMC)

La interfaz "registro de pacientes" permite la entrada de datos personales del paciente para hacer su registro a la base datos, en este módulo se podrán ver los pacientes que se encuentran almacenados registrados, así como la eliminación y actualización de los datos respectivamente.

Menú de opciones

Nutrición

Registro Paciente

Diagnóstico

Herramientas

Biblioteca

Consejo de salud

Registro de pacientes

Nuevo Guardar Modificar Eliminar

Nombre del paciente: N° de identificación Domicilio:

Sexo: Masculino Femenino Fecha de nacimiento: Teléfono celular:

Cancelar registro Guardar modificación Ocupación:

Buscar paciente: Ver datos Limpiar Tabla

Nº de identificación	Nombre	Sexo	Teléfono	Domicilio	Ocupación	fecha de nacimiento

Figura 2.17: Datos de registro de pacientes

La interfaz "evaluación del paciente 1" permite la entrada de datos para evaluar la antropometría del paciente, así como generar los resultados a la evaluación.

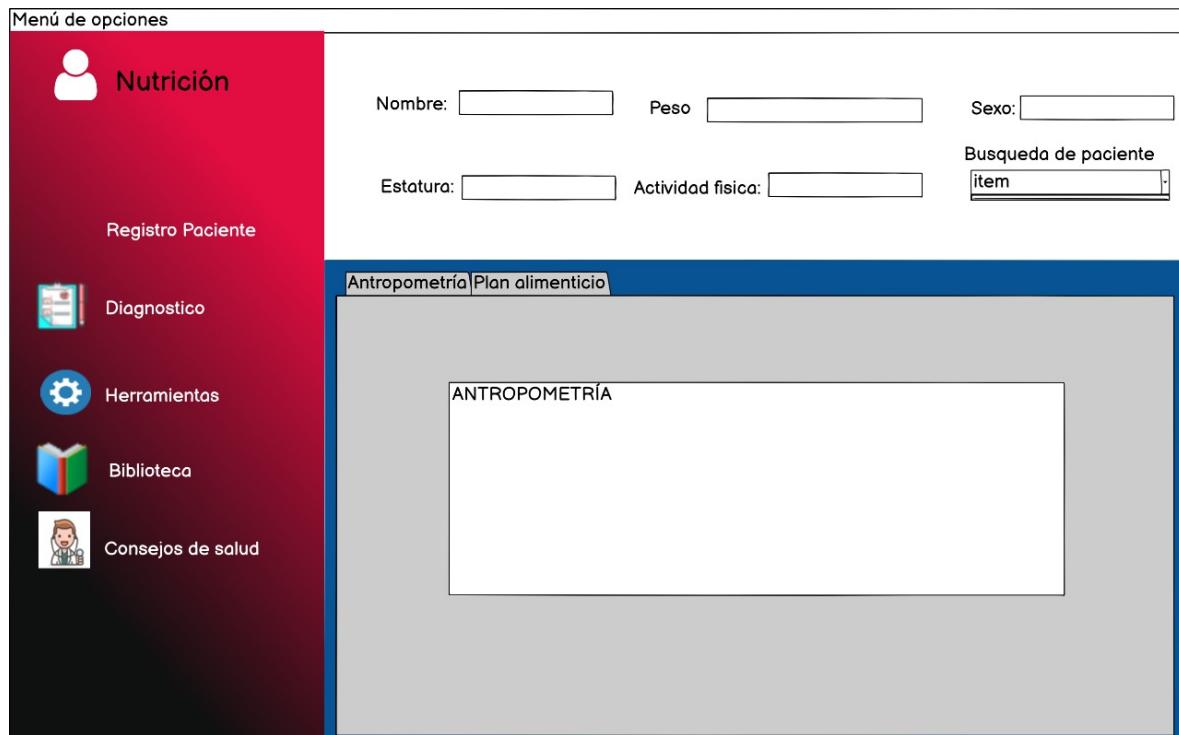


Figura 2.18: Evaluación del paciente

La interfaz "evaluación del paciente 2" permite la entrada de datos gestionar un plan alimenticio de acuerdo a las necesidades que el paciente tenga dado sus resultados en el apartado de antropometría.

Menú de opciones

Nutrición

Registro Paciente

Diagnóstico

Herramientas

Biblioteca

Consejos de salud

Nombre: _____ Peso: _____ Sexo: _____

Estatura: _____ Actividad física: _____ Busqueda de paciente
item

Antropometría | Plan alimenticio

EQUIVALEN	Desayuno	Colación matutin	Colación vespertin	Cena	Colación noctur

Figura 2.19: Evaluación del paciente 2

La interfaz "biblioteca de alimentos" mostrará una vista de los alimentos que hay, así como sus ingredientes y modo de preparación respectivamente.



Figura 2.20: Biblioteca de alimentos

Capítulo 3

Análisis

3.1. Diagramas UML

3.1.1. Diagramas de estructura

Diagrama de flujo de datos

Diagrama nivel 0

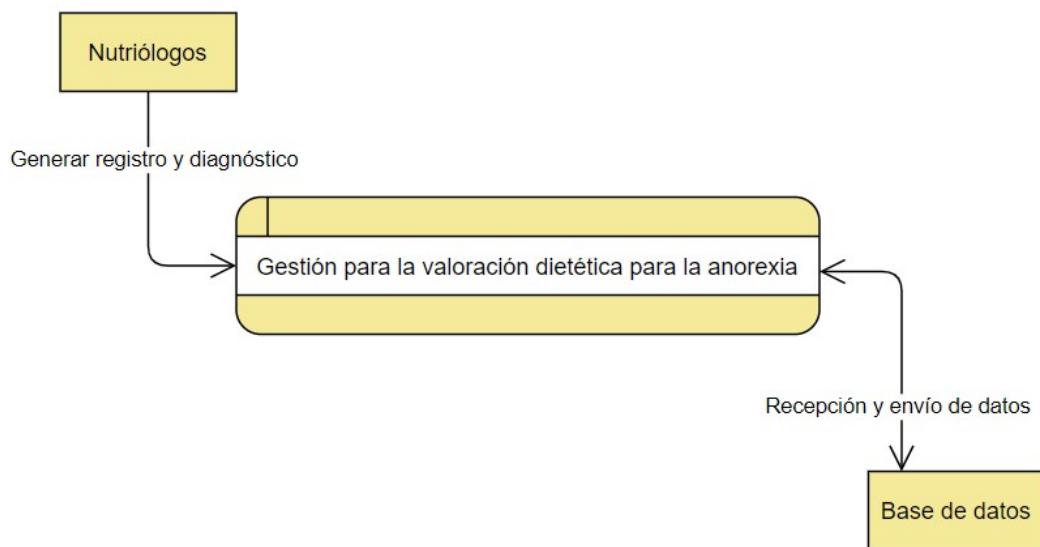


Figura 3.1: Diagrama nivel 0

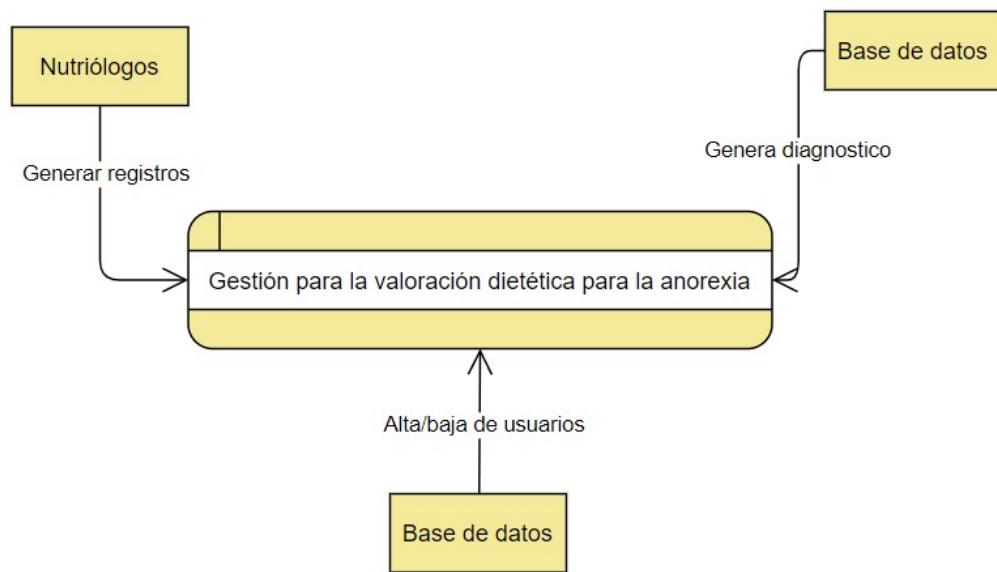
Diagrama nivel 1

Figura 3.2: Diagrama nivel 1

Diagrama de clases

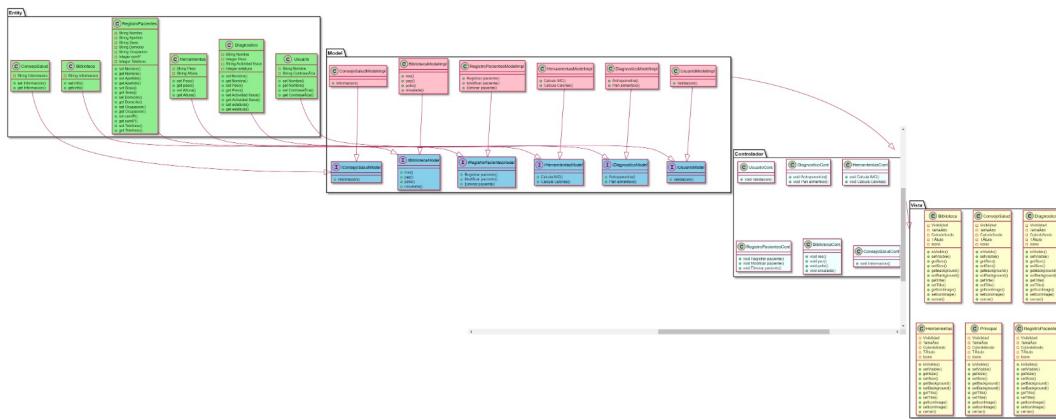


Figura 3.3: Diagrama de clases

Código PlantUML del diagrama de clases

```
@startuml
class Entity.Usuario #lightgreen{
    -String Nombre
    -String Contraseña
    +set Nombre()
    +get Nombre()
    +set Contraseña()
    +get Contraseña()
}
class Entity.Diagnostico #lightgreen{
    -String Nombre
    -Integer Peso
    -String Actividad fisica
    -Integer estatura
    +set Nombre()
    +get Nombre()
    +set Peso()
    +get Peso()
    +set Actividad fisica()
    +get Actividad fisica()
    +set estatura()
    +get estatura()
}
class Entity.Herramientas #lightgreen{
    -String Peso
    -String Altura
    +set Peso()
    +get peso()
    +set Altura()
    +get Altura()
}
class Entity.RegistroPacientes #lightgreen{
    -String Nombre
    -String Apellido
    -String Sexo
    -String Domicilio
    -String Ocupacion
    -Integer numIP
    -Integer Telefono
    +set Nombre()
```

```
+get Nombre()
+set Apellido()
+get Apellido()
+set Sexo()
+get Sexo()
+set Domicilio()
+get Domicilio()
+set Ocupacion()
+get Ocupacion()
+set numIP()
+get numIP()
+set Telefono()
+get Telefono()
}

class Entity.Biblioteca #lightgreen{
-String informacion
+set info()
+get info()
}

class Entity.ConsejoSalud #lightgreen{
-String Informacion
+set Informacion()
+get Informacion()
}

interface Model.IUsuarioModel #skyblue{
+Validacion()
}

interface Model.IDiagnosticoModel #skyblue{
+Antropometria()
+Plan alimenticio()
}

interface Model.IHerramientasModel #skyblue{
+Calcula IMC()
+Calcula Calorias()
}

interface Model.IRegistroPacientesModel #skyblue{
+Registrar paciente()
+Modificar paciente()
+Eliminar paciente()
}

interface Model.IBibliotecaModel #skyblue{
```

```
+res()
+pez()
+pollo()
+ensalada()
}
interface Model.IConsejoSaludModel #skyblue{
+Informacion()
}
class Model.UsuarioModelimpl #pink{
+Validacion()
}
class Model.DiagnosticoModelimpl #pink{
+Antropometria()
+Plan alimenticio()
}
class Model.HerramientasModelimpl #pink{
+Calcula IMC()
+Calcula Calorias()
}
class Model.RegistroPacientesModelimpl #pink{
+Registrar paciente()
+Modificar paciente()
+Eliminar paciente()
}
class Model.BibliotecaModelimpl #pink{
+res()
+pez()
+pollo()
+ensalada()
}
class Model.ConsejoSaludModelimpl #pink{
+Informacion()
}
class Controlador.UsuarioCont #azure{
+void Validacion()
}
class Controlador.DiagnosticoCont #azure{
+void Antropometria()
+void Plan alimenticio()
}
class Controlador.HerramientasCont #azure{
```

```
+void Calcula IMC()
+void Calcula Calorias()
}
class Controlador.RegistroPacientesCont #azure{
+void Registrar paciente()
+void Modificar paciente()
+void Eliminar paciente()
}
class Controlador.BibliotecaCont #azure{
+void res()
+void pez()
+void pollo()
+void ensalada()
}
class Controlador.ConsejoSaludCont #azure{
+void Informacion()
}
class Vista.Biblioteca{
-Visibilidad
-Tamaño
-Colordefondo
-Título
-Icono
+isVisible()
+setVisible()
+getSize()
+setSize()
+getBackground()
+setBackground()
+getTitle()
+setTitle()
+getIconImage()
+setIconImage()
+cerrar()
}
class Vista.ConsejoSalud{
-Visibilidad
-Tamaño
-Colordefondo
-Título
-Icono
```

```
+isVisible()
+setVisible()
+getSize()
+setSize()
+getBackground()
+setBackground()
+getTitle()
+setTitle()
+getIconImage()
+setIconImage()
+cerrar()
}
class Vista.Diagnostico{
-Visibilidad
-Tamaño
-Colordefondo
-Título
-Icono
+isVisible()
+setVisible()
+getSize()
+setSize()
+getBackground()
+setBackground()
+getTitle()
+setTitle()
+getIconImage()
+setIconImage()
+cerrar()
}
class Vista.Herramientas{
-Visibilidad
-Tamaño
-Colordefondo
-Título
-Icono
+isVisible()
+setVisible()
+getSize()
+setSize()
+getBackground()
```

```
+setBackground()
+getTitle()
+setTitle()
+getIconImage()
+setIconImage()
+cerrar()
}

class Vista.Principal{
-Visibilidad
-Tamaño
-Colordefondo
-Título
-Icono
+isVisible()
+setVisible()
+getSize()
+setSize()
+getBackground()
+setBackground()
+getTitle()
+setTitle()
+getIconImage()
+setIconImage()
+cerrar()
}

class Vista.RegistroPacientes{
-Visibilidad
-Tamaño
-Colordefondo
-Título
-Icono
+isVisible()
+setVisible()
+getSize()
+setSize()
+getBackground()
+setBackground()
+getTitle()
+setTitle()
+getIconImage()
+setIconImage()
```

```
+cerrar()
}
Entity.Usuario --|> Model.IUsuarioModel
Entity.Diagnostico --|> Model.IDiagnosticoModel
Entity.RegistroPacientes --|> Model.IRegistroPacientesModel
Entity.Biblioteca --|> Model.IBibliotecaModel
Entity.ConsejoSalud --|> Model.IConsejoSaludModel
Entity.Herramientas --|> Model.IHerramientasModel
Model.UsuarioModelimpl --|> Model.IUsuarioModel
Model.DiagnosticoModelimpl --|> Model.IDiagnosticoModel
Model.RegistroPacientesModelimpl --|> Model.IRegistroPacientesModel
Model.BibliotecaModelimpl --|> Model.IBibliotecaModel
Model.ConsejoSaludModelimpl --|> Model.IConsejoSaludModel
Model.HerramientasModelimpl --|> Model.IHerramientasModel
Model --|> Controlador
Controlador --|> Vista
@enduml
```

Diagrama de objetos

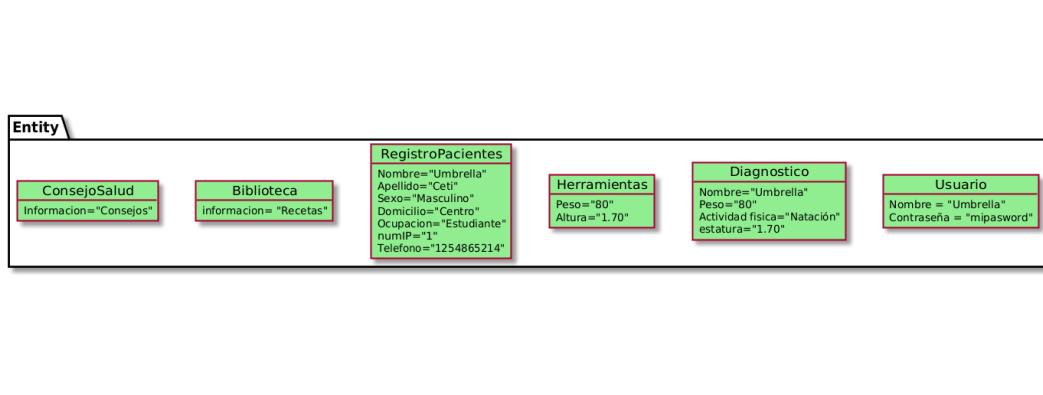


Figura 3.4: Diagrama de objetos

Código PlantUML del diagrama de objetos

```
@startuml
package Entity{
    object Usuario #lightgreen
    Usuario: Nombre = "Umbrella"
    Usuario: Contraseña = "mipassword"
}
package Entity{
    object Diagnostico #lightgreen
    Diagnostico: Nombre="Umbrella"
    Diagnostico: Peso="80"
    Diagnostico: Actividad fisica="Natación"
    Diagnostico: estatura="1.70"
}
package Entity{
    object Herramientas #lightgreen
    Herramientas: Peso="80"
    Herramientas: Altura="1.70"
}
package Entity{
    object RegistroPacientes #lightgreen
    RegistroPacientes: Nombre="Umbrella"
    RegistroPacientes: Apellido="Ceti"
    RegistroPacientes: Sexo="Masculino"
    RegistroPacientes: Domicilio="Centro"
    RegistroPacientes: Ocupacion="Estudiante"
    RegistroPacientes: numIP="1"
    RegistroPacientes: Telefono="1254865214"
}
package Entity{
    object Biblioteca #lightgreen
    Biblioteca: informacion= "Recetas"
}
package Entity{
    object ConsejoSalud #lightgreen
    ConsejoSalud: Informacion="Consejos"
}
@enduml
```

Diagrama de componenetes

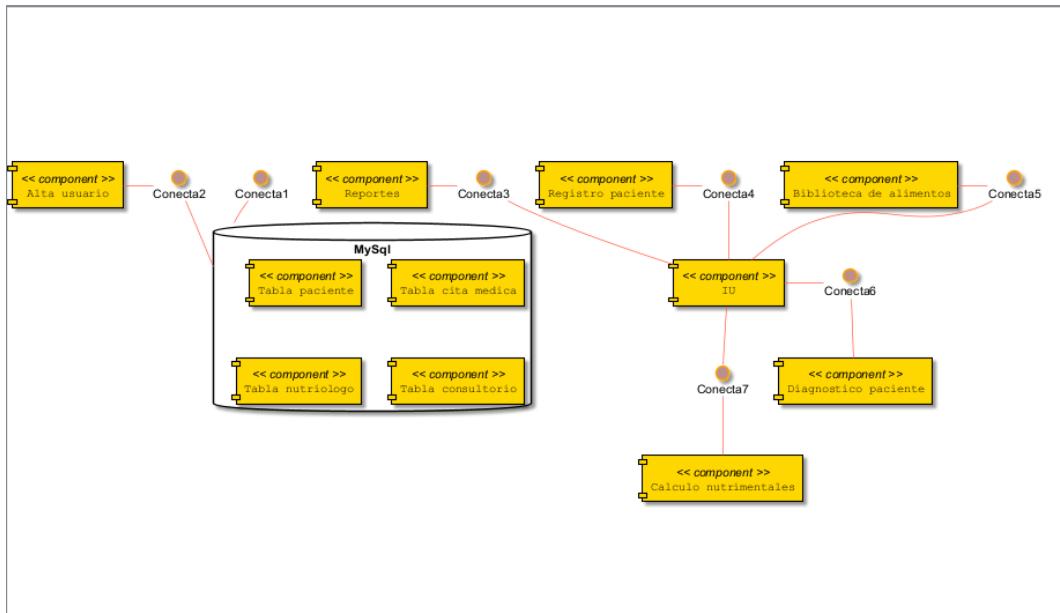


Figura 3.5: Diagrama de componenetes

Código PlantUML del Diagrama de componenetes

```

@startuml
skinparam interface {
    backgroundColor RosyBrown
    borderColor orange
}
skinparam component {
    FontSize 13
    BackgroundColor<<Apache>> Red
    BorderColor<<Apache>> #FF6655
    FontName Courier
    BorderColor black
    BackgroundColor gold
    ArrowFontName Impact
    ArrowColor #FF6655
    ArrowFontColor #777777
}
() "Conecta1" as DA
() "Conecta2" as DA
() "Conecta3" as DA
() "Conecta4" as DA
() "Conecta5" as DA
() "Conecta6" as DA
() "Conecta7" as DA

Conecta1 -- [ MySql ]
[Alta usuario]<< component >> - Conecta2
Conecta2 -- [ MySql ]
database " MySql " {
    [Tabla paciente]<< component >>
    [Tabla cita medica]<< component >>
    [Tabla nutriologo]<< component >>
    [Tabla consultorio]<< component >>
}

[Reportes]<< component >> - Conecta3
Conecta3-- [ IU ]
[ IU ]<< component >>
[ Registro paciente]<< component >> - Conecta4
Conecta4-- [ IU ]

```

```
[Biblioteca de alimentos]<< component >> - Conecta5
Conecta5--[IU]
[IU]-Conecta6
[Diagnostico paciente]<< component >>
Conecta6 --[Diagnostico paciente]
[Calculo nutrimentales]<< component >>
[IU]--Conecta7
Conecta7 -- [Calculo nutrimentales]
@enduml
```

Diagrama de despliegue

Diagrama de actividades

Diagrama de actividades del login

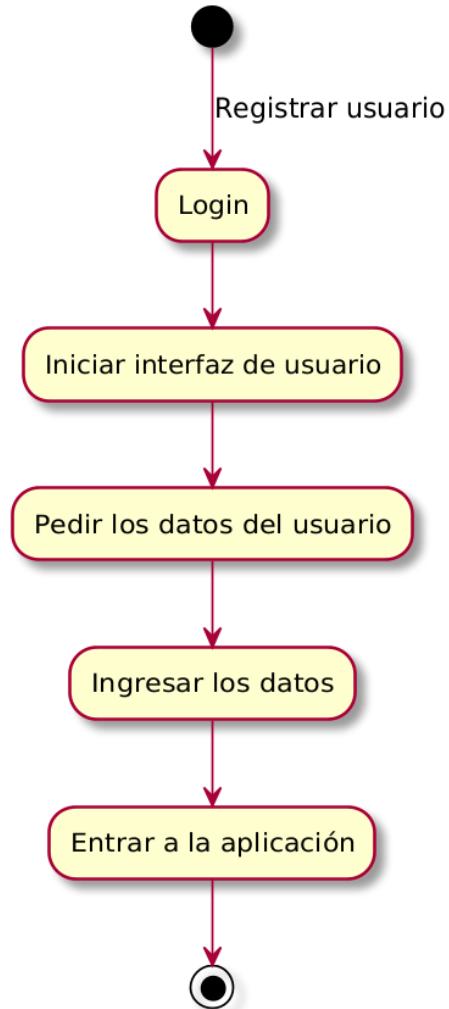


Figura 3.6: Diagrama de actividades del login

Código PlantUML del diagrama de actividades: login

```
@startuml
(*)--> [Registrar usuario] "Login"
--> "Iniciar interfaz de usuario"

--> "Pedir los datos del usuario"

--> "Ingresar los datos"
--> Entrar a la aplicación
--> (*)
@enduml
```

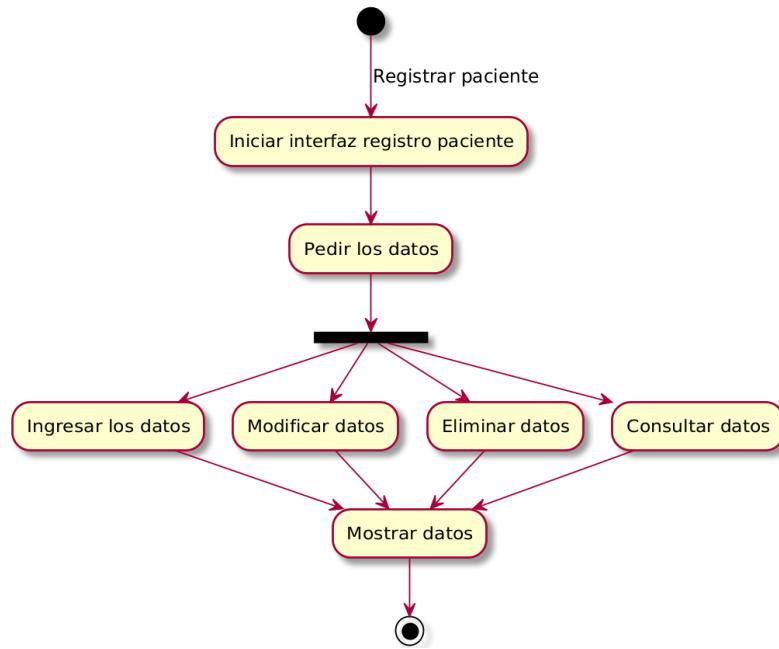
Diagrama de actividades del Registro paciente

Figura 3.7: Diagrama de actividades del Registro paciente

Código PlantUML del diagrama de actividades: Registro paciente

```
@startuml
(*)--> [Registrar paciente] "Iniciar interfaz registro paciente"
--> "Pedir los datos"
--> ===B1===
--> "Ingresar los datos"
--> "Mostrar datos"
==B1== --> "Modificar datos"
--> "Mostrar datos"
==B1== --> "Eliminar datos"
--> "Mostrar datos"
==B1== --> "Consultar datos"
--> "Mostrar datos"
--> (*)
@enduml
```

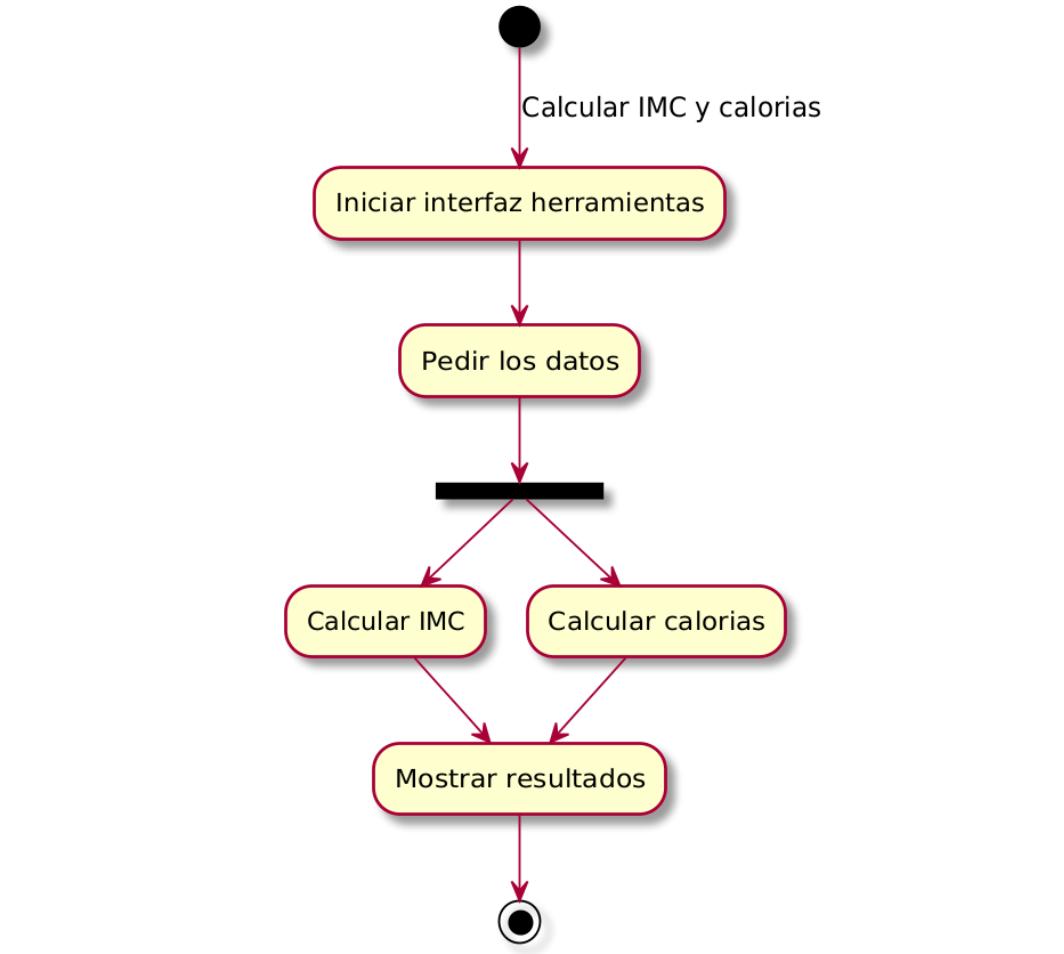
Diagrama de actividades de Herramientas

Figura 3.8: Diagrama de actividades de Herramientas

Código PlantUML del diagrama de actividades: Herramientas

```
@startuml
(*)--> [Calcular IMC y calorías] "Iniciar interfaz herramientas"
--> "Pedir los datos"
--> ===B1===
--> "Calcular IMC"
--> "Mostrar resultados"
==B1== --> "Calcular calorías"
--> "Mostrar resultados"
--> (*)
@enduml
```

Diagrama de actividades de Diagnóstico

Figura 3.9: Diagrama de actividades de Diagnóstico

Código PlantUML del diagrama de actividades: Diagnóstico

```
(*)--> [Registrar datos clinicos] "Iniciar interfaz diagnostico"  
--> "Pedir los datos"  
--> "Calcular datos clinicos"  
--> "Mostrar resultados clinicos"  
--> "Diseñar plan alimenticio"  
--> (*)  
@enduml
```

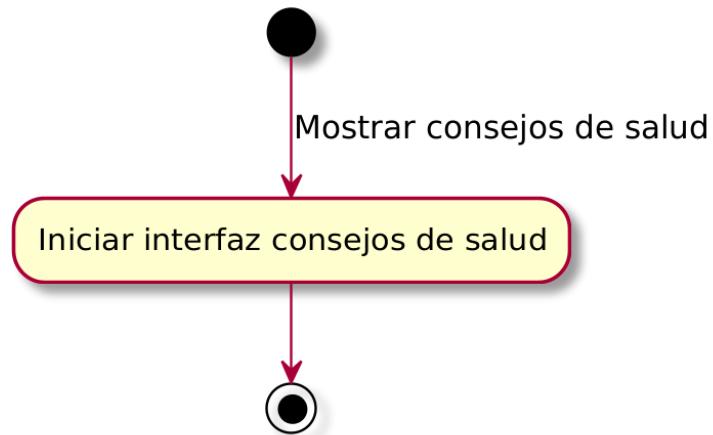
Diagrama de actividades de Consejos de salud

Figura 3.10: Diagrama de actividades de Consejos de salud

Código PlantUML del diagrama de actividades: Consejos de salud

```
@startuml
(*)--> [Mostrar consejos de salud] "Iniciar interfaz consejos de salud"
--> (*)
@enduml
```

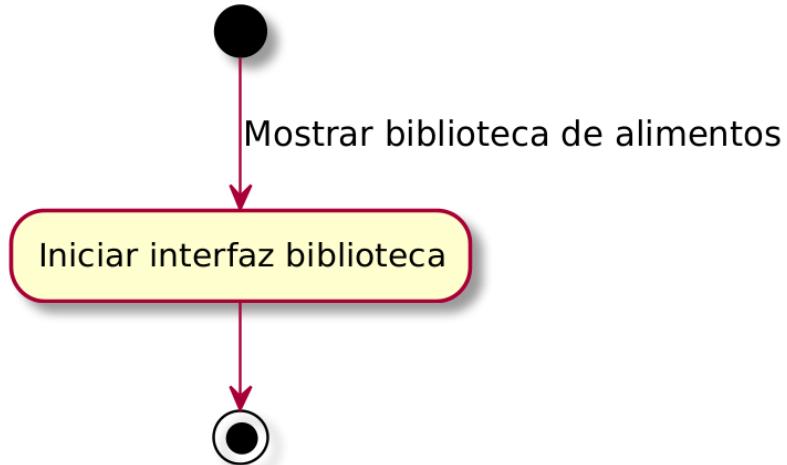
Diagrama de actividades de Biblioteca

Figura 3.11: Diagrama de actividades de Biblioteca

Código PlantUML del diagrama de actividades: Biblioteca

```
@startuml
(*)--> [Mostrar biblioteca de alimentos] "Iniciar interfaz biblioteca"
--> (*)
@enduml
```

3.1.2. Diagramas de comportamientos

Diagrama de caso de uso

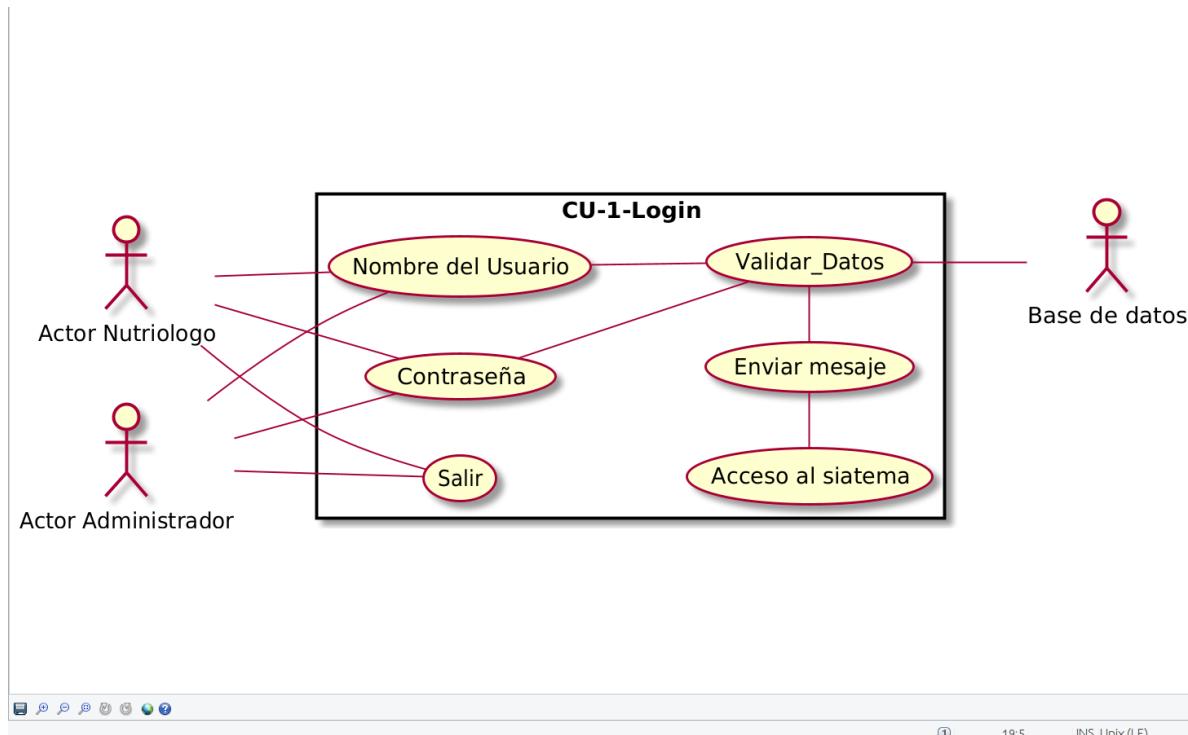


Figura 3.12: Caso de uso login

Código PlantUML del diagrama de caso de uso: login

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Actor Administrador" as D
actor "Actor Nutriologo" as N
actor "Base de datos" as BD
rectangle CU-1-Login{
    usecase "Nombre del Usuario" as Nom_usu
    usecase "Contraseña" as Con
    usecase "Salir" as S
    :Nom_usu: -- (Validar_Datos)
    :Con: --(Validar_Datos)
    usecase "Enviar mensaje" as m
    (m)-(Validar_Datos)
    usecase " Acceso al sistema" as a_s
    (a_s)-(m)
    N - (S)
    D - (S)
    N - (Con)
    D - (Con)
    N - (Nom_usu)
    D - (Nom_usu)
    (Validar_Datos) -- :BD:
}
@enduml
```

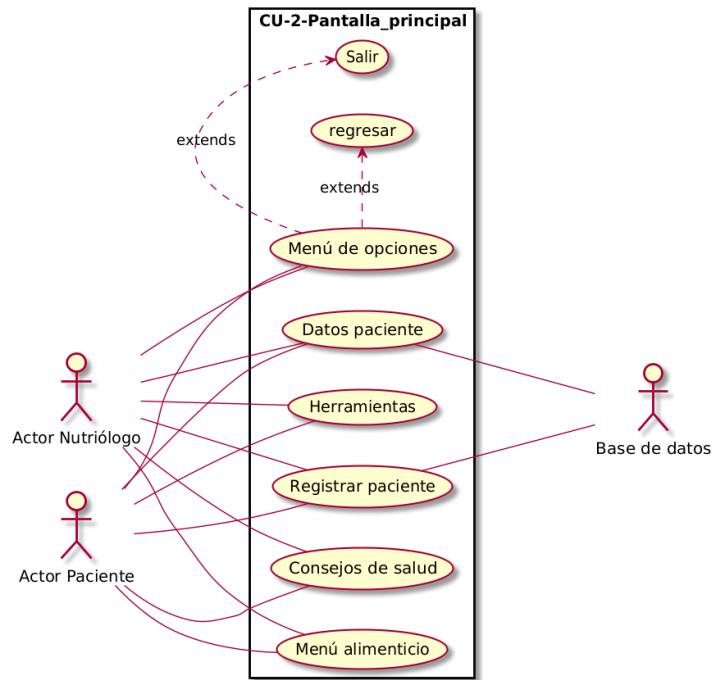


Figura 3.13: Caso de uso pantalla principal

Código PlantUML del diagrama de caso de uso: Pantalla principal

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Actor Nutriólogo" as N
actor "Actor Paciente" as P
actor "Base de datos" as BD
rectangle CU-2-Pantalla_principal{
    usecase "Registrar paciente" as rp
    N -- (rp)
    P -- (rp)
    (rp) --- :BD:
    usecase "Datos paciente" as dp
    N -- (dp)
    P -- (dp)
    (dp) --- :BD:
    usecase "Consejos de salud" as cs
    N -- (cs)
    P -- (cs)
    usecase "Herramientas" as h
    N -- (h)
    P -- (h)
    usecase "Menú alimenticio" as ma
    N -- (ma)
    P -- (ma)
    usecase "Menú de opciones" as mo
    N -- (mo)
    P -- (mo)
    (mo) .> (Salir) : extends
    (mo) .> (regresar) : extends
}
@enduml
```

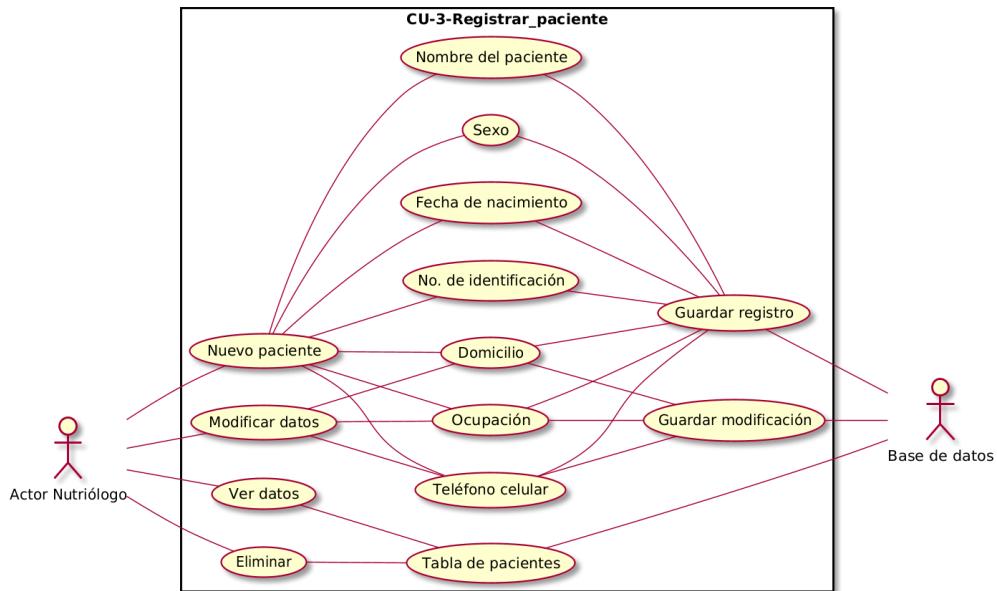


Figura 3.14: Caso de uso registro paciente

Código PlantUML del diagrama de caso de uso: registro paciente

```

@startuml
left to right direction
skinparam packageStyle rectangle
actor "Actor Nutriólogo" as N
actor "Base de datos" as BD
rectangle CU-3-Registrar_paciente{
    usecase "Nuevo paciente" as np
    usecase "Modificar datos" as md
    usecase "Ver datos" as vd
    usecase "Eliminar" as e

    (N)--(np)
    (N)--(md)
    (N)--(vd)
    (N)--(e)

    :np: --(Nombre del paciente)
    :np: --(No. de identificación)
    :np: --(Fecha de nacimiento)
    :np: --(Sexo)
    :np: --(Domicilio)
    :np: --(Teléfono celular)
    :np: --(Ocupación)

    :md: --(Domicilio)
    :md: --(Teléfono celular)
    :md: --(Ocupación)

    :vd: --(Tabla de pacientes)
    :e: --(Tabla de pacientes)

    :Nombre del paciente: --(Guardar registro)
    :No. de identificación: --(Guardar registro)
    :Fecha de nacimiento: --(Guardar registro)
    :Sexo: --(Guardar registro)
    :Domicilio: --(Guardar registro)
    :Teléfono celular: --(Guardar registro)
    :Ocupación: --(Guardar registro)

    :Domicilio: --(Guardar modificación)
}

```

```
:Teléfono celular: --(Guardar modificación)
:Ocupación: --(Guardar modificación)

(Guardar registro) -- BD
(Guardar modificación) -- BD
(Tabla de pacientes) -- BD
}

@enduml
```

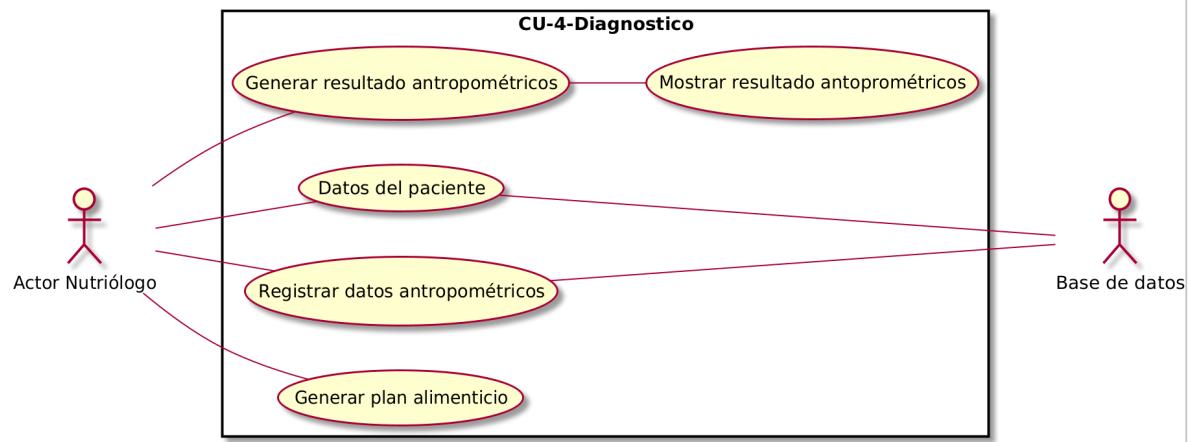


Figura 3.15: Caso de uso diagnóstico

Código PlantUML del diagrama de caso de uso: diagnóstico

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Actor Nutriólogo" as N
actor "Base de datos" as BD
rectangle CU-4-Diagnostico{
    usecase "Datos del paciente" as dp
    usecase "Registrar datos antropométricos" as rd
    usecase "Generar resultado antropométricos" as gr
    usecase "Generar plan alimenticio" as gp

    (N)--(dp)
    (N)--(rd)
    (N)--(gr)
    (N)--(gp)

    :gr: --(Mostrar resultado antropométricos)

    (dp) --- BD
    (rd) --- BD
}
@enduml
```

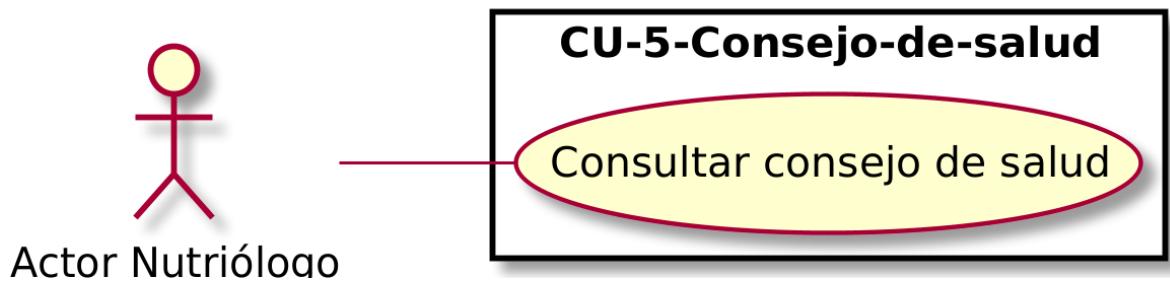


Figura 3.16: Caso de uso consejos de salud

Código PlantUML del diagrama de caso de uso: condejos de salud

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Actor Nutriólogo" as N
rectangle CU-5-Consejo-de-salud{
    usecase "Consultar consejo de salud" as cs
    (N)--(cs)
}
@enduml
```

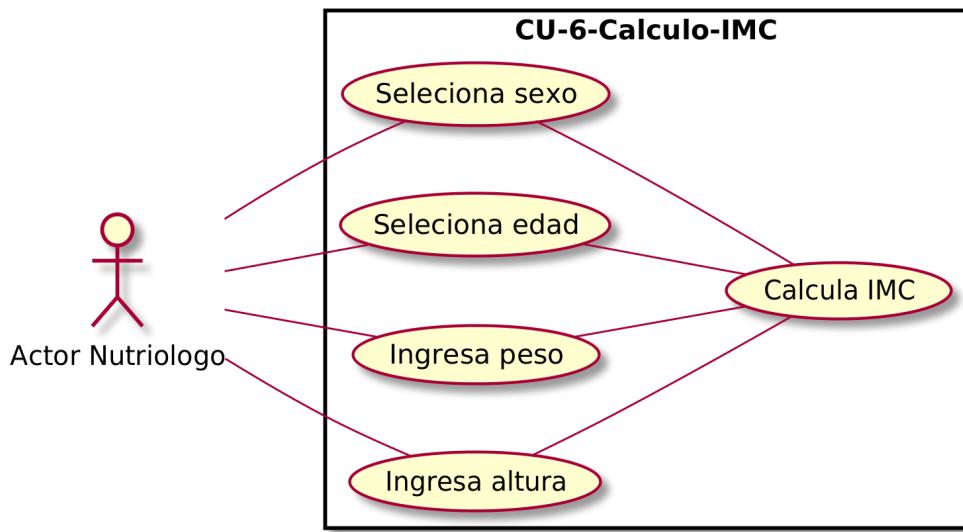


Figura 3.17: Caso de uso cálculo de IMC

Código PlantUML del diagrama de caso de uso:Cálculo de IMC

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Actor Nutriologo" as N
rectangle CU-6-Calculo-IMC{
    usecase "Selecciona sexo" as ss
    usecase "Selecciona edad" as se
    usecase "Ingresa peso" as ip
    usecase "Ingresa altura" as ia
    (N)--(ss)
    (N)--(se)
    (N)--(ip)
    (N)--(ia)
    :ss: -- (Calcula IMC)
    :se: --(Calcula IMC)
    :ip: -- (Calcula IMC)
    :ia: --(Calcula IMC)
}
@enduml
```

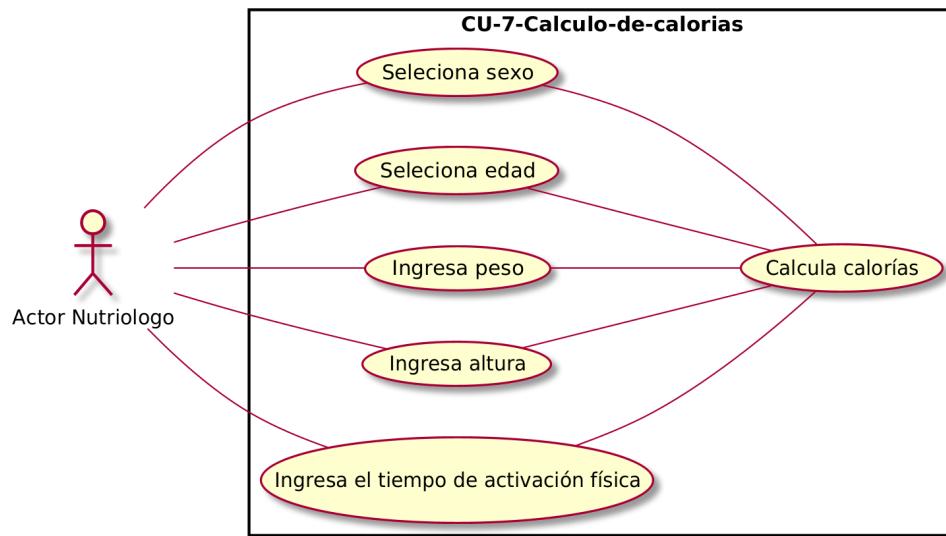


Figura 3.18: Caso de uso cálculo de calorías

Código PlantUML del diagrama de caso de uso:Cálculo de calorías

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Actor Nutriologo" as N
rectangle CU-7-Calculo-de-calorias{
    usecase "Selecciona sexo" as ss
    usecase "Selecciona edad" as se
    usecase "Ingresa peso" as ip
    usecase "Ingresa altura" as ia
    usecase "Ingresa el tiempo de activación física" as it
    (N)--(ss)
    (N)--(se)
    (N)--(ip)
    (N)--(ia)
    (N)--(it)
    :ss: --(Calcula calorías)
    :se: --(Calcula calorías)
    :ip: --(Calcula calorías)
    :ia: --(Calcula calorías)
    :it: --(Calcula calorías)
}
@enduml
```

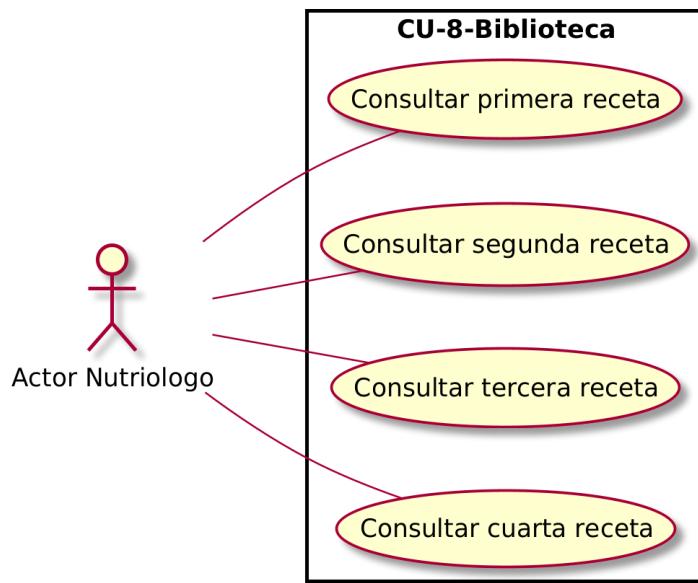


Figura 3.19: Caso de uso biblioteca de alimentos

Código PlantUML del diagrama de caso de uso: biblioteca de alimentos

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor "Actor Nutriologo" as N
rectangle CU-8-Biblioteca{
    usecase "Consultar primera receta" as c1
    usecase "Consultar segunda receta" as c2
    usecase "Consultar tercera receta" as c3
    usecase "Consultar cuarta receta" as c4

    (N)--(c1)
    (N)--(c2)
    (N)--(c3)
    (N)--(c4)
}
@enduml
```

Documentación de caso de uso

Id:	CU-1		
Nombre:	Login		
Creado por:	JPH	Actualizado por:	
Fecha creación:	30/11/2021		
Actores:	Nutriologo, Administrador		
Descripción:	El administrador y nutriologo iniciaran sesión para acceder a la ventana principal de la aplicación.		
Disparador:	Inicio sesión		
Precondiciones:	Ingresar usuario y contraseña para validar el perfil del usuario en la aplicación.		
Pos condición:	N/A		
Flujo normal:	<ol style="list-style-type: none"> 1. Ingresar el usuario 2. Ingresar la contraseña 3. El sistema verificará que los datos ingresados sean correctos con el registro de la base de datos. 4. Presionar el botón inicio. <p>4.1. Ir a caso de uso 1.</p>		
Flujos Alternas:	N/A		
Include:	N/A		
Frecuencia de uso:	Muy frecuente		
Requerimientos especiales:	Contraseña		
Supuestos:	N/A		
Issues y notas:	N/A		

Figura 3.20: Documentación login

Diagrama de estados

Id:	CU-2		
Nombre:	Pantalla principal		
Creado por:	ARR	Actualizado por:	
Fecha creación:	01/11/2021		
Actores:	Nutriologo, Administrador, Paciente.		
Descripción:	<p>El nutriologo y administrador teniendo acceso al sistema podrán evaluar y capturar información del paciente en tiempo.</p> <p>El paciente tendrá acceso al uso de herramientas, biblioteca y consejos de salud brindados por la aplicación.</p>		
Disparador:	<p>Seleccionar alguna de estas opciones:</p> <ol style="list-style-type: none"> 1. Evaluar 2. Capturar 3. Consejo de salud 4. Herramientas 5. Menú alimentación 6. Menú opciones 		
Precondiciones:	<p>Opción 1,2,6 administrador y nutriologo</p> <p>Opción 3,4,5,6 pacientes</p>		
Pos condición:	N/A		
Flujo normal:	<p>Administrador y paciente hacer:</p> <ol style="list-style-type: none"> 1. Presionar botón Evaluación paciente <ol style="list-style-type: none"> 1.1. Ir a caso de uso 3. 2. Presionar botón Datos paciente. <ol style="list-style-type: none"> 2.2. Ir a caso de uso 4. <p>Paciente hacer:</p> <ol style="list-style-type: none"> 1. Presionar botón Consejos de salud <ol style="list-style-type: none"> 1.1. Ir a caso de uso 5. 2. Presionar botón Herramientas. <ol style="list-style-type: none"> 2.2. Ir a caso de uso 6 3. Presionar botón Biblioteca <ol style="list-style-type: none"> 3.3. Ir a caso de uso 7 4. Para salir del programa, presionar botón menú de opciones <ol style="list-style-type: none"> 4.4. Presionar botón salir. 		
Flujos Alternas:	N/A		
Include:	N/A		
Frecuencia de uso:	Muy frecuente		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y notas:	N/A		

Figura 3.21: Documentación pantalla principal

Id:	CU-3		
Nombre:	Registrar pacientes		
Creado por:	MVR	Actualizado por:	
Fecha creación:	02/12/2021		
Actores:	Nutriologo		
Descripción:	Al nutriologo se le mostrará la ventana Registrar paciente, posteriormente hará un registro desde la acción nuevo. Se podrán hacer modificaciones, ver los datos y eliminar los datos por cada registro que haya.		
Disparador:	Presionar botón Registrar paciente En la ventana de registro: <ol style="list-style-type: none">1. Presionar botón Nuevo pacientes2. Presionar botón Modificar datos3. Presionar botón Ver datos4. Presionar botón Eliminar pacientes		
Precondiciones:	Ser nutriologo		
Pos condición:	N/A		
Flujo normal:	<ol style="list-style-type: none">1. Presionar botón Nuevo pacientes<ol style="list-style-type: none">1.1. Ingresar nombre del pacientes1.2. Ingresar No. Identificación1.3. Ingresar fecha de nacimiento1.4. Ingresar sexo1.5. Ingresar domicilio1.6. Ingresar No. teléfono1.7. Ingresar ocupación.2. Presionar botón Guardar.3. En caso de querer modificar, presionar botón Modificar.<ol style="list-style-type: none">3.1. Ingresar domicilio3.2. Ingresar No. teléfono3.3. Ingresar ocupación4. Presionar botón Guardar modificación5. Presionar botón ver datos para ver los registros guardados (los datos se visualizaran en una tabla)6. En caso de querer eliminar algún dato, seleccionar el datos y presionar el botón Eliminar.		
Flujos Alternas:	N/A		
Include:	N/A		
Frecuencia de uso:	Muy frecuente		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y notas:	N/A		

Figura 3.22: Documentación registro paciente

Id:	CU-4		
Nombre:	Diagnóstico paciente		
Creado por:	MHVR	Actualizado por:	
Fecha creación:	04/12/2021		
Actores:	Nutriologo		
Descripción:	El nutriologo en la ventana de diagnostico, tendrá el privilegio de obtener los datos del paciente, al igual que generar un diagnostico		
Disparador:	Seleccionar la opción diagnóstico de paciente.		
Precondiciones:	Ser nutriologo		
Pos condición:	N/A		
Flujo normal:	<p>En la ventana de Diagnostico de paciente, el nutriologo registrara la información del paciente al igual que generar un plan alimenticio.</p> <ol style="list-style-type: none"> 1. Ingresar datos del paciente <ol style="list-style-type: none"> 1.1. Nombre 1.2. Peso 1.3. Sexo 1.4. Estatura 1.5. Actividad física 2. Presionar el apartado de datos antropometricos para ingresar los datos que se solicita. 3. En el apartado de datos antropometricos se visualizara los resultados del diagnostico antropometrico. 4. Presionar el apartado de generar plan alimenticio, para visualizar el plan a seguir en su alimentación. 		
Flujos Alternas:	N/A		
Include:	N/A		
Frecuencia de uso:	Muy frecuente		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y notas:	N/A		

Figura 3.23: Documentación diagnóstico

Código PlantUML del diagrama de estados:login

```
@startuml
hide empty description
[*]--> Estado1
Estado1 : iniciar interfaz de login
Estado1 --> Estado2
Estado2 : Pedir los datos del usuario
Estado2 --> Estado3
Estado3 : Ingresar los datos
Estado3 --> Estado4
Estado4 : Entrar a la aplicación
Estado4 --> [*]
@enduml
```

Codigo PlantUML del diagrama de estados: registro paciente

```
@startuml
state fork_state <<fork>>
[*] --> Estado1
Estado1 --> fork_state
Estado1 : Iniciar interfaz registro paciente
fork_state --> Estado2
Estado2 : Ingresar los datos
fork_state --> Estado3
Estado3 : Modificar datos
fork_state --> Estado4
Estado4 : Eliminar datos
fork_state --> Estado5
Estado5 : Consultar datos
Estado2 --> Estado6
Estado6 : Mostrar datos
Estado3 --> Estado6
Estado4 --> Estado6
Estado5 --> Estado6
Estado6 --> [*]
@enduml
```

Código PlantUML del diagrama de estados: herramientas

```
@startuml
state fork_state <<fork>>
[*] --> Estado1
Estado1 --> Estado2
Estado1 : Iniciar interfaz herramientas
Estado2 --> fork_state
Estado2 : Pedir los datos
fork_state --> Estado3
Estado3 : Calcular IMC
fork_state --> Estado4
Estado4 : Calcular calorías
Estado3 --> Estado5
Estado5 : Mostrar resultados
Estado4 --> Estado5
Estado5 --> [*]
@enduml
```

Código PlantUML del diagrama de estados: Diagnóstico

```
@startuml
hide empty description
[*]--> Estado1
Estado1 : iniciar interfaz diagnostico
Estado1 --> Estado2
Estado2 : Pedir los datos
Estado2 --> Estado3
Estado3 : Calcular los datos clinicos
Estado3 --> Estado4
Estado4 : Mostrar resultados clinicos
Estado4 --> Estado5
Estado5 : Diseñar plan alimenticio
Estado5 --> [*]
@enduml
```

Código PlantUML del diagrama de estados: consejos salud

```
@startuml
hide empty description
[*]--> Estado1
Estado1 : Iniciar interfaz Consejos de salud
Estado1 --> [*]
@enduml
```

Codigo PlantUML del diagrama de estados: consejos salud

```
@startuml  
hide empty description  
[*]--> Estado1  
Estado1 : Iniciar interfaz Consejos de salud  
Estado1 --> [*]  
@enduml
```

Código PlantUML del diagrama de estados: biblioteca de alimentos

```
@startuml  
hide empty description  
[*]--> Estado1  
Estado1 : Iniciar interfaz biblioteca  
Estado1 --> [*]  
@enduml
```

Diagrama de secuencias

Código PlantUML del diagrama de secuencias: login

```
@startuml
actor Nutriologo #blue
participant Login #red
Nutriologo [#0000FF] ->0 Login : 1. Usuario y contraseña
participant BD #green
Login [#red] ->0 BD : 2. Conexion a BD
BD[#green] ->0 Login : 3. Resultado de validación
participant Interfaz #orange
Login [#red] ->0 Interfaz
Nutriologo [#red] ->0 Interfaz
@enduml
```

Codigo PlantUML del diagrama de secuencias: Pantalla principal

```
@startuml
actor Nutriologo #blue
participant Pantalla_principal #pink
Nutriologo [#0000FF] ->0 Pantalla_principal : 1. Usuario y contraseña
Nutriologo [#0000FF] ->0 Pantalla_principal : 2. Diagnostico
Nutriologo [#0000FF] ->0 Pantalla_principal : 3. Consejos de salud
Nutriologo [#0000FF] ->0 Pantalla_principal : 4. Biblioteca
Nutriologo [#0000FF] ->0 Pantalla_principal : 5. Menú de opciones
participant Interfaz #lightgreen
Pantalla_principal [#pink] ->0 Interfaz
Nutriologo[#blue] ->0 Interfaz
@enduml
```

Código PlantUML del diagrama de secuencias: Registro paciente

```
@startuml
actor Nutriologo #blue
participant Registro_paciente #pink
Nutriologo[#blue]->o Registro_paciente : 1. Agregar datos
Nutriologo[#blue]->o Registro_paciente : 2. Modificar datos
Nutriologo[#blue]->o Registro_paciente : 3. Ver datos
Nutriologo[#blue]->o Registro_paciente : 4. Eliminar datos
Registro_paciente[#pink]->o BD : Conexion a BD
participant BD
BD [#purple]->o Registro_paciente : Resultado del CRUD
participant Interfaz
Registro_paciente[#pink]->o Interfaz
Nutriologo[#blue]->o Interfaz
@enduml
```

Codigo PlantUML del diagrama de secuencias: Herramientas

```
@startuml
actor Nutriologo #blue
participant Calculo_de_IMC #pink
Nutriologo[#blue]->o Calculo_de_IMC : Seleccionar peso
Nutriologo[#blue]->o Calculo_de_IMC : Seleccionar edad
Nutriologo[#blue]->o Calculo_de_IMC : Ingresar peso
Nutriologo[#blue]->o Calculo_de_IMC : Ingresar altura
Calculo_de_IMC-> Interfaz : Mostrar resultado
Nutriologo[#blue]->o Interfaz
participant Interfaz
@enduml
```

Código PlantUML del diagrama de secuencias: Calculo de calorías

```
@startuml
actor Nutriologo #blue
participant Calculo_de_calorias #green
Nutriologo[#blue]->o Calculo_de_calorias : Seleccionar peso
Nutriologo[#blue]->o Calculo_de_calorias : Seleccionar edad
Nutriologo[#blue]->o Calculo_de_calorias : Ingresar peso
Nutriologo[#blue]->o Calculo_de_calorias : Ingresar altura
Nutriologo[#blue]->o Calculo_de_calorias : Ingresar tiempo de actividad fisica
Calculo_de_calorias[#green]->o Interfaz : Mostrar resultado
Nutriologo[#blue]->o Interfaz
participant Interfaz
@enduml
```

Codigo PlantUML del diagrama de secuencias: Calculo de calorias

```
@startuml
actor Nutriologo #blue
participant Calculo_de_calorias #green
Nutriologo[#blue]->o Calculo_de_calorias : Seleccionar peso
Nutriologo[#blue]->o Calculo_de_calorias : Seleccionar edad
Nutriologo[#blue]->o Calculo_de_calorias : Ingresar peso
Nutriologo[#blue]->o Calculo_de_calorias : Ingresar altura
Nutriologo[#blue]->o Calculo_de_calorias : Ingresar tiempo de actividad fisica
Calculo_de_calorias[#green]->o Interfaz : Mostrar resultado
Nutriologo[#blue]->o Interfaz
participant Interfaz
@enduml
```

Código PlantUML del diagrama de secuencias: consejo salud

```
@startuml
actor Nutriologo #blue
participant Consejos_de_salud #pink
Nutriologo[#blue]->o Consejos_de_salud : Ver consejos
Consejos_de_salud->o Interfaz
Nutriologo[#blue]->o Interfaz
participant Interfaz
@enduml
```

Codigo PlantUML del diagrama de secuencias: biblioteca

```
@startuml
actor Nutriologo #blue
participant Biblioteca #green
Nutriologo[#blue]->o Biblioteca : Busacar información
Biblioteca[#green]->o Interfaz : Mostrar primer receta
Biblioteca[#green]->o Interfaz : Mostrar segunda receta
Biblioteca[#green]->o Interfaz : Mostrar tercer receta
Biblioteca[#green]->o Interfaz : Mostrar cuarta receta
Nutriologo[#blue]->o Interfaz
participant Interfaz
@enduml
```

Diagrama de colaboración

Id:	CU-5		
Nombre:	Consejos de salud		
Creado por:	ARR	Actualizado por:	
Fecha creación:	02/12/2021		
Actores:	Paciente		
Descripción:	El paciente presionara el botón Consejos de salud para poder ingresar a la ventana.		
Disparador:	Presionar Consejos de salud		
Precondiciones:	Disponible para cualquier usuario		
Pos condición:	N/A		
Flujo normal:	En la ventana consejos de salud, el paciente solo visualizara la información.		
Flujos Alternas:	N/A		
Include:	N/A		
Frecuencia de uso:	Frecuente		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y notas:	N/A		

Figura 3.24: Documentación consejos de salud

Id:	CU-6		
Nombre:	Calculo de IMC		
Creado por:	JPH	Actualizado por:	
Fecha creación:	04/12/2021		
Actores:	Nutriologo		
Descripción:	El nutriologo en esta ventana, podra calcular el IMC de sus pacientes		
Disparador:	Seleccionar Calculo de IMC.		
Precondiciones:	Ser nutriologo		
Pos condición:	N/A		
Flujo normal:	<ol style="list-style-type: none"> 1. Presionar Calculo de IMC <ol style="list-style-type: none"> 1.1 Seleccionar sexo 1.2 Seleccionar edad 1.3 Ingresar peso 1.4 Ingresar altura.. 2. Presionar botón calcula IMC 		
Flujos Alternas:	N/A		
Include:	N/A		
Frecuencia de uso:	Frecuente		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y notas:	N/A		

Figura 3.25: Documentación cálculo de IMC

Id:	CU-7		
Nombre:	Calculo de calorías		
Creado por:	JPH	Actualizado por:	
Fecha creación:	04/12/2021		
Actores:	Nutriólogo		
Descripción:	El nutriólogo recopilará la información necesaria para calcular las calorías de consumo del paciente.		
Disparador:	Seleccionar cálculo de calorías.		
Precondiciones:	El nutriólogo conocerá el estado actual del paciente.		
Pos condición:	N/A		
Flujo normal:	<ol style="list-style-type: none"> 1. Presionar botón Calculo de calorías <ol style="list-style-type: none"> 1.1 Seleccionar sexo 1.2 Seleccionar edad 1.3 Ingresar peso 1.4 Ingresar altura.. 1.5 Ingresar ejercicio realizado 2. Presionar botón calculo de calorías 		
Flujos Alternas:	N/A		
Include:	N/A		
Frecuencia de uso:	Muy frecuente		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y notas:	N/A		

Figura 3.26: Documentación cálculo de calorías

Id:	CU-8		
Nombre:	Biblioteca		
Creado por:	ARR	Actualizado por:	
Fecha creación:	01/12/2021		
Actores:	Paciente		
Descripción:	El paciente obtendrá recetas de alimentos nutritivos.		
Disparador:	Seleccionar la opción biblioteca.		
Precondiciones:	Para todos los usuarios		
Pos condición:	N/A		
Flujo normal:	<ol style="list-style-type: none"> 1. Estando en la biblioteca de alimentos, el paciente podrá elegir el alimento que más deseé. 2. Una vez ejecutado el paso 1 se mostrará la información de los ingredientes y su modo de preparación. 		
Flujos Alternas:	N/A		
Include:	N/A		
Frecuencia de uso:	Frecuente		
Requerimientos especiales:	N/A		
Supuestos:	N/A		
Issues y notas:	N/A		

Figura 3.27: Documentación biblioteca de alimentos



Figura 3.28: Diagrama de estados: login

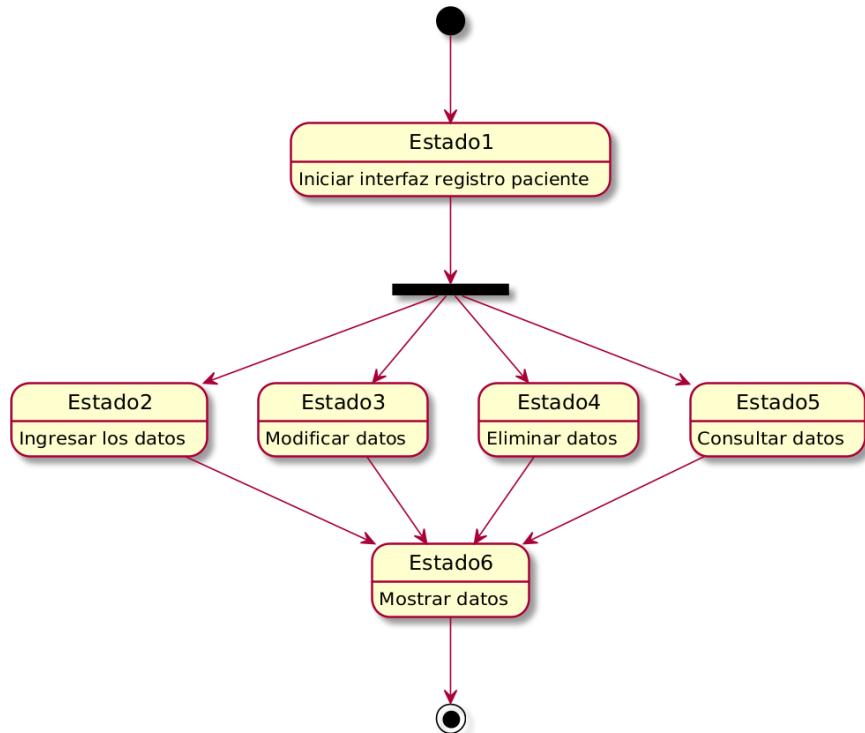


Figura 3.29: Diagrama de estados: registro paciente

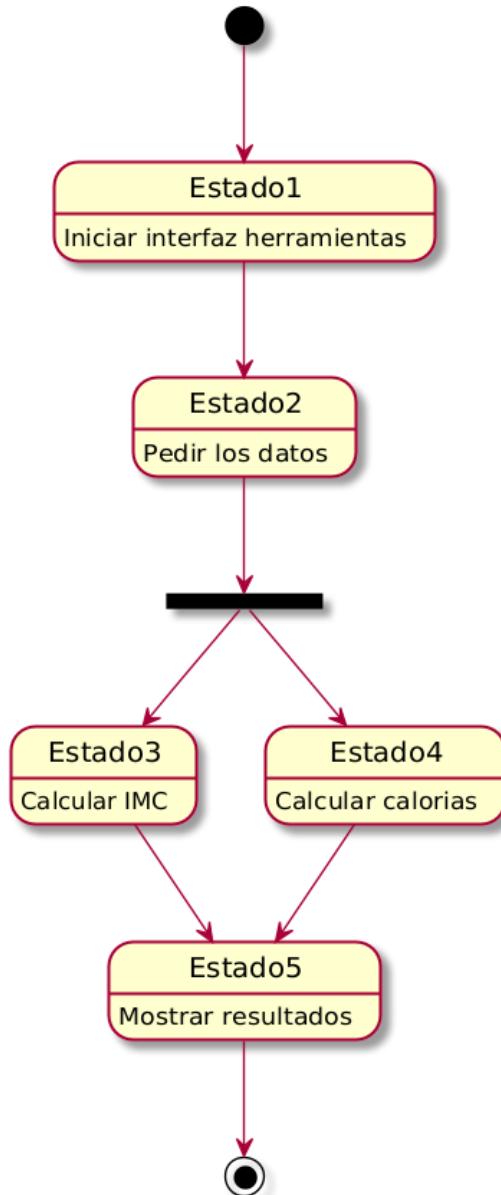


Figura 3.30: Diagrama de estados: herramientas

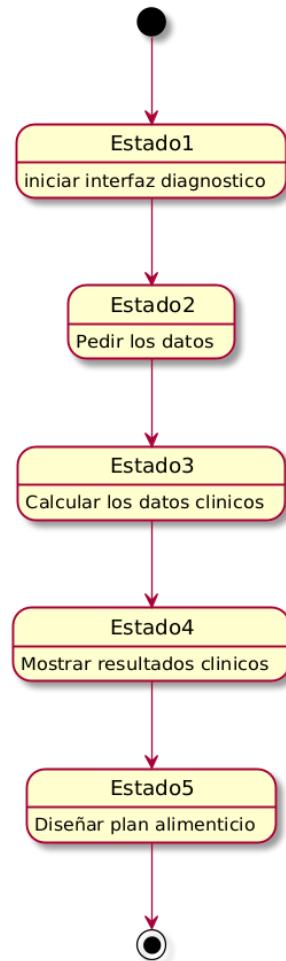


Figura 3.31: Diagrama de estados: diagnóstico

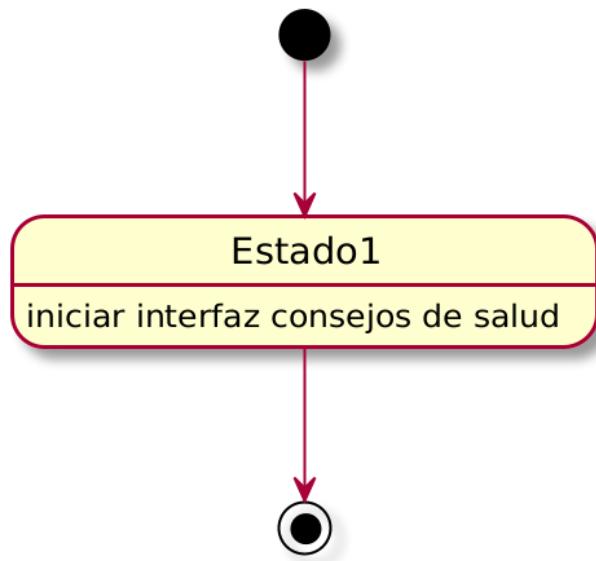


Figura 3.32: Diagrama de estados: consejos salud

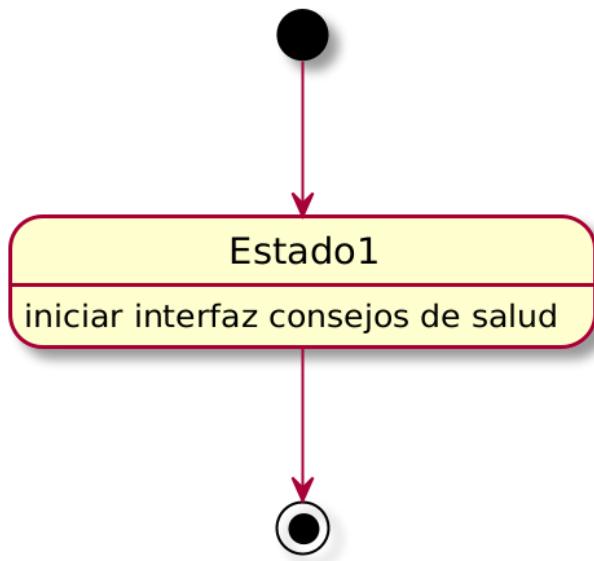


Figura 3.33: Diagrama de estados: consejos salud

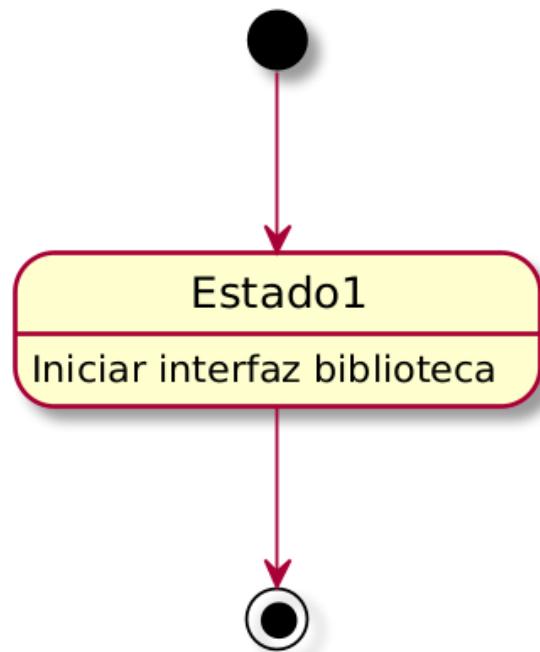


Figura 3.34: Diagrama de estados: biblioteca de alimentos

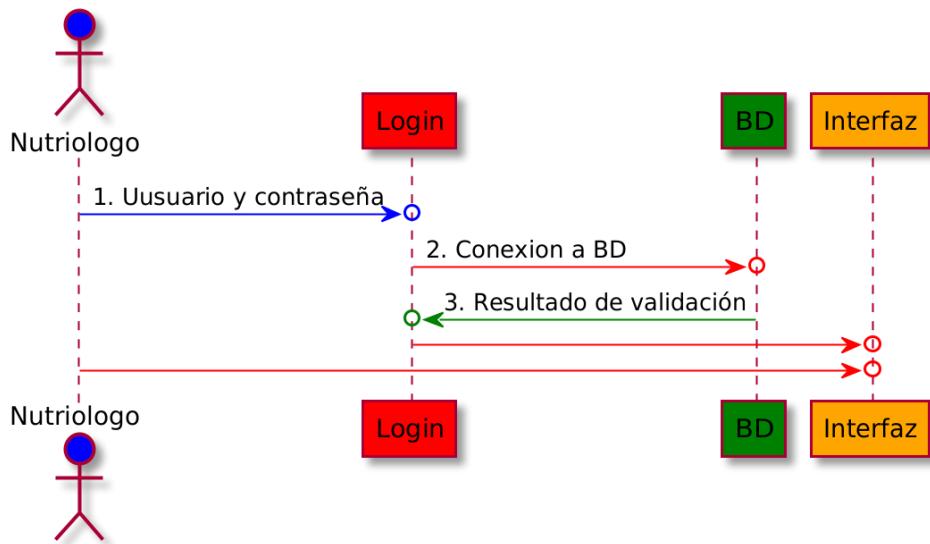


Figura 3.35: Diagrama de secuencias: login

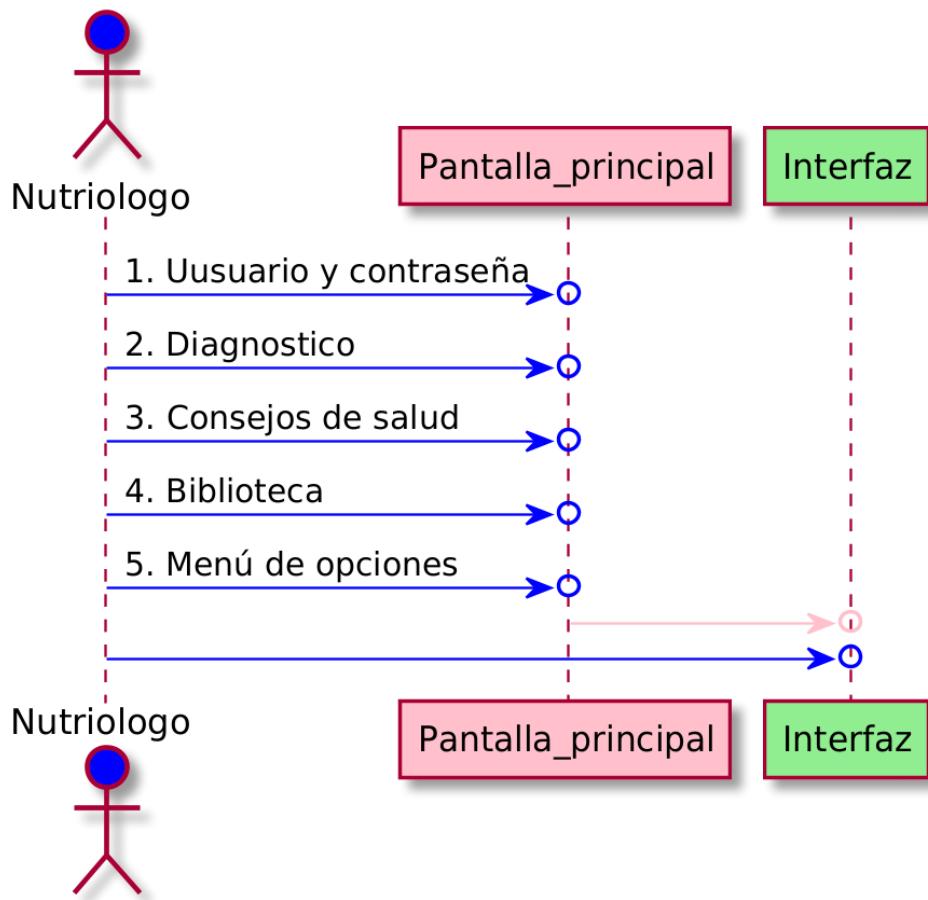


Figura 3.36: Diagrama de secuencias: Pantalla principal

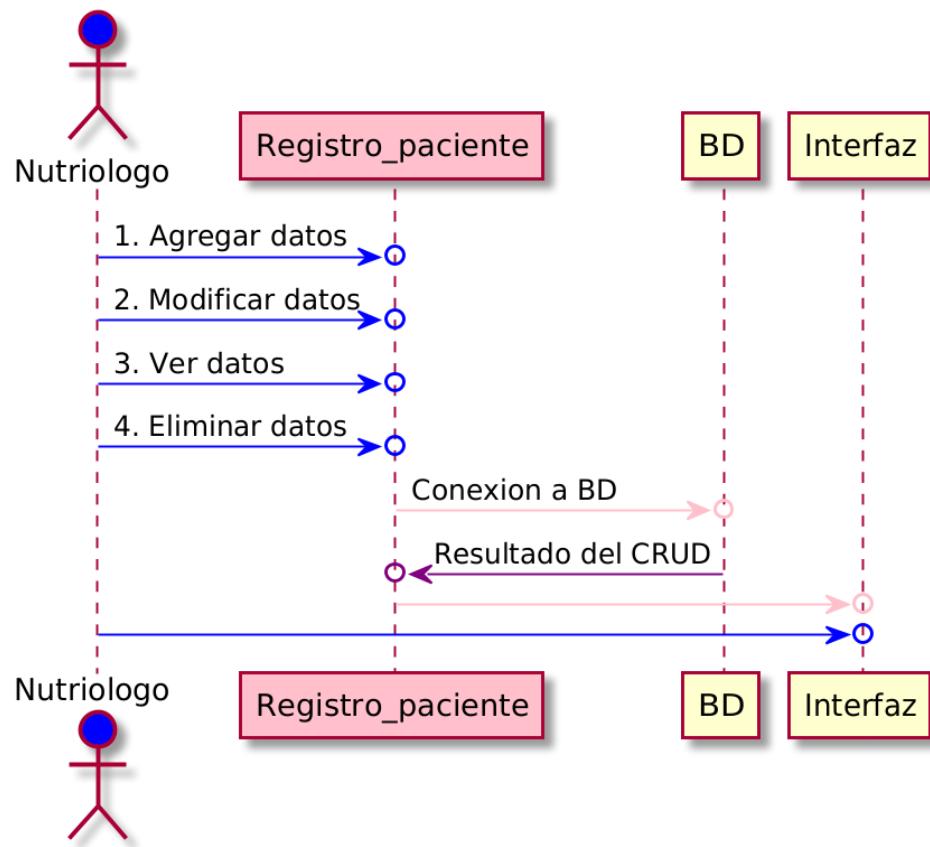


Figura 3.37: Diagrama de secuencias: Registro paciente

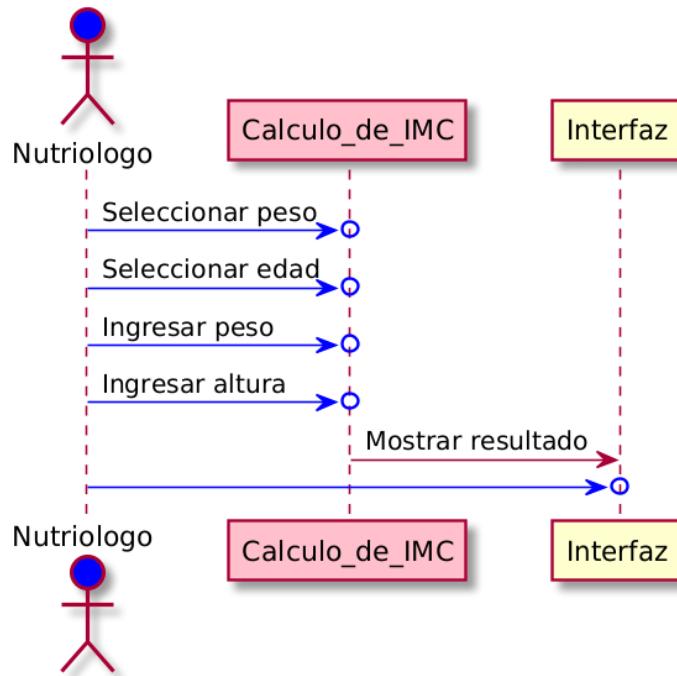


Figura 3.38: Diagrama de secuencias: Herramientas

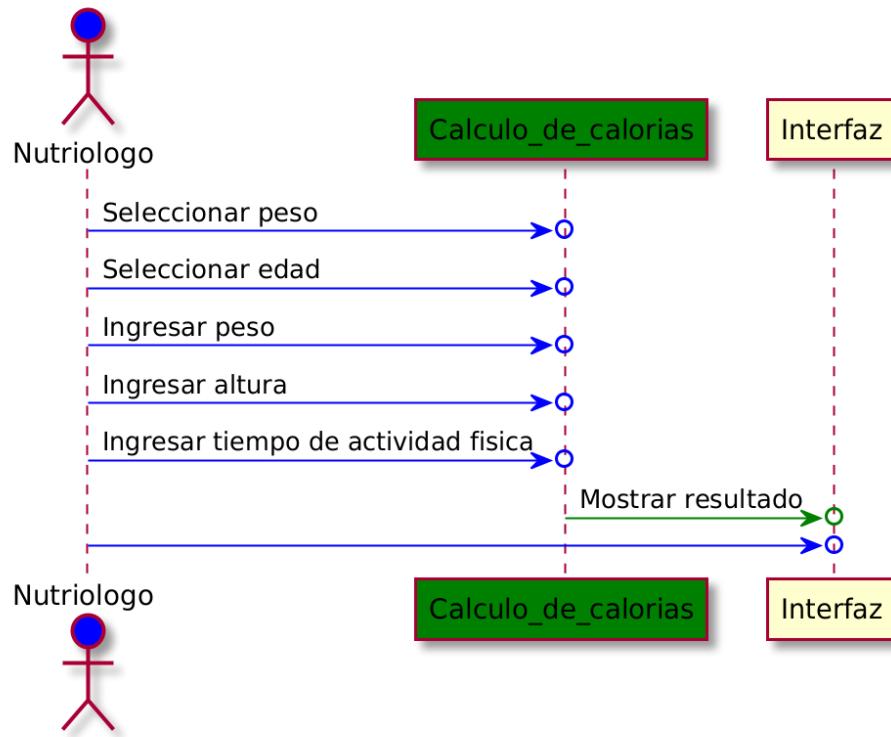


Figura 3.39: Diagrama de secuencias: Calculo de calorías

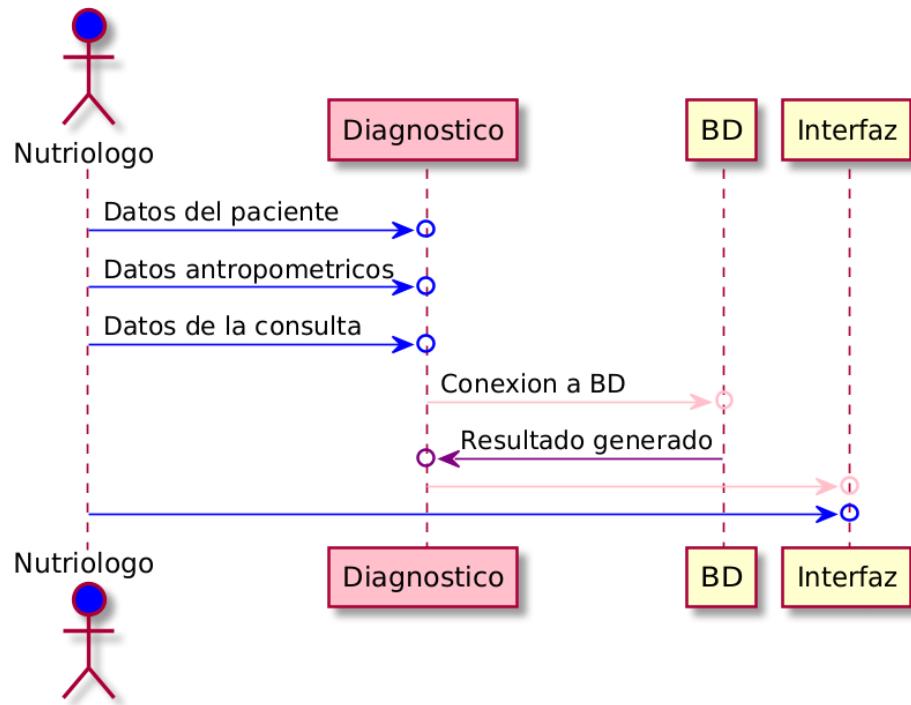


Figura 3.40: Diagrama de secuencias: diagnósticos

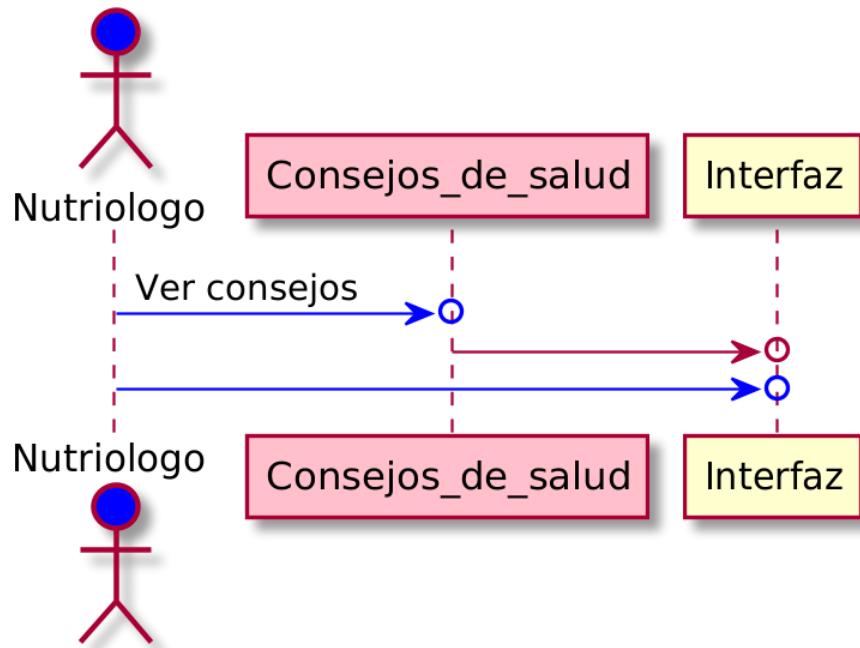


Figura 3.41: Diagrama de secuencias: consejo salud

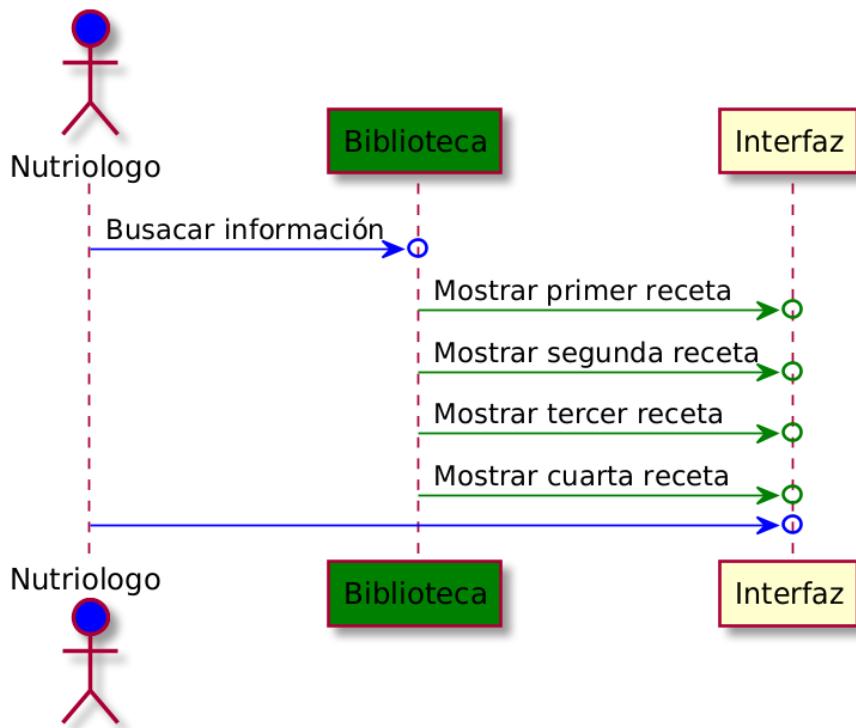


Figura 3.42: Diagrama de secuencias: biblioteca

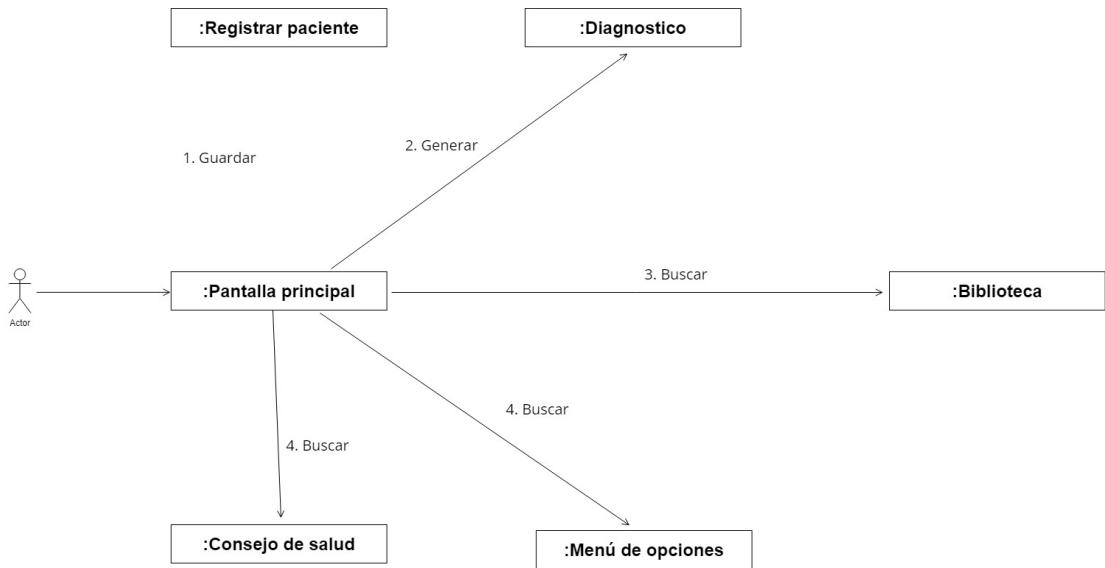
Visual Paradigm Online Free Edition



Visual Paradigm Online Free Edition

Figura 3.43: Diagrama de colaboración

Visual Paradigm Online Free Edition



Visual Paradigm Online Free Edition

Figura 3.44: Diagrama de colaboración

Capítulo 4

Diseño

4.1. Introducción

El diseño de software es quizá la etapa más importante y definitoria del proceso de desarrollo de software para que el producto que se obtenga sea de claridad. Esta etapa consiste, a grandes rasgos, en aplicar diferentes técnicas y metodologías con el fin de obtener un resultado lo suficientemente detallado como para que cualquier persona, dedicada a eso, pueda ser capaz de realizarlo de manera física o, dicho de otra manera, codificarlo. El diseño no sólo se refiere a la interfaz gráfica del software, como muchas veces se suele pensar cuando se escucha la palabra diseño, sino que implica un proceso específico gracias al cual se deben satisfacer los requisitos del sistema en desarrollo.

4.2. Descripción del sistema

El programa permite el registro clínico del paciente, como su historial de enfermedades, síntomas y exámenes de sangre. En base a estos datos, que siempre están abiertos a actualizaciones, el software analiza la cadena de información, generando indicativos y estimaciones para ayudar en la estrategia nutricional. Los profesionales usan el software de Nutrición pueden registrar modelos de minutaz que serán prescritos a sus pacientes. Esta funcionalidad le permite aplicar la consulta y que puede adaptarse fácilmente a las necesidades particulares de cada paciente. El programa de respaldo al nutricionista en sus análisis dietético. El software realiza análisis de nutrientes, cumple con las recomendaciones de ingesta diaria, calcula las interacciones nutricionales.

4.3. Arquitectura del sistema

4.3.1. Físico

Computadora portátil

Procesador Intel Core i7-1165G7 de 11a generación, no te costará trabajo realizar las actividades que requieran mayor exigencia, con fluidez y alta eficiencia. 8 GB en su memoria RAM para evitar interrupciones en multitareas o trabajo en línea. Unidad de estado sólido de 512 GB, podrás almacenar con seguridad tus documentos, o instalar tus programas preferidos consiguiendo un acceso a datos instantáneo.



Figura 4.1: computadora portátil

Teléfono móvil

El teléfono móvil es un dispositivo inalámbrico electrónico que permite tener acceso a la red de telefonía celular o móvil.



Figura 4.2: teléfono móvil

4.3.2. Lógico

Java (Lenguaje de programación)

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems. Su sintaxis es muy parecida a la de C o C++, e incorpora como propias algunas características que en otros lenguajes son extensiones: gestión de hilos, ejecución remota, etc. El código Java, una vez compilado, puede llevarse sin modificación alguna sobre cualquier máquina, y ejecutarlo. Esto se debe a que el código se ejecuta sobre una máquina hipotética o virtual, la Java Virtual Machine, que se encarga de interpretar el código (fichero compilado .class) y convertir a código particular de la CPU que se este utilizando (siempre que soporte dicha máquina virtual)

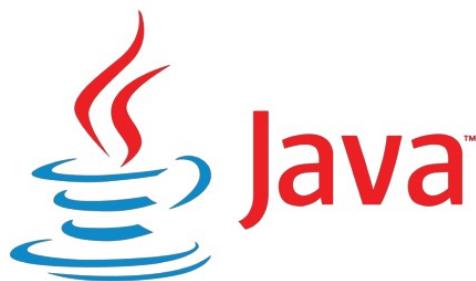


Figura 4.3: Java

Netbeans IDE (Integrated Developpment Environment)

Netbeans es un IDE o entorno de desarrollo integrado, que es gratuito y de código abierto, sirve para el desarrollo de aplicaciones web, corporativas, de escritorio y móviles que utilizan plataformas como Java y HTML, entre otras. El IDE se trata de un editor de código fuente, junto con recursos de construcción automática y un depurador. Igualmente, brinda la función de autocompletado inteligente de código o IntelliSense. En este caso el IDE de Netbeans ofrece un compilador y un interprete. Un IDE como Netbeans permite integrar los lenguajes de programación con las plataformas correspondientes a los sistemas operativos o entorno de programación.



Figura 4.4: Netbeans

PostgreSQL

PostgreSQL es un gestor que trabaja con bases de datos relacionales y que está orientado a objetos. Se trata de un programa de código abierto u open source, es decir, no está bajo el control de ninguna compañía particular, sino que cuenta con una comunidad de desarrolladores que trabajan en mejorar el programa de forma desinteresada.

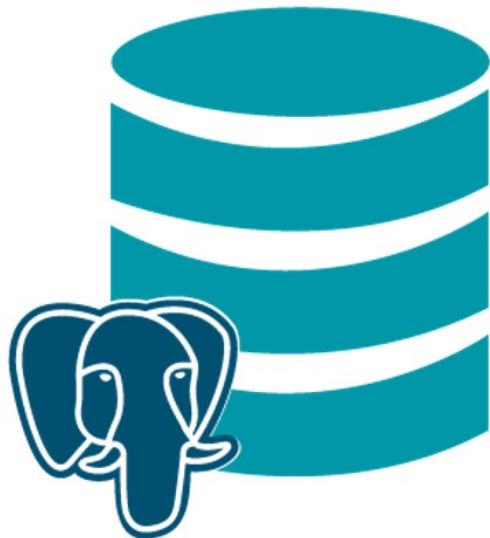


Figura 4.5: Netbeans

4.4. Estándar de codificación

Nomenclatura

El idioma por defecto a la hora de dar sentido funcional al nombre de clases, variables, constantes, etc. será una mezcla entre la nomenclatura tradicional en inglés y la nomenclatura funcional adoptada. Resumiendo, aquella codificación que por estandarización y/o aceptación se pueda escribir en inglés se mantendrá así por convenio, casos como: *insert*, *update*, *delete*, *create*, *retrieve*, *list*, *set*, *get*, *newInstance*, *Delegate*. Para la parte funcional se utilizará castellano, por lo tanto, la nomenclatura de los métodos será: *getListEmpresa* en sustitución de *getListCompany* o *insertBanco* en lugar de *insertarBanco*.

Paquetes

Por defecto todos los paquetes se escribirán en notación CamelCase y sin utilizar caracteres especiales. El paquete base queda definido como, en este paquete no se definirá ninguna clase. Se tendrá, así mismo, otro nivel extra dentro del paquete definido como el nombre del proyecto o del módulo.

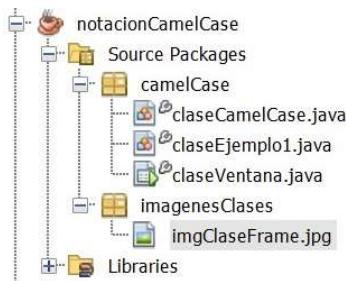


fig:Paquetes

Si existiera una parte común a varios de estos módulos, el nombre de los paquetes comenzará por:

La estructura en árbol de los paquetes siguientes a partir de este último.

Nombre de las interfaces

Por defecto todos los paquetes se escribirán en notación CamelCase y sin utilizar caracteres especiales. El paquete base queda definido como, en este paquete no se definirá ninguna clase. Se tendrá, así mismo, otro nivel extra dentro del paquete definido como el nombre del proyecto o del módulo.

```
public interface ejemploInterface {
    //Atributos
    //Métodos
}
```

fig:Nombre de las interfaces

Nombre de las clases

Los nombres de clases deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas (CamelCase). Debemos intentar mantener los nombres de clases simples y descriptivos.

```
public class ejemploClase {
    //Atributos
    //Métodos
}
```

fig:Nombre de las interfaces

Nombres específicos de gestiones

Nombres específicos de gestiones Cuando se trata de gestionar una entidad determinada (Ej. Usuario) se definen los nombres de clases, demás ficheros implicados con las siguientes reglas de las clases:

«*FuncionalidadGenerica*» «*Entidad*» «*Especificación de Clase*»

Ejemplo:

- Usuario
- usuarioAction
- findUsuarioAction

Métodos

Los métodos deberán ser verbos (en infinitivo), en mayúsculas y con la primera letra de cada palabra interna en mayúsculas (CamelCase). No se permiten caracteres especiales. El nombre del método suele ser un verbo o una frase verbal.

```
public void solicitarNumeros() {
    //Instrucciones
}
```

fig:Métodos

Indentación

El sangrado (también conocido como indentación) deberá aplicarse a toda estructura que esté lógicamente contenida dentro de otra. El sangrado será de un tabulador. Es suficiente entre 2 y 4 espacios. Para alguien que empieza a programar suele ser preferible unos 4 espacios, ya que se ve todo más claro.

Las líneas no tendrán en ningún caso demasiados caracteres que impidan que se pueda leer en una pantalla. Un número máximo recomendable suele estar entre unos 70 y 90 caracteres, incluyendo los espacios de sangrado. Si una línea debe ocupar más caracteres, tiene que dividirse en dos o más líneas. Para dividir una línea en varias, utiliza los siguientes principios:

- Tras una coma.
- Antes de un operador, que pasará a la línea siguiente.
- Una construcción de alto nivel (por ejemplo, una expresión con paréntesis).
- La nueva línea deberá alinearse con un sangrado lógico, respecto al punto de ruptura.

Variablos

Los nombres de las variables tanto de instancia como estáticas recibirán la notación de camel case, a continuación, se muestran algunos ejemplos.

```
int entero;
double decimal;
char caracter;
```

fig:Variables

Variablos

Se evitara en la medida de lo posible la utilización de caracteres especiales, así como nombre sin ningún tipo de significado funcional. Las excepciones son las variables utilizadas en bucles for, para esos casos se permite utilizar i, j, k, l y siempre en ese orden de anidamiento. El primer bucle siempre será el que tenga la variable i como iterador. (Esta variable se definirá para el bucle en cuestión, como se puede ver en el ejemplo).

```
for (int i = 0; i < 10; i++) {
    //implementación del código
}
```

fig:Variables for

Constantes

Los nombres de constantes de clases deberán escribirse en la notación en mayúsculas. Todas serán declaradas como public static final como se muestra en el ejemplo.

```
public static final int EJEMPLOCONSTANTE = 10;
```

fig:Constantes

Comentarios

Todos los ficheros fuente deben comenzar con un comentario en el que se lista el nombre de la clase, descripción, fecha de creación y fecha de actualización. Como se muestra en el ejemplo:

```
/*
 * Nombre de la clase: AplicEjemplo
 * Descripción: Ejemplo de los comentarios
 * Fecha de creación: 28-de-octubre-a-las-8:30-pm
 * Fecha de actualización: 28-de-octubre-a-las-10:00P-pm
 */
```

fig:Comentarios

Comentarios de implementación:

Delimitados por `/*...*/`, son para comentar nuestro código o para comentarios acerca de una implementación particular. Los comentarios deben contener sólo información que es relevante para la lectura y entendimiento del programa. Evitar cualquier comentario que pueda quedar desfasado a medida que el código evoluciona.

Se utilizarán los comentarios de una línea, que llevara los siguientes requisitos:

- Un comentario de una sola línea debe ir precedido de una línea en blanco.
- Pueden aparecer comentarios cortos de una única línea al nivel del código que siguen.
- Siempre el comentario se colocará en la primera línea.

Declaraciones

Para la declaración de las variables se utiliza una declaración de cada vez y no se permiten dejar variables locales sin inicializar salvo en el caso de que sean propiedades de un objeto vean. La codificación correcta sería:

```
for (int i = 0; i < 10; i++) { /*Esta función realiza...*/
    ...
}
```

fig:Declaraciones

La declaración de las variables locales a una clase, método o bloque de código se realizan al principio de este y no justo antes de necesitarse la utilización de la variable. La única excepción a esta regla son las variables que gestionan los bucles for. Las variables de avance de bucles for no podrán ser modificadas de ninguna manera fuera de la propia sentencia del bucle. La duplicidad de los nombres de variables en diferentes niveles dentro de la misma clase se tiene que evitar.

Sentencias

Las normas básicas son:

- Una únicamente sentencia por la línea de código.
- Todo bloque de sentencias entre llaves, aunque sea una sola sentencia después de un if.

```
public void ejemplo()
{
    //variable local ejemplo
    int ejemplo = 0;
    ejemplo = ejemplo + 5;
    System.out.println("ejemplo de declaracion local: " + ejemplo);
}
```

fig:Sentencias

La declaración de los bucles for serán usualmente de la forma: Son obligatorias las tres

```
if [expresion] {
    // Bloque 1
} else {
    // Bloque 2
}
```

fig:Sentencias

condiciones del bucle for: inicialización, condición de finalización y actualización del valor de la variable de avance. La variable de avance del bucle nunca podrá ser modificada dentro del propio bucle.

Propiedades

El acceso de las propiedades de una clase (no constantes) siempre mediante métodos de acceso get/set. La asignación de variables/propiedades no podrá ser consecutivas. Utilizar el operador

```
for (int i = 0; i < 10; i++) {
```

fig:Propiedades

de asignación en sitios donde se pueda confundir con el operador de igualdad, siempre y cuando hacer comentarios de la asignación para poder identificarla.

4.5. Diseño de datos

4.5.1. MC

En base a la información necesaria recabada se identificaron los atributos los cuales se asociaron a entidades y relaciones dentro de la base de datos. La base de datos nutrición está compuesta mediante veintiséis atributos en los cuales siete de ellos (idpaciente, Nombre, Sexo, Teléfono, Domicilio, Ocupación, Fechanac) pertenecen a la entidad paciente, cada atributo pertenece a un domino de valor codificado. Seis atributos forman parte de la entidad VisitaMedica la cual contiene (Idvisita, Diagnostico, FechaHora, idPaciente, IdMedico, IdConsultorio) pertenecen a un dominio con valor codificado. La entidad Médico se compone por cuatro atributos (idMédico, Nombre, Sexo, Domicilio) pertenecen a un dominio de valor codificado. Dos atributos se compone la entidad consultorio (idConsultorio, Nombre) pertenecen a un dominio de valor codificado. Todos los dominios con valor codificado conservan los valores almacenan que se almacenan de manera real en la base de datos. La entidad Paciente contiene una relación 1:M con la entidad VisitaMedica. La entidad Consultorio tiene una relación 1:M con la entidad VisitaMedica. La entidad Medico tiene una relación 1:M con VisitaMedica. Las demás relaciones que parten de la entidad VisitaMedica solo llevan una relación 1:1.

4.5.2. ML

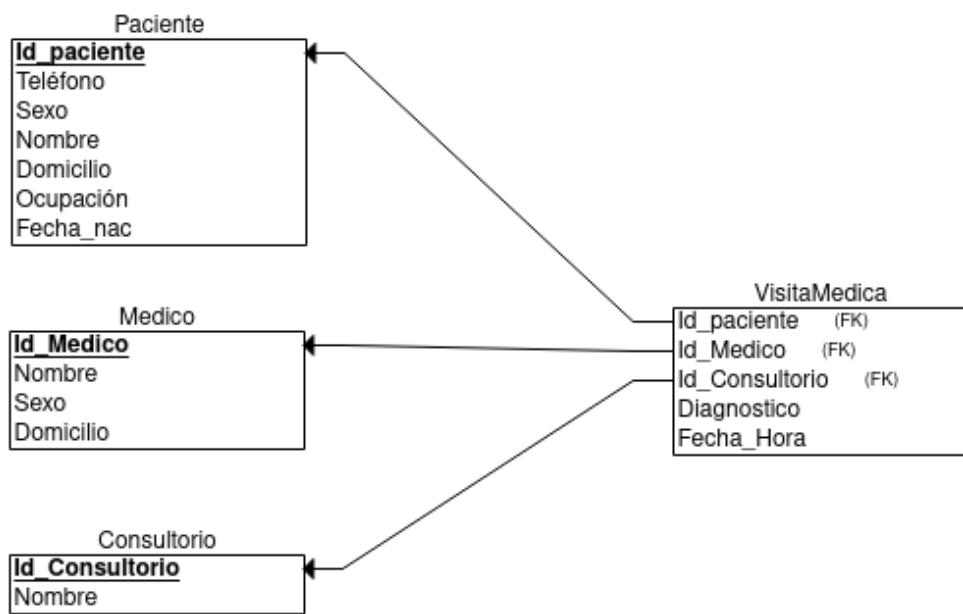


Figura 4.18: Esquema relacional

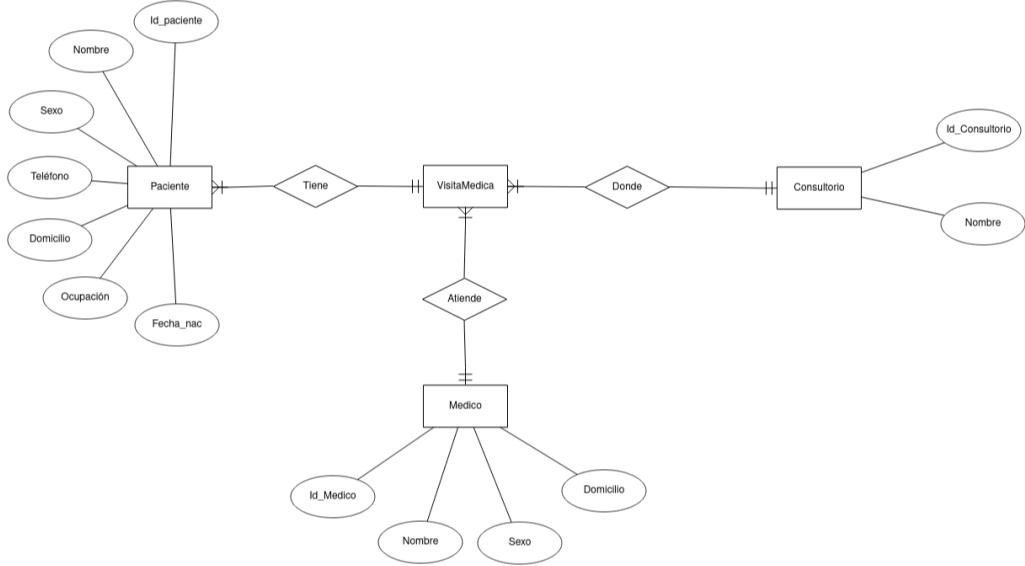


Figura 4.19: Diagrama entidad relación

4.5.3. MF

```

CREATE DATABASE nutricion;
create Domain idMedico
AS text
constraint idMedico_check
check((VALUE 'M'[0 - 9]4,4)::text));

CREATE TABLE Paciente(
Idpaciente integer,
Nombre varchar(50),
Sexo varchar(15),
Telefono integer,
Domicilio varchar(20),
Ocupacion varchar(30),
Fecha_nacimiento,
primarykey(Idpaciente));

CREATE TABLE Medico(
IdMedico idMedico,
Nombre varchar(50),
Sexo varchar(20),
Domicilio varchar(30),
primarykey(IdMedico));

CREATE TABLE Consultorio(
IdConsultorio integer,
Nombre varchar(20),
primarykey(IdConsultorio));

CREATE TABLE VisitaMedica(
Idvisita integer,
Diagnostico varchar(600),
Fecha_Hora date,
Idpaciente integer,
IdMedico idMedico,
IdConsultorio integer,
primarykey(Idvisita),
foreignkey(Idpaciente) references Paciente(Idpaciente) on delete restrict on update cascade,
foreignkey(IdMedico) references Medico(IdMedico) on delete restrict on update cascade,
foreignkey(IdConsultorio) references Consultorio(IdConsultorio) on delete restrict on update restrict);

```

4.5.4. Diccionario de datos

Tabla consultorio

	Columna	Tipo de dato	Descripción
PK	Id_Consultorio	Int	Identificación única del consultorio
	Nombre	Varchar (20)	Nombre del consultorio

Figura 4.20: Diccionario de la tabla consultorio

Tabla paciente

	Columna	Tipo de dato	Descripción
PK	Id_paciente	Integer	Clave única del paciente
	Nombre	Varchar (50)	Nombre completo del paciente
	Sexo	Varchar (15)	Sexo del paciente
	Teléfono	Integer	Teléfono de contacto del paciente
	Domicilio	Varchar (20)	Domicilio del paciente
	Ocupación	Varchar (30)	Ocupación del paciente
	Fecha_nac	Date	Fecha de nacimiento del paciente

Figura 4.21: Diccionario de la tabla pacientes

Tabla medico

	Columna	Tipo de dato	Descripción
PK	Id_Medico	Integer	Clave única del medico
	Nombre	Varchar (50)	Nombre completo del medico
	Sexo	Varchar (20)	Sexo del medico
	Domicilio	Varchar (30)	Domicilio del medico

Figura 4.22: Diccionario de la tabla médico

Tabla visita medica

	Columna	Tipo de dato	Descripción
PK	Id_visita	Int	Numero de identificación de la visita medica
	Diagnostico	Varchar (600)	Diagnostico completo de la visita
	Fecha_Horas	Date	Hora del diagnostico
FK	Id_paciente	Integer	Clave única del paciente
FK	Id_Medico	Integer	Calve única del medico
FK	Id_Consultorio	Integer	Identificación única del consultorio

Figura 4.23: Diccionario de la tabla visita médica

Capítulo 5

Conclusión

La implementación y obtención de requerimientos que fueron inicialmente analizados creamos que cumplen con la mayoría de las necesidades que los usuarios darían al especificar el propósito de la construcción del sistema informático, dirante el trancurso se aclararon dudas e inconsistencias en las definiciones. Por otro lado, la definición de requerimientos hace posible establecer alcances reales y funcionales para el usuario, ya que en muchas ocasiones los usuarios solicitan funcionalidades inalcanzables para los proyectos.

La arquitectura, los casos de uso y el diseño del sistema nos permitieron desarrollar clases u objetos lógicos bien diseñados, lo cual se refleja en el tiempo de ejecución de los módulos del sistema, es decir, el sistema se vuelve más eficiente, por otro lado nos ayudó a reducir los errores de programación y de seguridad que se presentan en el desarrollo de sistemas, es decir el sistema se vuelve más robusto, así también se mejoró el aprovechamiento de los recursos de los servidores donde se alojará la aplicación. Le permite, también, al desarrollador mayor entendimiento del código para posibles modificaciones o actualizaciones al sistema.

5.1. Definiciones, acrónimos, abreviaturas

Software

Es el conjunto de instrucciones que las computadoras emplean para manipular datos. Sin el software, la computadora sería un conjunto de medios sin utilizar. Al cargar los programas en una computadora, la máquina actuará como si recibiera una educación instantánea; de pronto «sabe» como pensar y como operar.

Requerimientos funcionales

Expresan la naturaleza del funcionamiento del sistema (Cómo interacciona el sistema con su entorno y cuáles (cómo interacciona el sistema con su entorno y cuáles (cómo interacciona el sistema con su entorno y cuáles (cómo interacciona el sistema con su entorno y cuáles van a ser su estado y funcionamiento).van a ser su estado y funcionamiento), es decir los requisitos funcionales definen qué debe hacer un sistema un sistema.

Requerimientos no funcionales

Restricciones sobre el espacio de posibles soluciones, es decir los requisitos no funcionales definen cómo debe ser el sistema.

JDK

El Java Development Kit, JDK por sus siglas en inglés, es un grupo de herramientas para el desarrollo de software provisto por Sun Microsystems, Inc. Incluye las herramientas necesarias para escribir, testear, y depurar aplicaciones y applets de Java.

Funcionalidad

Descripción de lo que el software debe hacer.

Interfaces externas

Cómo debe interactuar el sistema con las personas, el sistema de hardware, o con otros sistemas (software y hardware).

Interfaces externas

Indicación de la velocidad, disponibilidad, tiempos de respuesta, tiempos de recuperación, tiempos de determinadas funciones.

Cliente/usuario

Son todas las personas quienes hacen uso de los servicios que ofrece la empresa.

5.2. Anexo 2



Figura 5.1: Exposición

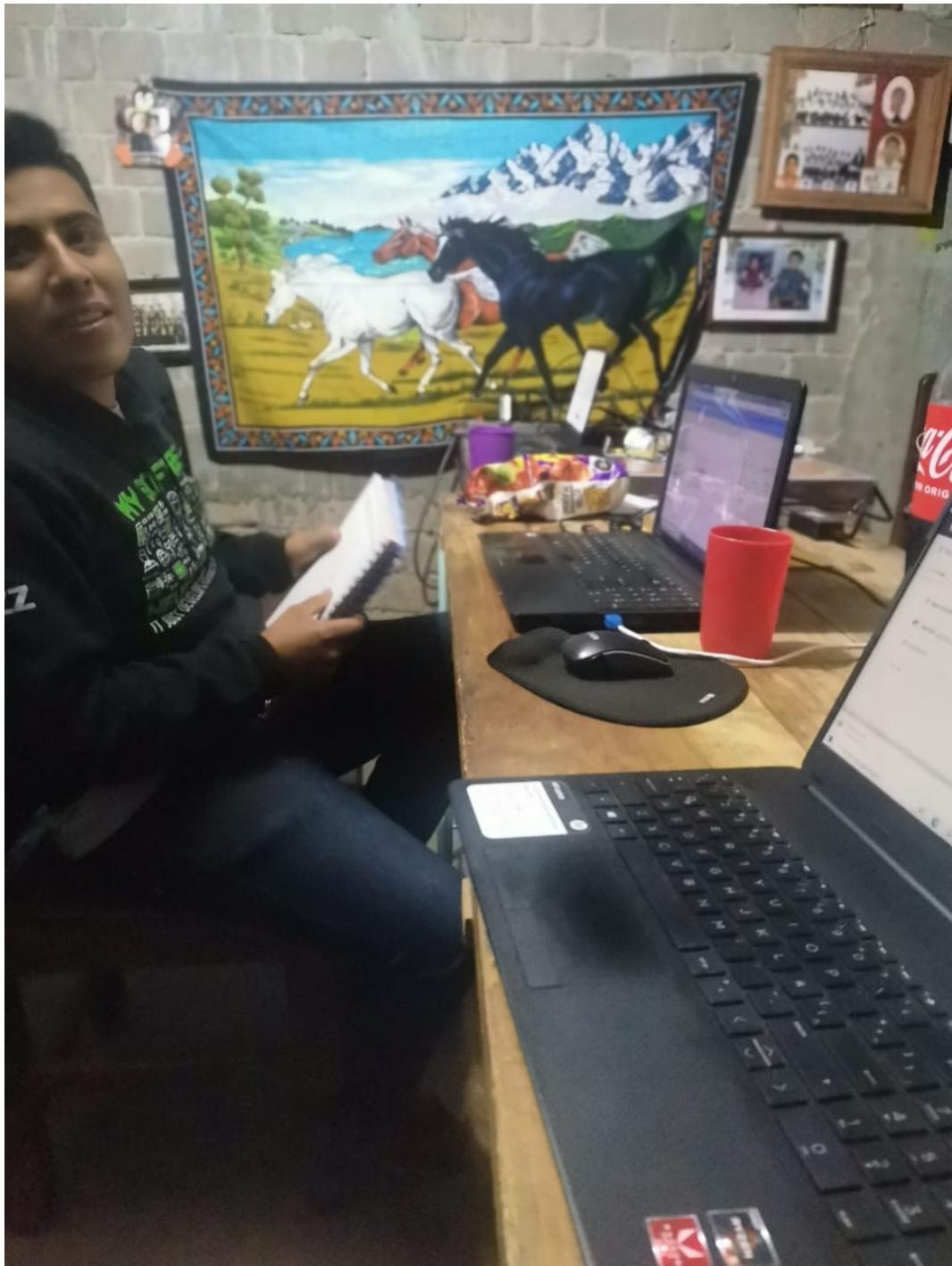


Figura 5.2: Noche de desvelo



Figura 5.3: Durmiendo

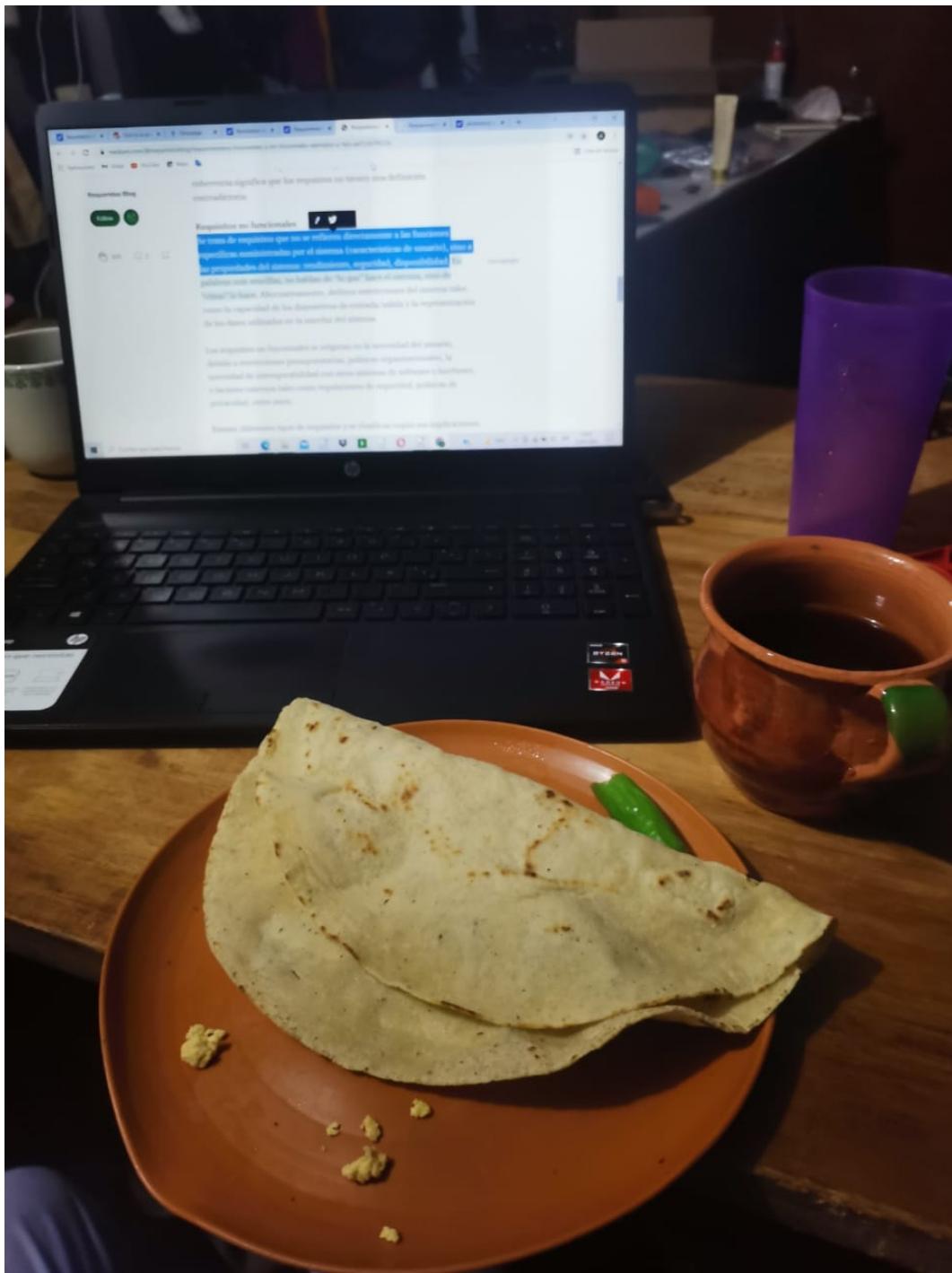


Figura 5.4: Cenando