

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

Indice de contenido

1. Introducción	5
2. Propósito	6
3. Ámbito del proyecto	7
4. Personal involucrado	8
5. Viabilidad técnica	9
6. Viabilidad económica	11
7. Mapa de navegación	12
8. Interfaces gráficas	13
9. Requisitos funcionales	23
10. Requisitos no funcionales	27
11. Estándar de codificación	28

Índice de figuras

7.1. Mapa de navegación	12
8.1. Login	13
8.2. Pantalla inicio	14
8.3. Consejos de salud	15
8.4. Herramientas(Calculadora de calorías)	16
8.5. Herramientas(Calcular tu IMC)	17
8.6. Datos de registro de pacientes	18
8.7. Paciente	19
8.8. Evaluación del paciente	20
8.9. Evaluación del paciente 2	21
8.10. Biblioteca de alimentos	22
9.1. RF01	23
9.2. RF02	24
9.3. RF03	24
9.4. RF04	25
9.5. RF05	25
9.6. RF06	26
9.7. RF07	26
10.1. RNF01	27
10.2. RNF02	27
10.3. RNF03	27
11.1. Paquetes	28
11.2. Nombre de las interfaces	29
11.3. Nombre de las clases	29
11.4. Métodos	29
11.5. Variables	30
11.6. Variables for	30
11.7. Constantes	31

11.8. Comentarios	31
11.9. Declaraciones	31
11.10Sentencias	32
11.11Sentencias	32
11.12Propiedades	32

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

Propósito

A partir de metodologías, técnicas y herramientas que sean útiles en los profesionales encargados del desarrollo de sistemas informáticos, se deberá cumplir con los procesos definidos previamente al inicio del desarrollo para la entrega de software de calidad.

Capítulo 3

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 ingesta 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.

Capítulo 4

Personal involucrado

Nombre	Cargo
M.T.C.A Rolando Pedro Gabriel	Usuario final
M.C. Lirio Ruiz Guerra	BDA
M.T.E. Everardo de Jesús Pacheco Antonio	Cliente
Mario Hecxai Valencia Reyes	Project Manager
Jazziel Pérez Hernández	Programador Senior
Alexi Daniel Ramírez Ruiz	Programador Jr

Capítulo 5

Viabilidad 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

Capítulo 6

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.

Capítulo 7

Mapa de navegación

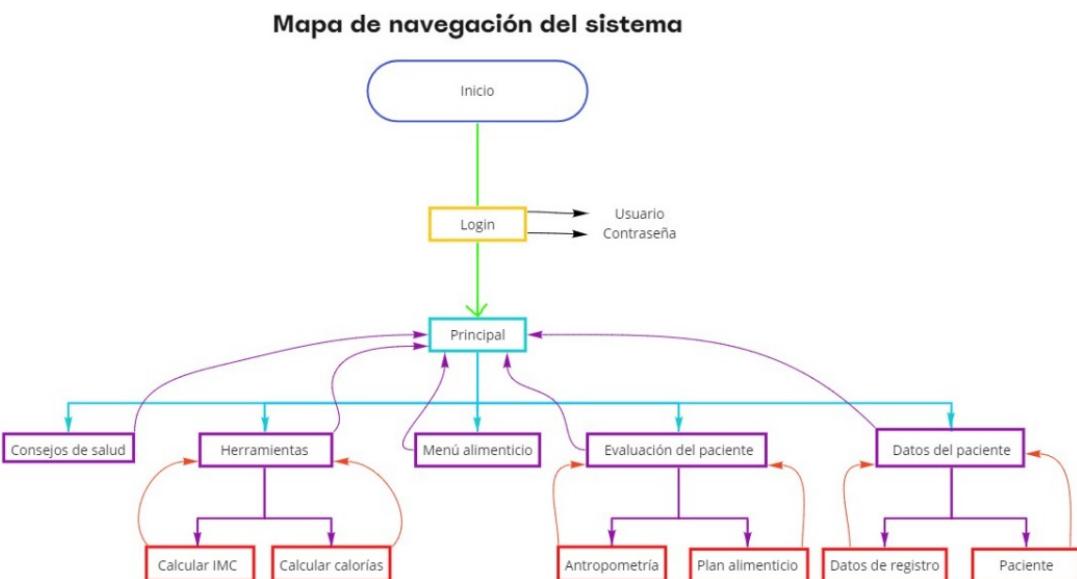


Figura 7.1: Mapa de navegación

Capítulo 8

Interfaces gráficas

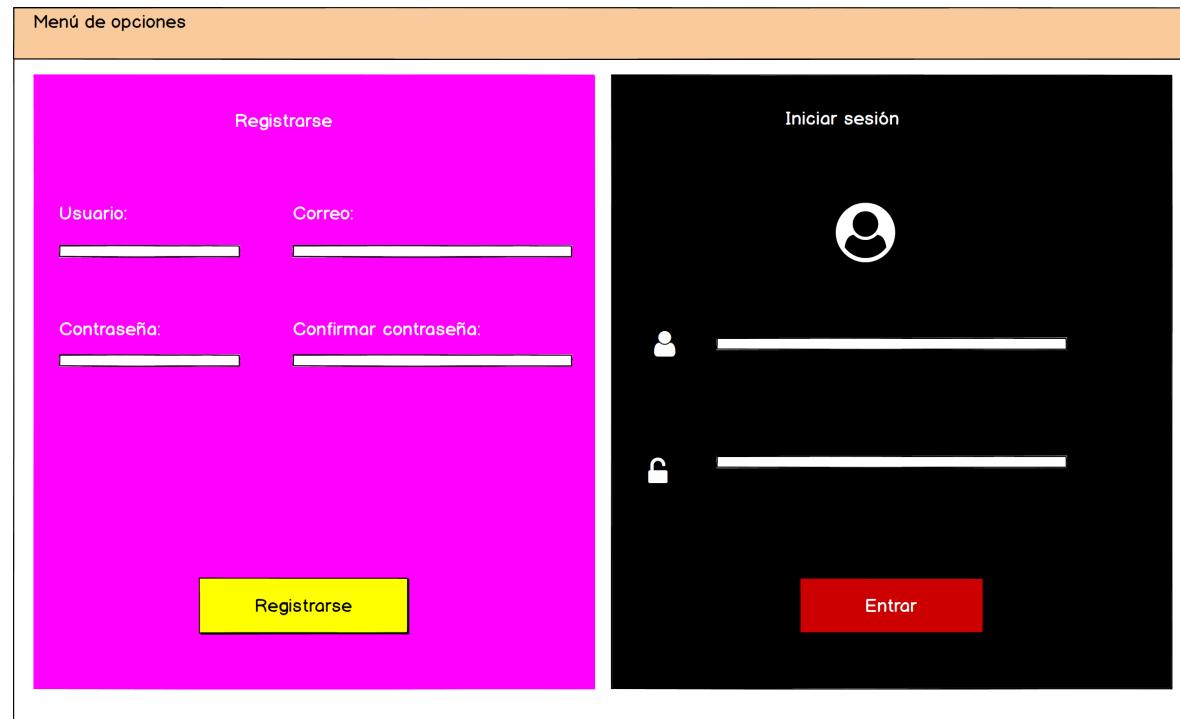


Figura 8.1: Login

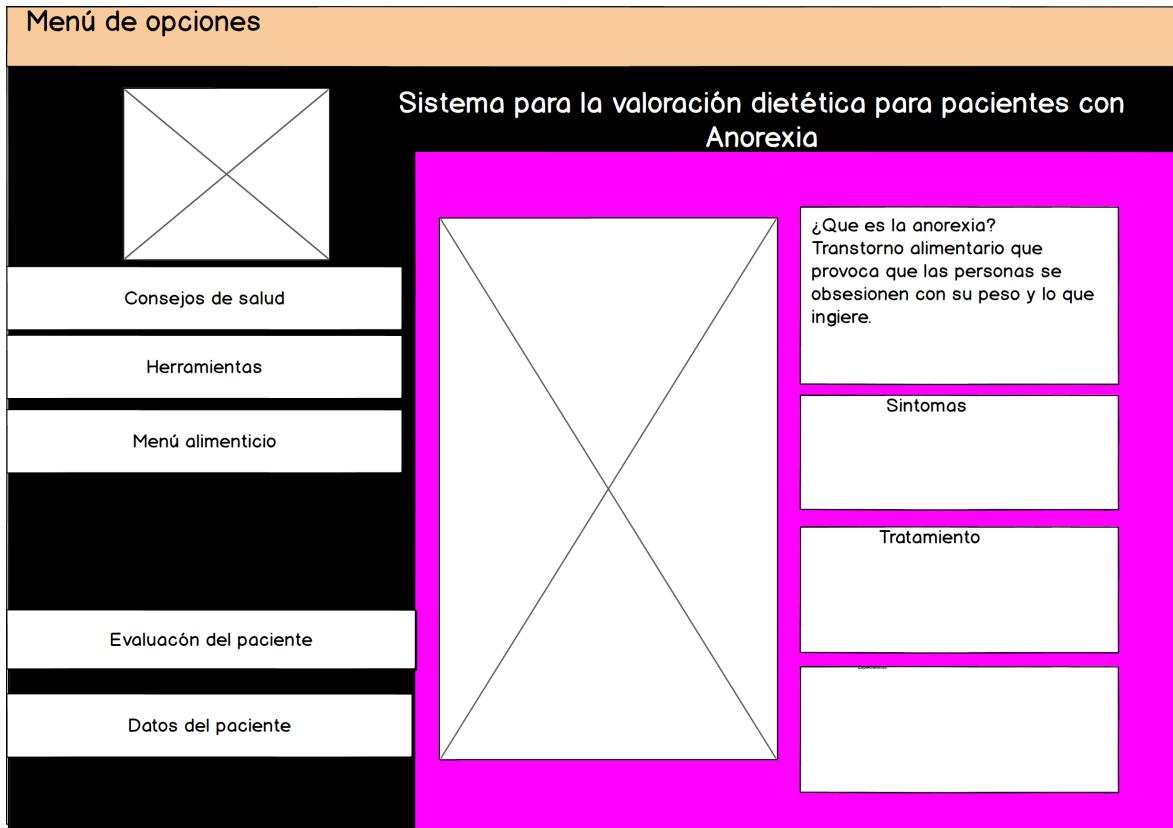
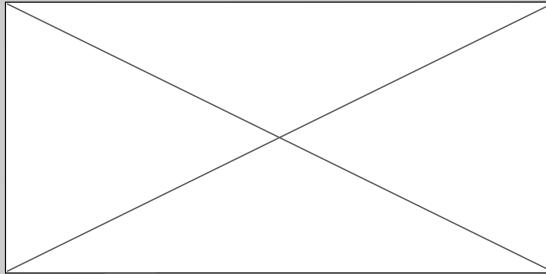


Figura 8.2: Pantalla inicio

Menú de opciones

Consejos de salud



Ambiente de la mesa

- Fomentar las comidas familiares.
- Favorecer que la ingesta se realice siempre en el lugar destinado a ello.
- Procurar un ambiente relajado, de conversación y de disfrute a la hora de comer.
- Desalentar la tendencia a comer en soledad.
- Educar en maneras en la mesa (uso de cubiertos, respeto a los demás comensales, postura, etc).
- Evitar tratar conflictos a la hora de comer ("ahora estamos comiendo...ya lo hablaremos en otro momento")

Dieta

- Procurar, en lo posible, una dieta variada e igual para todos los miembros de la familia.
- Evitar el hábito de tomar bebidas con sabores. El mejor líquido es el agua. Los zumos son un alimento.
- Establecer horarios y orden en las comidas, permitiendo cierta flexibilidad ocasional.
- No calificar las comidas de forma absoluta (buenas vs malas; basura vs sanas, etc.)
- No utilizar la comida como premio o amenaza.
- Es tan nocivo prohibir los dulces como mantener una permanente disponibilidad de los mismos.
- La idea de que hay que practicar la fuerza de voluntad para evitar comer excesivamente, es errónea y puede favorecer una percepción alterada de

Figura 8.3: Consejos de salud

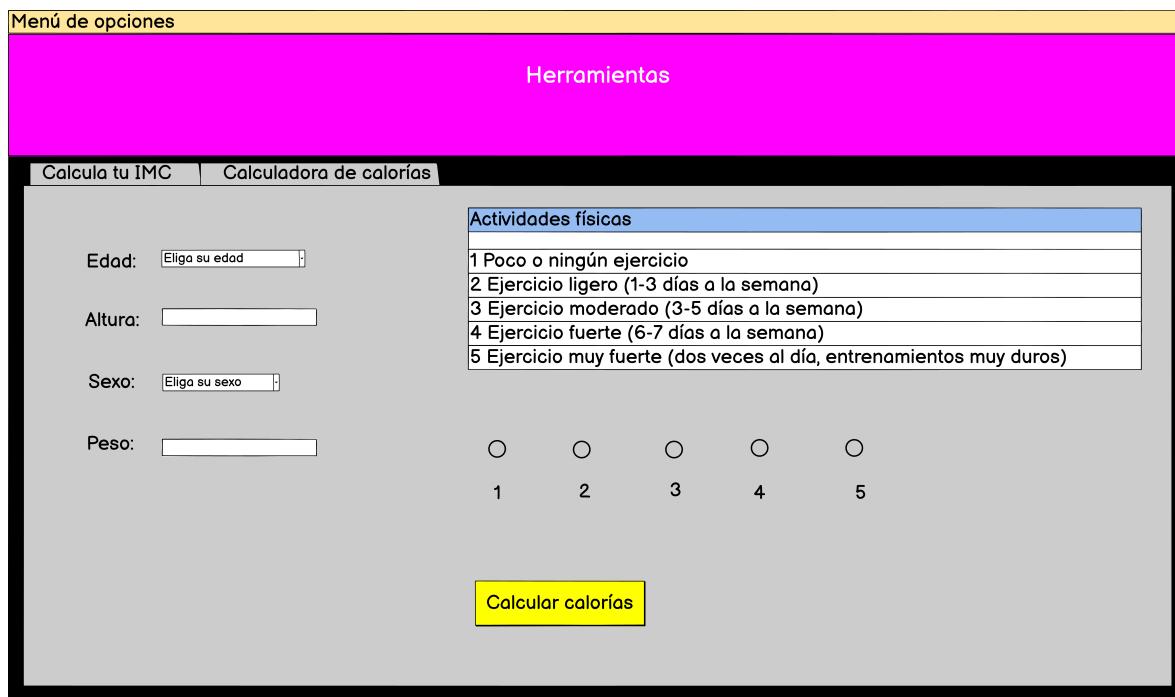


Figura 8.4: Herramientas(Calculadora de calorías)

Menú de opciones

Herramientas

[Calcula tu IMC](#) | [Calculadora de calorías](#)

Sexo:

Peso: Kgs

Edad:

Altura cms

Figura 8.5: Herramientas(Calcular tu IMC)

Menú de opciones

Registro de pacientes *Modificar/eliminar un paciente requiere confirmación

*Se requiere llenar todos los campos

Datos de registro | **Pacientes**

Nombre:	Seleccione el sexo:	Domicilio:
<input type="text"/>	<input type="radio"/> Mujer <input type="radio"/> Hombre	<input type="text"/>
Apellidos	Selecciona el día y mes de nacimiento:	Teléfono celular:
<input type="text"/>	<input type="text"/> 1 <input type="text"/> Enero <input type="text"/>	<input type="text"/>
Nº de identificación	Ingrasa el año de nacimiento:	Ocupación:
<input type="text"/>	<input type="text"/>	<input type="text"/>

Figura 8.6: Datos de registro de pacientes

Figura 8.7: Paciente

Figura 8.8: Evaluación del paciente

Figura 8.9: Evaluación del paciente 2

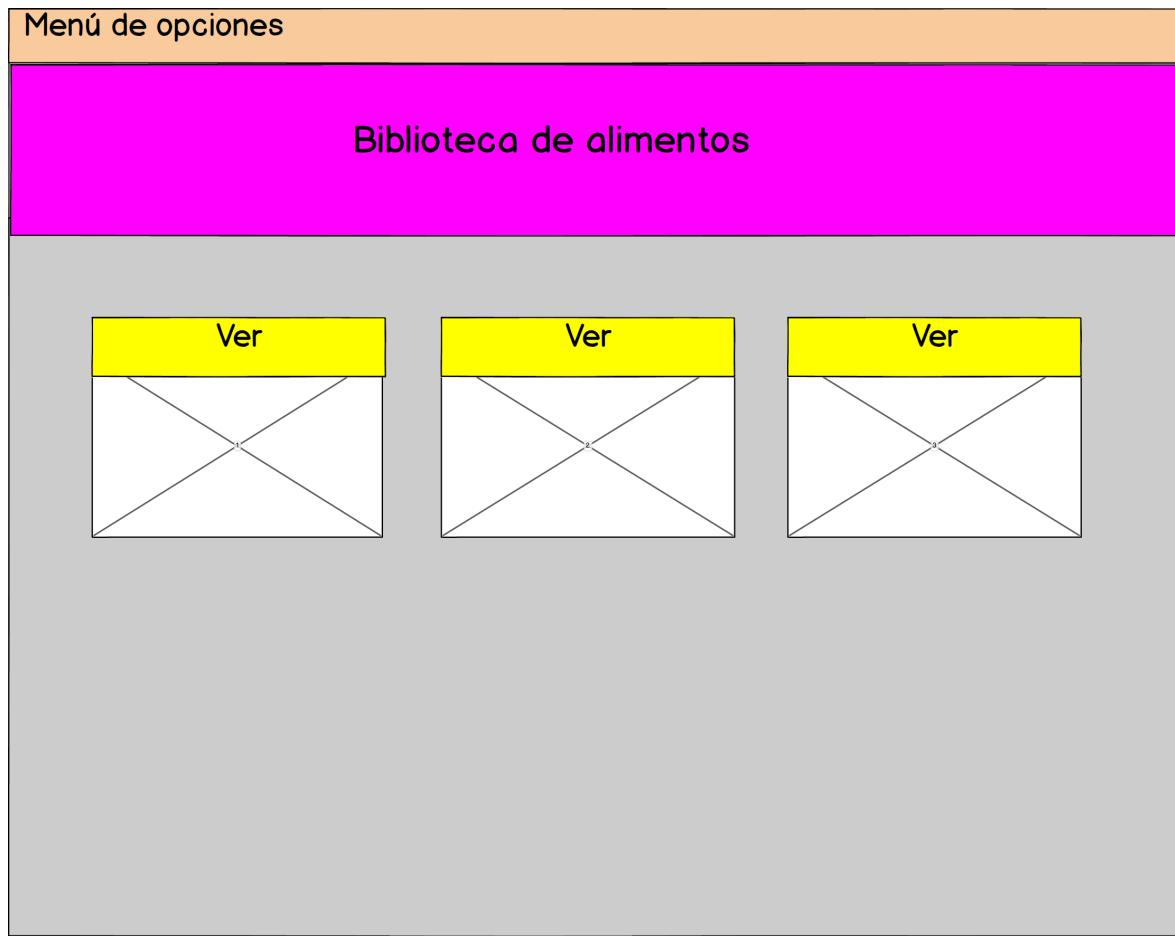


Figura 8.10: Biblioteca de alimentos

Capítulo 9

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 9.1: 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 9.2: 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 9.3: 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 9.4: 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 9.5: 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 9.6: 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 9.7: RF07

Capítulo 10

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 10.1: 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 10.2: RNF02

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

Figura 10.3: RNF03

Capítulo 11

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.

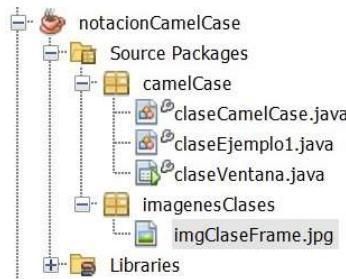


fig:Paquetes

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.

Variables

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

Variables

Se evitará 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: ApiEjemplo
 *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 llevará 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.