



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



Instituto Tecnológico de Acapulco

TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE ACAPULCO
MAESTRÍA EN SISTEMAS COMPUTACIONALES

**SISTEMA DE MENÚ INTERACTIVO APOYADO POR UN
ALGORITMO BASADO EN REGLAS DE ASOCIACIÓN PARA
LA ALIMENTACIÓN DE PACIENTES INTERNOS EN UN
HOSPITAL.**

CASO DE ESTUDIO: HOSPITAL DEL PACÍFICO

TESIS PROFESIONAL

**PARA OBTENER EL TÍTULO DE:
MAESTRO EN SISTEMAS COMPUTACIONALES**

PRESENTA:

ING. ROSA MARIBEL MARTÍNEZ MANZO

DIRECTOR:

M.I.D.S ALMA DELIA DE JESUS ISLAO

CODIRECTOR:

DR. EDUARDO DE LA CRUZ GÁMEZ

ACAPULCO DE JUAREZ, GRO. DICIEMBRE 2020,

El presente trabajo de tesis fue desarrollado en la *División de Estudios de Posgrado e Investigación* del *Instituto Tecnológico de Acapulco*, perteneciente al Programa Nacional de Posgrados de Calidad (PNPC-CONACYT).

Con domicilio para recibir y oír notificaciones en Av. Instituto Tecnológico de Acapulco S/N, Crucero del Cayaco, Acapulco, Guerrero, México. C.P. 39905.

Becario:	Rosa Maribel Martínez Manzo.
CVU:	851534.
Núm. de apoyo:	627019.
Grado:	Maestría



Descargo de Responsabilidad Institucional

La que suscribe declara que el presente documento titulado “Sistema de menú interactivo apoyado por un algoritmo basado en reglas de asociación para la alimentación de pacientes internos en un hospital. Caso de estudio Hospital del Pacífico” es un trabajo propio y original, el cual no ha sido utilizado anteriormente en institución alguna para propósitos de evaluación, publicación y/o obtención de algún grado académico.

Además, se adelanta que se han recogido todas las fuentes de información utilizadas, las cuales han sido citadas en la sección de referencias bibliográfica de este trabajo.

Acapulco, Gro., a 11 de diciembre de 2020.

Nombre: Ing. Rosa Maribel Martínez Manzo



Carta de cesión de derechos de autor

El que suscribe: Rosa Maribel Martínez Manzo, autor del trabajo escrito de evaluación profesional en la opción de Tesis Profesional de Maestría con el título “Sistema de menú interactivo apoyado por un algoritmo basado en reglas de asociación para la alimentación de pacientes internos en un hospital. Caso de estudio: Hospital del Pacífico”, por medio de la presente con fundamento en lo dispuesto en los artículos 5, 18, 24, 25, 27, 30, 32 y 148 de la Ley Federal de Derechos de Autor, así como los numerales 2.15.5 de los lineamientos para la Operación de los Estudios de Posgrado; manifiesto mi autoría y originalidad de la obra mencionada que se presentó en la División de Estudios de Posgrado e Investigación, para ser evaluada con el fin de obtener el Título Profesional de Maestro en Sistemas Computacionales.

Así mismo expreso mi conformidad de ceder los derechos de reproducción, difusión y circulación de esta obra, en forma NO EXCLUSIVA, al Tecnológico Nacional de México campus Acapulco; se podrá realizar a nivel nacional e internacional, de manera parcial o total a través de cualquier medio de información que sea susceptible para ello, en una o varias ocasiones, así como en cualquier soporte documental, todo ello siempre y cuando sus fines sean académicos, humanísticos, tecnológicos, históricos, artísticos, sociales, científicos u otra manifestación de la cultura.

Entendiendo que dicha cesión no genera obligación alguna para el Tecnológico Nacional de México campus Acapulco y que podrá o no ejercer los derechos cedidos. Por lo que el autor da su consentimiento para la publicación de su trabajo escrito de evaluación profesional.

Se firma presente en la ciudad de Acapulco de Juárez, estado de Guerrero a los 11 días del mes de diciembre de 2020.



Rosa Maribel Martínez Manzo



"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

Acapulco, Gro; a 7 de diciembre de 2020.

AUTORIZACIÓN DE IMPRESIÓN DE TESIS

Los abajo firmantes, miembros de la comisión revisora de tesis designada por la División de Estudios de Posgrado e Investigación del Tecnológico Nacional de México campus Acapulco para la evaluación de la tesis del alumno **Rosa Maribel Martínez Manzo**, manifiestan que después de haber revisado su tesis: **"Sistema de Menú Interactivo Apoyado por un Algoritmo Basado en Reglas de Asociación para la Alimentación de Pacientes Internos en un Hospital. Caso de Estudio: Hospital del Pacífico"** desarrollada bajo la dirección del DIRECTOR, y el CO-DIRECTOR, el trabajo se **ACEPTA** para proceder a su impresión.

ATENTAMENTE

M.I.D.S. Alma Delia de Jesús Islao
Cédula Profesional: 8604126

Dr. Eduardo de la Cruz Gámez
Cédula Profesional: 6526073

M.T.I. Jorge Carranza Gómez
Cédula Profesional: 5826790

Entera

Dr. Eduardo de la Cruz Gámez
Coordinador de la Maestría en Sistemas
Computacionales



SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO DE ACAPULCO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN



acapulco.edu.mx

Av. Instituto Tecnológico s/n Crucero del Cayaco C. P. 39905
de contacto:
depi_acapulco@tecnm.mx
Teléfonos: (744) 4429010
al 19 ext. 121
www.it-



CACECI
Consejo de Asesoría de la Comisión de Ingeniería, A.C.



Numero de registro: RPL-072
Fecha de inicio: 2017-04-10
Termino de la certificación: 2022-04-10



Instituto Tecnológico de Acapulco
División de Estudios de Posgrado e Investigación

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

Acapulco Gro., 8/Diciembre/2020

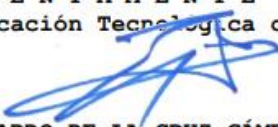
NO. OFICIO: DEPI-214/2020

ASUNTO:
AUTORIZACIÓN DE
IMPRESIÓN DE TESIS PROFESIONAL

C. ROSA MARIBEL MARTÍNEZ MANZO

De acuerdo al reglamento de los Institutos Tecnológicos, dependiente de la Secretaría de Educación Pública y habiendo cumplido con todos los requisitos normativos respecto a su trabajo para titulación, Opción Titulación Tesis Profesional, con el proyecto titulado: **SISTEMA DE MENÚ INTERACTIVO APOYADO POR UN ALGORITMO BASADO EN REGLAS DE ASOCIACIÓN PARA LA ALIMENTACIÓN DE PACIENTES INTERNOS EN UN HOSPITAL. CASO DE ESTUDIO: HOSPITAL DEL PACÍFICO**, se **CONCEDE** la **AUTORIZACIÓN** para que proceda a la impresión del mismo.

Sin otro particular por el momento, me es grato quedar de usted.

A T E N T A M E N T E "
Educación Tecnológica con Compromiso Social

EDUARDO DE LA CRUZ GÁMEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN



C.c.p. Expediente

EDG/stv



Av. Instituto Tecnológico s/n Crucero del Cayaco C.P. 39905
e-mail de contacto: depi_acapulco@tecnm.mx
Teléfonos: (744) 4429010 al 19 ext. 121
www.it-acapulco.edu.mx



www.diregco/Pvt-07
Fecha de emisión: 2017-09-10
Número de certificación: 2017-09-10

AGRADECIMIENTOS

A CONACYT por el apoyo económico brindado a lo largo de los dos años del posgrado y al Instituto Tecnológico de Acapulco por brindarnos la oportunidad de formar parte de la segunda generación de la Maestría en Sistemas Computacionales.

A mis maestros, quienes han fungido como guías en el estudio y quienes me han impulsado a superarme cada día más.

A mis compañeros de generación, con los cuales he aprendido muchas cosas que me servirán para aplicarlas en mi vida diaria, tanto personal como laboral, y con los que pase por tantas cosas para lograr nuestra meta conjunta.

DEDICATORIAS

A mis padres, que siempre han creído en mí y en mi capacidad para superarme y quienes han sido un apoyo invaluable tanto en este proceso como en mi vida, a mi padre Gustavo Martínez por brindarme su sabiduría, comprensión y tiempo y a mi madre Maribel Manzo por ser un soporte importante en mi vida, a pesar de todo siempre seremos una familia, los amo.

A todas aquellas personas, compañeros y amigos que me han ayudado a resolver mis dudas

A mi mejor amiga Linda Nelly Sales por ser la hermana biológica que nunca tuve, gracias por tu inmenso apoyo, por siempre estar ahí cuando necesité de alguien, por tus consejos incondicionales, juntas empezamos nuestras maestrías y juntas las terminamos

RESUMEN

En el presente trabajo se verá el desarrollo de un sistema de menú interactivo apoyado por el algoritmo Apriori necesario para la generación de reglas de asociación, las cuales ayudaron a indicar, definir y agrupar las diferentes comidas contenidas en las diversas dietas con las que cuenta el menú del Hospital del Pacífico ubicado en el puerto de Acapulco de Juárez, Guerrero.

La metodología que se utilizó fue SCRUM, de igual modo se empleó ASP.NET MVC Framework para poder diseñar un sistema computacional interactivo en la WEB, así mismo se muestran la arquitectura y desarrollo del proyecto, los resultados, la aplicación WEB implementada en el hospital, las conclusiones y los trabajos a futuro que tiene el mismo.

Palabras Clave: Algoritmos, Reglas, Asociación, Alimentación

ABSTRACT

In the present work we will see the development of an interactive menu system supported by the Apriori algorithm necessary for the generation of association rules, which helped to indicate, define and group the different meals contained in the diverse diets that the menu of the Hospital del Pacífico located in the port of Acapulco de Juárez, Guerrero.

The methodology that was used was SCRUM, in the same way ASP.NET MVC Framework was used to be able to design an interactive computing system on the WEB, likewise the architecture and development of the project, the results, the WEB application implemented in the hospital, the conclusions and future work that it has.

Keywords: Algorithms, Rules, Association, Alimentation

Contenido

Carta de cesión de derechos de autor	IV
Formato Autorización de impresión de tesis	V
Oficio Autorización de impresión de tesis profesional	VI
AGRADECIMIENTOS	VII
DEDICATORIAS	VIII
RESUMEN	IX
ABSTRACT	IX
Contenido	X
Índice de Figuras	XIII
Índice de tablas	XIV
INTRODUCCIÓN	XV
Capítulo 1 Introducción	1
1.1 Antecedentes del problema.....	1
1.2 Planteamiento del problema	4
1.3 Objetivo general.....	11
1.4 Objetivos específicos.....	11
1.5 Hipótesis.....	11
1.6 Justificación.....	12
1.7 Alcances y Limitaciones	13
Capítulo 2 Estado del arte	15
Capítulo 3 Marco teórico	26
3.1 Reglas de asociación	27
3.1.1 Algoritmo Apriori	27
3.1.2 Ejercicio en R	34
3.2 Lenguajes De Programación.....	50
3.2.1 Lenguaje C#	50
3.2.2 JavaScript	50
3.3 Herramientas De Desarrollo De Software.....	51
3.3.1 Microsoft Visual Studio	51
3.3.2 Enterprise Architect	51
3.4 Metodología De Desarrollo De Software	52

3.4.1 Scrum	52
3.5 MVC	56
3.5.1 Frontend	57
3.5.2 Backend	58
3.6 Base de datos	60
3.6.1 Motor de base de datos PostgreSQL	60
3.6.2 Sistema Gestor de Base de Datos pgAdmin III	60
Capítulo 4 Modelado del sistema	61
4.1 Definición de los requerimientos.....	61
4.1.1 Características del sistema	61
4.1.2 Requerimientos Funcionales	61
4.1.3 Requerimientos No Funcionales	62
4.1.4 Requerimientos del Software	62
4.1.5 Requerimientos del Hardware	63
4.2 Análisis y Diseño.....	64
4.2.1 Modelado de negocios	64
4.2.2 Diagrama de secuencia	66
4.2.3 Casos de uso	67
4.3 Diagrama de Modelado de la Base De Datos.....	70
4.4 Diagrama de Despliegue	74
4.5 Diagrama de transición de estados	75
4.6 Cronograma de actividades	76
Capítulo 5 Arquitectura y desarrollo del proyecto	77
5.1 Pacíficoapp.Dal (Data Acces Layer)	77
5.2 Pacíficoapp.Model (Modelos).....	80
5.3 Pacíficoapp.Security (Seguridad)	82
5.4 Pacíficoapp.Web (Presentación).....	84
5.5 Desarrollo de Módulos.....	89
5.5.1 Registro de pacientes	89
5.5.2 Comida diaria	92
5.5.3 Compra de ordenes	95
5.5.4 Órdenes del día en cocina	98
Capítulo 6 Resultados	99
	XI

6.1 Secciones.....	100
6.1.1 Como rol de administrador	100
6.1.2 Como Rol de Paciente	107
6.1.3 Como Rol de Personal	108
6.1.4 Como Rol de Cocina	109
6.2 Despliegue de aplicación en la empresa.....	110
CONCLUSIONES	118
Trabajo a futuro	120
ANEXO	121
BIBLIOGRAFIA	123

Índice de Figuras

Figura 3-1 Itemsets más frecuentes	41
Figura 3-2 Iteraciones en Scrum (ProyectosAgiles.org CC BY-SA)	55
Figura 3-3 Modelo Vista Controlador (codingornot.com)	56
Figura 4-1 Modelado de negocios	65
Figura 4-2 Diagrama de secuencia de la comanda	66
Figura 4-3 Caso de uso general	67
Figura 4-4 Administrador de roles	68
Figura 4-5 Caso de uso, interacción entre cocina y paciente	69
Figura 4-6 Base de datos completa	70
Figura 4-7 Bloque 1- Usuarios y roles	71
Figura 4-8 Bloque 2 - Catálogos de comidas	72
Figura 4-9 Bloque 3 - Paciente	73
Figura 4-10 Diagrama de distribución	74
Figura 4-11 Cronograma de Actividades	76
Figura 5-1 Capa de acceso a datos	78
Figura 5-2 Modelos	81
Figura 5-3 CatalogoComida.cs	82
Figura 5-4 Seguridad	82
Figura 5-5 Presentación al usuario	84
Figura 5-6 Primera migración	86
Figura 5-7 Acceso a servicios	87
Figura 5-8 Relación entre controladores y vistas	88
Figura 5-9 Primer registro al crear la base de datos y credenciales para la conexión de la BD	89
Figura 5-10 Obtención de pacientes	90
Figura 5-11 UsuarioController.cs	91
Figura 5-12 Consulta LINQ	93
Figura 5-13 ComidaDiariaController.cs	94
Figura 5-14 OrdenesController : Controller	95
Figura 5-15 Mis ordenes()	96
Figura 5-16 CocinaController.cs	97
Figura 5-17 Catalogo comida diaria	98
Figura 6-1 Portada inicial	99
Figura 6-2 Usuarios registrados	101
Figura 6-3 Tipos de puestos	102
Figura 6-4 Catálogo de comidas	103
Figura 6-5 Dietas que maneja el hospital	105
Figura 6-6 Registro de pacientes	105
Figura 6-7 Ordenes de cocina	106
Figura 6-8 Datos del paciente	107
Figura 6-9 Catalogo de comida por día	108

Figura 6-10 Listado de ingredientes	109
Figura 6-11 Registro de comidas	110
Figura 6-12 Router TP-Link modelo TD-W8960N	112
Figura 6-13 Caseta de almacenamiento	113
Figura 6-14 Cocina	113
Figura 6-15 Costado del edificio	114
Figura 6-16 Instalación del cable	115
Figura 6-17 Instalación del Router	116
Figura 6-18 Configuraciones del Router	117

Índice de tablas

Tabla 1-1 Dietas del Hospital del Pacífico	10
Tabla 3-1 Ejemplo, BD de un centro comercial	28
Tabla 3-2 Items individuales con su soporte.....	30
Tabla 3-3 Itemsets de tamaño $k=2$	31
Tabla 3-4 Itemsets frecuentes de tamaño $k=2$	31
Tabla 3-5 itemsets de tamaño $k=3$	32
Tabla 3-6 itemsets frecuentes de tamaño $k=3$	32
Tabla 3-7 itemsets frecuentes al final del procedimiento	32
Tabla 3-8 Reglas obtenidas de los itemsets frecuentes	33
Tabla 3-9 Operadores para filtrado de items.....	42

INTRODUCCIÓN

La alimentación hospitalaria tiene características muy especiales debido a que se relaciona con el aporte de nutrimentos específicos en personas en situaciones fisiopatológicas y en un estado anímico alterado por su situación y pronóstico de salud.

Si a esto le sumamos el hecho de que existen otras vulnerabilidades tales como las alergias e intolerancias ocasionadas por la ingesta de ciertos alimentos, así como el peligro que representan condiciones de salud tales como la diabetes, nos encontramos ante un problema que en el momento de proporcionar los alimentos a los pacientes de los hospitales debe ser tratado con mucho cuidado.

Aunado a esto debemos observar la condición del paciente, mismo que en ocasiones no se encuentra en condiciones de dar sus generales para que estas vulnerabilidades sean conocidas por el personal específico del hospital.

Esta problemática, analizada desde diversos ángulos sólo arroja que la solución a la que es susceptible sólo se encuentra en la interacción directa entre el paciente y el personal que prepara y lleva hasta éste sus respectivos alimentos.

Esta interacción puede ser simplificada con el auxilio de la tecnología, es decir, con la participación de una aplicación web que pueda encerrar todo el menú de alimentos que un hospital ofrece cada día aun cuando sea muy diverso para que el paciente pueda elegir los alimentos que desee consumir siempre y cuando éstos observen las características propias de la dieta que le ha impuesto su doctor.

Esta aplicación a su vez, manejaría de forma interactiva y dinámica el menú que tiene el hospital y así seleccionar el platillo que sea de su agrado de forma tal que el personal de cocina

pueda a su vez ver la posibilidad de satisfacer el pedido del paciente sin que “rompa su dieta”, esto, debido a que el personal manejaría un inventario diario de aquello con lo que la cocina y sus bodegas cuentan.

De esta forma, tenemos a la vista una serie de problemas que el dispositivo a través de la app debe resolver ¿cuáles herramientas de programación son las más adecuadas para erigir esta app? ¿de qué forma deben estar estructuradas? ¿su uso presentará problemas si diversos sistemas operativos son usados en la misma app?

La solución de estos y otros problemas es el planteamiento de este trabajo, la hipótesis radica en que existen procesos que pueden solucionarlos y la tesis observa la contemplación de la aplicación en funcionamiento.

Para todo ello es que he realizado todo un trabajo de investigación tanto de campo como teórico que ha cubierto la mayor parte de la actividad de esta tesis de posgrado, así como la erección de dicha aplicación que solucione los problemas de alimentación citados y la debida comprobación de su funcionamiento en el Hospital del Pacífico.

Capítulo 1 Introducción

1.1 Antecedentes del problema

El siguiente capítulo tiene como objetivo presentar el problema social que la aplicación que se tiene planeada laborar va a resolver.

Este problema, como se va a explicar a continuación expone una serie de situaciones mismas que implican un proceso de trabajo para la app debidamente especializado y que se pueden exponer de forma general como la satisfacción de las necesidades del personal de un nosocomio para hacer más eficiente el proceso diario de alimentación de su población de pacientes internos.

En dicho proceso tienen que observarse y resolverse problemas tales como los horarios, los espacios hospitalarios y los requerimientos alimenticios específicos de cada paciente, así como las dietas especiales que requieren para su recuperación enfermos que además de su estado precario de salud presentan características que los hacen más vulnerables tales como el padecimiento de enfermedades detonadas por Alergias, Intolerancias y la Diabetes, que son, tal como lo ha averiguado este trabajo de investigación aspectos de vital importancia que es imposible soslayar.

De esta manera las características de este problema se circunscriben a la forma en que éste se presenta para el personal y los pacientes del Hospital del Pacífico, sito en Acapulco Guerrero.

Hospital del Pacífico

El Hospital del Pacífico otorga servicios de atención médica y quirúrgica con calidad y seguridad, siendo sensible a las necesidades de las personas, promueve una cultura del cuidado de la salud y del medio ambiente, sustentado en la experiencia profesional y altamente competitiva de sus colaboradores.

El hospital cuenta con una estructura arquitectónica peculiar, siendo que el piso uno y dos son donde se encuentran las habitaciones de los pacientes y por el desnivel en el que se encuentra el terreno del hospital estos dos pisos están en una parte más baja que el nivel de calle, el piso tres es la entrada principal del hospital en el cual se encuentran la farmacia, la recepción, las oficinas administrativas y de recursos humanos, así como la sala de urgencias y la sala de espera, también se encuentran unas escaleras para el público en general que conducen al cuarto y quinto piso y unos elevadores que recorren los siete pisos de la torre médica, en la cual hay consultorios de diversos médicos y especialistas, en el área de urgencias también se encuentra otro elevador pero ese es para uso exclusivo del personal del hospital el cual recorre los cinco pisos propios del área del hospital, va desde el piso uno y dos que son las habitaciones y rayos x, pasando por tercer piso que es urgencias, cuarto piso que es pediatría y quinto piso que es el área de cocina y restaurante.

Las dietas en el hospital del Pacífico tienen que cumplir un papel muy importante en el proceso de recuperación del enfermo, por eso es importante que la gestión hospitalaria se encargue adecuadamente de la logística, planificación y elaboración de las dietas de acuerdo a las necesidades de cada uno de los pacientes.

Las Dietas que maneja el hospital son las siguientes:

- Dieta líquida: este régimen incluye líquidos claros, infusiones y zumos, es de bajo aporte calórico, por eso debe compensarse con sueroterapia. La dieta líquida está pensada para pacientes que salen del quirófano o que están por entrar en él.

- Dieta blanda: este tipo de dieta está conformada por alimentos cocidos o blandos y está pensada para pacientes que sienten pesadez tras la comida o con dificultades para masticar.

- Dieta blanda para diabético: este tipo de dieta está conformada por alimentos cocidos o blandos y está pensada para pacientes que sienten pesadez tras la comida o con dificultades para masticar aparte de que son sin azúcar por la condición de diabéticos.

- Dietas hiposódicas: Dieta baja en sal y alimentos altos en sodio. Están justificadas en pacientes anúricos (es la ausencia de orina en la vejiga), en anasarca (una forma de edema o acumulación de líquidos masiva y generalizada en todo el cuerpo) e hipertensión no controlada.

- Dieta a base de papillas: esta dieta se concentra en alimentos blandos hechos papilla para que el paciente pueda consumirlos sin problemas, por lo regular son de frutas o verduras de fácil absorción para el organismo.

- Dieta a complacencia: en esta dieta están disponibles todos los alimentos que se elaboran en el hospital.

- Dieta por sonda: esta dieta es diluida y licuada para que pueda ser administrada sin dificultad por un tubo.

- Dietas hipercalóricas: Altas en calorías y proteínas. Para pacientes con necesidades aumentadas por infecciones, traumas, desnutrición, estado crítico, quemados, entre otros.

- Dieta de 1800 kilocalorías: Es proporcionada en tres tomas en todo el día divididas dando un total de 1800 kilocalorías.

- Dieta astringente: por lo general este tipo de dieta se solicita para pacientes que sufren gastroenteritis, y está compuesta por arroz blanco, pollo, pescado y frutas cocidas, no debe contener ningún alimento fibroso ni irritantes intestinales (Medina-Hernández A, Huerta-Hernández RE et al, 2015).

1.2 Planteamiento del problema

Dentro del mundo de la comida hay muchos elementos a los que un porcentaje de personas son alérgicas. Se estima que las enfermedades alérgicas asociadas con alimentos se incrementan anualmente. Tienen una prevalencia de 2 a 4% en adultos y de 6 a 8% en niños (Medina-Hernández A, Huerta-Hernández RE et al, 2015).

Es importante no hacer restricciones dietéticas innecesarias sobre todo si las modificaciones llevan a bajos aportes en calorías y nutrientes que van a incidir negativamente sobre el estado nutricional (Clínica las Américas, 2017).

La alimentación hospitalaria tiene características muy especiales debido a que se relaciona con el aporte de nutrimentos específicos en personas en situaciones fisiopatológicas y en un estado anímico alterado por su situación y pronóstico de salud. En la actualidad se sigue planteando la hospitalización como una de las causas de desnutrición, en muchas ocasiones como elemento secundario o coadyuvante del deterioro de salud. Tal situación ocasiona graves trastornos inmunitarios y respuestas inadecuadas que dificultan los procesos de reparación y de rehabilitación ante enfermedades médicas o episodios quirúrgicos (IMSS, 2013).

Debido a los diferentes padecimientos de los pacientes, internados en el hospital, se requieren de varias dietas, si solo se tiene una cocina en el hospital se presentan los siguientes problemas:

Dietas

La dieta de un enfermo internado en el Hospital del Pacífico depende de la situación en la que se encuentre el paciente y es dictada por el médico que lo está atendiendo y el enfermero de guardia

tiene que asegurarse que no se le den alimentos que estén fuera de esa prescripción, pero siempre integrando una serie de elementos que sean nutritivos para la persona y le ayuden a recuperar sus fuerzas y energías más rápidamente.

Como ya se vio anteriormente el hospital maneja diversas dietas para los diversos problemas de los pacientes, estas se adecuan conforme las necesidades y problemas de cada paciente, es necesario que una dieta brinde los nutrimentos necesarios para la correcta alimentación del paciente y así contribuir a su pronta recuperación.

Alergias

Las alergias alimentarias se producen porqué nuestro sistema inmunitario percibe que una sustancia, en principio inofensiva para nuestro organismo, es nociva, y, en consecuencia, actúa de manera desproporcionada, provocando una serie de síntomas.

En ocasiones los pacientes son alérgicos a ciertos elementos que llevan estas comidas y no las pueden ingerir porque esta acción podría derivarles en más problemas, puesto que las alergias como ya se vio anteriormente si no es atendida de manera correcta y a tiempo puede llevar a la muerte.

Los síntomas de una alergia a un alimento incluyen: picazón e hinchazón de la boca, vómito, diarrea o cólicos abdominales y dolor, sarpullido o eccema, estornudos, tos, asma, sentir la garganta apretada y dificultad para respirar, disminución de la presión sanguínea, Shock anafiláctico, en casos extremos y si no se atiende a tiempo, la muerte. Sólo se pueden prevenir los síntomas de una reacción alérgica evitando el alimento que los cause (NIH, 2017).

Se han reportado alrededor de 175 alimentos alergénicos; sin embargo, ocho alimentos o grupos de alimentos –cacahuates, mariscos, leche, huevos, pescados, soya, trigo y nueces de

castilla, almendras y avellanas – son responsables del 90% de todas las alergias. Mientras que las alergias al huevo y a la leche de vaca suelen desaparecer, el resto de las alergias permanecen durante toda la vida.

Durante los últimos años, se ha visto un incremento de las enfermedades alérgicas, en particular de las causadas por los alimentos (Allergy Therapeutics, 2016).

Intolerancias

Una intolerancia es también una reacción adversa inducida por un alimento que no involucra al sistema inmunológico. La intolerancia es debida a un déficit de enzimas del organismo que impide metabolizarlo correctamente esto significa que el organismo no puede digerir correctamente algún componente del alimento. Las intolerancias son más comunes que las alergias y no representan una amenaza grave (Allergy Therapeutics, 2016). Sin embargo, las intolerancias también son muy recurrentes en la población.

Los síntomas de la intolerancia se suelen localizar en el tubo digestivo. Se manifiestan, en la mayoría de los casos con dolor abdominal, gases, vómitos, diarrea, retortijones y náusea.

Las consecuencias de estas no son tan extremas como las de las alergias dejando al paciente solo con temor a ingerir los alimentos y que estos les causen más problemas de los que ya tienen, evitando la pronta recuperación del paciente por no tener opciones para sustituir esos elementos que les causan problemas.

Los principales causantes de intolerancias alimentarias suelen ser ausencia de una enzima necesaria para digerir completamente un alimento, sensibilidad a los aditivos de los alimentos, colon irritable, estrés recurrente o factores psicológicos

Diabéticos

Así como las personas con problemas de alergias y de intolerancias, las personas con diabetes requieren una dieta específica, esto también se deriva en el tipo de diabetes que tengan, pidiendo así una mayor regulación sobre lo que pueden o no comer.

Con las personas diabéticas se tiene que formular una dieta acorde con su condición y que aparte le permita obtener los nutrimentos necesarios para su recuperación.

Estas dietas tienen que cuidar principalmente el consumo de azúcares para la regularización de la glucosa, entonces se tienen que buscar elementos que sustituyan estos azúcares y proporcionen los nutrimentos necesarios para el paciente.

Otros

Dentro de estos podemos señalar las dietas que necesitan los bebés, niños, adultos mayores, personas con problemas de obesidad y las personas con problemas no tan comunes o raros, de los que se tiene un registro muy bajo de incidencia.

Todos los problemas anteriormente mencionados provocan que los pacientes con esas necesidades alimenticias aún más especiales que la mayoría de los mismos se alimenten de una manera que repercute en su estado anímico al no recibir todos los valores nutrimentales necesarios para su pronta recuperación y provocando en ocasiones que los pacientes tengan recaídas o se les presenten problemas de otra índole.

También causa confusión entre el personal de cocina y el de entrega de alimentos porque si se le dice que un paciente no puede comer ciertas cosas genera un conflicto por que en muchas ocasiones los familiares del paciente le dicen lo que no puede comer al personal que entrega la

comida y estos a su vez o se les olvida o no reciben respuesta por parte de la cocina provocando así que el paciente se quede sin comer o solo coma lo que pueda de lo que le llevaron con anterioridad.

Así mismo si se atiende una petición de esta índole en la que implique el cambio de algún elemento por otro, se entra en conflicto con el inventario de la cocina puesto que éste está controlado rigurosamente.

El proceso de atención nutricional hospitalaria requiere de un análisis y evaluación de la producción de los alimentos. Es indispensable establecer y vigilar indicadores de evaluación en gestión y control del presupuesto, control de costos por comida y por total de alimentos producidos y control estadístico de dietas producidas

Los requerimientos están basados en el siguiente proceso de llegada y registro del paciente:

- El paciente llega.
- Se registra.
- Se le hace un estudio y encuesta inicial y el medico dicta su dieta respecto al padecimiento que presenta.
- Se hace el registro del paciente en el sistema y se le asigna su tipo dieta.
- El enfermero de guardia va a la habitación del paciente con la Tablet para que este haga la selección de los alimentos según la dieta que este ya tenga registrada, si el paciente está incapacitado para realizar la selección, lo hará el familiar asignado o el enfermero según sea el caso
- Una vez seleccionado el alimento se manda la petición a la cocina.

- En la sesión de cocina aparece la orden realizada y se procede a su preparación conforme se indique en el sistema y se le baja al paciente.

Al ingresar el paciente el enfermero de guardia le hace un estudio sobre el motivo de su estancia en el hospital, así como las enfermedades que padece independientemente de la razón de su visita, (las que ya sufre del diario) se toma en cuenta el por qué ingreso al hospital y de ahí se le informa al médico que lo vaya a atender y este indica la dieta que debe llevar en caso de que se tenga que quedar el paciente, una vez ingresado el paciente y ya sabiendo el tipo de dieta que lleva se registra en el sistema con sus datos básicos: nombre completo, edad, sexo, padecimiento, numero de cama, tipo de dieta, notas extras, nombre de usuario, contraseña y familiar a cargo.

En el caso de la cocina el proceso para el levantamiento de comandas que actualmente desarrollan es el siguiente: se manejan tres horarios para alimentar a los pacientes, el desayuno, la comida y la merienda cada uno con un horario específico, el desayuno a las 8 am, la comida a las 2 pm y la merienda a las 7 pm. Esto indica que para llevar las comidas a tiempo el personal de cocina tiene que llegar a las 7 am y pasar por las comandas de desayunos que se tomaron la noche anterior de los pacientes a recepción y prepararlos para tenerlos listos a la hora que se requieren, después en la comida y merienda el personal de cocina tiene que bajar a las habitaciones a decir el menú y dar las opciones según la dieta del paciente apoyados por el enfermero de guardia que vigila que los alimentos que quiera el paciente cumplan con la dieta establecida por el médico.

El hospital ya cuenta con dietas específicas para dar a los pacientes y estas dietas constan de ciertos alimentos que en su momento se preparan para el consumo de ese paciente, estas dietas se pueden ver en la Tabla 1.1 a continuación.

1	Líquida	Solo líquidos como Té de manzanilla, jugo de manzana, gelatina de piña
2	Blanda normal	Consta de pechuga de pollo, sopa, caldo de pollo, huevos al gusto, sincronizadas, sándwich, jugo, gelatina de sabores
3	Blanda para diabético	Contiene huevo (claras con espinacas o nopales), sopas de pasta, caldos con verdura, té, gelatina light, verdura al vapor, queso panela, fruta: pera, manzana. Todo sin azúcar o edulcorantes
4	Hiposódica normal	Tiene carne, caldos, sopas sin sal, sándwich de panela con pollo, té, jugo, gelatina de sabor
5	Papillas	Pueden ser verduras con pollo, arroz, frutas: manzana, pera, guayaba, papaya. Verduras: zanahoria, calabaza, chayote
6	Dieta a complacencia	Es a petición del paciente, pechuga, cecina, tacos dorados, sándwich, sincronizadas, huevos al gusto con jamón o salchicha, fruta, verduras, jugo de sabor y gelatina de sabor
7	Por sonda	Es diluida y licuada para que pase por el tubo sin dificultad (caldo de verdura, sopa con pollo, verduras, fruta licuadas con agua), jugo solo líquidos
8	Hipercalórica	Es a base de proteínas y carbohidratos, pechuga, pescado, carne, arroz, huevo, té, jugo, gelatinas
9	1800 kilocalorías	Es proporcionada en tres tomas en todo el día divididas dando un total de 1800 kcal., con base en pollo, verduras o frutas que sean licuadas.
10	Astringente	Es a base de pollo, carne, caldos, sopa, verduras, frutas, jugo, gelatina, te, sándwich de panela o pollo, (no lácteos, no grasas, o chiles ni irritantes)

Tabla 1-1 Dietas del Hospital del Pacífico

El hospital no cuenta con un departamento de sistemas como tal si no que requieren el apoyo de un ingeniero externo cada vez que algún equipo sufre un desperfecto, por lo cual el hospital no cuenta con la infraestructura de internet libre necesaria tanto en cocina como en las

habitaciones de los pacientes para la implementación del proyecto, por lo que se aplazará su puesta en marcha hasta que la administración del hospital implemente las condiciones correctas para el óptimo funcionamiento de la aplicación.

1.3 Objetivo general

Diseñar un sistema computacional interactivo apoyado por un algoritmo con reglas de asociación que proporcione al personal médico y de apoyo interno las diferentes posibilidades de cambios en las comidas conforme a las dietas que cada paciente tenga.

1.4 Objetivos específicos

- Desarrollar la aplicación web de un menú interactivo con un diseño responsivo para que pueda ser visualizado en cualquier dispositivo móvil.
- Diseñar y desarrollar un sistema de costeo de menús.
- Realizar una Base de Datos para el control y actualización de inventarios y menús.

1.5 Hipótesis

La implementación de un sistema computacional que va a trabajar en dispositivos móviles desde la habitación de un paciente para solicitar su comida con restricciones de tal forma que solo

le muestran las recetas correspondientes a la dieta que le asigno el médico y de la cual pudiese solicitar cambios en los elementos que no puede consumir por cuestiones médicas o de gustos.

1.6 Justificación

Impacto Social

Para brindar una mejor atención a los pacientes en cuanto a su alimentación, logrando que estos seleccionen desde su cama o habitación y por medio de dispositivos móviles ya sean smartphones o tablets sus alimentos tomando en cuenta su dieta ya preestablecida, pero sin poner en riesgo la salud ni el tiempo de recuperación.

Impacto Ambiental /Económico

Con la utilización de esta aplicación se reducirá la utilización de hojas de papel y tinta de impresoras, las cuales se ocupan para la elaboración de las comandas y las notas de venta, si estas se manejan a través del sistema supone un ahorro tanto económico como un apoyo ambiental.

Así como también se reduciría el uso del elevador por parte del personal de cocina el cual usan al ir a levantar las comandas a las habitaciones obteniendo así un ahorro de energía eléctrica.

Se evitará el desperdiciar comida por llevarle al paciente comida que contenga algo que no pueda comer.

Impacto Tecnológico

El uso de nuevas tecnologías web, alta disponibilidad y persistencia de datos que permitan una mayor ergonomía de uso y movilidad entre los pacientes debido al uso de la tecnología

responsive design que es el diseño Web adaptable, el cual responde a las necesidades de los usuarios y los dispositivos que estén usando.

1.7 Alcances y Limitaciones

Los alcances y limitaciones para el proceso de desarrollo del sistema involucran lo siguiente:

Como Alcances

- El sistema pretende que el paciente quede registrado con la dieta que le fue asignada al conocer su condición, misma que se le fue asignada por el médico que lo revisó.
- Se busca la optimización del proceso del levantamiento de comandas para los pacientes, puesto que actualmente es muy laborioso.
- Que el sistema cuente con un costeador de precio de la comida conforme a los ingredientes utilizados en la receta, así como también de un precio sugerido aumentando un porcentaje al costo por porción de cada receta.
- Creación de catálogos de ingredientes, comidas, enfermedades y dietas. Para lo anterior se necesita la creación de una base de datos para los ingredientes y las comidas según las dietas que maneja el hospital.

Como limitaciones del proyecto:

- El sistema no está vinculado al programa de administración general del hospital (Maines).
- El sistema no vincula a los pacientes entre ellos o con los médicos o enfermeros.
- El sistema no genera facturas, solo muestra el consumo de comidas del paciente y por lo tanto el total de su consumo en el restaurante.
- El sistema no recalcula el precio de las comidas si se le llega a cambiar algún ingrediente
- Los cambios de un ingrediente por otro con respecto a su gramaje dentro de la receta tendrán que ser avalados por un nutriólogo.

Capítulo 2 Estado del arte

El artículo “Metodología para el desarrollo de aplicaciones móviles” que presentan Mantilla et al. (Mantilla, 2013), presenta la evolución de los servicios de telefonía móvil en Latinoamérica contextualizados en las diferentes generaciones tecnológicas, las características del Software para dispositivos móviles y una propuesta de método de trabajo para el desarrollo de aplicaciones para móviles. El método se basa en la conceptualización de las tecnologías y las metodologías ágiles para el desarrollo de Software, y su objetivo principal es facilitar la creación de nuevas aplicaciones y servicios exitosos.

El método se desarrolla en cinco etapas: etapa de análisis, donde se obtienen y clasifican los requerimientos y se personaliza el servicio; etapa de diseño, momento en el que se define el escenario tecnológico y se estructura la solución por medio de algún diagrama o esquema, integrando tiempos y recursos; etapa de desarrollo, cuando se implementa el diseño en un producto de Software; etapa de prueba de Funcionamiento, donde se emula y simula el producto ajustando detalles, se instala en equipos reales y se evalúa el rendimiento, y posteriormente se evalúa el potencial de éxito; y finalmente, en la etapa de entrega, se define el canal de distribución de la aplicación, con el propósito de adecuar la aplicación al mismo. Además, el artículo presenta los resultados del desarrollo de un servicio de salud para Android y J2ME utilizando el método propuesto, el servicio está dirigido a pacientes que requieren o deseen tener un control periódico de las medidas corporales de tensión arterial y glucosa, servicio que obtuvo un potencial de éxito en los usuarios de prueba.

De este artículo me sirve la forma en que la metodología se describe para la realización de las aplicaciones móviles, puesto que lo que plantean resume de manera eficaz el desarrollo de una interfaz fácil y entendible para el usuario y también está dirigida al sector salud que es en el que se desarrolla mi proyecto

El artículo “Aprovechamiento del hardware de los dispositivos móviles para la construcción de nuevas aplicaciones” (Rodríguez et al., 2014), presenta el alto grado de inserción de dispositivos móviles hace que resulte de interés implementar aplicaciones que aprovechen el hardware de los mismos. Esta línea está enfocada en dispositivos móviles de alta gama, por ejemplo, teléfonos inteligentes, los cuales proveen una gran cantidad de sensores. Dentro del proyecto se planificó tanto el desarrollo de aplicaciones nativas como también aplicaciones web móviles usando HTML 5. Dado que con HTML 5 gradualmente está incorporando acceso a los sensores directamente desde la web.

Lo que proponen en este artículo está el sacar el mayor provecho posible a los celulares, como lo indica en la introducción está orientado a dispositivos de alta gama.

De este artículo me llamó la atención que dentro de los estudios que realizan en celulares de este tipo puedo tomar de referencia esos resultados y los procesos para compararlos con los resultados que tengo que obtener del análisis de los dispositivos de gama media que es en los que se desarrollará mi proyecto, así como la comparación entre aplicaciones nativas y en WEB.

En el artículo “Uso de herramienta libre para la generación de reglas de asociación, facilitando la gestión eficiente de incidentes e inventarios.” (Ing. Corso et al., 2012). El propósito de este trabajo es la presentación de una propuesta metodológica dentro de la rama de la minería de datos, relacionada con la generación de patrones de comportamientos de los incidentes de los equipos informáticos, detectados en el contexto del Laboratorio de Sistemas de Información, perteneciente al Departamento de Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional Facultad Regional Córdoba. A partir de la generación del modelo de conocimiento, el objetivo es brindar información estadística relacionada con los aspectos que tienen mayor incidencia en el reporte de incidentes de equipos informáticos, permitiendo elaborar un plan de prevención para disminuir la generación de reportes de incidentes y recomendaciones para una gestión eficiente de los inventarios relacionado con los componentes y piezas utilizando en el proceso de mantenimiento.

El artículo presenta una metodología con el uso de la herramienta KNIME para la generación de reglas de asociación y con eso facilitar la gestión de incidentes e inventarios, habla sobre las características de esa herramienta el tipo de datos que lee y el funcionamiento general y como es el procesamiento de los datos en la misma para la generación de las reglas de asociación.

Lo que plantea este artículo me da la posibilidad de conocer una nueva herramienta para tener una opción más a ocupar para la generación de las reglas de asociación, puesto que actualmente solo conozco la herramienta “RapidMiner” y “Weka”.

En el artículo “Métodos de programación segura en java para aplicaciones móviles en Android” (Pimienta et al., 2014). Con el auge del uso de los teléfonos móviles se debe tomar en cuenta la seguridad y se debe tener una programación que garantice que el uso de aplicaciones sea seguro, dando al usuario la certeza de que sus datos están protegidos. Es por ello que los programadores no sólo deben tener en cuenta que las aplicaciones cumplan con los requerimientos del proyecto, sino que al mismo tiempo con los estándares de seguridad que ofrecen distintas instituciones para garantizar la integridad, confidencialidad y disponibilidad de los datos del usuario, ya que una pequeña vulnerabilidad en el código del programa puede no sólo comprometer la información que esa aplicación maneja, sino todos los datos que se encuentran almacenados en el dispositivo.

Lo que me sirve de este artículo es que presenta los diferentes tipos de seguridad y en que consiste cada una de ellas y así como cada una es importante por si sola también se complementan entre sí, dentro de la seguridad de software está la seguridad en Java y en Android, la programación segura engloba la validación de entrada, las restricciones de credenciales, la autenticación, la autorización, la asignación de permisos.

En el artículo “Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. Estado actual” (Balaguera, 2013). Las metodologías ágiles han ganado popularidad desde hace algunos años, ya que constituyen una buena solución para proyectos a corto plazo, en especial, aquellos proyectos en donde los requisitos están cambiando constantemente, un ejemplo de esto son las aplicaciones para dispositivos móviles, debido a que éstas tienen que satisfacer una serie de características y condicionantes especiales, tales como: canal, movilidad, portabilidad, capacidades específicas de las terminales, entre otras, y aun cuando existen miles de aplicaciones

para dispositivos móviles que corren en diferentes sistemas operativos IOs, Android, BlackBerry y Windows Mobile; éstas llenan las expectativas de los usuarios hasta cierto punto por su escasa calidad en el desarrollo, ya que el uso de metodologías de desarrollo de software no se considera importante en este ámbito, por tanto, los desarrollos para dispositivos móviles, hasta el momento, se han venido realizando, principalmente, de manera desordenada y en la mayoría de los casos por desarrolladores individuales que no aplican métodos de ingeniería de software que garanticen su mantenibilidad y por lo tanto su calidad.

Dentro de los temas que desarrolla se encuentran las apreciaciones teóricas, la definición de las metodologías de desarrollo, entre ellas las metodologías ágiles como las siguientes: Extreme Programing (XP), Scrum, Test Driven Development (TDD), también el desarrollo de aplicaciones para dispositivos móviles, las generalidades del desarrollo de aplicaciones para dispositivos móviles, los sistemas operativos para dispositivos móviles tales como: IOS de Apple, BlackBerry OS de BlackBerry (antes RIM), Windows phone de Microsoft, así como las metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles en las que explica las siguientes: Metodologías usadas actualmente para el desarrollo de aplicaciones móviles, Mobile-D, Hybrid Methodology Design y Mobile Development Process Spiral

Lo que me dejó este artículo es que es necesaria la aplicación de métodos de ingeniería de software que garanticen la mantenibilidad y calidad de las aplicaciones móviles para su correcto funcionamiento a lo largo de su uso, y que así mismo su actualización y corrección de errores sea más ordenada y más rápida de realizar, puesto que ya se cuenta con un procedimiento a seguir.

En el artículo “Algoritmos de clasificación y redes neuronales en la observación automatizada de registros” (González, 2015). El objetivo del artículo es mostrar los resultados después de analizar los datos obtenidos a través de una plataforma *on-line*, mediante diferentes técnicas de clasificación y aprendizaje orientadas al descubrimiento del conocimiento. Se aplican técnicas de “Minería de Datos” para obtener relaciones de fiabilidad que informen del interés de los usuarios por cumplimentar de manera rigurosa el cuestionario on-line atendiendo al modo de realizar el mismo. Aunque existen técnicas que nos permiten observar el comportamiento de los usuarios mientras realizan el cuestionario, en este caso se emplean “Redes Neuronales Artificiales” para predecir el comportamiento de aquellos, atendiendo a variables obtenidas al realizar el cuestionario.

Las reglas de asociación también pueden ser usadas en otros contextos, incluyendo análisis Web, detección de intrusos, producción continua y bioinformática. Un problema asociado a esta técnica de descubrimiento del conocimiento es que, en todas las aplicaciones, el resultado será abultado; es decir, encontraremos un grandísimo conjunto de reglas de asociación. Sin embargo, es importante realizar un análisis minucioso con el fin de realizar un proceso de eliminación de aquellas reglas obvias para el conocimiento, y mantener aquellas que son de interés para el estudio.

Para seleccionar las reglas interesantes de todo el conjunto generado por la aplicación, existen algunas restricciones en varias medidas de significancia e interés que pueden ser usadas. Las restricciones más conocidas son umbrales mínimos de soporte y confianza. El Soporte $Supp(X)$ de un *itemset* X (*itemset* es un conjunto de atributos) es definido como la proporción de transacciones en el conjunto de datos que contienen al *itemset* X. La confianza de una regla también es necesaria para la consideración de ésta como válida

En relación a la obtención de las Reglas de Asociación, WEKA mantiene como herramienta de análisis el algoritmo a priori. Hay que señalar que cualquier algoritmo o procedimiento aplicado para la obtención de Reglas de Asociación puede generar una cantidad importante de éstas.

Lo que presenta el artículo me puede servir en la manera en la que analizan los datos y los procesan para la generación de las redes de asociación implementando también otros métodos de minería de datos y de inteligencia artificial, cosas que probablemente tenga que aplicar para mi proyecto también.

En el artículo “Patrones de actividades humanas mineras de Big Data Smart para aplicaciones de atención médica” (Abdulsalam et al., 2017). Hoy en día, hay una migración cada vez mayor de personas a las áreas urbanas. El servicio de atención médica es uno de los aspectos más desafiantes que se ve muy afectado por la gran afluencia de personas a los centros de las ciudades.

La demanda de recursos de atención médica se verá muy afectada a futuro por esta gran afluencia de personas a los centros de las ciudades. Este cambio demográfico sin precedentes impone una carga enorme a las ciudades para repensar los enfoques tradicionales de proporcionar servicios de salud a los residentes. En respuesta a las nuevas necesidades y desafíos, las ciudades están adoptando una transformación digital masiva en un esfuerzo por apoyar a las comunidades urbanas sostenibles y proporcionar un entorno más saludable.

En dicha transformación, millones de hogares están siendo equipados con dispositivos inteligentes (por ejemplo, medidores inteligentes, sensores.) que generan volúmenes masivos de datos de grano fino e indexados que se pueden analizar para respaldar los servicios de atención médica.

El avance de las tecnologías de minería de datos grandes, que proporcionan medios para procesar una gran cantidad de datos para obtener información procesable, pueden ayudarnos a comprender cómo las personas llevan a cabo su vida. Por ejemplo, monitorear los cambios en el uso del artefacto dentro de un hogar inteligente puede usarse para determinar indirectamente el bienestar de la persona en base a los datos históricos. Dado que los hábitos de las personas se identifican principalmente por las rutinas diarias, descubrir estas rutinas nos permite reconocer actividades anómalas que pueden indicar las dificultades de las personas para cuidarse, como no preparar la comida o no usar la ducha / baño.

La correlación subyacente entre el uso del dispositivo dentro del hogar inteligente y las actividades de rutina puede ser utilizada por las aplicaciones de atención médica para detectar posibles problemas de salud. Esto no solo aliviará la carga de los sistemas de atención médica, sino que también proporcionará un servicio de monitoreo las 24 horas que identifica automáticamente los comportamientos normales y anormales para los pacientes que viven independientemente o aquellos con condiciones autolimitantes (por ejemplo, ancianos y pacientes con discapacidades cognitivas).

En este artículo muestran el estudio realizado y los resultados obtenidos, así como su procesamiento por la minería de datos, en el cual muestran esquemas, gráficas, tablas de datos, algoritmos, fórmulas utilizadas que se utilizaron para el desarrollo del proyecto y también para

mostrar sus resultados de manera más entendible, esto posiblemente me sea de ayuda para tener una base sobre la cual presentar mis resultados y la manera en que los puedo manipular para la obtención de los resultados finales una vez que estos ya hayan pasado por los diversos procesos de minería de datos.

En el artículo “Identificación de reglas recurrentes de asociación en la predicción de defectos de software” (Watanabe et al. 2016). Las reglas extraídas se usan generalmente para mejorar las actividades actuales mediante la identificación de patrones recurrentes en el conjunto de datos anterior. Si se espera que un patrón se repita en el futuro, las acciones se pueden planificar en respuesta al patrón. De lo contrario, si un patrón no se repite en el futuro, dicho patrón es inútil y el plan de acción se vuelve incluso dañino. En nuestro caso, buscamos patrones recurrentes de módulos propensos a defectos en la versión anterior de un producto de software para que los módulos que coinciden con el patrón en la próxima versión sean probados exhaustivamente antes de liberarlos en el campo.

El objetivo de este estudio es encontrar dichos criterios para identificar las reglas de asociación altamente recurrentes a partir de una enorme cantidad de reglas que generalmente se extraen de un gran conjunto de datos. En términos generales, las reglas recurrentes altas deben tener valores de soporte altos, es decir, las reglas deben coincidir con frecuencia con los casos en el conjunto de datos, porque es probable que el mismo evento ocurra una y otra vez si sucedió muchas veces en el pasado. Sin embargo, no hay pautas sobre el límite inferior del valor de soporte que debe enfocarse. Además, dado que el valor de soporte depende en gran medida del tamaño del conjunto de datos, creemos que este valor no es una medida adecuada para distinguir las reglas

recurrentes altas en conjuntos de datos variables. En este documento, se define "tasa de disminución de la confianza (RDR)", que es una medida para evaluar la recurrencia esperada de una regla. Luego, utilizando el conjunto de datos de defectos de Eclipse Mylyn, el objetivo es aclarar experimentalmente la línea base para distinguir entre las reglas RDR altas y las RDR bajas en función del número de transacciones y la confianza de una regla. En el experimento, se usa el número de transacciones en lugar de los valores de soporte de las reglas.

En este documento se definió la recurrencia de las reglas de asociación y se propuso una métrica llamada tasa de disminución de la confianza (CDR) para cuantificar la recurrencia de una regla. También se propuso un método para identificar reglas recurrentes altas / bajas basadas en la validación cruzada en el conjunto de datos dado para extraer las reglas de asociación. A través de un estudio de caso con el conjunto de datos de Eclipse Mylyn, se mostro cómo funciona el método propuesto y se puede usar para identificar reglas recurrentes bajas. En el caso de Mylyn, podríamos identificar las siguientes características relacionadas con la recurrencia de las reglas: A medida que las transacciones se vuelven más grandes, la CDR disminuye, lo que significa que las reglas se vuelven altamente recurrentes en promedio.

El artículo “*icuARM: Un sistema de apoyo a la toma de decisiones clínicas de la UCI que utiliza la regla de asociación en minería de datos*” (Cheng et al., 2013), habla de la toma de decisiones en tiempo real basada en la evidencia para pacientes críticamente enfermos en la UCI (Unidad de Cuidados Intensivos) se ha vuelto más difícil debido a que el volumen y la complejidad de los datos han ido aumentando a lo largo de los años. Por lo tanto, para ayudar a los médicos a tomar decisiones óptimas, existe una necesidad crítica de aplicar tecnología de la información

moderna y análisis de datos avanzados para extraer información de datos clínicos heterogéneos. En este estudio, se investiga y desarrolla un sistema de apoyo a la decisión clínica en tiempo real icuARM para ayudar a los médicos a generar reglas de apoyo a decisiones cuantitativas y en tiempo real para la UCI basadas en una gran base de datos de pacientes de la UCI MIMICII. Se adoptan las métricas de " apoyo " y " confianza " adecuadas para la aplicación clínica de la UCI a partir de la extracción convencional de reglas de asociación. Además, se define y desarrollan dos nuevas métricas basadas en reglas " importancia " y " dominancia " y un " efecto de medida ". Se desarrolló una interfaz gráfica de usuario interactiva y fácil de usar que permite a los médicos realizar una minería de datos flexible en tiempo real para la toma de decisiones personalizada. Se probó icuARM en dos casos que investigaron las asociaciones entre estadías prolongadas en la UCI y datos demográficos de pacientes, la presencia de una o más trastornos o enfermedades preexistentes y uso de medicamentos. Los resultados no solo reforzaron la evidencia actual de toma de decisiones, sino que también revelaron nuevos conocimientos al predecir las características de una estadía prolongada en la UCI.

En el artículo viene el pseudocódigo de un algoritmo para la generación de reglas de asociación el cual se puede implementar en mi sistema, también cuenta con fórmulas y diagramas que muestran los procesos a realizar, así como gráficas y tablas de información que muestran los resultados de ese estudio.

Capítulo 3 Marco teórico

Los modelos de aprendizaje supervisado son aquellos en los que se aprenden funciones, relaciones que asocian entradas con salidas, por lo que se ajustan a un conjunto de ejemplos de los que conocemos la relación entre la entrada y la salida deseada. Este hecho incluso llega a proporcionar una de las clasificaciones más habituales en el tipo de algoritmos que se desarrollan, así, dependiendo del tipo de salida, suele darse una subcategoría que diferencia entre modelos de clasificación, si la salida es un valor categórico (por ejemplo, una enumeración, o un conjunto finito de clases), y modelos de regresión, si la salida es un valor de un espacio continuo.

Los modelos de aprendizaje no supervisado son aquellos en los que no estamos interesados en ajustar pares (entrada, salida), sino en aumentar el conocimiento estructural de los datos disponibles (y posibles datos futuros que provengan del mismo fenómeno), por ejemplo, dando una agrupación de los datos según su similaridad (clustering), simplificando la estructura de los mismos manteniendo sus características fundamentales (como en los procesos de reducción de la dimensionalidad), o extrayendo la estructura interna con la que se distribuyen los datos en su espacio original (aprendizaje topológico). (Sevilla, 2020)

3.1 Reglas de asociación

Los algoritmos de reglas de asociación tienen como objetivo encontrar relaciones dentro un conjunto de transacciones, en concreto, *items* o atributos que tienden a ocurrir de forma conjunta. En este contexto, el término transacción hace referencia a cada grupo de eventos que están asociados de alguna forma (Amat, 2018).

A cada uno de los eventos o elementos que forman parte de una transacción se le conoce como *item* y a un conjunto de ellos *itemset*. Una transacción puede estar formada por uno o varios *items*, en el caso de ser varios, cada posible subconjunto de ellos es un *itemset* distinto. Por ejemplo, la transacción $T = \{A, B, C\}$ está formada por 3 *items* (A , B y C) y sus posibles *itemsets* son: $\{A, B, C\}$, $\{A, B\}$, $\{B, C\}$, $\{A, C\}$, $\{A\}$, $\{B\}$ y $\{C\}$.

Una regla de asociación se define como una implicación del tipo “si X entonces Y ” ($X \Rightarrow Y$), donde X e Y son *itemsets* o *items* individuales. El lado izquierdo de la regla recibe el nombre de antecedente o *left-hand-side (LHS)* y el lado derecho el nombre de consecuente o *right-hand-side (RHS)*. Por ejemplo, la regla $\{A, B\} \Rightarrow \{C\}$ significa que, cuando ocurren A y B , también ocurre C .

3.1.1 Algoritmo Apriori

Apriori fue uno de los primeros algoritmos desarrollados para la búsqueda de reglas de asociación y sigue siendo uno de los más empleados, tiene dos etapas:

- Identificar todos los *itemsets* que ocurren con una frecuencia por encima de un determinado límite (*itemsets* frecuentes).
- Convertir esos *itemsets* frecuentes en reglas de asociación.

Con la finalidad de ilustrar el funcionamiento del algoritmo, se emplea un ejemplo sencillo. Supóngase la siguiente base de datos de un centro comercial en la que cada fila es una transacción. En este caso, el término transacción hace referencia a todos los productos comprados bajo un mismo ticket (misma cesta de la compra). Las letras A, B, C y D hacen referencia a 4 productos (*items*) distintos.

Transacción
{A, B, C, D}
{A, B, D}
{A, B}
{B, C, D}
{B, C}
{C, D}
{B, D}

Tabla 3-1 Ejemplo, BD de un centro comercial

Antes de entrar en los detalles del algoritmo, conviene definir una serie de conceptos:

- Soporte: El soporte del *item* o *itemset* X es el número de transacciones que contienen X dividido entre el total de transacciones.
- Confianza: La confianza de una regla “Si X entonces Y” se define acorde a la ecuación

$$\text{Confianza}(X \Rightarrow Y) = \text{Soporte}(\text{unión}(X, Y)) / \text{Soporte}(X),$$

donde $\text{unión}(X, Y)$ es el *itemset* que contienen todos los *items* de X y de Y. La confianza se interpreta como la probabilidad $P(Y|X)$, es decir, la probabilidad de que una transacción que contiene los *items* de X, también contenga los *items* de Y.

Volviendo al ejemplo del centro comercial, puede observarse que, el artículo A, aparece en 3 de las 7 transacciones, el artículo B en 6 y ambos artículos juntos en 3. El soporte del *item* {A} es por lo tanto del 43%, el del *item* {B} del 86% y del *itemset* {A, B} del 43%. De las 3 transacciones que incluyen A, las 3 incluyen B, por lo tanto, la regla “clientes que compran el artículo A también compran B”, se cumple, acorde a los datos, un 100% de las veces. Esto significa que la confianza de la regla $\{A \Rightarrow B\}$ es del 100%.

Encontrar *itemsets* frecuentes (*itemsets* con una frecuencia mayor o igual a un determinado soporte mínimo) no es un proceso trivial debido a la explosión combinatoria de posibilidades, sin embargo, una vez identificados, es relativamente directo generar reglas de asociación que presenten una confianza mínima. El algoritmo *Apriori* hace una búsqueda exhaustiva por niveles de complejidad (de menor a mayor tamaño de *itemsets*). Para reducir el espacio de búsqueda aplica la norma de “si un *itemset* no es frecuente, ninguno de sus *supersets* (*itemsets* de mayor tamaño que contengan al primero) puede ser frecuente”. Visto de otra forma, si un conjunto es infrecuente, entonces, todos los conjuntos donde este último se encuentre, también son infrecuentes. Por ejemplo, si el *itemset* {A, B} es infrecuente, entonces, {A, B, C} y {A, B, E} también son infrecuentes ya que todos ellos contienen {A, B}.

El funcionamiento del algoritmo es sencillo, se inicia identificando los *items* individuales que aparecen en el total de transacciones con una frecuencia por encima de un mínimo

establecido por el usuario. A continuación, se sigue una estrategia *bottom-up* en la que se extienden los candidatos añadiendo un nuevo *item* y se eliminan aquellos que contienen un subconjunto infrecuente o que no alcanzan el soporte mínimo. Este proceso se repite hasta que el algoritmo no encuentra más ampliaciones exitosas de los *itemsets* previos o cuando se alcanza un tamaño máximo.

Se procede a identificar los *itemsets* frecuentes y, a partir de ellos, crear reglas de asociación.

Para este problema se considera que un *item* o *itemset* es frecuente si aparece en un mínimo de 3 transacciones, es decir, su soporte debe de ser igual o superior a $3/7 = 0.43$. Se inicia el algoritmo identificando todos los *items* individuales (*itemsets* de un único *item*) y calculando su soporte.

Itemset (k=1)	Ocurrencias	Soporte
{A}	3	0.43
{B}	6	0.86
{C}	4	0.57
{D}	5	0.71

Tabla 3-2 Items individuales con su soporte

Todos los *itemsets* de tamaño $k = 1$ tienen un soporte igual o superior al mínimo establecido, por lo que todos superan la fase de filtrado.

A continuación, se generan todos los posibles *itemsets* de tamaño $k = 2$ que se pueden crear con los *itemsets* que han superado el paso anterior y se calcula su soporte.

Itemset (k=2)	Ocurrencias	Soporte
{A, B}	3	0.43
{A, C}	1	0.14
{A, D}	2	0.29
{B, C}	3	0.43
{B, D}	4	0.57
{C, D}	3	0.43

Tabla 3-3 Itemsets de tamaño $k=2$

Los *itemsets* {A, B}, {B, C}, {B, D} y {C, D} superan el límite de soporte, por lo que son frecuentes. Los *itemsets* {A, C} y {A, D} no superan el soporte mínimo por lo que se descartan. Además, cualquier futuro *itemset* que los contenga también será descartado ya que no puede ser frecuente por el hecho de que contiene un subconjunto infrecuente.

Itemset (k=2)	Ocurrencias	Soporte
{A, B}	3	0.43
{B, C}	3	0.43
{B, D}	4	0.57
{C, D}	3	0.43

Tabla 3-4 Itemsets frecuentes de tamaño $k=2$

Se repite el proceso, esta vez creando *itemsets* de tamaño $k = 3$.

Itemset (k=3)
{A, B, C}
{A, B, D}
{B, C, D}
{C, D, A}

Tabla 3-5 itemsets de tamaño $k=3$

Los *itemsets* {A, B, C}, {A, B, D} y {C, D, A} contienen subconjuntos infrecuentes, por lo que son descartados. Para los restantes se calcula su soporte.

Itemset (k=3)	Ocurrencias	Soporte
{B, C, D}	2	0.29

Tabla 3-6 itemsets frecuentes de tamaño $k=3$

El *items* {B, C, D} no supera el soporte mínimo por lo que se considera infrecuente. Al no haber ningún nuevo *itemset* frecuente, se detiene el algoritmo.

Como resultado de la búsqueda se han identificado los siguientes *itemsets* frecuentes:

Itemset frecuentes
{A, B}
{B, C}
{B, D}
{C, D}

Tabla 3-7 itemsets frecuentes al final del procedimiento

El siguiente paso es crear las reglas de asociación a partir de cada uno de los *itemsets* frecuentes. De nuevo, se produce una explosión combinatoria de posibles reglas ya que, de cada *itemset* frecuente, se generan tantas reglas como posibles particiones binarias. En concreto, el proceso seguido es el siguiente:

1. Por cada *itemset* frecuente I , obtener todos los posibles subsets de I .

1.1 Para cada subset s de I , crear la regla “ $s \Rightarrow (I-s)$ ”

2. Descartar todas las reglas que no superen un mínimo de confianza.

Supóngase que se desean únicamente reglas con una confianza igual o superior a 0.7, es decir, que la regla se cumpla un 70% de las veces. Tal y como se describió anteriormente, la confianza de una regla se calcula como el soporte del *itemset* formado por todos los *items* que participan en la regla, dividido por el soporte del *itemset* formado por los *items* del antecedente.

Reglas	Confianza	Confianza
{A} => {B}	soporte{A, B} / soporte {A}	0.43 / 0.43 = 1
{B} => {A}	soporte{A, B} / soporte {B}	0.43 / 0.86 = 0.5
{B} => {C}	soporte{B, C} / soporte {B}	0.43 / 0.86 = 0.5
{C} => {B}	soporte{B, C} / soporte {C}	0.43 / 0.57 = 0.75
{B} => {D}	soporte{B, D} / soporte {B}	0.43 / 0.86 = 0.5
{D} => {B}	soporte{B, D} / soporte {D}	0.43 / 0.71 = 0.6
{C} => {D}	soporte{C, D} / soporte {C}	0.43 / 0.57 = 0.75
{D} => {C}	soporte{C, D} / soporte {D}	0.43 / 0.71 = 0.6

Tabla 3-8 Reglas obtenidas de los *itemsets* frecuentes

De todas las posibles reglas, únicamente $\{C\} \Rightarrow \{D\}$ y $\{C\} \Rightarrow \{B\}$ superan el límite de confianza.

La principal desventaja de algoritmo *Apriori* es el número de veces que se tienen que escanear los datos en busca de los *itemsets* frecuentes, en concreto, el algoritmo escanea todas las transacciones un total de $k_{max}+1$, donde k_{max} es el tamaño máximo de *itemset* permitido. Esto hace que el algoritmo *Apriori* no pueda aplicarse en situaciones con millones de registros.

3.1.2 Ejercicio en R

El siguiente ejercicio tiene como objeto ejemplificar la manera en que el algoritmo a priori funciona ya en la programación, el lenguaje que se utiliza es R y RStudio es su IDE dedicado a la computación estadística y gráficos.

Para este ejercicio ejecutado en programación R se emplea el set de datos *dd* (ver anexo) con el paquete *arules*, que contiene el registro de algunos artículos de comida que se manejan en el Hospital del Pacífico, en total se dispone de 64 transacciones formadas por combinaciones de 15 ingredientes, cada ingrediente tiene su valor por porción de 100 gr del mismo, los elementos que se tomaron en cuenta para su valoración son la cantidad de Hidratos, Proteínas, Grasas, Agua, Fibra y Kcal que contienen por cada 100 grs. El objeto *dd* almacena la información en un formato .csv (delimitado por comas).

Cada línea del archivo contiene la información de un *ítem* y el identificador de la transacción a la que pertenece.

```

> library(arules)
>
> library(tidyverse)
> datos <- read_csv(file = "~/articulos/dd.csv", col_names = TRUE)
> head(datos)

# A tibble: 6 x 2
  valores articulo
  <chr>    <chr>
1 H 9     fresa
2 H 9     limon
3 H 12    melon
4 H 13    pera
5 H 1.5   aguacate
6 H 4     nuez

```

Con la función *read.transactions* se pueden leer directamente los datos de archivos tipo texto y almacenarlos en un objeto tipo *transactions*, que es la estructura de almacenamiento que emplea *arules*.

```

> # IMPORTACIÓN DIRECTA DE LOS DATOS A UN OBJETO TIPO TRANSACTION
> # =====
===
> library(arules)
> transacciones <- read.transactions(file = "~/articulos/dd.csv",
+                                   format = "single",
+                                   sep = ",",
+                                   header = TRUE,
+                                   cols = c("valores", "articulo"),
+                                   rm.duplicates = TRUE)
> transacciones
transactions in sparse format with
 64 transactions (rows) and
 15 items (columns)

```

También es posible convertir un objeto *dataframe* en uno de tipo *transactions* con la función *as(dataframe, "transactions")*. Para lograrlo, primero hay que convertir el *dataframe* en una lista en la que cada elemento contiene los *items* de una transacción. Este proceso puede ser

muy lento si el *dataframe* tiene muchos registros, por lo que suele ser mejor crear un archivo de texto con los datos e importarlo mediante *read.transactions()*.

```
> # CONVERSIÓN DE UN DATAFRAME A UN OBJETO TIPO TRANSACTION
> # =====
> # Se convierte el dataframe a una lista en la que cada elemento contiene
los items de una transacción

> datos_split <- split(x = datos$articulo, f = datos$valores)
> transacciones <- as(datos_split, Class = "transactions")
> transacciones
transactions in sparse format with
 64 transactions (rows) and
 15 items (columns)
```

Exploración de items

Uno de los primeros análisis que conviene realizar cuando se trabaja con transacciones es explorar su contenido y tamaño, es decir, el número de *items* que las forman y cuáles son. Mediante la función *inspect()* se muestran los *items* que forman cada transacción.

```
> inspect(transacciones [1:5])
  items      transactionID
[1] {huevo crudo} A 112
[2] {avellana}    A 16
[3] {nuez}        A 18
[4] {cerdo,pollo} A 71
[5] {aguacate}   A 80
```

Para extraer el tamaño de cada transacción se emplea la función *size()*.

```
> tamanos <- size(transacciones)
> summary(tamanos)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.00   1.00   1.00   1.25   1.00   3.00
```

```
> quantile(tamanyos, probs = seq(0,1,0.1))
 0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
  1   1   1   1   1   1   1   1   2   2   3
```

El siguiente análisis básico consiste en identificar cuáles son los *items* más frecuentes (los que tienen mayor soporte) dentro del conjunto de todas las transacciones. Con la función *itemFrequency()* se puede extraer esta información de un objeto tipo *transactions*. El nombre de esta función puede causar confusión. Por “frecuencia” se hace referencia al soporte de cada *item*, que es la fracción de transacciones que contienen dicho *item* respecto al total de todas las transacciones. Esto es distinto a la frecuencia de un *item* respecto al total de *items*, de ahí que la suma de todos los soportes no sea 1.

```
> frecuencia_items <- itemFrequency(x = transacciones, type = "relative")
> frecuencia_items %>% sort(decreasing = TRUE) %>% head(5)
aguacate almendra avellana  fresa  limon
0.09375  0.09375  0.09375  0.09375  0.09375
```

Si se indica el argumento *type = "absolute"*, la función *itemFrequency()* devuelve el número de transacciones en las que aparece cada *item*.

```
> frecuencia_items <- itemFrequency(x = transacciones, type = "absolute")
> frecuencia_items %>% sort(decreasing = TRUE) %>% head(5)
aguacate almendra avellana  fresa  limon
      6      6      6      6      6
```

Itemsets

Con la función *apriori()* se puede aplicar el algoritmo *Apriori* a un objeto de tipo *transactions* y extraer tanto *itemsets* frecuentes como reglas de asociación que superen un determinado soporte y confianza.

Se procede a extraer aquellos *itemsets*, incluidos los formados por un único *item*, que hayan sido encontrados al menos 0.5 veces. En un caso real, este valor sería excesivamente bajo si se tiene en cuenta la cantidad total de transacciones, sin embargo, se emplea 0.5 para que en los resultados aparezcan un número suficiente de *itemsets* y reglas de asociación que permitan mostrar las posibilidades de análisis que ofrece el paquete *arules*.

```
> soporte <- 0.5 / dim(transacciones)[1]
> itemsets <- apriori(data = transacciones,
                     parameter = list(support = soporte,
                                     minlen = 1,
                                     maxlen = 20,
                                     target = "frequent itemset"))
```

Apriori

Parameter specification:

```
confidence minval smax arem aval originalsupport maxtime support minlen axlen
NA          0.1   1  none FALSE      TRUE           5   0.0078125  1    20
          target  ext
frequent itemsets FALSE
```

Algorithmic control:

```
filter tree heap memopt load sort verbose
0.1 TRUE TRUE  FALSE TRUE   2    TRUE
```

Absolute minimum support count: 0

```
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[15 item(s), 64 transaction(s)] done [0.00s].
sorting and recoding items ... [15 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [28 set(s)] done [0.00s].
creating S4 object ... done [0.00s].
```

```
> summary(itemsets)
```

```

set of 28 itemsets

most frequent items:
  limon almendra   cerdo   fresa   melon (Other)
    5         4         4         4         4         22

element (itemset/transaction) length distribution:sizes
 1 2 3
15 11 2

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1.000  1.000   1.000   1.536  2.000   3.000

summary of quality measures:
  support          count
Min.   :0.01562   Min.   :1.000
1st Qu.:0.01562   1st Qu.:1.000
Median :0.06250   Median :4.000
Mean   :0.05580   Mean   :3.571
3rd Qu.:0.09375   3rd Qu.:6.000
Max.   :0.09375   Max.   :6.000

includes transaction ID lists: FALSE

mining info:
      data ntransactions  support confidence
transacciones           64 0.0078125          1

```

Se han encontrado un total de 28 *itemsets* frecuentes que superan el soporte mínimo de 0.5, En el siguiente listado se muestran los 28 *itemsets* que conforman los más frecuentes con mayor soporte que, como cabe esperar, son los formados por *items* individuales (los *itemsets* de menor tamaño).

```

> top_itemsets <- sort(itemsets, by = "support", decreasing = TRUE)
> inspect(top_itemsets)

```

```

  items          support count
[1] {avellana}    0.093750 6
[2] {nuez}        0.093750 6
[3] {aguacate}    0.093750 6
[4] {zanahoria}   0.093750 6
[5] {tomate}       0.093750 6
[6] {almendra}    0.093750 6
[7] {fresa}       0.093750 6
[8] {limon}       0.093750 6
[9] {leche}       0.078125 5
[10] {cebolla}    0.078125 5
[11] {melon}      0.078125 5

```

[12]	{pera}	0.078125	5
[13]	{huevo crudo}	0.062500	4
[14]	{cerdo}	0.062500	4
[15]	{pollo}	0.062500	4
[16]	{melon,pera}	0.062500	4
[17]	{cerdo,pollo}	0.046875	3
[18]	{fresa,limon}	0.046875	3
[19]	{aguacate,cebolla}	0.015625	1
[20]	{avellana,nuez}	0.015625	1
[21]	{fresa,zanahoria}	0.015625	1
[22]	{fresa,tomate}	0.015625	1
[23]	{almendra,cerdo}	0.015625	1
[24]	{almendra,pollo}	0.015625	1
[25]	{limon,melon}	0.015625	1
[26]	{limon,pera}	0.015625	1
[27]	{almendra,cerdo,pollo}	0.015625	1
[28]	{limon,melon,pera}	0.015625	1

```
# Para representarlos con ggplot se convierte a dataframe
as(top_itemsets, Class = "data.frame") %>%
  ggplot(aes(x = reorder(items, support), y = support)) +
  geom_col() +
  coord_flip() +
  labs(title = "Itemsets más frecuentes", x = "itemsets") +
  theme_bw()
```

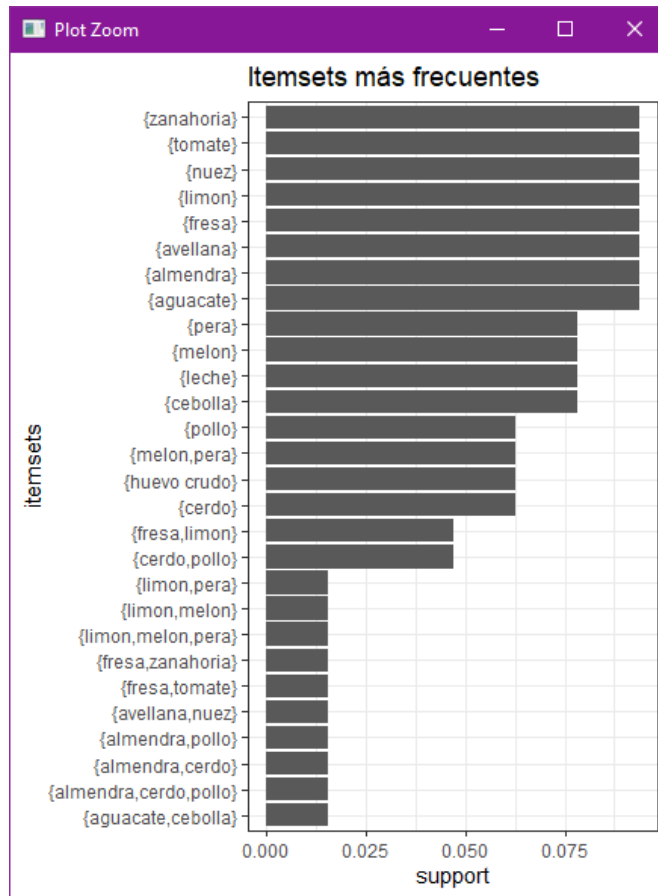



Figura 3-1 Itemsets más frecuentes

Filtrado de itemsets

Una vez que los *itemsets* frecuentes han sido identificados mediante el algoritmo *Apriori*, pueden ser filtrados con la función *subset()*. Esta función recibe dos argumentos: un objeto *itemset* o *rules* y una condición lógica que tienen que cumplir las reglas/*itemsets* para ser seleccionados. La siguiente tabla muestra los operadores permitidos:

Operador	Significado
&	AND
%in%	contiene cualquier de los siguientes elementos
%ain%	contiene todos de los siguientes elementos
%pin%	contiene parcialmente los siguientes elementos

Tabla 3-9 Operadores para filtrado de items

Como esta función tiene el mismo nombre que una función del paquete básico de R, para evitar errores, es conveniente especificar el paquete donde se encuentra.

Se procede a identificar aquellos *itemsets* frecuentes que contienen el *item* “limon”.

```
> itemsets_filtrado <- arules::subset(itemsets,subset = items %ain% "limon")
> itemsets_filtrado
set of 5 itemsets

> # Se muestran
> inspect(itemsets_filtrado[1:5])
  items          support count
[1] {limon}          0.093750  6
[2] {limon,melon}    0.015625  1
[3] {limon,pera}     0.015625  1
[4] {fresa,limon}    0.046875  3
[5] {limon,melon,pera} 0.015625  1
```

Se repite el proceso, pero, esta vez, con aquellos *itemsets* que contienen “limon” y “melon”

```
> itemsets_filtrado <- arules::subset(itemsets,subset = items %ain% c("limon",
,"melon"))
> itemsets_filtrado
set of 2 itemsets

> # Se muestran
> inspect(itemsets_filtrado)
```

```

      items          support count
[1] {limon,melon}    0.015625  1
[2] {limon,melon,pera} 0.015625  1

```

Puede observarse que muchos *itemsets* están a su vez contenidos en *itemsets* de orden superior, es decir, existen *itemsets* que son *subsets* de otros. Para identificar cuáles son, o cuales no lo son, se puede emplear la función *is.subset()*. Encontrar los *itemsets* que son *subsets* de otros *itemsets* implica comparar todos los pares de *itemsets* y determinar si uno está contenido en el otro. La función *is.subset()* realiza comparaciones entre dos conjuntos de *itemsets* y devuelve una matriz lógica que determina si el *itemset* de la fila está contenido en cada *itemset* de las columnas.

```

> # Para encontrar los subsets dentro de un conjunto de itemsets, se compara
el conjunto de itemsets con sigo mismo.
> subsets <- is.subset(x = itemsets, y = itemsets, sparse = FALSE)

```

Para conocer el total de *itemsets* que son *subsets* de otros *itemsets* se cuenta el número total de *TRUE* en la matriz resultante.

```

> sum(subsets)
[1] 62

```

Reglas de asociación

Para crear las reglas de asociación se sigue el mismo proceso que para obtener *itemsets* frecuentes, pero, además de especificar un soporte mínimo, se tiene que establecer una confianza mínima para que una regla se incluya en los resultados. En este caso, se emplea una confianza mínima del 70%.

```
> soporte <- 0.05 / dim(transacciones)[1]
> reglas <- apriori(data = transacciones,
                    parameter = list(support = soporte,
                                     confidence = 0.70,
                                     # Se especifica que se creen reglas
                                     target = "rules"))
```

Apriori

Parameter specification:

```
confidence minval smax arem  aval originalSupport maxtime  support  minlen
      0.7      0.1    1  none  FALSE      TRUE          5      0.00078125  1
maxlen target  ext
      10  rules  FALSE
```

Algorithmic control:

```
filter tree heap memopt load sort verbose
  0.1 TRUE TRUE  FALSE TRUE    2    TRUE
```

Absolute minimum support count: 0

```
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[15 item(s), 64 transaction(s)] done [0.00s].
sorting and recoding items ... [15 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [8 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
```

```
> summary(reglas)
```

set of 8 rules

rule length distribution (lhs + rhs):sizes

```
2 3
4 4
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2.0	2.0	2.5	2.5	3.0	3.0

summary of quality measures:

support		confidence		lift		count	
Min.	:0.01562	Min.	:0.7500	Min.	:10.24	Min.	:1.00
1st Qu.	:0.01562	1st Qu.	:0.7875	1st Qu.	:11.56	1st Qu.	:1.00
Median	:0.03125	Median	:0.9000	Median	:12.40	Median	:2.00
Mean	:0.03516	Mean	:0.8875	Mean	:12.76	Mean	:2.25
3rd Qu.	:0.05078	3rd Qu.	:1.0000	3rd Qu.	:13.60	3rd Qu.	:3.25
Max.	:0.06250	Max.	:1.0000	Max.	:16.00	Max.	:4.00

mining info:

	data	ntransactions	support	confidence
transacciones		64	0.00078125	0.7

Se han identificado un total de 8 reglas. Como se muestra a continuación se enlistan las reglas que se crearon con el algoritmo a priori, una vez que el algoritmo analizó los valores por porción de 100 gr de cada ingrediente contenido en el dataframe y cuyas clasificaciones fueron Hidratos, Proteínas, Grasas, Agua, Fibra y Kcal se obtienen como resultado las siguientes reglas, en las que se puede observar que en el primer grupo de ingredientes para intercambio entre ellos es conformado por cerdo, pollo y almendra por sus similitudes en las clasificaciones, en el segundo grupo se encuentran el limón, pera y melón, lo anterior indica que si por alguna razón no se tiene o no se puede comer alguno de esos ingredientes se puede cambiar por otro dentro de su mismo grupo.

```
> inspect(sort(x = reglas, decreasing = TRUE, by = "confidence"))
```

	lhs	rhs	support	confidence	lift	count
[1]	{almendra,cerdo}	=> {pollo}	0.015625	1.00	16.00	1
[2]	{almendra,pollo}	=> {cerdo}	0.015625	1.00	16.00	1
[3]	{limon,melon}	=> {pera}	0.015625	1.00	12.80	1
[4]	{limon,pera}	=> {melon}	0.015625	1.00	12.80	1
[5]	{melon}	=> {pera}	0.062500	0.80	10.24	4
[6]	{pera}	=> {melon}	0.062500	0.80	10.24	4
[7]	{cerdo}	=> {pollo}	0.046875	0.75	12.00	3
[8]	{pollo}	=> {cerdo}	0.046875	0.75	12.00	3

Evaluación de las reglas

Además de la confianza y el soporte, existen otras métricas que permiten cuantificar la calidad de las reglas y la probabilidad de que reflejen relaciones reales. Algunas de las más empleadas son:

Lift: el estadístico *lift* compara la frecuencia observada de una regla con la frecuencia esperada simplemente por azar (si la regla no existe realmente). El valor *lift* de una regla “si X, entonces Y” se obtiene acorde a la siguiente ecuación:

$$\text{soporte}(\text{union}(X,Y)) / \text{soporte}(X) * \text{soporte}(Y)$$

Cuanto más se aleje el valor de *lift* de 1, más evidencias de que la regla no se debe a un artefacto aleatorio, es decir, mayor la evidencia de que la regla representa un patrón real.

Coverage: es el soporte de la parte izquierda de la regla (antecedente). Se interpreta como la frecuencia con la que el antecedente aparece en el conjunto de transacciones.

Fisher exact test: devuelve el *p-value* asociado a la probabilidad de observar la regla solo por azar.

Con la función *interestMeasure()* se pueden calcular más de 20 métricas distintas para un conjunto de reglas creadas con la función *apriori()*.

```
> metricas <- interestMeasure(reglas,measure =  
c("coverage","fishersExactTest"),transactions = transacciones)  
> metricas  
  coverage fishersExactTest  
1 0.062500    3.793030e-04  
2 0.062500    3.793030e-04  
3 0.078125    3.882216e-05  
4 0.078125    3.882216e-05  
5 0.015625    6.250000e-02  
6 0.015625    6.250000e-02  
7 0.015625    7.812500e-02  
8 0.015625    7.812500e-02
```

Estas nuevas métricas pueden añadirse al objeto que contiene las reglas.

```
> quality(reglas) <- cbind(quality(reglas), metricas)  
> # inspect(sort(x = reglas, decreasing = TRUE, by = "confidence"))
```

```

> df_reglas <- as(reglas, Class = "data.frame")
> df_reglas %>% as.tibble() %>% arrange(desc(confidence)) %>% head()

# A tibble: 6 x 7
  rules                                support confidence lift count coverage fishersExactTest
  <fct>                                <dbl>         <dbl> <dbl> <int>   <dbl>         <dbl>
1 {almendra,cerdo} => {pollo}  0.0156        1     16     1   0.0156        0.0625
2 {almendra,pollo} => {cerdo}  0.0156        1     16     1   0.0156        0.0625
3 {limon,melon} => {pera}      0.0156        1    12.8   1   0.0156        0.0781
4 {limon,pera} => {melon}      0.0156        1    12.8   1   0.0156        0.0781
5 {melon} => {pera}            0.0625        0.8   10.2   4   0.0781        0.0000388
6 {pera} => {melon}           0.0625        0.8   10.2   4   0.0781        0.0000388

```

Seguendo el ejercicio se tienen las siguientes recetas que contienen los ingredientes de las reglas.

Para preparar una ensalada de verdura con pollo se enlistan los siguientes ingredientes: lechuga, jitomate, cebolla, aguacate, pepino, zanahoria, pollo y aceite de oliva. En este caso si no se puede consumir el pollo se puede sustituir por la almendra ya que el cerdo no es apto para una ensalada.

Para preparar un coctel de fruta se tienen los siguientes ingredientes: plátano, manzana, melón, piña, papaya y sandía. En este caso si no se tiene melón porque no es temporada se puede sustituir por la pera.

Filtrado de reglas

Cuando se crean reglas de asociación, pueden ser interesantes únicamente aquellas que contienen un determinado conjunto de *items* en el antecedente o en el consecuente. Con *arules* existen varias formas de seleccionar solo determinadas reglas.

Restringir las reglas que se crean

Es posible restringir los *items* que aparecen en el lado izquierdo y/o derecho de las reglas a la hora de crearlas, por ejemplo, supóngase que solo son de interés reglas que muestren productos que se vendan junto con melon. Esto significa que el *item* melon, debe aparecer en el lado derecho (rhs).

```
> soporte <- 0.05 / dim(transacciones)[1]
> reglas <- apriori(data = transacciones,
                    parameter = list(support = soporte,
                                     confidence = 0.70,
                                     # Se especifica que se creen reglas
                                     target = "rules"),
                    appearance = list(rhs = "melon"))
```

Apriori

Parameter specification:

confidence	minval	smax	arem	aval	originalsupport	maxtime	support	minlen
0.7	0.1	1	none	FALSE	TRUE	5	0.00078125	1
10	rules	FALSE						

Algorithmic control:

filter	tree	heap	memopt	load	sort	verbose
0.1	TRUE	TRUE	FALSE	TRUE	2	TRUE

Absolute minimum support count: 0

```
set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[15 item(s), 64 transaction(s)] done [0.00s].
sorting and recoding items ... [15 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [2 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
```

```
> summary(reglas)
```

set of 2 rules

rule length distribution (lhs + rhs):sizes

```
2 3
1 1
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
------	---------	--------	------	---------	------

2.00 2.25 2.50 2.50 2.75 3.00

summary of quality measures:

support	confidence	lift	count
Min. :0.01562	Min. :0.80	Min. :10.24	Min. :1.00
1st Qu.:0.02734	1st Qu.:0.85	1st Qu.:10.88	1st Qu.:1.75
Median :0.03906	Median :0.90	Median :11.52	Median :2.50
Mean :0.03906	Mean :0.90	Mean :11.52	Mean :2.50
3rd Qu.:0.05078	3rd Qu.:0.95	3rd Qu.:12.16	3rd Qu.:3.25
Max. :0.06250	Max. :1.00	Max. :12.80	Max. :4.00

mining info:

data	ntransactions	support	confidence
transacciones	64	0.00078125	0.7

> inspect(reglas)

	lhs	rhs	support	confidence	lift	count
[1]	{pera}	=> {melon}	0.062500	0.8	10.24	4
[2]	{limon,pera}	=> {melon}	0.015625	1.0	12.80	1

Los lenguajes, programas de desarrollo de software y metodologías de desarrollo de software que a continuación se describen tienen en común que todos sirven para el desarrollo de aplicaciones WEB y para el desarrollo óptimo y eficaz de sistemas en los que se apoyaran dichas aplicaciones, se dará una breve reseña de ellos y se mostrarán sus principales características y después se analizará el más adecuado para la realización del proyecto, los resultados de este análisis se mostrarán en el Marco Conceptual que procederán a esta sección.

3.2 Lenguajes De Programación

3.2.1 Lenguaje C#

C# combina las mejores ideas de lenguajes como C, C++ y Java con las mejoras de productividad de .NET Framework de Microsoft y brinda una experiencia de codificación muy productiva tanto para los nuevos programadores como para los veteranos (Deitel y Deitel, 2004).

C# es el idioma de Windows Mobile. Es muy similar a C++ y Java. Microsoft ha adoptado algunas de las características de Java para simplificar su arquitectura, manteniendo el C++ como diseño.

C# es un lenguaje elegante y seguro orientado a objetos que permite a los desarrolladores crear una variedad de aplicaciones seguras y robustas que se ejecutan en .NET Framework. Puede usar C# para crear aplicaciones cliente de Windows, servicios web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de bases de datos y mucho, mucho más. Visual C# proporciona un editor de código avanzado, diseñadores de interfaz de usuario convenientes, depurador integrado y muchas otras herramientas para facilitar el desarrollo de aplicaciones basadas en el lenguaje C# y .NET Framework.

3.2.2 JavaScript

JavaScript tiene una larga historia que se remonta a los inicios de la World Wide Web. Un lenguaje muy popular front-end y el servidor, permite los desarrolladores web hacer todo lo posible de mejorar la experiencia del usuario de sus sitios web a la construcción de aplicaciones web completa.

Hoy en día, hay varios frameworks de JavaScript dirigidos específicamente a plataformas de desarrollo móvil, como Ionic 2 y React Native. Es muy fácil de desarrollar aplicaciones móviles multiplataforma con estos frameworks y librerías. Esto significa que sólo se tiene que escribir una sola versión de su aplicación, y funcionará en iOS o Android (Visual Studio, 2018).

3.3 Herramientas De Desarrollo De Software

3.3.1 Microsoft Visual Studio

El entorno de desarrollo integrado de Visual Studio es una plataforma de lanzamiento creativa que puede usar para editar, depurar y crear código, y luego publicar una aplicación. Un IDE (Integrated Development Environment / Entorno de Desarrollo Integrado) es un programa rico en funciones que se puede utilizar para muchos aspectos del desarrollo de software. Además del editor y depurador estándar que proporcionan la mayoría de los IDE, Visual Studio incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más funciones para facilitar el proceso de desarrollo de software (Docs.Microsoft, 2019).

3.3.2 Enterprise Architect

Enterprise Architect es una herramienta gráfica multiusuario diseñada para ayudar a los equipos de desarrollo de software a construir sistemas robustos y mantenibles.

Herramientas de modelado para desarrollo de software e ingeniería. El desarrollo de software es un proceso complejo y a menudo difícil que requiere la síntesis de muchas disciplinas.

Desde el modelado y el diseño hasta la generación de código, gestión de proyectos, pruebas, implementación, gestión de cambios y más, una herramienta de modelado basada en UML como Enterprise Architect se ha convertido en una parte esencial de la gestión de esa complejidad. (Sparxsystems, 2019)

3.4 Metodología De Desarrollo De Software

3.4.1 Scrum

Scrum es una metodología ágil y flexible para gestionar el desarrollo de software, cuyo principal objetivo es maximizar el retorno de la inversión para su empresa (ROI). Se basa en construir primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación.

Con la metodología Scrum el cliente se entusiasma y se compromete con el proyecto dado que lo ve crecer iteración a iteración. Asimismo, le permite en cualquier momento realinear el software con los objetivos de negocio de su empresa, ya que puede introducir cambios funcionales o de prioridad en el inicio de cada nueva iteración sin ningún problema.

Esta metódica de trabajo promueve la innovación, motivación y compromiso del equipo que forma parte del proyecto, por lo que los profesionales encuentran un ámbito propicio para desarrollar sus capacidades (Softeng, 2017).

Beneficios

Cumplimiento de expectativas: El cliente establece sus expectativas indicando el valor que le aporta cada requisito / historia del proyecto, el equipo los estima y con esta información el *Product Owner* establece su prioridad. De manera regular, en las demos de *Sprint* el *Product Owner* comprueba que efectivamente los requisitos se han cumplido y transmite el *feedback* al equipo.

Flexibilidad a cambios: Alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.

Reducción del Time to Market: El cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo.

Mayor calidad del software: La metódica de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a la obtención de un software de calidad superior.

Mayor productividad: Se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.

Maximiza el retorno de la inversión (ROI): Producción de software únicamente con las prestaciones que aportan mayor valor de negocio gracias a la priorización por retorno de inversión.

Predicciones de tiempos: Mediante esta metodología se conoce la velocidad media del equipo por sprint (los llamados puntos historia), con lo que consecuentemente, es posible estimar fácilmente para cuando se dispondrá de una determinada funcionalidad que todavía está en el

Backlog.

Reducción de riesgos: El hecho de llevar a cabo las funcionalidades de más valor en primer lugar y de conocer la velocidad con que el equipo avanza en el proyecto, permite despejar riesgos eficazmente de manera anticipada (Softeng.es, 2017).

Scrum es una metodología ágil y flexible para gestionar el desarrollo de software, cuyo principal objetivo es maximizar el retorno de la inversión para su empresa (ROI). Se basa en construir primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación.

Con la metodología Scrum el cliente se entusiasma y se compromete con el proyecto dado que lo ve crecer iteración a iteración. Asimismo, le permite en cualquier momento realinear el software con los objetivos de negocio de su empresa, ya que puede introducir cambios funcionales o de prioridad en el inicio de cada nueva iteración sin ningún problema.

Esta metódica de trabajo promueve la innovación, motivación y compromiso del equipo que forma parte del proyecto, por lo que los profesionales encuentran un ámbito propicio para desarrollar sus capacidades.

En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija (iteraciones que normalmente son de 2 semanas, aunque en algunos equipos son de 3 y hasta 4 semanas) Se muestra como ejemplo la Figura 3.2. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente (Product Owner) prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas. (ProyectosAgiles.org CC BY-SA 2017)

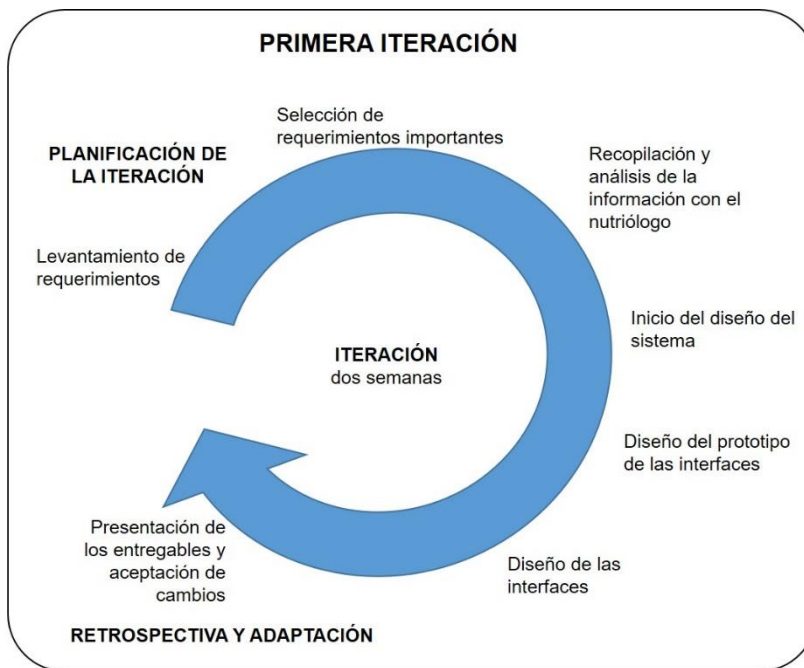


Figura 3-2 Iteraciones en Scrum (ProyectosAgiles.org CC BY-SA)

Una vez analizadas los lenguajes, programas de desarrollo de software y metodologías de desarrollo de software descritos en la sección anterior se llegó a la conclusión de que el proyecto se desarrollará en lenguaje C# puesto que es el lenguaje principal que maneja el programa Visual Studio para la realización de aplicaciones web y la mejor metodología para el desarrollo del proyecto es Scrum, en siguiente se verán unas breves descripciones.

3.5 MVC

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos de interacción con éste.

El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno (Alicante, 2019). En la Figura 3.3 se puede ver de manera gráfica cómo funciona el MVC.

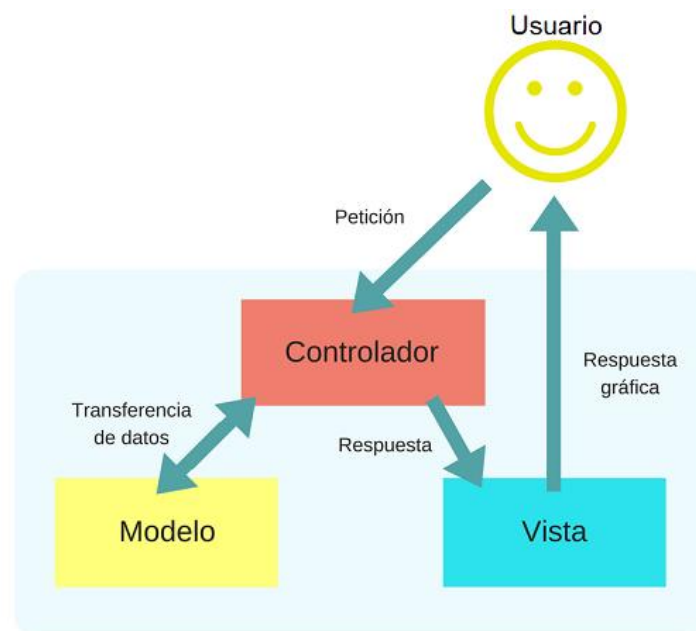


Figura 3-3 Modelo Vista Controlador (codingornot.com)

3.5.1 Frontend

El Frontend se enfoca en el usuario final, en todo con lo que se puede interactuar y lo que se ve mientras se navega por la página web, se busca causar una buena impresión y agradar al usuario, para lo cual se utiliza HTML, CSS y JAVASCRIPT. Se busca que el usuario tenga una buena experiencia, inmersión y usabilidad, hoy en día existen una gran variedad de frameworks, preprocesadores y librerías que nos ayudarán en esta tarea. Para un frontend la creatividad es el recurso más valioso, ya que tendrá que tomar fuentes, colores, imágenes y todos los recursos de los cuales se disponga para crear sitios agradables que se vean bien en todos los dispositivos y resoluciones (Docs.microsoft, 2019).

3.5.1.1 Bootstrap ver. 4.1.3.1

Bootstrap es un kit de herramientas de código abierto para desarrollar con HTML, CSS y JS. Visual Studio facilita la identificación de las clases que provienen del marco CSS Bootstrap al mostrar el logotipo en la lista de finalización. Incluye componentes CSS y JavaScript para elementos comunes de interfaz de aplicaciones web. Incluye estilos para elementos de navegación, formularios, botones y un sistema de diseño de cuadrícula sensible que permite que los diseños del sitio se ajusten dinámicamente a dispositivos que tienen diferentes tamaños de pantalla, como teléfonos y tabletas. Al usar el sistema de diseño Bootstrap, puede desarrollar un solo sitio que presente una interfaz adecuada para todos los dispositivos que sus clientes puedan usar (Visual Studio, 2019).

3.5.1.2 Responsive Design

Diseño Web Adaptable, originalmente definido por Ethan Marcotte en A List Apart, responde a las necesidades de los usuarios y los dispositivos que estén usando. Los cambios de diseño según el tamaño y las capacidades del dispositivo. Por ejemplo, en un teléfono los usuarios deberían ver el contenido que se muestra en una única columna; una tablet puede mostrar el mismo contenido en dos columnas (Marcotte, 2011).

3.5.1.3 JQuery

jQuery es una biblioteca de JavaScript rápida, pequeña y rica en funciones. Hace que cosas como el desplazamiento y la manipulación de documentos HTML, el manejo de eventos, la animación y Ajax sean mucho más simples con una API fácil de usar que funciona en una multitud de navegadores. Con una combinación de versatilidad y extensibilidad (jquery.com, 2019).

3.5.2 Backend

Backend enfocado en hacer que todo lo que está detrás de un sitio web funcione correctamente. Toma los datos, los procesa y los envía al usuario, además de encargarse de las consultas o peticiones a la Base de Datos, la conexión con el servidor, entre otras tareas que debe realizar en su día a día. Cuenta con una serie de lenguajes y herramientas que le ayudan a cumplir

con su trabajo como PHP, Ruby, Python, JavaScript, SQL, MongoDB, MySQL, etc, estos son usados para crear sitios dinámicos. Como en muchos sitios la información se encuentra en constante cambio o actualización, una buena capacidad de respuesta y una velocidad óptima del sitio son responsabilidades que un backend debe de afrontar (Docs.microsoft, 2019).

3.5.2.1 ASP.NET Core

ASP.NET Core es un marco multiplataforma de código abierto y de alto rendimiento que tiene como finalidad compilar modernas aplicaciones conectadas a Internet y basadas en la nube

ASP.NET Core MVC ofrece una manera basada en patrones de crear sitios web dinámicos que permitan una clara separación de intereses. Proporciona control total sobre el mercado, admite el desarrollo controlado por pruebas y usa los estándares web más recientes (Docs.microsoft, 2019).

3.5.2.2 LINQ

Lenguaje integrado para consultas en C# Language-Integrated Query (LINQ) en inglés es el nombre de un conjunto de tecnologías basadas en la integración de capacidades de consulta directamente en el lenguaje C #.

Las expresiones de consulta se escriben en una sintaxis de consulta declarativa. Al utilizar la sintaxis de consulta, puede realizar operaciones de filtrado, ordenación y agrupación en orígenes

de datos con un mínimo de código. Utiliza los mismos patrones básicos de expresión de consulta para consultar y transformar datos en bases de datos SQL, conjuntos de datos ADO .NET, documentos y secuencias XML y colecciones .NET (Docs.microsoft, 2019).

3.6 Base de datos

3.6.1 Motor de base de datos PostgreSQL

PostgreSQL escrito en la Universidad de California en Berkeley es un potente sistema de base de datos relacional de objetos de código abierto, con muy buen rendimiento, aplicable a cualquier plataforma y con solidez de características (postgresql.org, 2019).

3.6.2 Sistema Gestor de Base de Datos pgAdmin III

pgAdmin 3 es la plataforma de administración y desarrollo de código abierto más popular y rica en características para PostgreSQL, la base de datos de código abierto más avanzada del mundo. La aplicación se puede usar en plataformas Linux, FreeBSD, Solaris, macOS y Windows para administrar PostgreSQL 8.4 a 9.5 que se ejecutan en cualquier plataforma (pgadmin.org, 2019).

Capítulo 4 Modelado del sistema

4.1 Definición de los requerimientos

4.1.1 Características del sistema

El sistema cuenta con el menú tanto para pacientes como para personal del hospital y público en general. Cabe destacar que cada persona que labore en el hospital se le registrará y se le asignará un rol para que puedan acceder al menú y hacer su pedido, en cuanto a los pacientes se registran junto con sus enfermedades y padecimientos, se le asigna una dieta conforme las indicaciones del médico. El sistema cuenta con diferentes roles de usuario y dependiendo de estas se tiene acceso a diferentes secciones del software.

4.1.2 Requerimientos Funcionales

Dentro de los requerimientos funcionales se describen las interacciones que tendrán los usuarios con el software, así como las condiciones necesarias para su correcto funcionamiento

- Que en el área de cocina se tenga internet acceso al sistema.
- Que en el área de las habitaciones tengan internet y acceso al sistema.
- Que el programa sea capaz de otorgar al paciente los alimentos a seleccionar según su dieta asignada.

- Que el programa haga el cálculo de costeo por receta y por porción para el mejor control de los gastos e ingresos en el área de cocina.
- Que el sistema tenga varios roles para los distintos trabajadores del hospital.
- Que solamente el usuario con rol de administrador tenga acceso a las secciones de registro de alimentos y dietas, así como el control de los precios de los mismos.
- Que sea fácil e intuitivo tanto para los pacientes como para los trabajadores.
- Que se pueda acceder al sistema desde cualquier dispositivo con internet.

4.1.3 Requerimientos No Funcionales.

Son los complementarios, como por ejemplo restricciones en el diseño o la implementación o los estándares de calidad.

- Que tenga una interfaz llamativa, pero a la vez acorde al tema y seriedad del hospital.
- Que la red wifi del hospital sea restringida por contraseña.
- La aplicación web debe poseer un diseño “Responsivo” a fin de garantizar la adecuada visualización en múltiples computadores personales, dispositivos tableta y teléfonos inteligentes.

4.1.4 Requerimientos del Software

Estos son los requerimientos lógicos que tiene la computadora que sirve de servidor para el alojamiento de la aplicación.

Características del Servidor

- Sistema operativo Windows 8.1 Pro de 64 bits, con procesador Intel Pentium G3250 a 3.20 GHz, RAM de 8 GB.

Programas necesitados:

- Microsoft Visual Studio Community 2017 versión 15.9.4 con Microsoft.Net Framework versión 4.8.03752 con versión de APS.NET Core 2.2 con Entity Framework.
- PostgreSQL versión 9.5.18 para Windows de 64 bits como motor de base de datos.
- pgAdmin III como gestor de base de datos.

4.1.5 Requerimientos del Hardware

Estos son los requerimientos físicos con los que cuenta la computadora que funge como servidor, así como de la infraestructura necesaria para la implementación de la app.

- Computadora marca HP con 8 GB de memoria RAM con un regulador Nobreak, conexión por cable al rack alimentado por internet infinitum de 100 Mbps.
- Un router TP-Link modelo TD-W8960N con 300 Mbps, así como la infraestructura del cableado en el área de cocina. Se utilizaron 30 metros de cable UTP categoría 6 para subir el internet desde el tercer piso al quinto piso.
- Dos router y dos amplificadores de señal para el primer y segundo piso y con eso cubrir la necesidad de internet en estos pisos.

4.2 Análisis y Diseño

Para esta sección se utilizó el programa Enterprise Architect como apoyo para el modelado de negocios, los casos de uso y los diversos diagramas que se encuentran en esta.

4.2.1 Modelado de negocios

En la Figura 4.1 se muestra el diagrama de negocios el cual dicta de forma simplificada y sencilla el flujo que debe seguir el programa desde que se inicia la aplicación para la selección de los alimentos hasta la entrega de los mismos en la habitación del paciente.

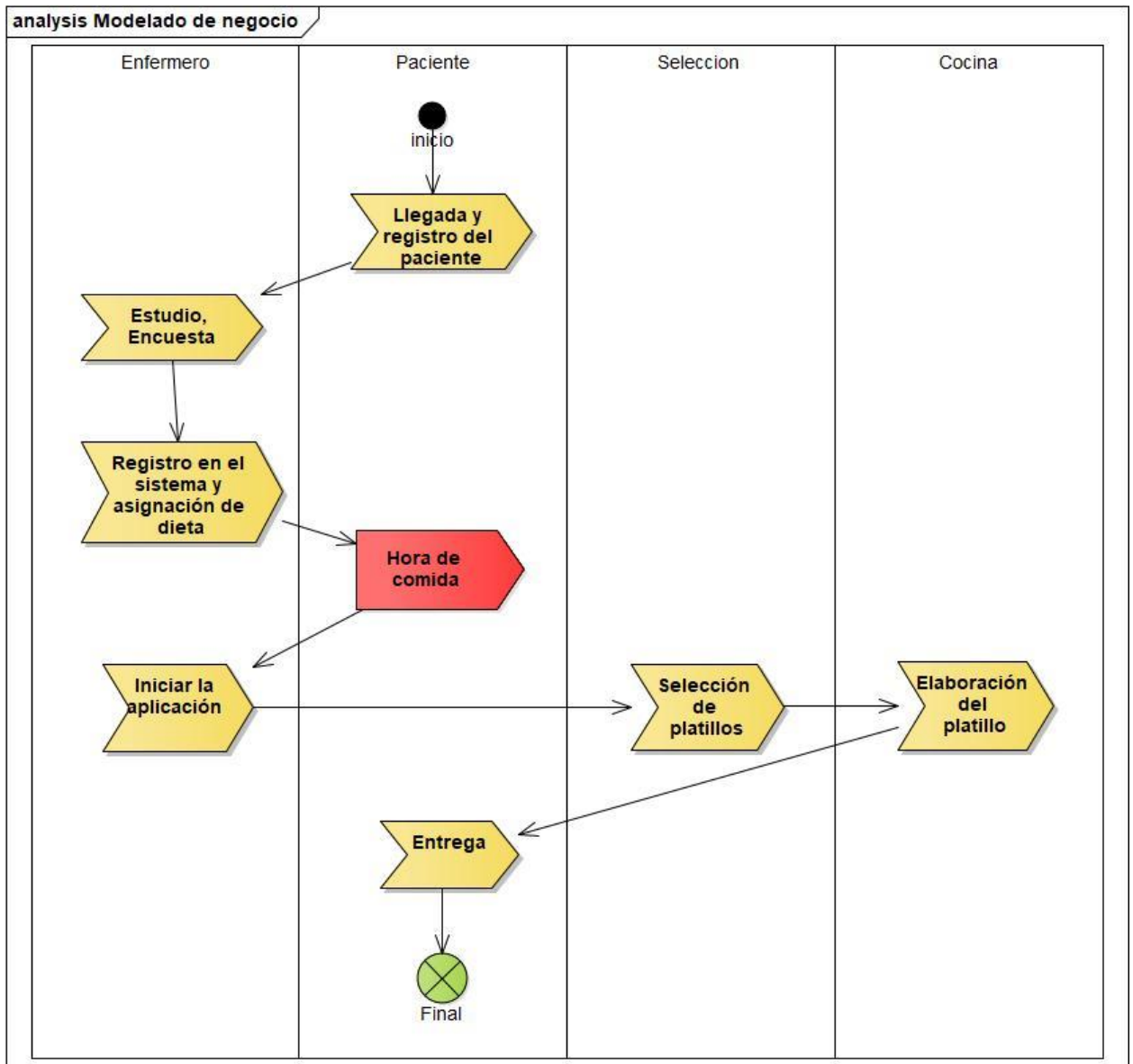


Figura 4-1 Modelado de negocios

4.2.2 Diagrama de secuencia

En este diagrama el cual es representado en la Figura 4.2 se muestra la secuencia de la comanda desde que el enfermero inicia la aplicación para la selección de la comida, como ésta se va generando conforme le van agregando los alimentos y como se mantiene abierta hasta que la comida es entregada.

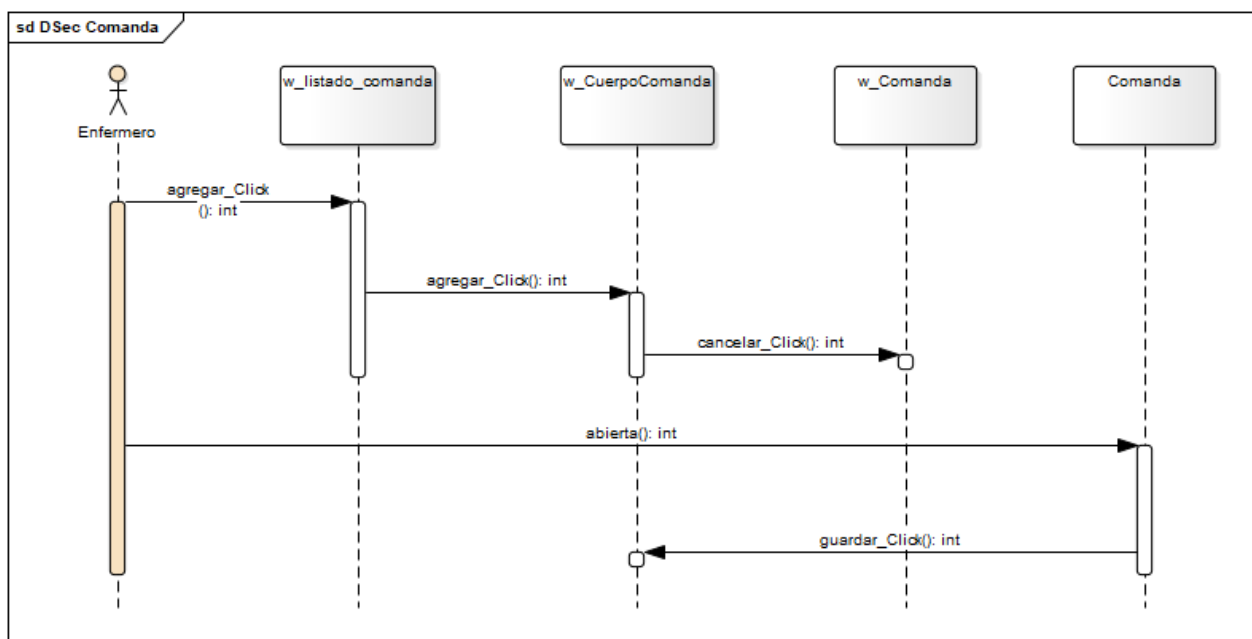


Figura 4-2 Diagrama de secuencia de la comanda

4.2.3 Casos de uso

En este diagrama se pueden visualizar a los diferentes actores que utilizarán el sistema, así como los diferentes roles que se necesitan para los distintos accesos y permisos de usuarios. La Figura 4.3 muestra la aplicación de distribución de comida con base en la enfermedad del paciente.



Figura 4-3 Caso de uso general

En este caso de uso de la Figura 4.4 se visualizan los diversos actores principales que participan en la alimentación de los pacientes, así como los accesos al sistema. En primera parte tenemos al administrador que es el que tiene acceso a todas las funcionalidades del sistema y el que otorga los permisos a los diferentes usuarios. Este se encarga de registrar a la encargada de cocina y a los enfermeros para que tengan acceso al sistema y a su vez los enfermeros puedan dar de alta a los pacientes en el sistema y asignarles la dieta que les dio el doctor para el tratamiento de estos.

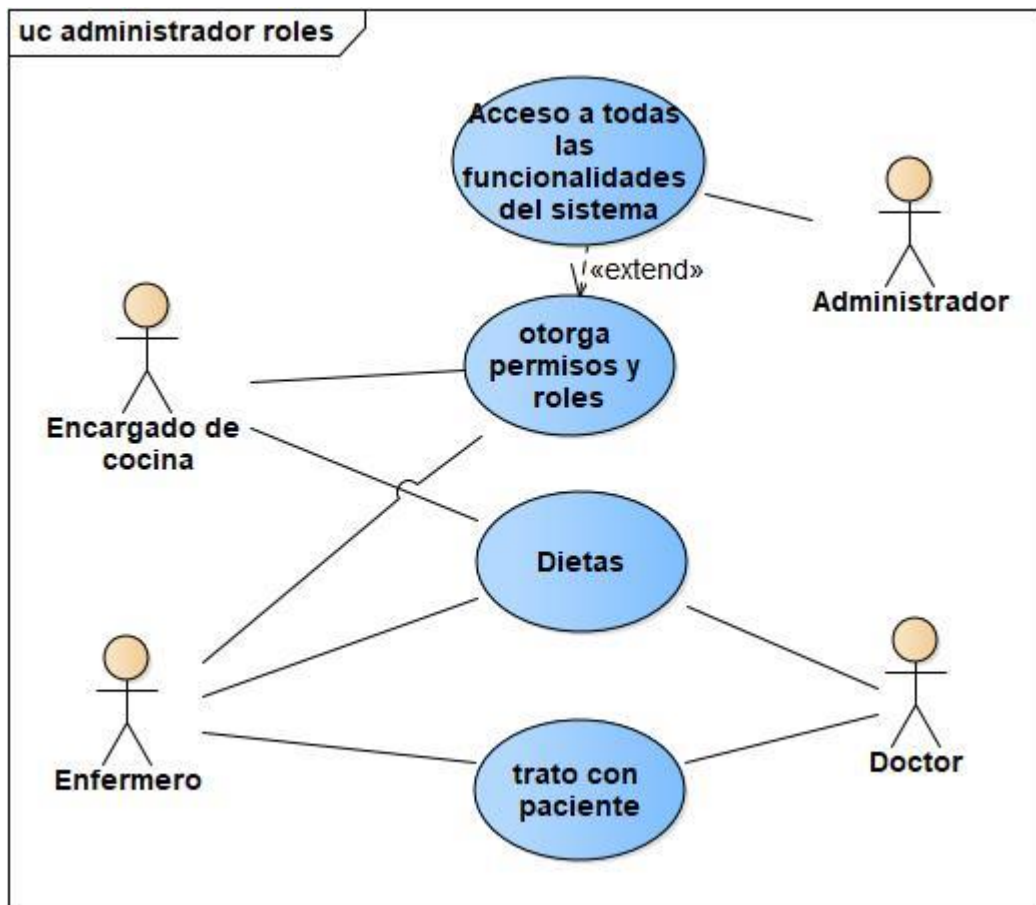


Figura 4-4 Administrador de roles

En este caso de uso Figura 4.5 se visualiza el flujo que se tiene desde el registro de las comidas y las dietas para que el paciente los pueda seleccionar, la encargada de cocina registra las comidas, así como a qué dieta pertenecen. Los enfermeros una vez que han registrado al paciente en el sistema tienen que estar al pendiente de lo que selecciona éste para comer y una vez que lo autoriza se procede a mandar la orden a cocina para su elaboración y una vez que esté lista se le lleva al paciente.

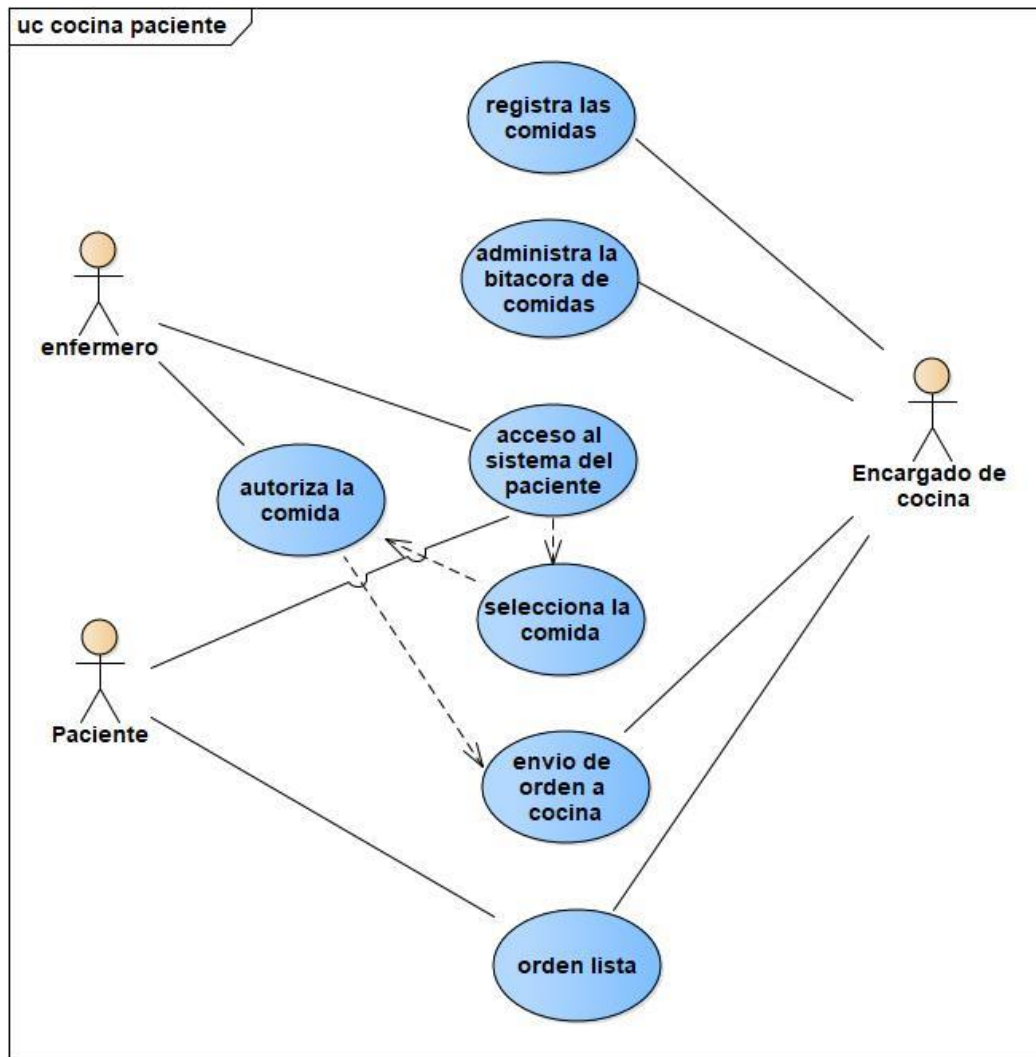


Figura 4-5 Caso de uso, interacción entre cocina y paciente

4.3 Diagrama de Modelado de la Base De Datos

En la Figura 4.6 se muestra como está estructurada la base de datos, es manejado en bloques para la fácil localización de los elementos en caso de que se necesite modificar, así como se seccionaran adelante para que se pueda visualizar mejor.

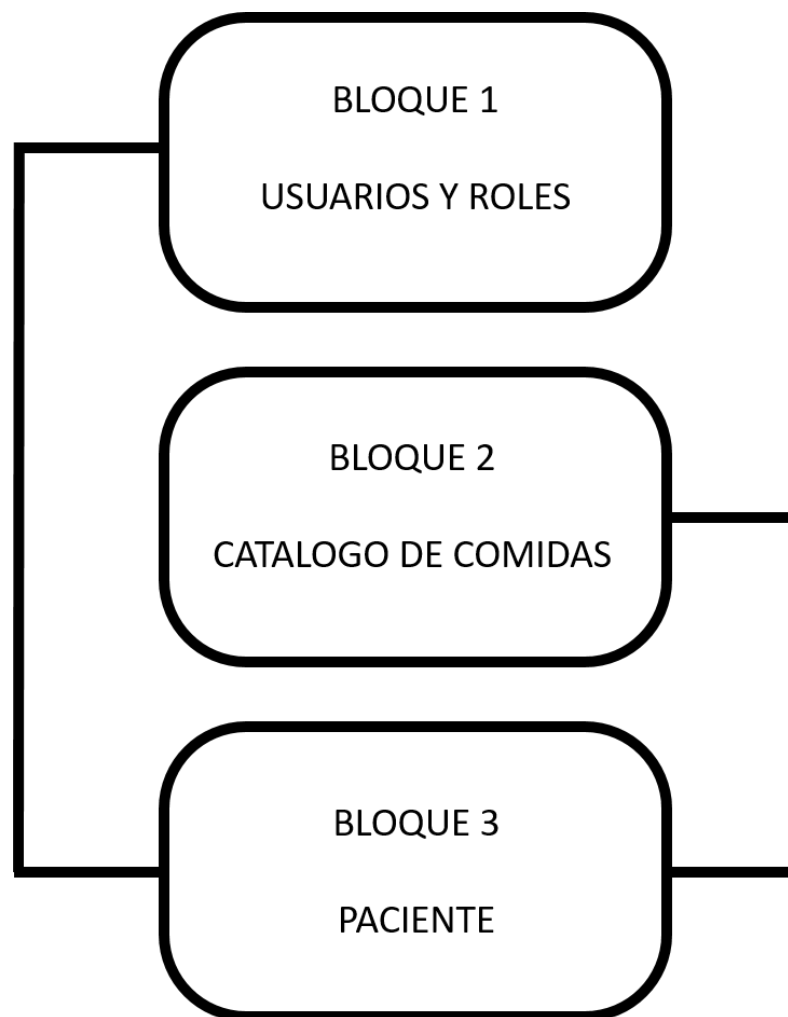


Figura 4-6 Base de datos completa

La base de datos está conformada por 23 tablas y está estructurada por tres bloques. El primer bloque visto en la Figura 4.7 que es el que vemos a continuación es el de los usuarios y roles.

En este vemos ocho tablas relacionadas entre sí para guardar los registros y permisos para dar acceso a los tipos de roles y de puestos. En este bloque la tabla principal es *asp_net_users* que es la que hace el registro de los usuarios, a esta se ligán los tipos de usuario, los roles, los puestos y el acceso.

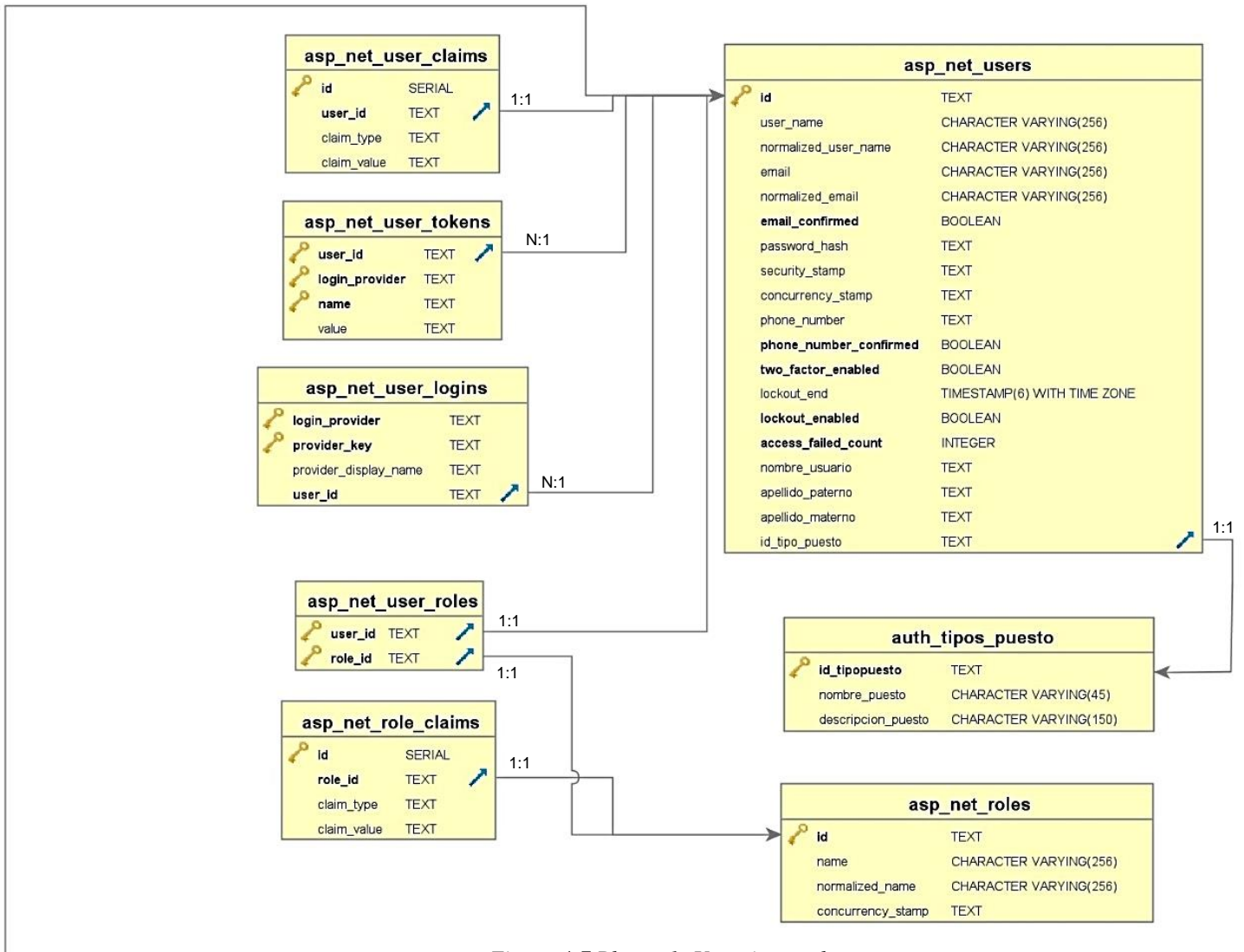


Figura 4-7 Bloque 1- Usuarios y roles

El segundo bloque Figura 4.8 es la parte dedicada al catálogo de comidas y de sus ingredientes, también está ligada a la parte de dietas que está en el bloque tres que es el del paciente.

En esta sección hay dos tablas principales: la tabla *com_catologo_comidas* y la tabla *receta_ingrediente* y cada una tiene sus relaciones con las demás tablas, en estas se ven el carro de compra que guarda el listado de las ordenes hechas a cocina, y el registro de los ingredientes de cada receta. *com_catologo_comidas* está relacionada con *pac_catologo_tipo_dietas* de la siguiente sección.

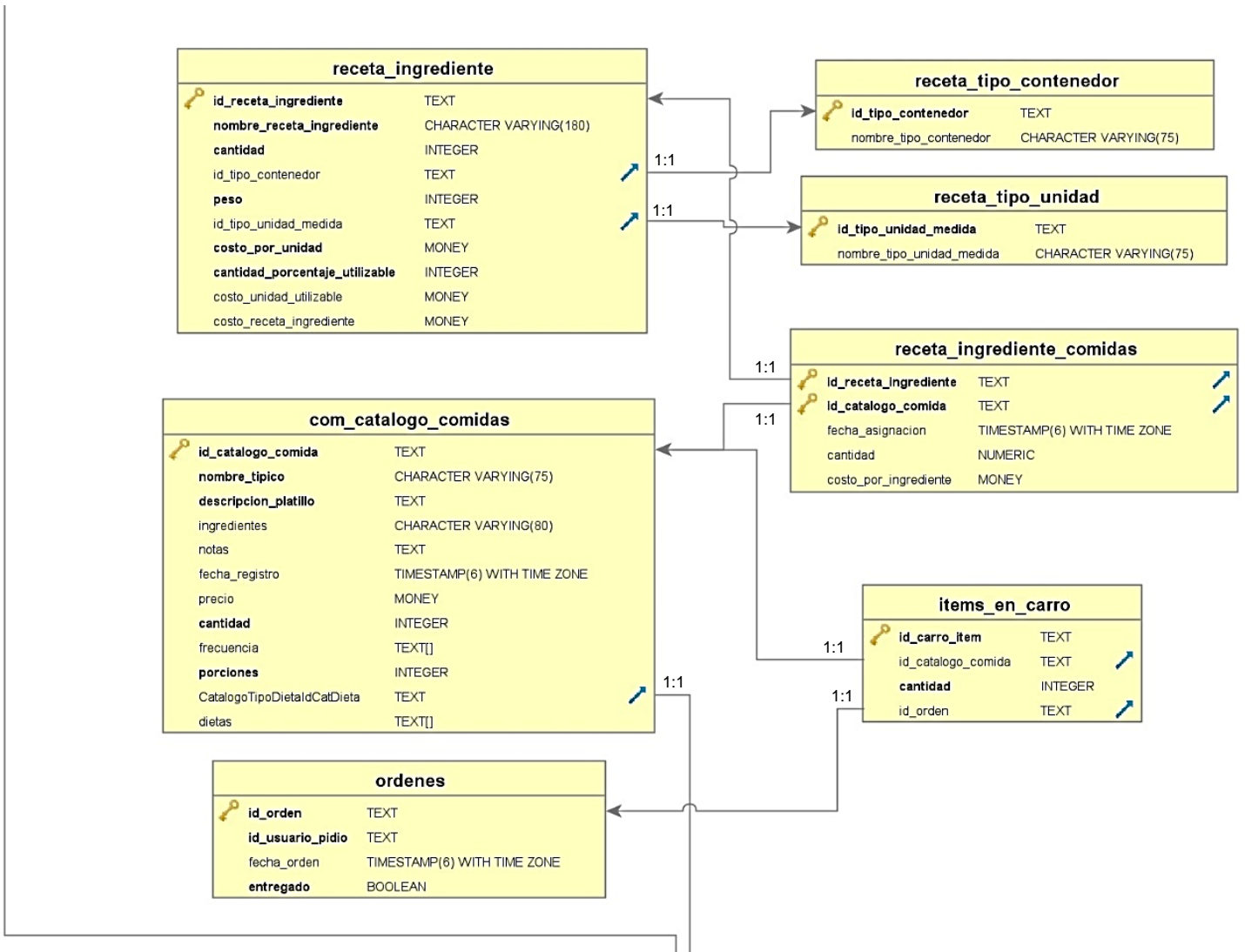


Figura 4-8 Bloque 2 - Catálogos de comidas

Este es el bloque tres, la sección del paciente, Figura 4.9, el registro del mismo, el registro de dieta que debe llevar, sus enfermedades o padecimientos previos o con los que ya tiene que lidiar diariamente, sus familiares, en esta sección la tabla principal es *pac_pacientes* en donde se guarda el registro del paciente cuando se hace por primera vez y ésta está ligada a *asp_net_users* de la primera sección y a *com_catalogo_comida* de la segunda sección

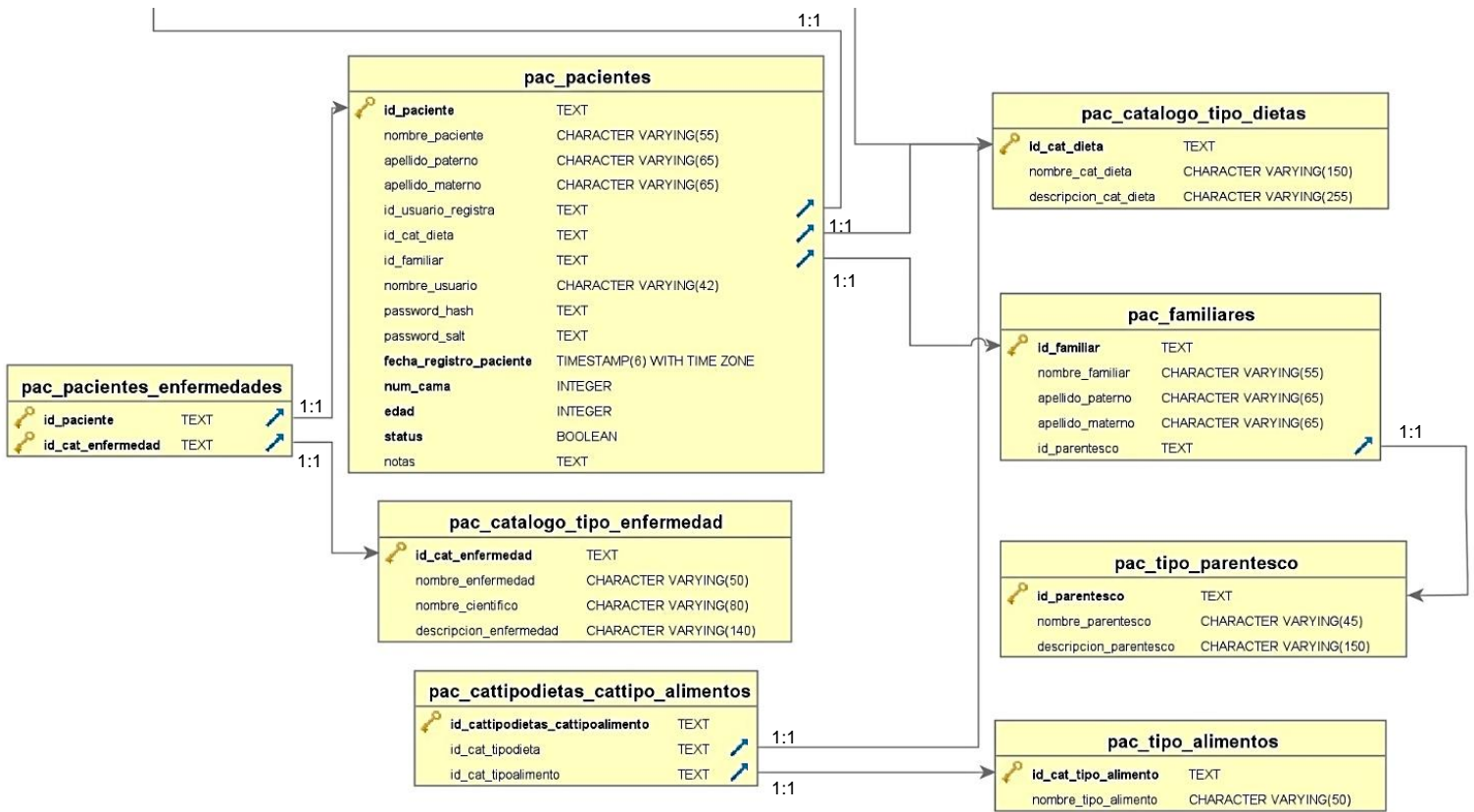


Figura 4-9 Bloque 3 - Paciente

4.4 Diagrama de Despliegue

En el diagrama mostrado en la Figura 4.10 se ve la distribución de los dispositivos y su interrelación, la computadora que servirá de servidor de la aplicación será una computadora HP con Windows 8.1 con acceso a internet y con el PostgreSQL para el manejo de la base de datos. Este estará ligado a un servidor web para que a través del internet se pueda acceder al sistema, este acceso será por dispositivos móviles, la aplicación al estar programada con un diseño responsivo puede adaptarse a cualquier pantalla, ya sea de computadora, laptop, teléfono celular o tablet, sin importar el sistema operativo que utilice cada dispositivo.

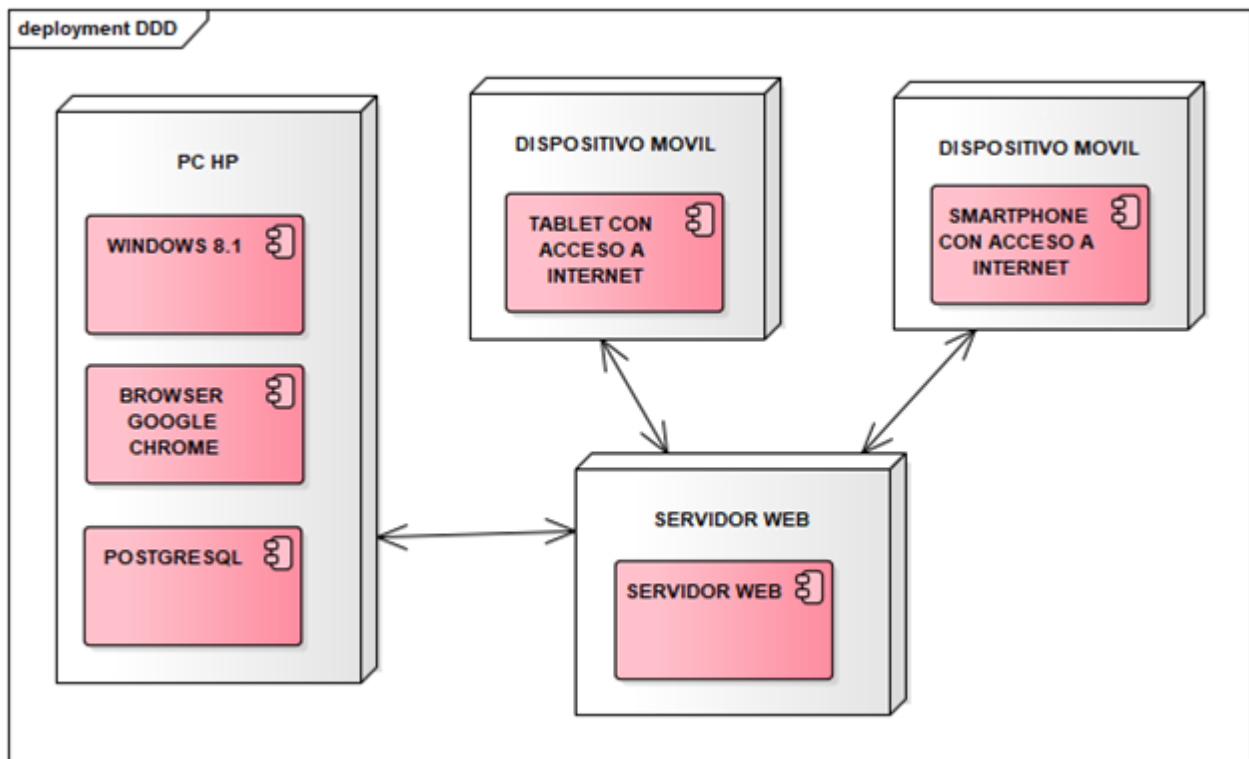


Figura 4-10 Diagrama de distribución

4.5 Diagrama de transición de estados

En este ejemplo vemos la transición de estados de la comanda Figura 4.11 realizada cuando se pide la comida, desde que se inicia cuando se ingresa al sistema hasta cuando el personal de cocina confirma la entrega al paciente desde su perfil.

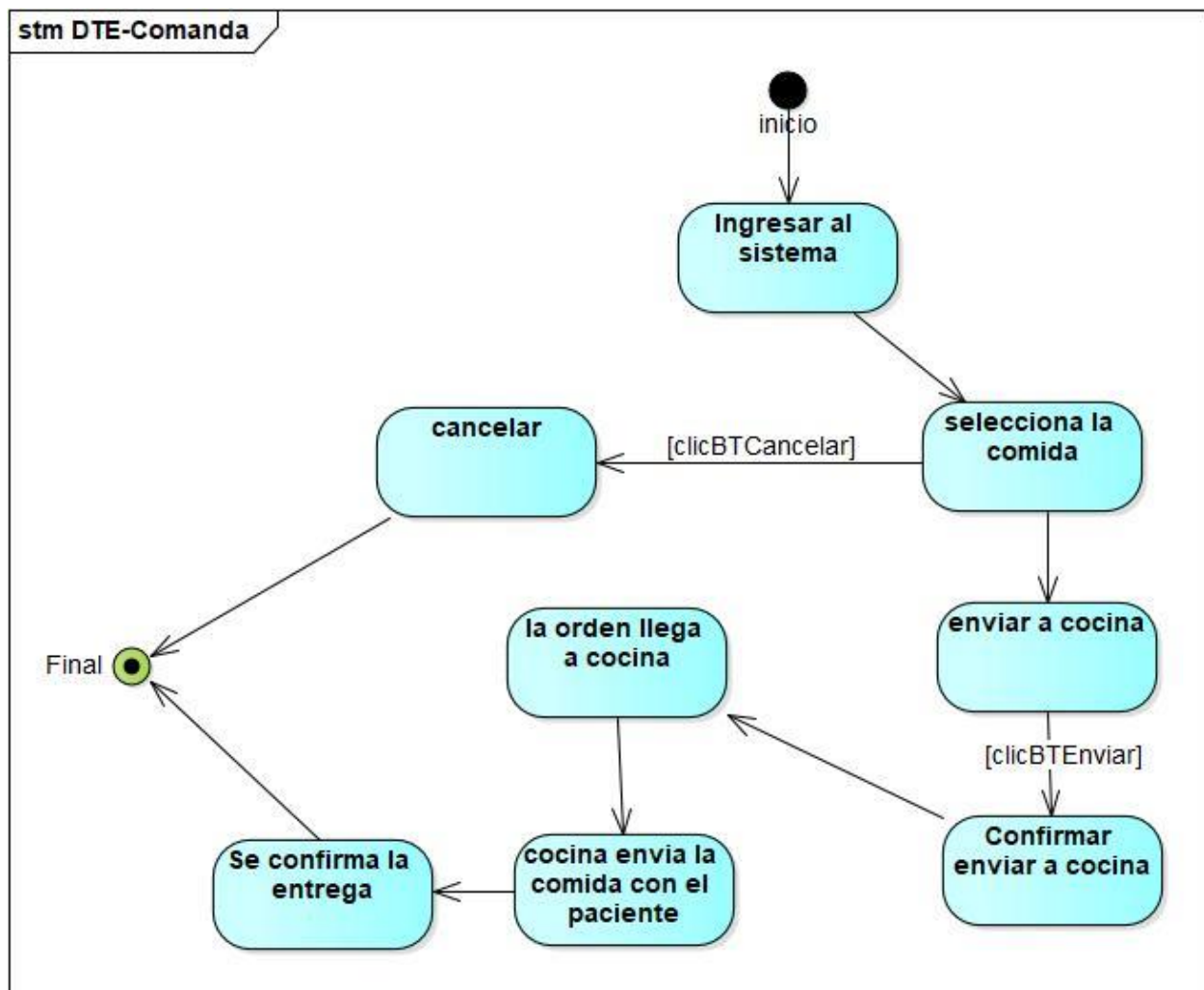


Figura 4-11 Diagrama de transición de estados - Comanda

4.6 Cronograma de actividades

A continuación, se muestra el cronograma de actividades realizado en Microsoft Project 2016 para mostrar el tiempo de cada tarea y las diversas relaciones entre las mismas, viendo el tiempo de duración de cada una, así como sus antecesoras y el equipo encargado de desarrollarlas.

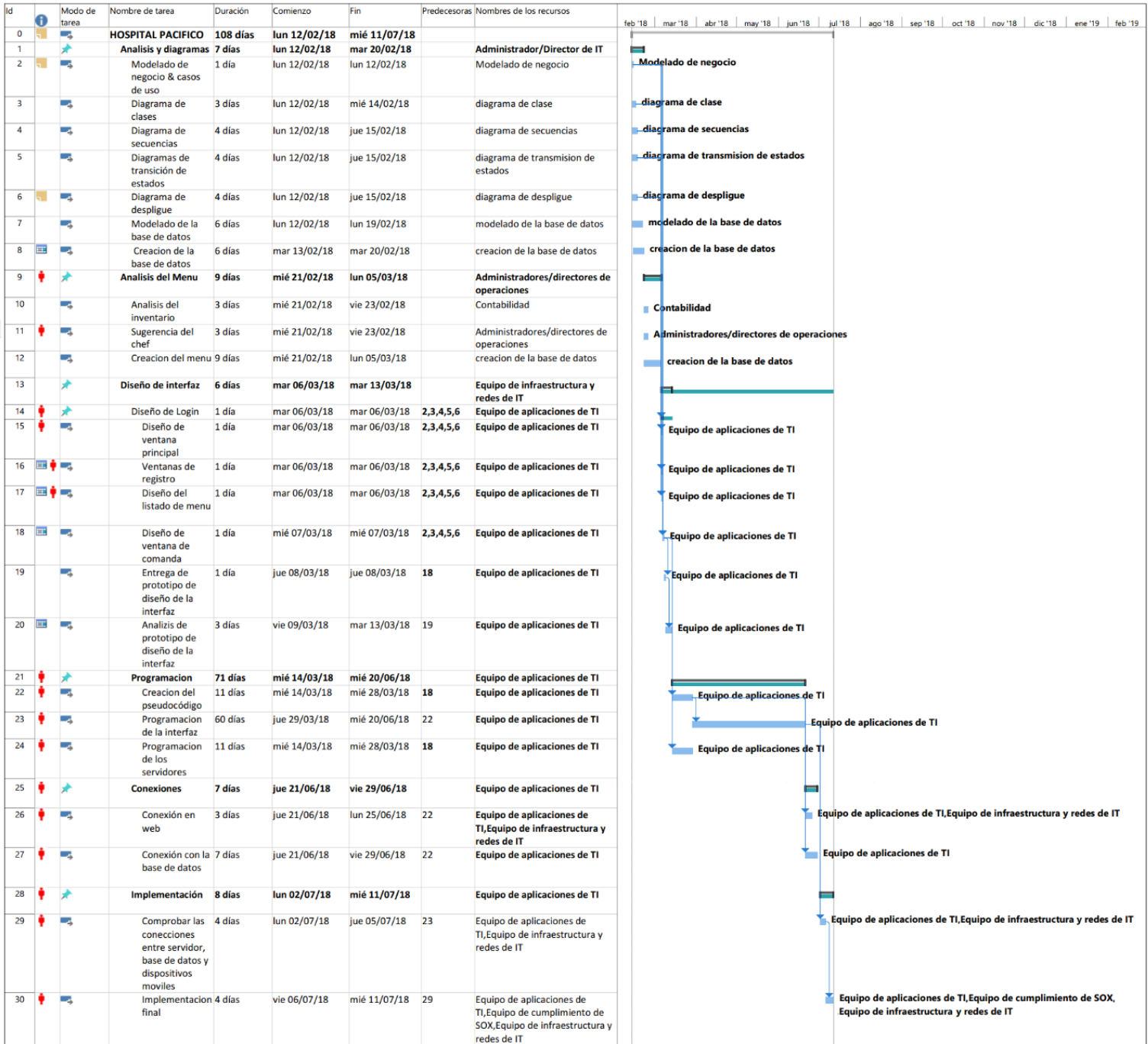


Figura 4-12 Cronograma de Actividades

Capítulo 5 Arquitectura y desarrollo del proyecto

El proyecto está conformado por una arquitectura de cuatro capas las cuales se describen a continuación, más adelante se encuentra la parte del desarrollo de los módulos, se encuentran las secciones en donde se describen algunos códigos que son clave en el programa para las relaciones entre pacientes, dietas, roles y usuarios.

5.1 Pacificoapp.Dal (Data Acces Layer)

La capa de acceso a datos o DAL (del inglés data access layer) es una capa del programa que proporciona acceso simplificado a los datos almacenados en la base de datos.

La forma más común de definir la capa de datos es mediante el uso de lo que a veces se hace referencia como un objeto de datos universal (UDO), que está escrito en el lenguaje de programación JavaScript. Los tipos de datos contenidos en una capa de datos pueden ser numerosos y variados, y consisten en elementos como la información de transacciones de comercio electrónico, los datos de comportamiento web y el uso de aplicaciones móviles.

En la Figura 5.1 se muestra la estructura de esta capa del programa, los elementos que contiene y las relaciones con las otras capas del mismo, así como las clases propias de esta sección.

Una instancia de DbContext representa una combinación de los patrones de Unidad de trabajo y Repositorio, de modo que se puede usar para consultar desde una base de datos y agrupar cambios que luego se escribirán como una unidad.

Patrón de Repositorio Genérico / Repository Generic

El repositorio media entre el dominio y las capas de mapeo de datos, actuando como una colección en memoria de objetos de dominio (programmingwithwolfgang.com, 2019).

Objetivos del patrón de repositorio

- Desacoplar el código comercial del acceso a datos. Como resultado, el Marco de persistencia se puede cambiar sin un gran esfuerzo.
- Separación de intereses.
- Minimizar la lógica de consulta duplicada.
- Testabilidad.

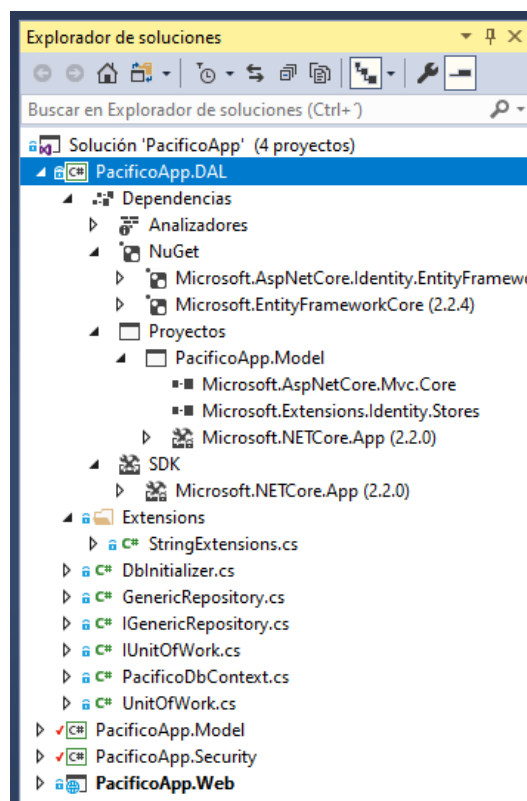


Figura 5-1 Capa de acceso a datos

El patrón de repositorio se usa a menudo cuando una aplicación realiza operaciones de acceso a datos. Estas operaciones pueden realizarse en una base de datos, servicio web o almacenamiento de archivos. El repositorio encapsula estas operaciones para que no le importe a la lógica de negocios donde se realizan las operaciones. A la aplicación no le importa si se cargan desde una base de datos o un servicio web.

El repositorio debe verse como una colección en memoria y debe tener métodos genéricos como Agregar, Eliminar o Encontrar por Id. Con tales métodos genéricos, el repositorio se puede reutilizar fácilmente en diferentes aplicaciones. Además del repositorio genérico, se implementan uno o más repositorios específicos, que heredan del repositorio genérico. Estos repositorios especializados tienen métodos que la aplicación necesita.

- Unidad de Trabajo / Unit of Work

Mantiene una lista de objetos afectados por una transacción comercial y coordina la escritura de los cambios.

Objetivos del patrón de unidad de trabajo

- Aumenta el nivel de abstracción y mantiene la lógica empresarial libre de código de acceso a datos.
- Mayor capacidad de mantenimiento, flexibilidad y capacidad de prueba.
- Más clases e interfaces, pero menos código duplicado.
- La lógica de negocios está más lejos de los datos porque el repositorio abstrae la infraestructura. Esto tiene el efecto de que podría ser más difícil optimizar ciertas operaciones que se realizan contra la fuente de datos.

5.2 Pacificoapp.Model (Modelos)

El modelo representa los datos específicos del dominio y la lógica empresarial en la arquitectura MVC. Mantiene los datos de la aplicación. Los objetos modelo recuperan y almacenan el estado del modelo en el almacén de persistencia como una base de datos. La clase de modelo contiene datos en propiedades públicas. Todas las clases Modelo residen en la carpeta Modelo en la estructura de carpetas MVC (jossjack.wordpress.com, 2019).

- DTO (data transfer object)

Un DTO es un objeto que define cómo se enviarán los datos a través de la red.

- Entities

Una entidad es la representación de un elemento del mundo real dentro de Object Relational Mapping (ORM) como Entity Framework. Esta representación se asignará a una tabla en una base de datos y sus atributos se transformarán en columnas (stackoverflow.com, 2019).

En la Figura 5.2 se ven todas las clases de modelos de la base de datos y estas también estarán ligadas a la capa *WEB* en la sección de controladores, pues serán llamadas e instanciadas para su visualización.

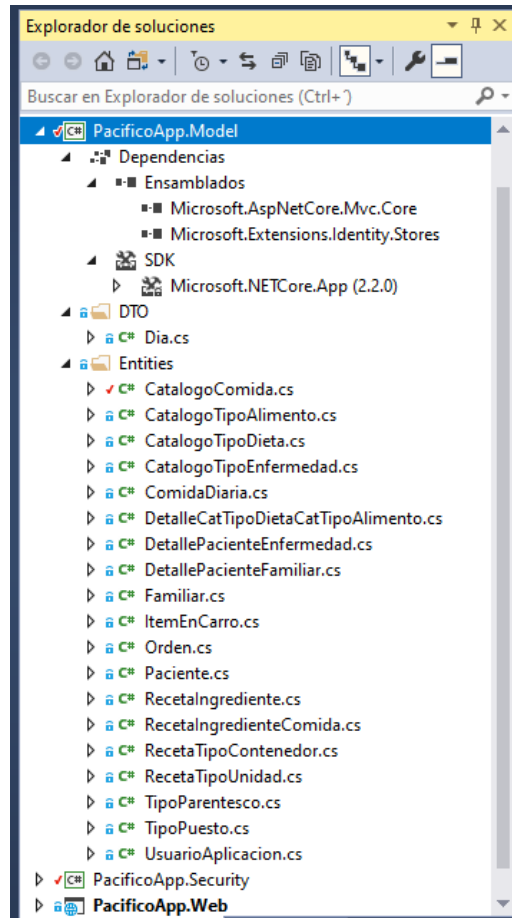


Figura 5-2 Modelos

Como se puede ver en la Figura 5.3 tomando de ejemplo la clase *CatalogoComida.cs* se muestran los campos que se requerirán para dar de alta en la base de datos, en esta se describen las características que tendrán cada uno de los formularios a rellenar y los mensajes de error en caso de que no se cumplan con las características que ahí mismo describen, el tipo de variable que son así como las longitudes de sus caracteres.

```

5
6 namespace PacificoApp.Model.Entities
7 {
8     public class CatalogoComida
9     {
10         //[BindNever]
11         public string IdCatalogoComida { get; set; }
12
13         [Display(Name = "Nombre típico")]
14         [Required(ErrorMessage = "Nombre típico requerido")]
15         [StringLength(75, ErrorMessage = "El '{0}' debería ser como mínimo {2} y como máximo {1} caracteres", MinimumLength = 3)]
16         public string NombreTipico { get; set; }
17
18         [Display(Name = "Descripción")]
19         [Required(ErrorMessage = "Descripción de platillo requerido")]
20         [StringLength(255, ErrorMessage = "El '{0}' debería ser como mínimo {2} y como máximo {1} caracteres", MinimumLength = 10)]
21         public string DescripcionPlatillo { get; set; }
22
23         [Display(Name = "Ingredientes")]
24         [StringLength(155, ErrorMessage = "El '{0}' debería ser como mínimo {2} y como máximo {1} caracteres", MinimumLength = 10)]
25         public string Ingredientes { get; set; }
26
27         [Display(Name = "Notas")]
28         [StringLength(155, ErrorMessage = "El '{0}' debería ser como mínimo {2} y como máximo {1} caracteres", MinimumLength = 10)]
29         public string Notas { get; set; }
30
31         [BindNever]
32         public string IdCatalogoComida { get; set; }
33     }
34 }

```

Figura 5-3 CatalogoComida.cs

5.3 Pacificoapp.Security (Seguridad)

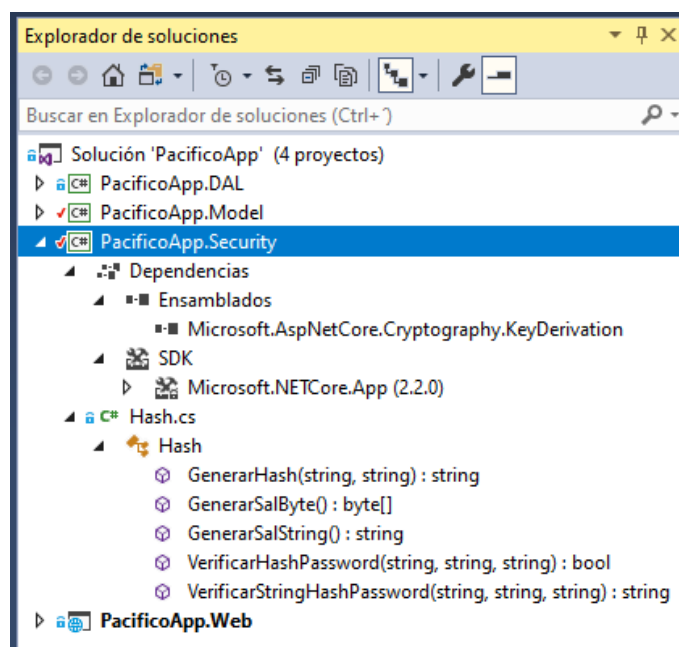


Figura 5-4 Seguridad

En esta sección Figura 5.4 se muestran los elementos necesarios para la seguridad del sistema y se describen a continuación.

- Dependencias

La inyección de dependencia es una técnica que ayuda a crear aplicaciones flexibles y simplifica las pruebas unitarias. *.NET Core* trae la inyección de dependencias de fábrica.

El término inyección de dependencia (DI) describe un enfoque para crear componentes acoplados libremente, que MVC usa automáticamente. Esto significa que los controladores y otros componentes no necesitan tener ningún conocimiento de cómo se crean los tipos que requieren.

- Hash.cs

El hash se usa para identificar una función criptográfica, es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija. Independientemente de la longitud de los datos de entrada, el valor hash de salida tendrá siempre la misma longitud.

Estas funciones sirven para asegurar la autenticidad de datos, almacenar de forma segura contraseñas, y la firma de documentos electrónicos (academy.bit2me.com, 2019).

Contiene una longitud de 40 caracteres y en este proyecto es utilizado para encriptar la información en la base de datos, cada valor agregado es transformado en un hash y toma el valor de 40 caracteres, cabe destacar que estos siempre serán únicos del registro, así si llega a haber dos nombres o contraseñas iguales para diferentes usuarios cada uno de ellos está respaldado por su propio hash.

5.4 Pacificoapp.Web (Presentación)

En esta capa se concentran todos los recursos necesarios para llevar al usuario por una grata navegación por el sitio, en la Figura 5.5 se observan todos esos elementos.

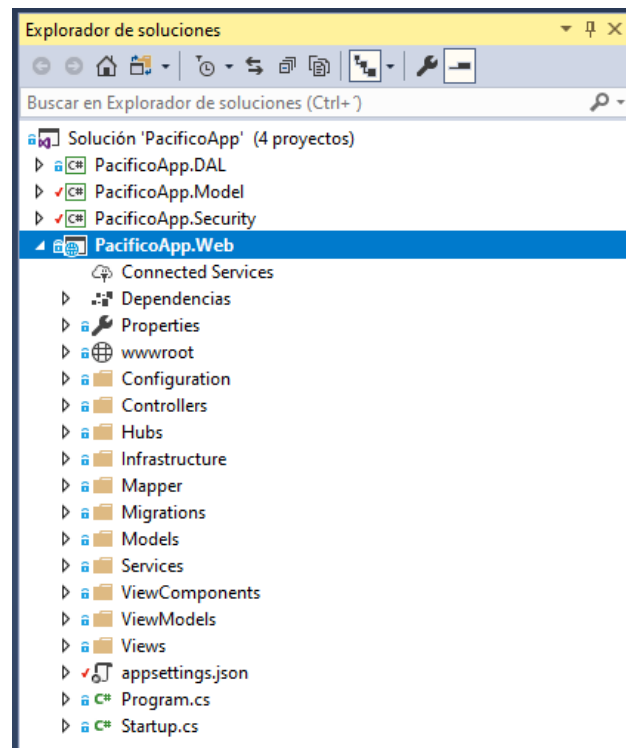


Figura 5-5 Presentación al usuario

- Controladores

La carpeta Controllers contiene archivos de clase para los controladores. Controladores maneja la solicitud de los usuarios y devuelve una respuesta. MVC requiere que el nombre de todos los archivos del controlador termine con "*Controller*" de otro modo no podrá ser llamado de manera correcta (mvc-folder-structure, 2019).

El controlador en la arquitectura MVC maneja cualquier solicitud de URL entrante. El controlador es una clase, derivada de la clase base *System.Web.Mvc.Controller*. La clase de controlador contiene métodos públicos llamados métodos de acción. El controlador y su método de acción maneja las solicitudes entrantes del navegador, recupera los datos del modelo necesarios y devuelve las respuestas apropiadas.

Cabe destacar que para que estos funcionen correctamente sus clases base ya deben estar creadas en la sección de modelos.

- Migraciones (BD)

Para la creación de esta carpeta se realizó lo siguiente : en la ventana Consola de Package Manager, se ingresó el siguiente comando: actualizar base de datos El comando *update-database* ejecuta el método *Up* para crear la base de datos y luego ejecuta el método *Seed* para llenar la base de datos, en este caso el único registro para llenar es el del usuario administrador, una vez generada la base aparece la carpeta Migrations junto con todas sus clases así como con los atributos de cada tabla generada. Figura 5.6.

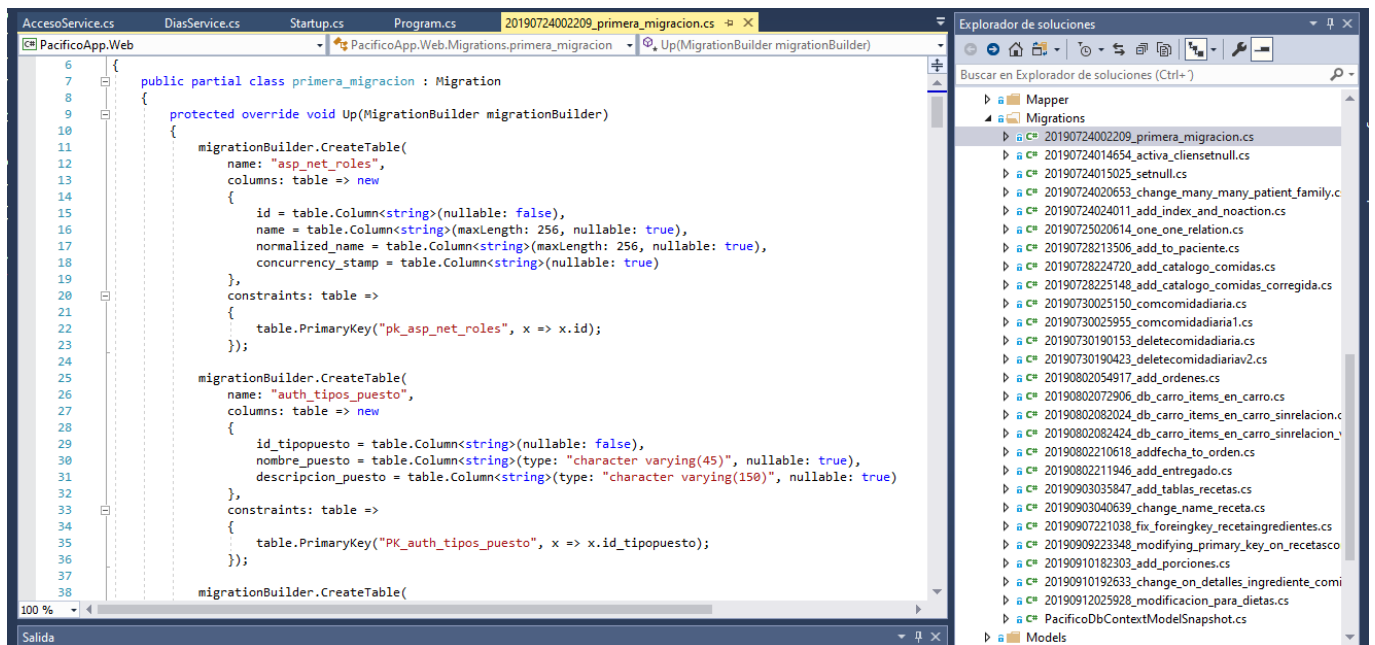


Figura 5-6 Primera migración

- Servicios: En esta sección están los permisos y las autenticaciones del hash.

En esta sección está el acceso a servicios, la autenticación de los mismos y el registro de los días de servicio. En la Figura 5.7 se muestra el archivo *AccesoService.cs* como ejemplo de la autenticación del paciente.

- Vistas de modelos

La vista es una interfaz de usuario. La vista muestra datos del modelo al usuario y también le permite modificar los datos.

Las vistas ASP.NET MVC se almacenan en la carpeta Vistas. Los diferentes métodos de acción de una sola clase de controlador pueden representar diferentes vistas, por lo que la carpeta

Vistas contiene una carpeta separada para cada controlador con el mismo nombre que el controlador, con el fin de acomodar múltiples vistas (Docs.microsoft, 2019).

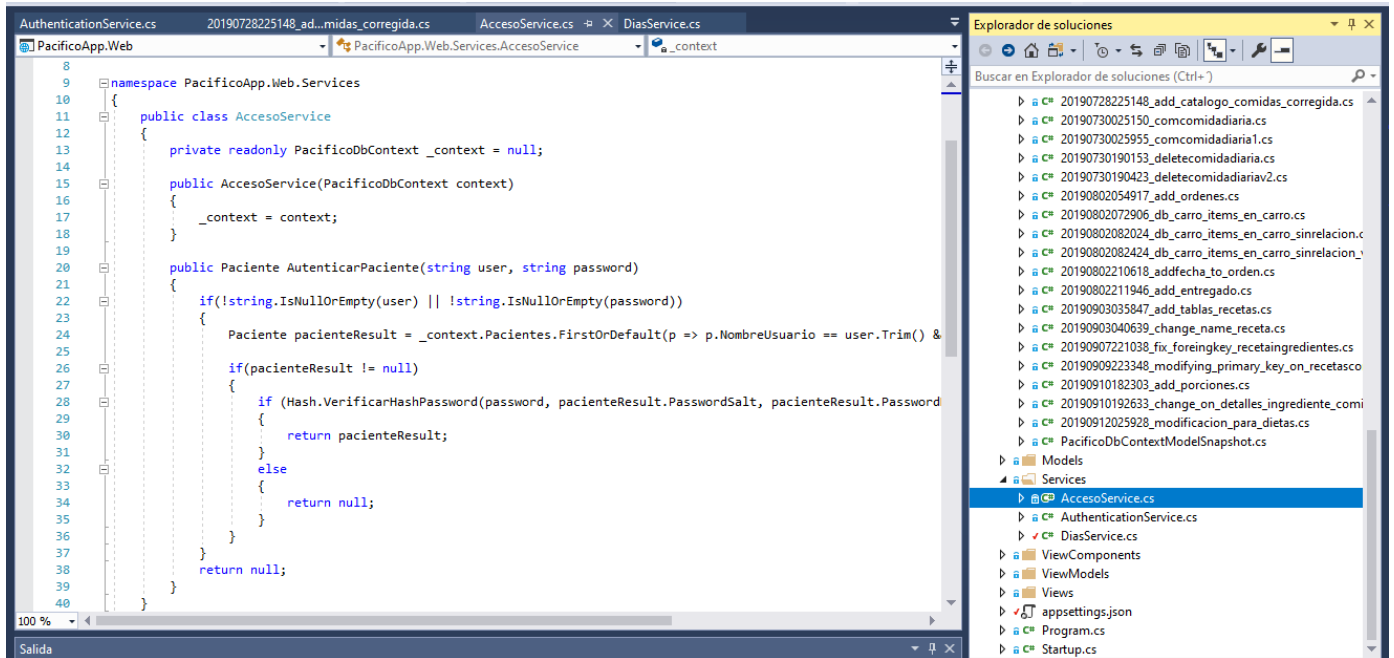


Figura 5-7 Acceso a servicios

Por ejemplo, Figura 5.8 las vistas, que se procesarán desde cualquiera de los métodos de acción de `CatalogoTiposDieta`, reside en `Vistas > CatalogoTipoDieta`. Del mismo modo, las vistas que se mostrarán desde `CocinaControler`, se ubicarán en la carpeta `Vistas > Cocina` como se muestra a continuación.

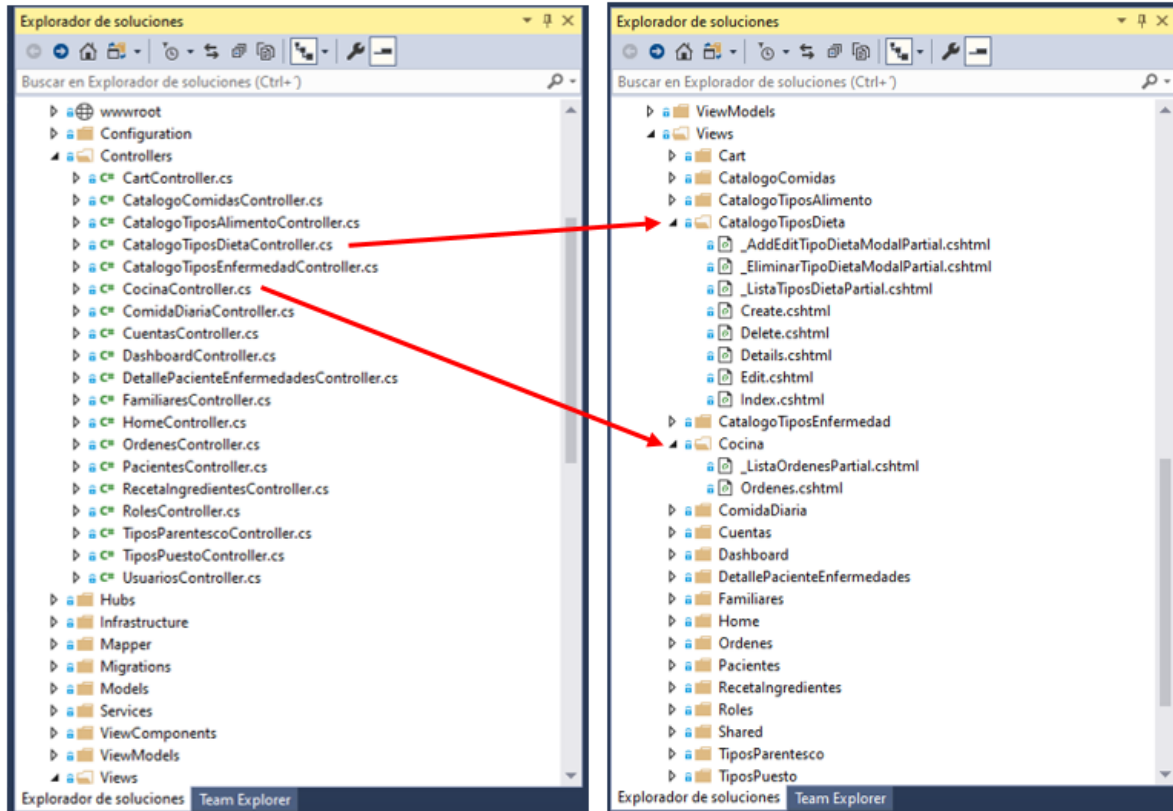


Figura 5-8 Relación entre controladores y vistas

En la parte de las configuraciones tenemos en el archivo *appsettings.json* Figura 5.9, las credenciales para el primer ingreso a la aplicación, al momento de generar la base de datos en la consola nuget se hace el registro de un usuario administrador con los siguientes datos, estos se guardan y encriptan en la base de datos, se hace este primer registro para que no se tenga que ir a la base de datos y ejecutar un *query* para la insercion del primer usuario, este proceso ahorra ese paso y se facilita el primer acceso al sistema.


```
1 {
2   "ConfigAdministrador": {
3     "UserName": "rosita@pacificoapp.com",
4     "NormalizedUserName": "ROSITA@PACIFICOAPP.COM",
5     "Email": "rosita@pacificoapp.com",
6     "Password": "Pacifico",
7     // "Password": "Qwerty123$", reset
8     "Nombre": "Rosa Maribel",
9     "ApellidoPaterno": "Martinez",
10    "ApellidoMaterno": "Manzo"
11  },
12  "Settings": {
13    "PacificoCookie": "PacificoAppCookieAuthenticationScheme"
14  },
15  "ConnectionStrings": {
16    "DefaultConnection": "Host=localhost;Database=dbpacifico;Username=postgres;Password=2408"
17  },
18  "Logging": {
19    "LogLevel": {
20      "Default": "Warning"
21    }
22  },
23  "AllowedHosts": "*"
24 }
25
```

Figura 5-9 Primer registro al crear la base de datos y credenciales para la conexión de la BD

5.5 Desarrollo de Módulos

5.5.1 Registro de pacientes

En esta sección se ve el código para el registro de pacientes, como se ve en la Figura 5.10 la variable *pacificoDbContext* espera al contexto de pacientes que también incluye los catálogos de dieta, los usuarios de la aplicación y la lista de la orden del paciente, se crea una nueva variable para la vista del modelo *PacienteViewModel* y esta se mapea en la lista de *pacificoDbContext*, si el resultado es diferente de nulo entonces se despliegan los atributos para registro del paciente

como son su nombre, apellidos, nombre de usuario, fecha de registro, usuario y tipo de dieta, y esto se regresa a la vista del modelo.

```
// GET: Pacientes
public async Task<IActionResult> Index(string id = null)
{
    var pacificoDbContext = await _context.Pacientes
        .Include(p => p.CatalogoTiposDieta)
        .Include(p => p.UsuariosAplicacion)
        .AsNoTracking().OrderBy(p => p.NombrePaciente).ToListAsync();

    PacienteViewModel pacienteViewModel = new PacienteViewModel();
    pacienteViewModel.Pacientes = _mapper.Map<List<PacienteViewModel>>(pacificoDbContext);

    if (id != null)
    {
        PacienteViewModel defaultPatient = pacienteViewModel.Pacientes.FirstOrDefault(p => p.IdPaciente == id.Trim());
        ViewData["PacienteId"] = defaultPatient.IdPaciente;
        pacienteViewModel.NombrePaciente = defaultPatient.NombrePaciente;
        pacienteViewModel.ApellidoPaterno = defaultPatient.ApellidoPaterno;
        pacienteViewModel.ApellidoMaterno = defaultPatient.ApellidoMaterno;
        pacienteViewModel.NombreUsuario = defaultPatient.NombreUsuario;
        pacienteViewModel.FechaRegistro = defaultPatient.FechaRegistro;
        pacienteViewModel.UsuarioRegistra = defaultPatient.UsuarioRegistra;
        pacienteViewModel.TipoNombreCatDieta = defaultPatient.TipoNombreCatDieta;
    }

    return View(pacienteViewModel);
}
```

Figura 5-10 Obtención de pacientes

Alta de usuarios

Para la creación de los usuarios se tiene el siguiente código, como se ve en la Figura 5.11 se tiene la variable *usuarioAplicacion* que espera el correo del usuario y la variable *IdentityResult* crea una nueva variable que recibe el resultado de la identidad, si el resultado es nulo significa que es un nuevo usuario y entonces pide los siguientes datos, el correo para registro, el nombre de usuario, apellido paterno, apellido materno y el tipo de puesto, con esto se registra un nuevo usuario, si el usuario es agregado exitosamente entonces aparece el mensaje de “nuevo usuario agregado correctamente” y se redirecciona al índice, si el usuario ya está registrado aparece el mensaje “Error el correo de usuario ya existe” y se regresa de nuevo al registro.

```
appsettings.json # PacificoDbContext.cs # 20190724002209_primera_migracion.cs # UsuariosController.cs # X CatalogoComida.cs #
PacificoApp.Web # PacificoApp.Web.Controllers.UsuariosController # Create()
89 // POST: Usuarios/Create
90 [HttpPost]
91 [ValidateAntiForgeryToken]
92 1 referencia | 0 cambios | 0 autores, 0 cambios | 0 solicitudes | 0 excepciones
93 public async Task<IActionResult> Create(/*IFormCollection collection*/ UsuarioAplicacionViewModel model)
94 {
95     try
96     {
97         if (ModelState.IsValid)
98         {
99             // TODO: Add insert logic here
100             UsuarioAplicacion user = await _userManager.FindByEmailAsync(model.CorreoElectronico.Trim());
101             IdentityResult identityResult = new IdentityResult();
102
103             if (user == null)
104             {
105                 UsuarioAplicacion nuevoUser = new UsuarioAplicacion()
106                 {
107                     UserName = model.CorreoElectronico,
108                     NombreUsuario = model.NombreUsuario.Trim().ToUpper(),
109                     ApellidoPaterno = model.ApellidoPaterno.Trim().ToUpper(),
110                     ApellidoMaterno = model.ApellidoMaterno.Trim().ToUpper(),
111                     Email = model.CorreoElectronico,
112                     IdTipoPuesto = model.IdTipoPuesto.Trim()
113                 };
114
115                 identityResult = await _userManager.CreateAsync(nuevoUser, model.Password);
116
117                 if (identityResult.Succeeded)
118                 {
119                     _toastNotification.AddSuccessToastMessage("Nuevo usuario agregado correctamente");
120                     return RedirectToAction(nameof(Index));
121                 }
122             }
123             else
124             {
125                 _toastNotification.AddErrorToastMessage("Error el correo del usuario ya existe");
126                 return RedirectToAction(nameof(UsuariosController.Create), model);
127             }
128         }
129     }
130     catch (Exception error)
131     {
132         return NotFound(error.Message);
133     }
134
135     PopulateTipoPuestoDropDownList(model.IdTipoPuesto);
136     return View(model);
137 }
138
```

100 %

Listo LIn 87 Col 10 Car 10 INS

Figura 5-11 UsuarioController.cs

5.5.2 Comida diaria

La relación se encuentra en que al paciente se le asigna una dieta conforme el medico lo indica y esa dieta se agrega a su registro entonces en el apartado donde se registraron las comidas también se le asigna uno o varios tipos de dieta a la comida, esa es la disponibilidad de la comida conforme a la dieta, el paciente una vez teniendo asignada la dieta no puede escoger un alimento que no esté dentro de su dieta, lo mismo está relacionado con el tipo de enfermedad, así mismo el carro de compra está ligado a la selección de la dieta.

En la parte de los tipos de dietas que pertenecen a los pacientes y los tipos de dietas con los alimentos, llevan estructuras de bifurcación anidados como es la cláusula *if* junto con la cláusula *else*, y los datos son obtenidos a través de una consulta que se realiza mediante *LINQ*, con el filtro *where*, delimitamos la comida diaria correspondiente al usuario, ya que el usuario dentro de sus registros contiene un *flag* o bandera que ayudara a diferenciar de qué tipo de dietas tiene, posteriormente la información se rellena en el contenedor que es una colección que indica que tipo de alimentos tiene permitido ver el usuario, en su sesión principal para que él pueda seleccionarlo.

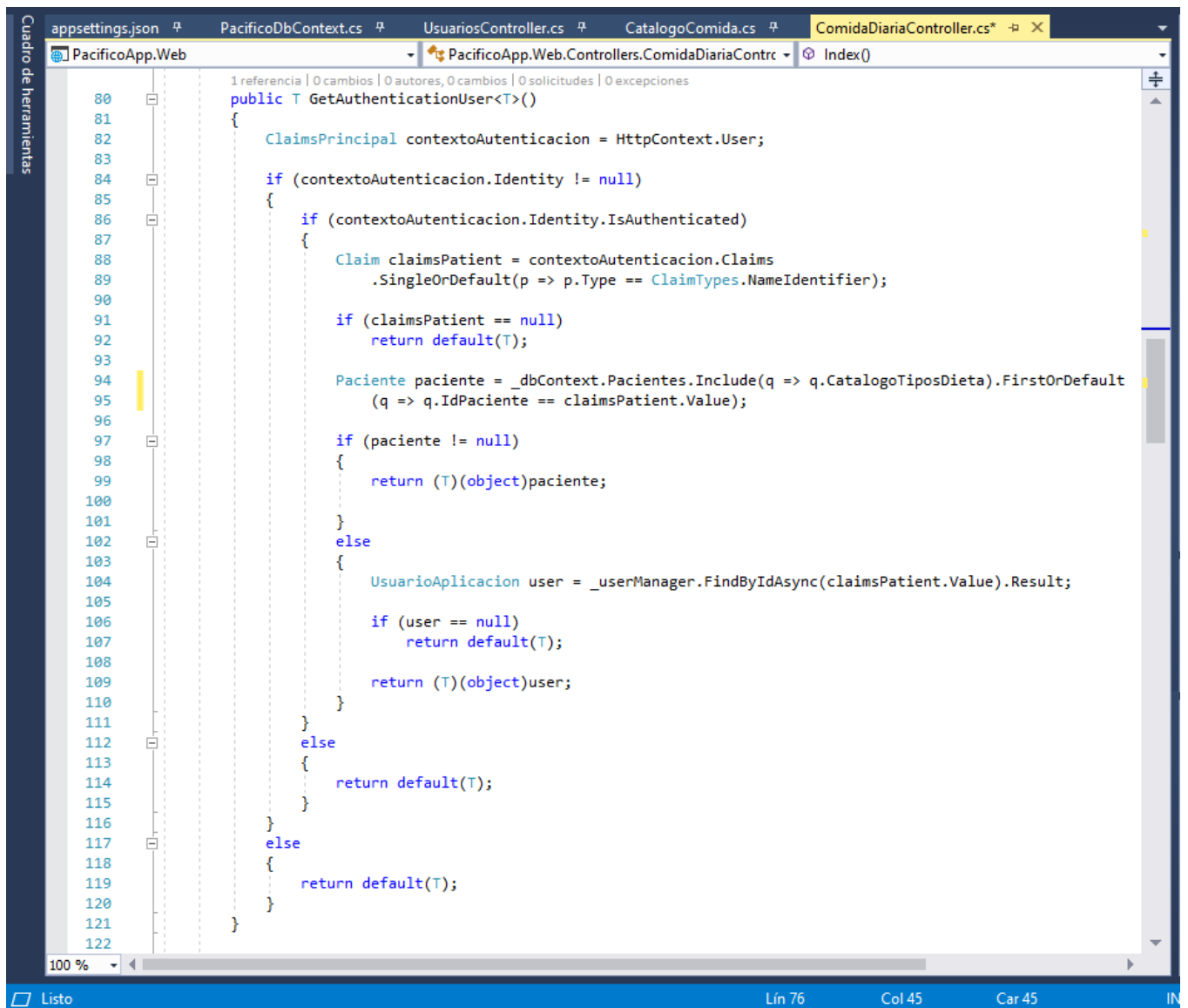
```
32 // GET: ComidaDiaria
33 3 referencias | 0 cambios | 0 autores, 0 cambios | 0 solicitudes | 0 excepciones
34 public async Task<ActionResult> Index()
35 {
36     DateTime tiempoHoy = DateTime.Now;
37     int diaHoy = Convert.ToInt16(tiempoHoy.DayOfWeek)-1;
38
39     IEnumerable<CatalogoComida> catalogoComida = new List<CatalogoComida>();
40
41     Dia diaSeleccionado = DiasService.ListaDias().Find(e => e.Numero == diaHoy);
42
43     var catalogo = await _dbContext.CatalogoComidas
44         .Where(q => q.Frecuencia.Contains(diaSeleccionado.Nombre))
45         .OrderBy(q => q.NombreTipico)
46         .ToListAsync();
47
48     ComidaDiaraViewModel viewModel = new ComidaDiaraViewModel();
49     try
50     {
51         var getUserContext = this.GetAuthenticationUser<object>();
52
53         if (getUserContext != null)
54         {
55             if (getUserContext.GetType() == typeof(Paciente))
56             {
57                 string nombreDieta = ((Paciente)getUserContext).CatalogoTiposDieta.NombreCatDieta;
58                 catalogoComida = catalogo.Where(q => q.Dietas.Contains(nombreDieta));
59                 viewModel.CatalogoComidas = catalogoComida;
60                 viewModel.Dias = diaSeleccionado;
61             }
62
63             if (getUserContext.GetType() == typeof(UsuarioAplicacion))
64             {
65                 viewModel.CatalogoComidas = catalogo;
66                 viewModel.Dias = diaSeleccionado;
67             }
68         }
69     } catch (Exception)
70     {
71     }
72     throw;
73 }
74
75 return View("Index", viewModel);
76 }
77
78
```

100 %

Listo Lin 27 Col 10 Car 10

Figura 5-12 Consulta LINQ

Por lo que considero que estamos usando una expresión de asociación basada en el tipo de enfermedad que tiene el paciente y en el tipo de dieta que puede consumir, realizado abstractamente mediante las consultas de *LINQ*, para denotarlo puede observar la Figura 5.12, en la cual explícitamente se puede observar el código para la consulta principal de *LINQ* y a través del tipo de autenticación de usuario decidir si el paciente puede tener o no la lista de alimentos en su ventana principal.



```
appsettings.json PacíficoDbContext.cs UsuariosController.cs CatalogoComida.cs ComidaDiariaController.cs* X
PacíficoApp.Web PacíficoApp.Web.Controllers.ComidaDiariaContr Index()
1 referencia | 0 cambios | 0 autores, 0 cambios | 0 solicitudes | 0 excepciones
80 public T GetAuthenticationUser<T>()
81 {
82     ClaimsPrincipal contextoAutenticacion = HttpContext.User;
83
84     if (contextoAutenticacion.Identity != null)
85     {
86         if (contextoAutenticacion.Identity.IsAuthenticated)
87         {
88             Claim claimsPatient = contextoAutenticacion.Claims
89                 .SingleOrDefault(p => p.Type == ClaimTypes.NameIdentifier);
90
91             if (claimsPatient == null)
92                 return default(T);
93
94             Paciente paciente = _dbContext.Pacientes.Include(q => q.CatalogoTiposDieta).FirstOrDefault
95                 (q => q.IdPaciente == claimsPatient.Value);
96
97             if (paciente != null)
98             {
99                 return (T)(object)paciente;
100             }
101         }
102         else
103         {
104             UsuarioAplicacion user = _userManager.FindByIdAsync(claimsPatient.Value).Result;
105
106             if (user == null)
107                 return default(T);
108
109             return (T)(object)user;
110         }
111     }
112     else
113     {
114         return default(T);
115     }
116 }
117 else
118 {
119     return default(T);
120 }
121 }
122 }
```

100 %

Listo Lin 76 Col 45 Car 45 IN

Figura 5-13 ComidaDiariaController.cs

En el código anterior el método de consulta de comida diaria *index*, que se encuentra en el controlador *ComidaDiaraController*, que está apoyado en el método de autenticación de usuarios como se puede observar en la Figura 5.13 *GetAuthenticationUser<T>()*, y que devuelve un tipo de usuario genérico que es pasado como referencia, el cual obtiene el tipo de usuario en el contexto de la sesión y devuelve únicamente dos tipos de usuario; usuario que administra la aplicación y el paciente registrado previamente en la base de datos.

5.5.3 Compra de ordenes

En el código mostrado en la Figura 5.14 se puede observar la programación para el inicio de la clase *OrdenesController* que recibe variables solo de lectura, el *PacificoDbContext* la cual es privada, la variable pública *carro* que también es solo de lectura, la variable de manipulación de usuario que también es privada y solo de lectura y por último el *hubContext* que también es privada, en el método *OrdenesController* que recibe las variables arriba mencionadas.

```
[Authorize(Roles = "Administrador, Paciente, Cocina, Personal")]
1 referencia | 0 cambios | 0 autores, 0 cambios
public class OrdenesController : Controller
{
    private readonly PacificoDbContext _context;
    public readonly Carro _carro;
    private readonly UserManager<UsuarioAplicacion> _userManager = null;
    private IHubContext<SignalServer> _hubContext;

    0 referencias | 0 cambios | 0 autores, 0 cambios | 0 excepciones
    public OrdenesController(PacificoDbContext context, Carro carro, UserManager<UsuarioAplicacion>
        userManager, IHubContext<SignalServer> hubContext)
    {
        _context = context;
        _carro = carro;
        _userManager = userManager;
        _hubContext = hubContext;
    }
}
```

Figura 5-14 *OrdenesController : Controller*

En la figura 5.15 vemos el código para el método *MisOrdenes()* el cual recibe variables vacías de tipo *string*: el *idGenerico*, el nombre de usuario y el número de cama, lo primero que hace es recibir el valor de la variable de autenticación *getAuthenticationUser* si esta es nula entonces regresa que no ha sido encontrada, de otra manera entra a la siguiente fase, el tipo de usuario que está en el sistema, si el usuario es un paciente se obtienen sus atributos, si el usuario tiene otro tipo de rol también pide su id y su nombre de usuario.

```
public IActionResult MisOrdenes()
{
    string idGenerico = string.Empty;
    string nombreUser = string.Empty;
    string numCama = string.Empty;

    var getAuthentication = GetAuthenticationUser<object>();

    if (getAuthentication == null)
        return NotFound();

    if (getAuthentication.GetType() == typeof(Paciente))
    {
        idGenerico = ((Paciente)getAuthentication).IdPaciente;
        nombreUser = ((Paciente)getAuthentication).NombreCompleto;
        numCama = ((Paciente)getAuthentication).NumCama.ToString();
    }

    if (getAuthentication.GetType() == typeof(UsuarioAplicacion))
    {
        idGenerico = ((UsuarioAplicacion)getAuthentication).Id;
        nombreUser = ((UsuarioAplicacion)getAuthentication).NombreCompleto;
    }

    if (string.IsNullOrEmpty(idGenerico))
        return NotFound();

    IQueryable<ItemEnCarro> ordenes = _context.ItemsEnCarro
        .Include(q => q.Ordenes)
        .Include(q => q.CatalogoComida)
        .Where(q => q.Ordenes.IdUsuarioPidio == idGenerico)
        .OrderBy(q => q.Ordenes.FechaOrden);

    OrdenViewModel viewModel = new OrdenViewModel();

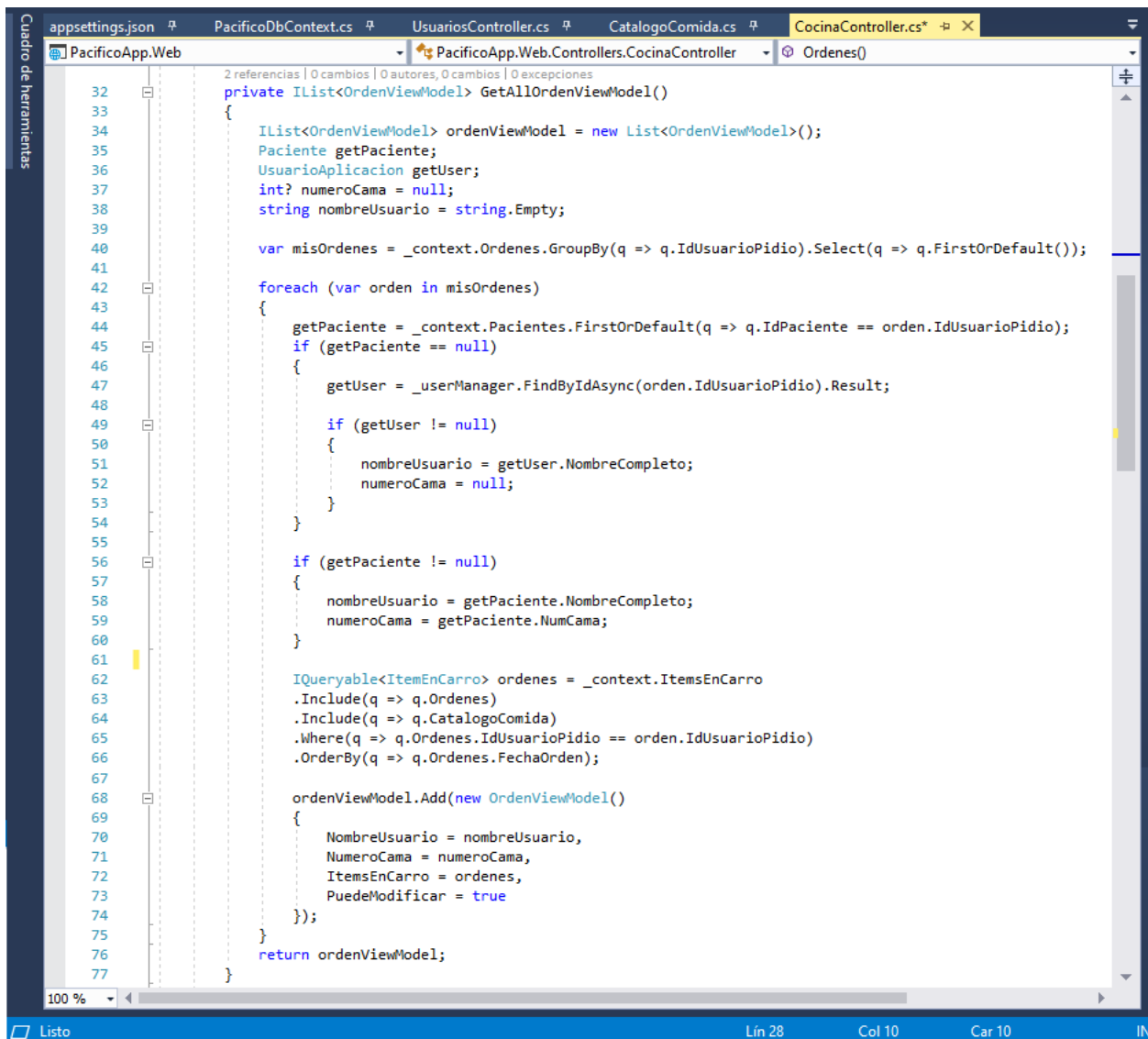
    if (numCama == string.Empty)
    {
        viewModel.NombreUsuario = nombreUser;
        viewModel.ItemsEnCarro = ordenes;
        viewModel.PuedeModificar = false;
    }
    else
    {
        viewModel.NombreUsuario = nombreUser;
        viewModel.ItemsEnCarro = ordenes;
        viewModel.NumeroCama = Convert.ToInt32(numCama);
        viewModel.PuedeModificar = false;
    }

    return View(viewModel);
}
```

Figura 5-15 Mis ordenes()

En código siguiente el método *GetAllOrdenViewModel()* mostrado en la Figura 5.16, de la clase *CocinaController.cs* obtiene el contexto de los pacientes y recorre los elementos de la colección *misOrdenes* para obtener al paciente.

Cuando se hace la consulta *LINQ* de ítems del carro esta manda a llamar las ordenes, el catálogo de comida donde las ordenes son las comidas que el usuario pidió y las ordena por fecha y las muestra en la vista con el nombre de usuario, el número de cama y el listado de las ordenes que tiene ese paciente.



```
32 private IList<OrdenViewModel> GetAllOrdenViewModel()
33 {
34     IList<OrdenViewModel> ordenViewModel = new List<OrdenViewModel>();
35     Paciente getPaciente;
36     UsuarioAplicacion getUser;
37     int? numeroCama = null;
38     string nombreUsuario = string.Empty;
39
40     var misOrdenes = _context.Ordenes.GroupBy(q => q.IdUsuarioPidio).Select(q => q.FirstOrDefault());
41
42     foreach (var orden in misOrdenes)
43     {
44         getPaciente = _context.Pacientes.FirstOrDefault(q => q.IdPaciente == orden.IdUsuarioPidio);
45         if (getPaciente == null)
46         {
47             getUser = _userManager.FindByIdAsync(orden.IdUsuarioPidio).Result;
48
49             if (getUser != null)
50             {
51                 nombreUsuario = getUser.NombreCompleto;
52                 numeroCama = null;
53             }
54         }
55
56         if (getPaciente != null)
57         {
58             nombreUsuario = getPaciente.NombreCompleto;
59             numeroCama = getPaciente.NumCama;
60         }
61
62         IQueryable<ItemEnCarro> ordenes = _context.ItemsEnCarro
63             .Include(q => q.Ordenes)
64             .Include(q => q.CatalogoComida)
65             .Where(q => q.Ordenes.IdUsuarioPidio == orden.IdUsuarioPidio)
66             .OrderBy(q => q.Ordenes.FechaOrden);
67
68         ordenViewModel.Add(new OrdenViewModel()
69         {
70             NombreUsuario = nombreUsuario,
71             NumeroCama = numeroCama,
72             ItemsEnCarro = ordenes,
73             PuedeModificar = true
74         });
75     }
76     return ordenViewModel;
77 }
```

Figura 5-16 *CocinaController.cs*

5.5.4 Órdenes del día en cocina

En la figura 5.17 se tiene el código para mostrar el catálogo de comidas este arroja como resultado de vista las comidas que son del día y dependiendo del tipo de usuario que sea es lo que se le va a permitir visualizar, en caso de ser paciente esto se restringe a su dieta previamente registrada y solo se mostraran los alimentos que sean de ésta y que además estén disponibles el día en que se está haciendo la petición, si son otros usuarios como administradores y personal estos no tienen más restricciones en la comida más que la que seleccionan sea del día.

```
public async Task<ActionResult> Index()
{
    DateTime tiempoHoy = DateTime.Now;
    int diaHoy = Convert.ToInt16(tiempoHoy.DayOfWeek)-1;

    IEnumerable<CatalogoComida> catalogoComida = new List<CatalogoComida>();

    Dia diaSeleccionado = DiasService.ListaDias().Find(e => e.Numero == diaHoy);

    var catalogo = await _dbContext.CatalogoComidas
        .Where(q => q.Frecuencia.Contains(diaSeleccionado.Nombre))
        .OrderBy(q => q.NombreTipico)
        .ToListAsync();

    ComidaDiaraViewModel viewModel = new ComidaDiaraViewModel();
    try
    {
        var getUserContext = this.GetAuthenticationUser<object>();

        if (getUserContext != null)
        {
            if (getUserContext.GetType() == typeof(Paciente))
            {
                string nombreDieta = ((Paciente)getUserContext).CatalogoTiposDieta.NombreCatDieta;
                catalogoComida = catalogo.Where(q => q.Dietas.Contains(nombreDieta));
                viewModel.CatalogoComidas = catalogoComida;
                viewModel.Dias = diaSeleccionado;
            }

            if (getUserContext.GetType() == typeof(UsuarioAplicacion))
            {
                viewModel.CatalogoComidas = catalogo;
                viewModel.Dias = diaSeleccionado;
            }
        }
    }
    catch (Exception)
    {
        throw;
    }

    return View("Index", viewModel);
}
```

Figura 5-17 Catalogo comida diaria

Capítulo 6 Resultados

En este capítulo se ve el programa como resultado de todo lo anterior, es la parte que ya visualizan los usuarios finales de la aplicación, se muestran los diferentes roles y se explican cada una de las secciones y opciones que tiene el sistema.

Como pantalla principal como se muestra en la Figura 6.1 está la presentación se da la bienvenida al usuario, y se le muestra un video sacado de sus redes sociales, así como un mapa para la ubicación y un listado de servicios.

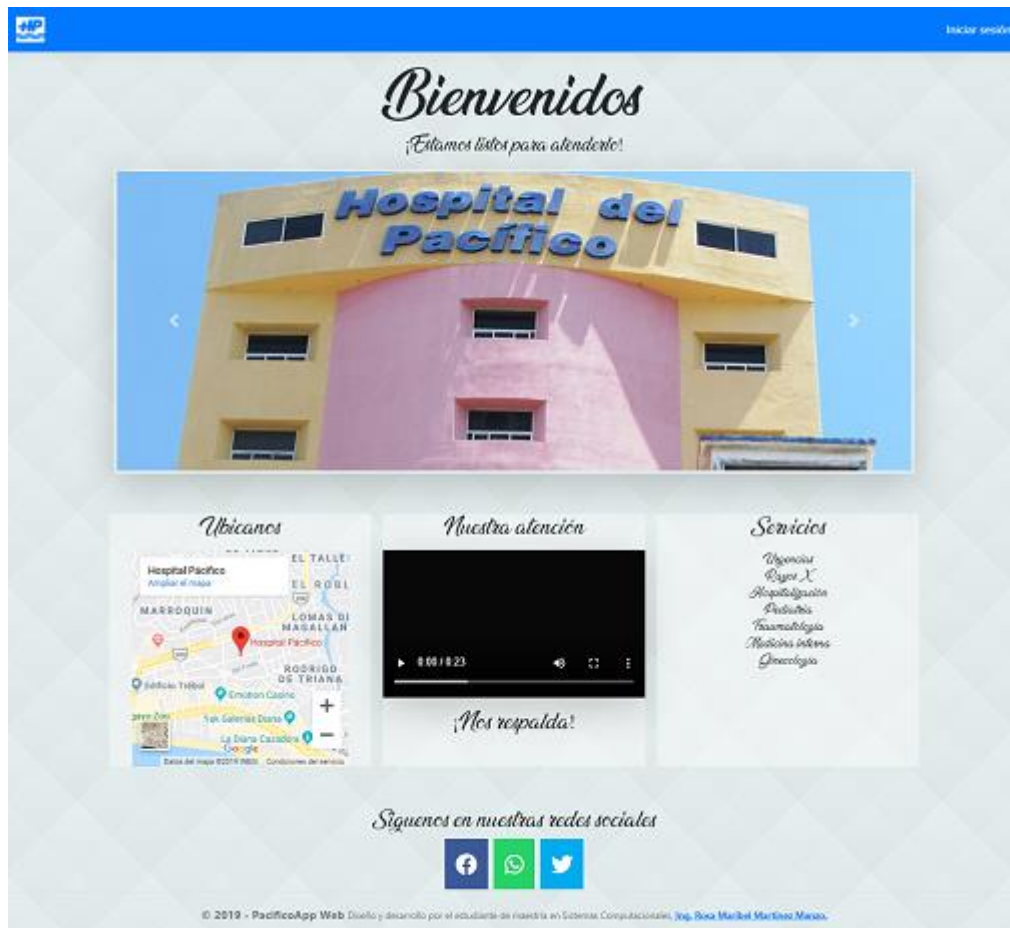


Figura 6-1 Portada inicial

6.1 Secciones

6.1.1 Como rol de administrador

En esta sección se verá el acceso al sistema como administrador, este rol tiene todos los privilegios para hacer cambios y registros tanto de pacientes como de personal para que utilicen el sistema, otorga permisos a otros usuarios dependiendo del rol con el que fueron registrados y tiene control sobre cambio de precios.

Comida Diaria

Mi Compra

Se despliega una lista en donde viene las siguientes casillas: cantidad, el platillo o comida seleccionado, el precio unitario y el subtotal, así como los botones de “regresar al menú” y “enviar a cocina”

Configuración:

- **Acceso al sistema**
- **Usuarios**

Se muestra el listado de los usuarios que tienen acceso al sistema, viene la opción de registro de nuevo usuario, el listado cuenta con la información básica como es el nombre completo del usuario, el puesto que desempeña, el correo electrónico y el tipo de permiso que posee dentro del sistema, así como las opciones de modificar el registro, la eliminación del usuario y los permisos de usuario.

El registro de nuevo usuario contiene un formulario para rellenar lo siguiente: nombre, apellidos materno y paterno de la persona, así como el puesto que desempeña, el correo electrónico y la contraseña de acceso al sistema, la cual deberá de conocer solo él.

En la sección de permisos de usuario muestra de igual manera el nombre completo y correo electrónico del usuario, así como la opción del reseteo de la contraseña de ingreso al sistema en caso de que el usuario la olvide, también cuenta con el listado de roles de acceso al sistema para que se le asignen al usuario, tales roles son: Administrador, Paciente, Personal y Cocina. Figura 6.2










Nombre de usuario	Apellido paterno	Apellido materno	Puesto	Correo electrónico	Permisos	
EMILY	EMILY	EMILY	COCINA	emily@gmail.com	• Cocina	  
MARIBEL	MANZO	ZAPATA	ADMINISTRADOR	maribel@gmail.com	• Personal	  
ROSA MARIBEL	MARTINEZ	MANZO	ADMINISTRADOR	rosita@pacificoapp.com	• Administrador	  

Figura 6-2 Usuarios registrados

- **Tipo de puesto**

En esta sección encontramos la opción de registrar un nuevo puesto, así como un listado de los puestos mostrando el nombre y descripción de estos con la opción de editarlos o eliminarlos. Figura 6.3

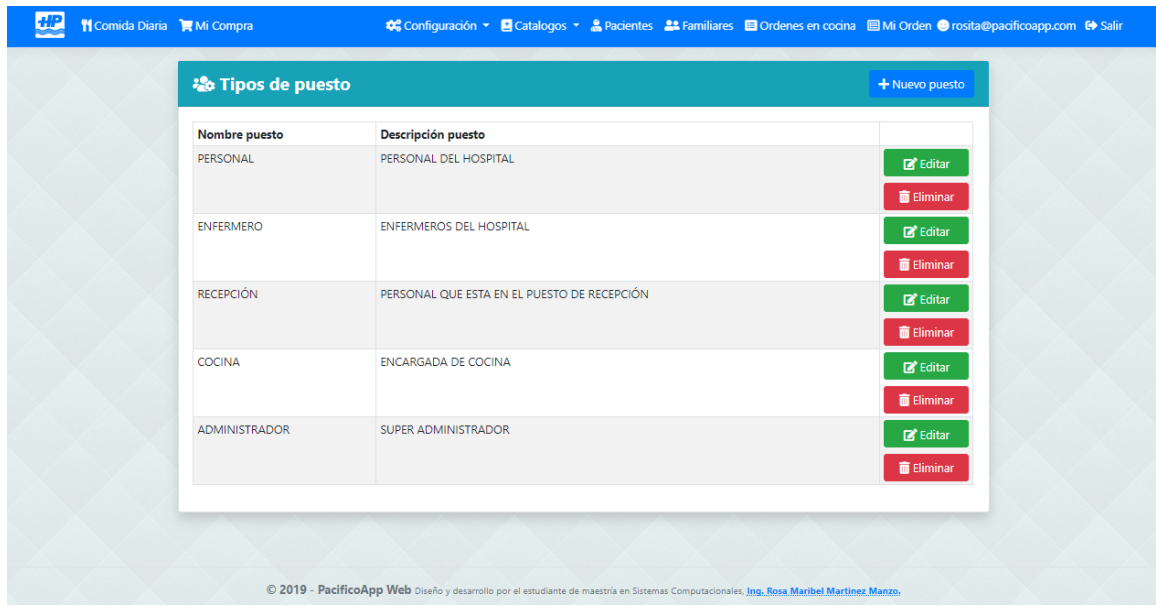


Figura 6-3 Tipos de puestos

- **Roles de usuario**

Muestra el listado de los roles, tales roles son: Administrador, Paciente, Personal y Cocina.

Catálogos:

- **Comidas**

- **Platillos y Comidas**

En esta sección se enlistan los diferentes platillos contenidos en el menú independientemente de la dieta a la que pertenezcan, se tiene la opción de registro de una nueva comida, y en la lista se muestra el nombre típico del platillo, la descripción y el precio, así como la opción de modificar el registro y los detalles de la comida y la eliminación de la misma. Figura 6.4

Al momento de registrar una nueva comida se piden los siguientes datos: el nombre típico, una breve descripción, se puede elegir los tipos de dietas a las que pertenecen, el precio, las porciones una sección de notas extras y la opción de poner los días en los que se prepara esta comida.

En los detalles de la comida aparece el nombre típico ya registrado, así como la descripción, las notas, las dietas a las que pertenece, el precio, las porciones y la fecha de registro, el costo por porción y el costo de la receta por total de porciones, esta sección también cuenta con la posibilidad de asignar los ingredientes que esta contiene, así como la cantidad y costo por ingrediente, el programa calcula el precio del ingrediente utilizado en esa comida en específico respecto a la cantidad de este que se utilice en la receta, para esto los ingredientes ya deben estar previamente registrados para solo seleccionar en el listado de ingredientes el que la receta utiliza, poner la cantidad que se utiliza y el precio por kilo para que el programa haga el cálculo.

Nombre típico	Descripción	Precio
CALDO DE RES	CALDO DE RES	60.00
TACOS DORADOS CON CONSOMÉ 2	RICOS TACOS DORADOS CON CONSOME	45.00
TACOS DORADOS CON CONSOMÉ3	ROQUISIMOS TACOS DE CONSOME CON VERDURA	46.00
TORTITAS DE PAPA	TORTITAS DE PAPA EN CALDO DE JITOMATE	50.00
TOSTADAS DE TINGA	RICAS TOSTADAS DE TINGA CON VERDURA CREMA Y QUESO	50.00
TACOS DORADOS CON CONSOMÉ	TACOS DORADOS CON CONSOMÉ	45.00
ENCHILADAS ROJAS CON POLLO	ENCHILADAS ROJAS CON POLLO	30.00

© 2019 - PacificoApp Web Diseño y desarrollo por el estudiante de maestría en Sistemas Computacionales, Ing. Rosa Maribel Martínez Manzo.

Figura 6-4 Catálogo de comidas

- **Comida Diaria**

Muestra el listado de todas las comidas que se sirven diario se muestra el nombre, la descripción y la opción de agregarlos al carrito de pedidos para hacer la orden.

- **Ingredientes**

En esta sección se encuentra el listado de todos los ingredientes utilizados para todas las recetas con las que se hacen las comidas y las dietas, trae un listado con el nombre del ingrediente, la cantidad, el peso, el costo por unidad, la cantidad utilizable del ingrediente, el costo de la unidad utilizable, el costo en receta del ingrediente, el contenedor y la unidad de medida o peso, así como la opción de editar o eliminar dicho ingrediente, al momento de registrar un nuevo ingrediente se piden todos los datos anteriores y estos son mostrados en el listado.

- **Enfermedades y Padecimientos**

Aquí se muestran las enfermedades o padecimientos más comunes que las personas tienen al momento de ingresar al hospital, tiene la opción de agregar tantos como sean necesarios, de igual manera después de su registro tiene la opción de ver los detalles, editarlos o eliminar la enfermedad.

- **Dietas**

En esta sección se registran las dietas con las que cuenta el hospital, se agrega su nombre y su descripción y de igual manera se pueden editar o eliminar. Figura 6.5

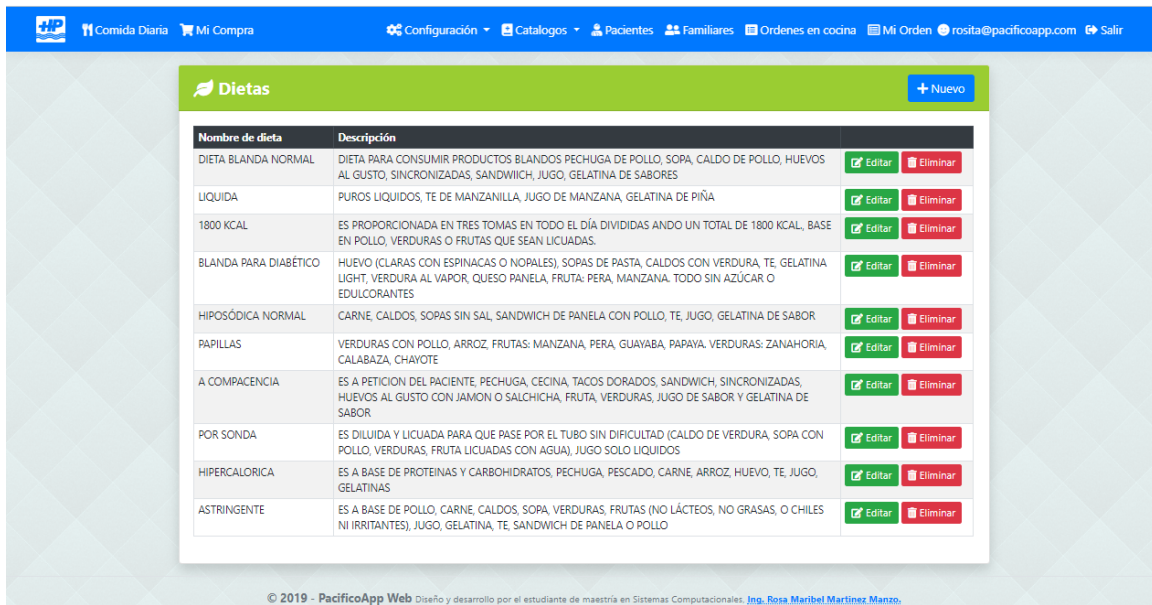


Figura 6-5 Dietas que maneja el hospital

Pacientes

Al registrar al paciente se le pide nombre completo, edad, numero de cama, tipo de dieta, alguna nota, el nombre de usuario y una contraseña para acceso al sistema como se muestra en la Figura 6.6.

Registrar paciente

Nombre del paciente: Apellido paterno: Apellido materno:

Tipo de dieta:

Número de cama: Edad:

Notas:

Nombre de usuario:

Contraseña: Confirmar contraseña:

Figura 6-6 Registro de pacientes

Una vez registrado el paciente en la sección de detalles se muestran todos los datos de registro, así como el usuario que lo registró, la fecha y la hora de registro, su status, su nombre de usuario y la opción de agregar su enfermedad o padecimiento y a su familiar a cargo.

Familiares

En este apartado aparece el listado de los familiares ya registrados en la sección de pacientes, viene su nombre completo, el parentesco y el paciente asociado, tiene la opción de editar la información y/o borrar el registro.

Ordenes en Cocina

En esta sección viene el listado de todos los platillos que se han pedido a cocina, en este vienen conceptos como la cantidad de platillos, el nombre del platillo, el precio unitario, el subtotal, total, la fecha y hora en que se realizó el pedido y la opción de ingresar si se entregó o no la comida. Figura 6.7.

Ordenes en cocina						
Nombre del usuario				Número de cama		
ROSA MARIBEL MARTINEZ MANZO						
Cantidad	Nombre platillo o comida	Precio unitario	Subtotal	Fecha	Entregado	
1	CALDO DE RES	\$60.00	\$60.00	viernes, 13 septiembre 2019 17:33	No	
			Total:	0		
Nombre del usuario				Número de cama		
LINDA NELLY SALES LUGARDO				101		
Cantidad	Nombre platillo o comida	Precio unitario	Subtotal	Fecha	Entregado	
1	TOSTADAS DE TINGA	\$50.00	\$50.00	viernes, 15 noviembre 2019 00:35	No	
			Total:	0		

Figura 6-7 Ordenes de cocina

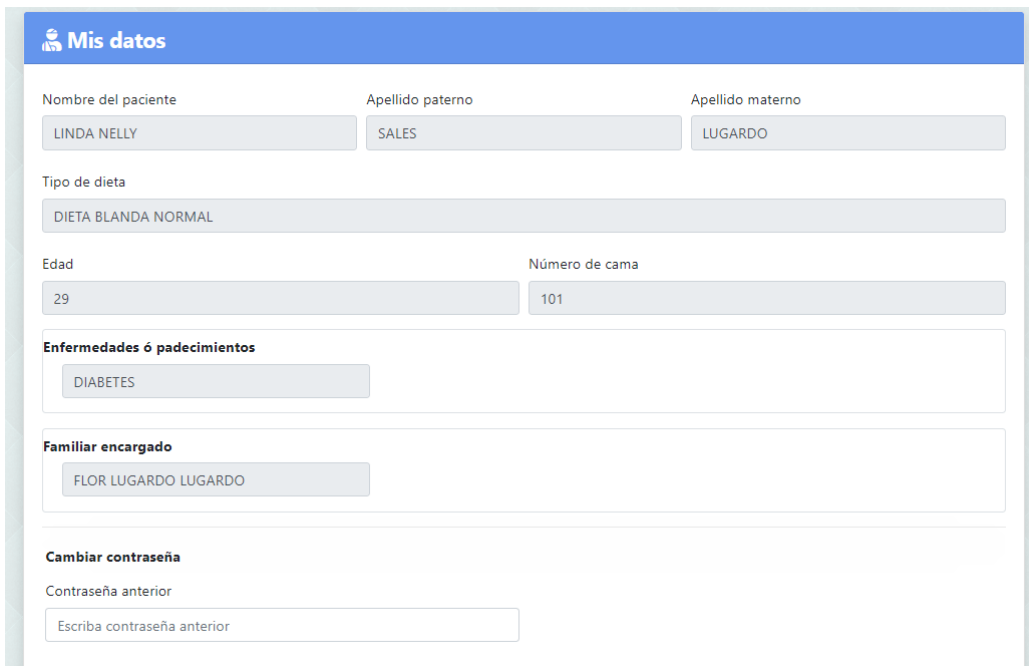
Mi Orden

En esta sección viene el listado de los platillos que se han pedido a cocina en esa sesión con ese usuario, en este vienen conceptos como la cantidad de platillos, el nombre del platillo, el precio unitario, el subtotal, total, la fecha y hora en que se realizó el pedido y la opción de ingresar si se entregó o no la comida.

La Sesión y Salir

Muestra el correo del usuario que está actualmente activo dentro del sistema, al darle clic a este se muestran los datos que lleno de registro y da la posibilidad de resetear su contraseña o cambiarla por otra, a un lado de esta sección se encuentra el botón de salir el cual regresa la navegación a la pantalla principal de inicio de sesión.

6.1.2 Como Rol de Paciente



The screenshot shows a web form titled "Mis datos" (My data) for a patient. The form is organized into several sections with input fields:

- Nombre del paciente:** LINDA NELLY
- Apellido paterno:** SALES
- Apellido materno:** LUGARDO
- Tipo de dieta:** DIETA BLANDA NORMAL
- Edad:** 29
- Número de cama:** 101
- Enfermedades ó padecimientos:** DIABETES
- Familiar encargado:** FLOR LUGARDO LUGARDO
- Cambiar contraseña:** Contraseña anterior: Escriba contraseña anterior

Figura 6-8 Datos del paciente

Se muestran los datos que se pidieron al registro del paciente y la opción de cambiar la contraseña, también tiene una sección de compra y de orden, en estas viene el nombre del paciente, el número de habitación y la comida que pidió Figura 6.8

6.1.3 Como Rol de Personal

Se visualizan los datos de la persona que es trabajador del hospital y tiene acceso al registro de pacientes, así como al de los familiares y la sección de la comida y su orden. Como se puede observar en la Figura 6.9 el catálogo de comidas está abierto para el personal del hospital.

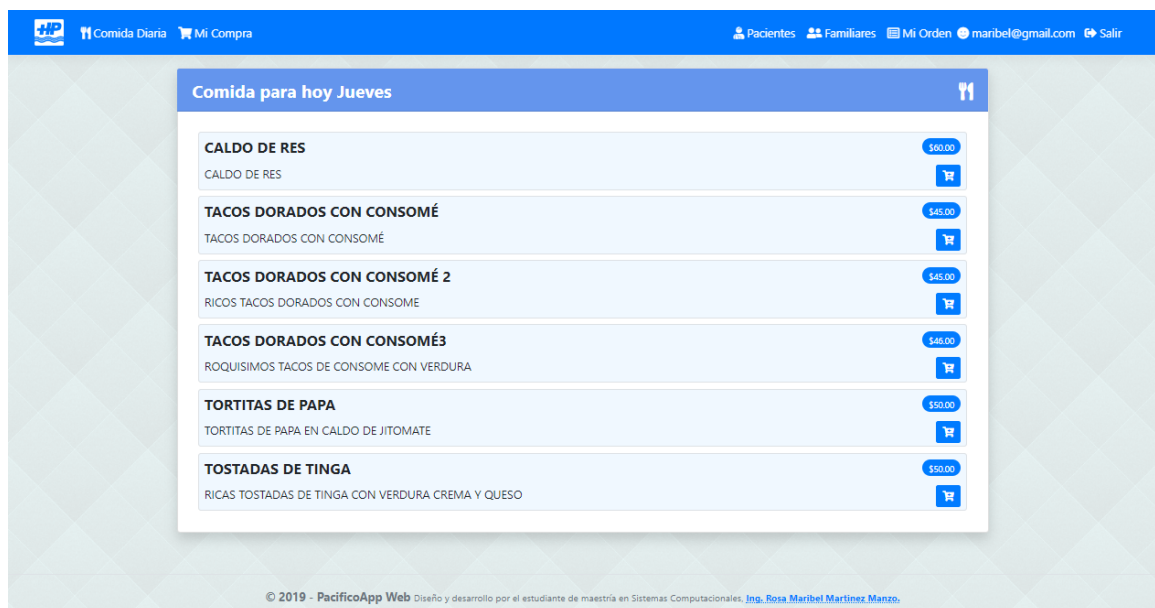
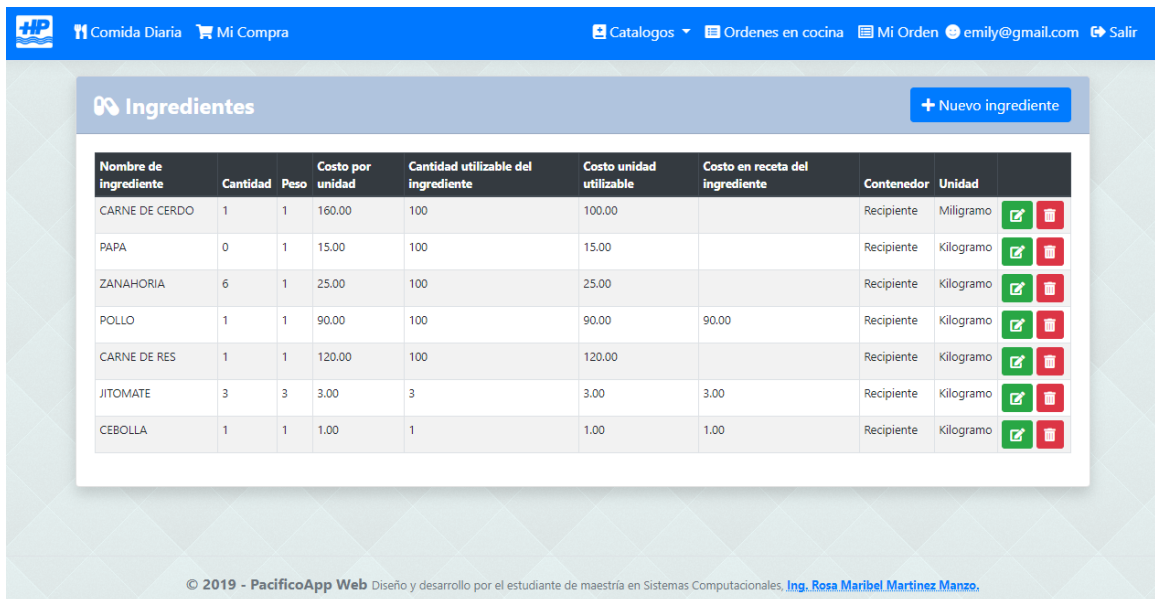
















Figura 6-9 Catalogo de comida por día

6.1.4 Como Rol de Cocina

Se visualizan los datos de la persona encargada del área de cocina, así como a las secciones de comida diaria, mi compra, mi orden y los catálogos junto con todas sus funciones, puede modificar y/o agregar comidas e ingredientes, Figura 6.10.

En seguida en la Figura 6.11 se ve el registro de una nueva comida, se piden datos como nombre típico, descripción, las porciones que se tienen pensadas hacer en el día y una sección de notas, así como los apartados para que se seleccionen las dietas a las que va a pertenecer la comida y los días que estará disponible.



Nombre de ingrediente	Cantidad	Peso	Costo por unidad	Cantidad utilizable del ingrediente	Costo unidad utilizable	Costo en receta del ingrediente	Contenedor	Unidad	
CARNE DE CERDO	1	1	160.00	100	100.00		Recipiente	Miligramo	 
PAPA	0	1	15.00	100	15.00		Recipiente	Kilogramo	 
ZANAHORIA	6	1	25.00	100	25.00		Recipiente	Kilogramo	 
POLLO	1	1	90.00	100	90.00	90.00	Recipiente	Kilogramo	 
CARNE DE RES	1	1	120.00	100	120.00		Recipiente	Kilogramo	 
JITOMATE	3	3	3.00	3	3.00	3.00	Recipiente	Kilogramo	 
CEBOLLA	1	1	1.00	1	1.00	1.00	Recipiente	Kilogramo	 

© 2019 - PacificoApp Web Diseño y desarrollo por el estudiante de maestría en Sistemas Computacionales, Ing. Rosa Maribel Martínez Manzo.

Figura 6-10 Listado de ingredientes

The screenshot shows a web application interface for registering a dish or food item. The interface is titled "Registrar platillo o comida" and includes the following fields and options:

- Nombre típico:** A text input field.
- Descripción:** A larger text area for detailed description.
- Eliga los tipos de dietas:** A set of checkboxes for diet types:
 - DIETA BLANDA NORMAL
 - LIQUIDA
 - 1800 KCAL
 - BLANDA PARA DIABÉTICO
 - HIPOSÓDICA NORMAL
 - PAPILLAS
 - A COMPACENCIA
 - POR SONDA
 - HIPERCALORICA
 - ASTRINGENTE
- Porciones:** A text input field.
- Notas:** A larger text area for additional notes.
- Frecuencia de exhibición:** A set of checkboxes for days of the week:
 - Lunes
 - Martes
 - Miércoles
 - Jueves
 - Viernes
 - Sabado
 - Domingo

At the bottom right, there are two buttons: "Regresar" (Return) and "Guardar" (Save). The footer of the page contains the copyright information: "© 2019 - PacíficoApp Web Diseño y desarrollo por el estudiante de maestría en Sistemas Computacionales, [Ina. Rosa Maribel Martínez Manzo](#)."

Figura 6-11 Registro de comidas

6.2 Despliegue de aplicación en la empresa

Instalación en el hospital

La computadora que tienen como servidor está ubicada en la oficina del administrador junto al rack que alimenta de internet a las diferentes áreas del hospital.

Al instalarse los programas necesarios e instalar el sistema y hacer las primeras pruebas para correr el programa hubo varios problemas derivados de la falta de librerías necesarias para la correcta implementación y ejecución del programa, se le instalaron estas librerías y diversas herramientas del programa para corregir los errores.

En cuanto a la infraestructura física para el internet el hospital no contaba con internet en el área de cocina y en las habitaciones de los pacientes, se encuentran varias redes wifi pero son privadas así que se tuvo que recurrir al cableado de dichas áreas y a la colocación de repetidores de señal para que estas fueran cubiertas por el internet proporcionado por el hospital.

El hospital cuenta con una estructura arquitectónica peculiar, siendo que el piso uno y dos son donde se encuentran las habitaciones de los pacientes y por el desnivel en el que se encuentra el terreno del hospital estos dos pisos están en una parte más baja que el nivel de calle, el piso tres es la entrada principal del hospital en el cual se encuentran la farmacia, la recepción, las oficinas administrativas y de recursos humanos, así como la sala de urgencias y la sala de espera, también se encuentran unas escaleras para el público en general que conducen al cuarto y quinto piso y unos elevadores que recorren los siete pisos de la torre médica, en la cual hay consultorios de diversos médicos y especialistas, en el área de urgencias también se encuentra otro elevador pero ese es para uso exclusivo del personal del hospital el cual recorre los cinco pisos propios del área del hospital, va desde el piso uno y dos que son las habitaciones y rayos x, pasando por tercer piso que es urgencias, cuarto piso que es pediatría y quinto piso que es el área de cocina y restaurante, es en esa área donde se unificó el internet.

Instalación de la Infraestructura

Para la implementación del proyecto y para su correcto funcionamiento era necesario que todas las áreas involucradas tuvieran internet, la cocina y restaurante en el piso cinco, la recepción y el área de urgencias que están en el piso tres, y las habitaciones de los pacientes que están en el piso uno y dos.

Para esto en el piso cinco se colocó un modem router TP-Link modelo TD-W8960N Figura 6.12, en la caseta del almacén de cocina, el cual está cerca tanto de la cocina como de las mesas del restaurante, se decidió esa localización por que ahí se protege de la grasa, el sol, la lluvia y el polvo que pudieran ocasionar un desgaste más rápido del dispositivo, para que el router pudiera dar servicio de internet inalámbrico se tuvo que hacer la instalación de un cable que va desde el tercer piso que es donde está el rack y el modem que surte de internet a esa área así como la computadora que sirve de servidor para el proyecto, se necesitaron treinta metros de cable del tercer al quinto piso, en las imágenes siguientes se muestra la instalación del modem



Figura 6-12 Router TP-Link modelo TD-W8960N

Esta es la caseta de almacenamiento Figura 6.13 en la que se instaló el router, se encuentra a cinco metros de la cocina ya siete del restaurante



Figura 6-13 Caseta de almacenamiento

Del otro lado de la caseta se encuentra la cocina Figura 6.14.



Figura 6-14 Cocina

Para subir el cable se necesitó perforar la pared de la oficina administrativa que es donde se encuentra el rack y subirlo por la parte lateral derecha del hospital hacia el quinto piso que es donde está la cocina y restaurante. Figura 6.15

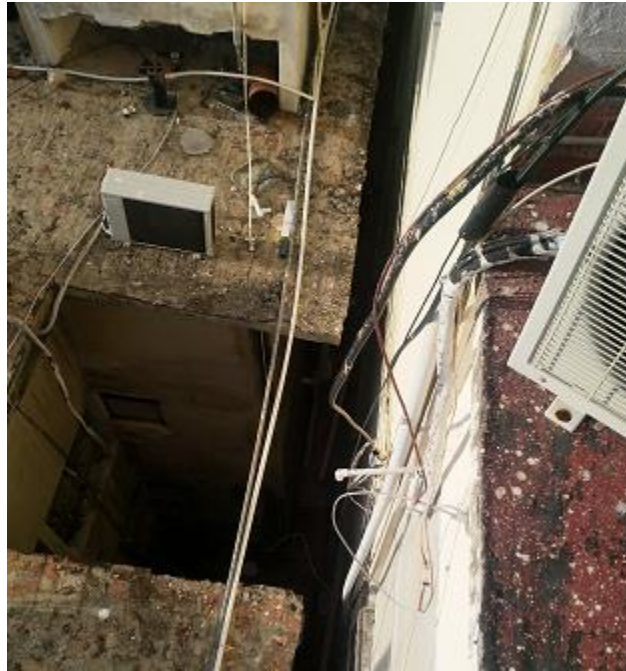


Figura 6-15 Costado del edificio

Para ingresarlo a la caseta de almacenamiento se aprovechó un tubo que antes servía para llevar electricidad a un aire acondicionado que actualmente está en desuso, se guio el cable hacia el techo y por encima de las ventanas hacia el punto donde se conectaría con el router, se afianzo a la pared con armellas. Figura 6.16



Figura 6-16 Instalación del cable

Router TP-Link TD-W8960N con velocidad inalámbrica de hasta 300 Mbps con un rango de señal inalámbrica quince metros aproximadamente a la redonda Figura 6.17



Figura 6-17 Instalación del Router

Dentro de las configuraciones del router Figura 6.18 tenemos lo siguiente; el nombre de la red es HospitalApp, el tipo de autenticación es WPA2 y la contraseña también se le cambio por seguridad, esta contraseña solo la tiene el administrador, la contadora y la jefa de cocina, por órdenes del jefe del hospital a esta red no pueden tener acceso las trabajadoras de la cocina, cocineras y meseras, para evitar que se distraigan de sus labores, de la misma manera se configuro para que tuviera una sola dirección IP estática y pudiera proporcionar un rango de 100 direcciones dinámicas durante 24 horas para la conexión de diversos dispositivos.

300Mbps Wireless N ADSL2+ Modem Router
Model No. TD-W8960N

Quick Setup - Summary

WAN Configurations

WAN Type:	Ethernet WAN
Layer2 Information:	LAN4/WAN
WAN Link Type:	IPoE
WAN IP Setting:	Dynamic IP
DNS Setting:	Obtain Automatically

Note1: Some WAN Connection(s) or Layer2 interface(s) may be replaced by new one!

Note2: The Virtual Server Rules of some WAN Connection(s) may be deleted!

Wi-Fi Configurations

Wireless Network Name:	HospitalApp
Network Authentication:	WPA2-Personal
Wireless NetWork Key:	[REDACTED]

Figura 6-18 Configuraciones del Router

300Mbps Wireless N ADSL2+ Modem Router
Model No. TD-W8960N

Local Area Network (LAN) Setup

Configure the DSL Modem Router IP Address and Subnet Mask for LAN interface. GroupName: Default ▾

IP Address: 192.172.1.1

Subnet Mask: 255.255.255.0

Enable IGMP Snooping

- Standard Mode
- Blocking Mode

Enable LAN side firewall

NOTE: If "LAN side firewall" is enabled, all PCs in the LAN will not able to manage the Router. Please make sure y

Disable DHCP Server

Enable DHCP Server

Start IP Address: 192.172.1.100

End IP Address: 192.172.1.200

Leased Time (hour): 24 (1-48)

Figura 6-19 Configuraciones del Router

CONCLUSIONES

El desarrollo y la implementación del presente proyecto me ha servido para ampliar y desarrollar mis habilidades para el entendimiento de problemas sociales y a su vez buscarles una solución tecnológica con la cual se beneficien y den los resultados óptimos que la empresa necesita para abrir su camino hacia el mundo tecnológico.

El hacer un trabajo conforme los requerimientos específicos y especiales de una empresa no es trabajo fácil pues requiere que en este caso yo como desarrolladora entienda la visión del cliente y lo que quiere para los usuarios finales de la aplicación, son necesarias muchas pruebas y errores para llegar a lo que el cliente tiene en su visión del proyecto ya terminado, en este caso se trabajó conjuntamente con el personal de cocina así como con el personal de las oficinas administrativas y el doctor en jefe del hospital para cumplir con las especificaciones del proyecto y se llegó a un resultado satisfactorio para ambas partes.

A lo largo del tiempo de las estancias y del desarrollo del mismo se han hecho gestiones en dos etapas que son llevar el internet a cocina en el quinto piso y llevar internet a los primer y segundo pisos que son donde se encuentran las habitaciones de los pacientes para que la infraestructura de red quede completa, pero desafortunadamente solo se completó en una primera etapa.

En la presentación del proyecto ya terminado al personal del hospital que estará más involucrado en la utilización del mismo se demostró su funcionamiento y se resolvieron dudas sobre su manipulación, quedo instalado de manera local en la computadora que ellos tienen como servidor en la espera de que se complete la segunda etapa del cableado en los pisos de las

habitaciones para que su implementación se complete, por lo mientras en lo demostrado, el personal quedo complacido del funcionamiento del programa.

Trabajo a futuro

Como trabajo a futuro queda el completar la infraestructura de red dentro del hospital para la implementación completa de la aplicación, así como también la integración de este proyecto junto con un sistema que maneja la farmacia y que el hospital cuente con un sistema completo para el monitoreo completo de los pacientes, desde que se registran, lo que consumen de servicios hospitalarios, las medicinas e insumos de farmacia y su consumo en el restaurante, también para que el hospital ya cuente con una base de datos generalizada con toda esta información sobre sus pacientes en caso de que tengan que regresar por alguna razón integrando así un expediente médico digital con todo el historial que este ha generado a lo largo de su o sus estadías en el hospital y que esto ayude a los médicos a dar un mejor y más eficaz diagnóstico sobre el paciente y en caso de que se integre personal nuevo al hospital ya tengan como antecedente toda esa información para que le puedan brindar una atención de calidad al paciente.

ANEXO

Archivo set de datos *dd.csv* (delimitado por comas), en los valores, las letras tienen el siguiente significado, H: Hidratos, P: Proteínas, G: Grasas, A: Agua, F: Fibra, K: Kcal, los valores numéricos corresponden al contenido del elemento por cada 100 gramos de los artículos.

valores	articulo
H 9	fresa
H 9	limon
H 12	melon
H 13	pera
H 1.5	aguacate
H 4	nuez
H 3.5	almendra
H 5.5	avellana
H 5.8	cebolla
H 3.7	tomate
H 8.7	zanahoria
H 4.8	leche
P 1	fresa
P1	limon
P 0.5	melon
P 0.5	pera
P 1.5	aguacate
P 14	nuez
P 20	almendra
P 14	avellana
P 1.5	cebolla
P 1.1	tomate
P 1	zanahoria
P 20	pollo
P 20	cerdo
P 18.5	huevo crudo
P 3	leche
G 0.5	fresa
G 0.5	limon
G 14	aguacate
G 59	nuez

G 53.5	almendra
G 54	avellana
G 0.1	tomate
G 0.2	zanahoria
G 9	pollo
G 7	cerdo
G 16	huevo crudo
G 3.6	leche
A 83	fresa
A 88	limon
A 85	melon
A 85	pera
A 80	aguacate
A 18	nuez
A 9	almendra
A 16	avellana
A 90	cebolla
A 93	tomate
A 86	zanahoria
A 71	pollo
A 71	cerdo
A 112	huevo crudo
A 89	leche
F 1.5	fresa
F 1	limon
F 1	melon
F 1	pera
F 2.5	aguacate
F 5	nuez
F 14	almendra
F 10	avellana
F 2	cebolla
F 1.5	tomate
F 3.4	zanahoria
K 62	fresa
K 62	limon
K 52	melon
K 52	pera
K 138	aguacate
K 603	nuez
575	almendra

BIBLIOGRAFIA

- Abdulsalam Yassine, (. i. (2017). Patrones de actividades humanas mineras de Big Data Smart para aplicaciones de atención médica. *IEEE Access (Volume: 5) ISSN: 2169-3536, 13131-13141*
<https://ieeexplore.ieee.org/abstract/document/7959184>.
- Advantage, S. y. (2017). *Metodología Scrum para desarrollo de software*. Obtenido de Aplicaciones complejas: <https://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum.html>
- Alicante, U. d. (2019). *Servicio de informatica ASP.NET MVC 3 Framework*. Obtenido de Modelo Vista Controlador (MVC): <https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html>
- Amat Rodrigo, J. (Junio de 2018). *Reglas de asociación y algoritmo Apriori con R*. Obtenido de https://www.cienciadedatos.net/documentos/43_reglas_de_asociacion
- Americas, C. L. (2017). *Alimentación del paciente hospitalizado*. Obtenido de <http://conexionlasamericas.com/marzo%202014/paginas/Alimentaci%C3%B3n-del-paciente-hospitalizado.html>
- Andoid, S. (2018). *Android*. Obtenido de Set up for Android Development: <https://source.android.com/setup/>
- Azure, M. (2017). *Learn Azure at your own pace*. Obtenido de Terminology: <https://docs.microsoft.com/en-us/azure/api-management/api-management-terminology>
- Balaguera, Y. D. (2013). Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. Estado actual. *Journal Technology, Volumen 12, Número 2, 111-124*.
- Bit2me, a. b. (2019). *Que es un Hash*. Obtenido de academy by Bit2me: <https://academy.bit2me.com/que-es-hash/>
- Cheng Chih-Wen (member, i. N. (2013). icuARM: Un sistema de apoyo a la toma de decisiones clínicas de la UCI que utiliza la regla de asociación en minería de datos. *10.1109/JTEHM.2013.2290113*.
- Deitel, D. (2004). *Como Programar C, C++ y Java*. Estado de México: Pearson Educación de México, S.A. de C.V.
- Docs.microsoft. (2018). *Create a theme for your portal*. Obtenido de Implement portal templates y using Bootstrap: <https://docs.microsoft.com/en-us/dynamics365/portals/create-theme>
- Docs.microsoft. (2018). *Welcome to the Visual Studio IDE*. Obtenido de Microsoft: <https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019>
- Docs.microsoft. (2019). *Introducción al lenguaje C# y .NET Framework*. Obtenido de Microsoft: <https://docs.microsoft.com/es-es/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>

- Docs.microsoft. (2019). *LanguagenIntegrated Query*. Obtenido de Microsoft:
<https://docs.microsoft.com/en-us/dotnet/csharp/linq/>
- Educativo, F. b. (2017). *Tipos de dietas hospitalarias*. Obtenido de Fude by educativo:
<http://www.educativo.net/articulos/tipos-de-dietas-hospitalarias-1074.html>
- Educativo, F. b. (2018). *Tipos de dietas hospitalarias*. Obtenido de
<http://www.educativo.net/articulos/tipos-de-dietas-hospitalarias-1074.html>
- Foundation, T. j. (2019). *jQuery*. Obtenido de What is Query: <https://jquery.com>
- Gasca Mantilla, M. C. (27 de agosto de 2013). *Metodología para el desarrollo de aplicaciones móviles*. Obtenido de Sistema de Información Científica Red de Revistas Científicas de América Latina y el Caribe.: <http://www.redalyc.org/articulo.oa?id=257030546003>
- González-Ruiz, S. G.-G.-B.-M. (2015). Algoritmos de clasificación y redes neuronales en la observación automatizada de registros. . *Cuadernos de Psicología del Deporte*, 15(1), , 31-40
<https://dx.doi.org/10.4321/S1578-84232015000100003>.
- IMSS. (2015). Guía de Práctica Clínica Servicios de Alimentación. *Seguridad Alimentaria para el paciente hospitalizado*. Ciudad de México, México.
- Ing. Corso, C. L. (2012). *Uso de herramienta libre para la generación de reglas de asociación, facilitando la gestión eficiente de incidentes e inventarios*. Obtenido de Departamento de Ingeniería en Sistemas de Información. Laboratorio de Sistemas de Información ISSN: 1850-2857- Página 179 - 190: http://41jaiio.sadio.org.ar/sites/default/files/16_JSL_2012.pdf
- JACK, S. I. (2014). *Sistemas Informatica y Electrónica*. Obtenido de Patrón de Diseño MVC (Modelo Vista Controlador) y DAO (Data Access Object): <https://jossjack.wordpress.com/2014/06/22/patron-de-diseno-mvc-modelo-vista-controlador-y-dao-data-access-object/>
- Mantilla Maira, C. A. (04 de 05 de 2014). *Metodología para el desarrollo de aplicaciones móviles*. Obtenido de Revista Tecnura:
<https://revistas.udistrital.edu.co/index.php/Tecnura/article/view/6972>
- Marcotte, E. (2010). *Diseño WEB Responsivo*. Obtenido de [diseowebresponsivo.org](http://xn--diseowebresponsivo-q0b.org/): <http://xn--diseowebresponsivo-q0b.org/>
- Marcotte, E. (2011). *Responsive WEB Design*. New York, New York: A Book Apart.
- Medina-Hernández A, H.-H. R.-M.-S. (2015). Perfil clínico-epidemiológico de pacientes con sospecha de alergia alimentaria en México. Estudio Mexipreval. *Revista Alergia México*, 28-40.
- MedlinePlus. (2017). *Alergia a los alimentos*. Obtenido de MedlinePlus:
<https://medlineplus.gov/spanish/foodallergy.html>
- Microsoft. (2019). *Lado Cliente*. Obtenido de Visual Studio :
<https://visualstudio.microsoft.com/vs/features/web/frameworks/?rr=https%3A%2F%2Fwww.google.com%2F>

- Microsoft. (2019). *Visual Studio*. Obtenido de Lenguajes para la WEB :
<https://visualstudio.microsoft.com/es/vs/features/web/languages/>
- Ofner, W. (2018). *Repository and Unit of Work Pattern*. Obtenido de Programming with Wolfgang :
<https://www.programmingwithwolfgang.com/repository-and-unit-of-work-pattern/>
- Oracle. (2019). *Java*. Obtenido de ¿Que es la tecnología Java y para que la necesito?:
https://www.java.com/es/download/faq/whatis_java.xml
- Overflow, S. (2019). *Stack Overflow*. Obtenido de What is an ORM, how does it work, and how should I use one?: <https://stackoverflow.com/questions/1279613/what-is-an-orm-how-does-it-work-and-how-should-i-use-one>
- Overflow, S. (2019). *What is a difference between a Model and an Entity*. Obtenido de Stack Overflow:
<https://stackoverflow.com/questions/8743995/what-is-difference-between-a-model-and-an-entity>
- Pimienta García Rodrigo, G. A. (2014). Métodos de programación segura en java para aplicaciones móviles en Android. *revista Ciencia Ergo Sum ISSN: 1405-0269, vol. 21, núm. 3, 243-248*.
- PostgreSQL. (2019). *New to PostgreSQL*. Obtenido de PostgreSQL: <https://www.postgresql.org/>
- Rocío Andrea Rodríguez, P. M. (2014). Aprovechamiento del hardware de los dispositivos móviles para la construcción de nuevas aplicaciones. *WICC 2014 XVI Workshop de Investigadores en Ciencias de la Computación, 676 – 680*.
- Rodríguez, R. A. (Mayo de 2014). *Aprovechamiento del hardware de los dispositivos móviles para la construcción de nuevas aplicaciones*. Obtenido de Repositorio Institucional de la UNLP - Red de Universidades con Carreras en Informática (RedUNCI):
<http://sedici.unlp.edu.ar/handle/10915/42655>
- S.L. González-Ruiz, I. G.-G.-B.-M. (2015). Algoritmos de clasificación y redes neuronales en la observación automatizada de registros. *CPD vol.15 no.1 Murcia*.
- Sevilla, F. S. (Diciembre de 2020). *Dpto. de Ciencias de la Computación e Inteligencia Artificial*. Obtenido de Universidad de Sevilla: <http://www.cs.us.es/~fsancho/?e=77>
- Softeng. (2017). *Metodología Scrum para desarrollo de software - aplicaciones complejas*. Obtenido de <https://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum.html>
- Softqanetwork. (2017). *Softqanetwork*. Obtenido de Metodología MoProSoft:
<http://www.softqanetwork.com/moprosoft-modelo-de-procesos-de-software>
- Software, P. d. (2017). *Procesos de Software*. Obtenido de Metodología RUP:
<https://procesosdesoftware.wikispaces.com/METODOLOGIA+RUP>
- Soloriio, M. (2013). *Metodología en cascada*. Obtenido de Metodología en cascada:
<http://metodologiaencascada.blogspot.mx/>
- Sparxsystems. (2019). *Enterprise Architect*. Obtenido de <http://www.sparxsystems.com.ar/products/ea/>

- THERAPEUTICS, A. (2016). Principales diferencias entre alergias alimentarias e intolerancias alimentarias. *Revista A la Carta*.
- THERAPEUTICS, A. (2016). Revista A la Carta Principales diferencias entre alergias alimentarias e intolerancias alimentarias. *ALLERGY THERAPEUTICS*.
- TutorialsTeacher. (2018). *ASP.NET MVC Folder Structure*. Obtenido de TutorialsTeacher:
<https://www.tutorialsteacher.com/mvc/mvc-folder-structure>
- Watanabe Takashi, Y. K. (2016). Identificación de reglas recurrentes de asociación en la predicción de defectos de software. *978-1-5090-0806-3/16, IEEE ICIS 2016*.
- Xatakandroid. (2017). *Xatakandroid*. Obtenido de Que es Android:
<https://www.xatakandroid.com/sistema-operativo/que-es-android>