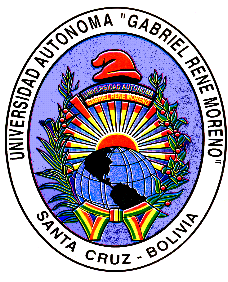
**UNIVERSIDAD AUTÓNOMA “GABRIEL RENE MORENO”**

**FACULTAD DE INTEGRAL DEL NORTE**

**CARRERA DE INGENIERIA EN SISTEMAS**



**MATERIA: INGENIERIA DE SOFTWARE II**

**TEMA: “SISTEMA DE INFORMACION DE ODONTOGRAMA PARA LA CLINICA DENTAL “ODONTO VIERA”**

**AUTOR:** Widen Valles Álvarez

Luis Fernando Quinteros Santos

Luis Rodrigo Abasto Torrez

**Docente:** Ing. Magno Edwin Calizaya A.

**Santa Cruz, Bolivia 10 de abril del 2019**

**Cesión de derechos**

Declaro que el proyecto, titulado “**DESARROLLO DE UN SISTEMA INFORMACIONAL DE ODONTOGRAMA PARA LA CLINICA DENTAL ODONTO VIERA**” es de propia autoría; que no ha sido presentado previamente al tribunal de la Facultad Integral del Norte, presentada para aprobar la materia de Ingeniería de Software II; y que se ha consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaro que cedo mi derecho de propiedad Intelectual correspondiente a este trabajo, a la UAGRM Facultad Integral del Norte, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

**Dedicatoria**

Caminé con paso firme y constante, con profunda fe, y absoluta convicción de que alcanzaría mi meta más preciada; hoy que al fin puedo vivirla y trazarme muchas más quiero expresar mi agradecimiento primeramente a Dios por darme vida y salud para vivir este momento. ¡Gracias Señor por llenar mi vida de tantas bendiciones! Gran motivo me inspira a dedicar y agradecer a todos quienes me ayudaron: A mi mamá con quien cuento para todo logro y siempre deposita en mi toda la confianza y el amor para hacer más placentero y hermoso el camino. A mis abuelos quienes fueron y serán “La Raíz y el Tronco", el mejor papá y la mejor mamá, gracias por sus cuidados.

**Agradecimientos**

Todo mi agradecimiento a nuestras familiares en especial a nuestros padres y hermanos por su apoyo, motivación y cariño.

Agradezco a todos mis amigos de la Universidad por haber compartido esta aventura tan bonita, por su solidaridad y apoyo por compartir su conocimiento.

Agradezco a nuestros docentes Ing. Edwin Calizalla, Ing. Víctor Córdova y nuestro Ing. Edwin Calle por su gran enseñanza y confianza que nos dieron.

Agradezco a la Facultad Integral del Norte, por ser una gran institución, de la cual me llevo un gran aprendizaje y una muy bonita experiencia. Agradezco a todas aquellas personas especiales en mi vida que de una u otra forma me han brindado su apoyo incondicional desde el inicio hasta el final de la realización de mi carrera.

Tabla de contenido

[**CAPITULO I: PERFIL DEL PROYECTO** 4](#_Toc12332386)

[**1.1** **Introducción** 5](#_Toc12332387)

[**1.2** **Resumen** 6](#_Toc12332388)

[**1.3** **Abstract** 7](#_Toc12332389)

[**1.4 Antecedentes del problema** 7](#_Toc12332390)

[**Antecedentes institucionales** 7](#_Toc12332391)

[**Misión y Visión** 8](#_Toc12332392)

[**1.5. Planteamiento del problema** 8](#_Toc12332393)

[**Campo de acción** 8](#_Toc12332394)

[**Objeto de estudio** 9](#_Toc12332395)

[**Formulación del problema científico.** 9](#_Toc12332396)

[**Matriz foda** 10](#_Toc12332397)

[**Árbol de problema** 10](#_Toc12332398)

[**1.6. OBJETIVOS** 11](#_Toc12332399)

[**Objetivo General** 11](#_Toc12332400)

[**Objetivos específicos** 11](#_Toc12332401)

[**1.7 Hipótesis o idea a defender** 12](#_Toc12332402)

[**1.8 Definición de variables** 12](#_Toc12332403)

[**Odontograma** 12](#_Toc12332404)

[**Sistema de odontograma** 12](#_Toc12332405)

[**1.9 Justificación del proyecto o investigación** 12](#_Toc12332406)

[**Justificación teórica** 12](#_Toc12332407)

[**Justificación social** 13](#_Toc12332408)

[**Justificación económica** 13](#_Toc12332409)

[**Justificación técnica y tecnológica** 13](#_Toc12332410)

[**1.10 Delimitación o alcance** 13](#_Toc12332411)

[**Delimitación temática o sustantiva (lo pueden dividir en módulos)** 14](#_Toc12332412)

[**Delimitación temporal** 14](#_Toc12332413)

[**Delimitación geográfica** 14](#_Toc12332414)

[**1.11 Metodología de la investigación** 14](#_Toc12332415)

[**Tipo de investigación** 14](#_Toc12332416)

[**Métodos de investigación** 14](#_Toc12332417)

[**Metodología de desarrollo** 15](#_Toc12332418)

[**Lenguaje de modelado** 15](#_Toc12332419)

[**Paradigmas o modelos de programación** 15](#_Toc12332420)

[**Herramientas de desarrollo** 16](#_Toc12332421)

[**Tecnica ie investigacion** 16](#_Toc12332422)

[**Instrumentos de investigación** 16](#_Toc12332423)

[**1.12 Cronograma** 17](#_Toc12332424)

[**CAPITULO II: MARCO TEORICO** 18](#_Toc12332425)

[**2.1 Sql server 2017** 19](#_Toc12332426)

[***2.2 Visual studio*** 25](#_Toc12332427)

[**2.3 Lenguaje de programación C#** 31](#_Toc12332428)

[**2.3.1 Historia de C#** 32](#_Toc12332429)

[**2.4 Programación En 3 Capas** 34](#_Toc12332430)

[**a)** **Capa de presentación** 35](#_Toc12332431)

[**b)** **Capa negocio** 35](#_Toc12332432)

[**c)** **Capa de datos** 35](#_Toc12332433)

[**2.5 Programacion extrema xp** 36](#_Toc12332434)

[**2.6 Entity Framework** 37](#_Toc12332435)

[**2.7 Odontograma** 41](#_Toc12332436)

[**¿Cuántos Dientes Tenemos?** 42](#_Toc12332437)

[**CAPITULO III: MARCO APLICATIVO** 44](#_Toc12332438)

[**3.1 Requerimientos** 45](#_Toc12332439)

[**3.1.1 Requerimientos funcionales (product backlog)** 45](#_Toc12332440)

[**3.1.2 Requerimientos no funcionales** 48](#_Toc12332441)

[**3.2 Desarrollo del Software** 53](#_Toc12332442)

[**3.2.1 Especificación de Requerimientos de Software** 53](#_Toc12332443)

[**3.2.1 Historias de Usuario** 53](#_Toc12332444)

[**3.3 Diseño** 61](#_Toc12332445)

[**3.3.1 Arquitectura de la solución para el Modelo de Capas** 61](#_Toc12332446)

[**3.3.2 Descripción de la arquitectura** 61](#_Toc12332447)

[**3.3.3 Arquitectura de la solución basada en la tecnologia .Net** 62](#_Toc12332448)

[**3.3.4 Descripción de Capas** 63](#_Toc12332449)

[**3.3.5 ADO.NET** 64](#_Toc12332450)

[**3.3.4 Diseño de la base de datos** 65](#_Toc12332451)

[**3.4 Implementación** 70](#_Toc12332452)

[**Aplicación de escritorio** 70](#_Toc12332453)

[**3.5 Pruebas** 111](#_Toc12332454)

[**Pruebas del software** 111](#_Toc12332455)

[**3.6 Recomendaciones** 117](#_Toc12332456)

[**3.7 Conclusiones** 118](#_Toc12332457)

[**3.8 Referencia bibliográfica** 118](#_Toc12332458)

[**3.9 Bibligrafia** 119](#_Toc12332459)

[**3.10 Anexos** 119](#_Toc12332460)

# **CAPITULO I: PERFIL DEL PROYECTO**

* 1. **Introducción**

El Software de gestión hoy en día es muy fundamental para las empresas del sector salud y entidades porque existen un mejor control de la información de las actividades que realiza. En el presente proyecto se buscará mejorar el funcionamiento de las clínicas dentales mediante el manejo de los sistemas informacional de odontograma que no se contaba con el software.

La información se constituye como uno de los activos más valiosos para toda institución, que de ella depende la toma de decisiones que puede afectar o ayudar en un papel fundamental para el alcance de sus objetivos y metas.

El software a medida es de gran utilidad para el desarrollo normal de las actividades de la empresa y lograr mantener un control continuo sobre las actividades comerciales, financieras y administrativas permitiendo de esta manera crear un medio estructurado informático que contribuye al sostenimiento de la misma.

El sistema mencionado llevara a dar soluciones que faciliten la operación de las tareas mediante un software que pueda satisfacer las necesidades del usuario en menor tiempo, brindando así una mejor calidad en los servicios.

A si mismo tener un control estricto. De esta manera hacerlo más eficiente su uso, adaptándolo a las necesidades en el ámbito de la gestión de la información, con el fin de obtener una satisfacción total por parte del cliente.

* 1. **Resumen**

El presente trabajo de investigación se basó en el Desarrollo de un Sistema información informático para el Control del manejo del historial clínico y uso de odontograma para los tratamientos dentales de los pacientes. Para lo cual fue necesario estudiar el funcionamiento actual de dicha área y determinar las problemáticas que se presentaban en cuanto a las operaciones que se realizan en la clínica dental; para luego definir los requerimientos de información del sistema en base a dicho problema; procediéndose después a diseñar una arquitectura sólida que cumpliera con todos los requerimientos establecidos, hasta finalmente obtener el prototipo inicial de la aplicación. Dicho trabajo siguió un tipo de investigación Proyectiva, con un nivel comprensivo, empleándose como técnica de recolección de los datos la revisión documental, la entrevista no estructurada y la observación directa. El desarrollo del sistema se fundamentó en la metodología XP conjuntamente con el lenguaje unificado UML, usando herramientas de software tales como C#, como manejador de base de datos SQL SERVER. De esta manera se pudo concluir que con el desarrollo del sistema se generan beneficios, como reducción de tiempo, riesgo de pérdida en cuanto a información y control en las operaciones que se llevan al momento de realizar tratamientos dentales.

* 1. **Abstract**

This research work was based on the Development of a computer information System for the Control of the management of the clinical history and use of Odontograma for the dental treatments of the patients. For which it was necessary To study the current functioning of this area and To determine the problems that were presented in relation to the operations that are carried out in the dental clinic; To then define the system information requirements based on that problem; Then proceeding to design a solid architecture that fulfilled all the requirements established, until finally obtaining the initial prototype of the application. This work followed a type of Projective research, with a comprehensive level, using as a data collection technique the documentary review, the unstructured interview and the direct observation. The development of the system was based on the XP methodology in conjunction with the Unified UML language.

## **1.4 Antecedentes del problema**

### **Antecedentes institucionales**

Debido al crecimiento poblacional de la localidad de montero se dio la idea de crear un consultorio dental para satisfacer la demanda de la sociedad.

Desde ese punto nació la idea de crear el consultorio dental “**ODONTO VIERA”** que está brindando servicio desde el 2009 que se encuentra en la calle arenales y mariscal santa cruz de la localidad de Montero.

Cuando el consultorio empezó a funcionar se realizaba el registro de los tratamientos en formularios tras que pasaba los años los clientes aumentaban y el manejo de los formularios se hacía más difícil de administrar

Cuando el instituto empezó a funcionar solo contaba con un registro de alumnos, docentes y administrativos en el cual se registraba en libros.

El consultorio dental “ODONTO VIERA” lleva funcionando 10 años en la población, desde 2009, en todo el tiempo que lleva funcionando no ha tenido ningún antecedente de incumplimiento.

### **Misión y Visión**

Ser una clínica odontológica reconocida en Bolivia por la prestación de servicios odontológicos de calidad y por su responsabilidad social en la salud buco dental de la población.

En “Odontóloga Uriarte Sirona” estamos centrados en la capacitación de profesionales de la odontología para que puedan practicar una odontología de mayor calidad, más rápida y más segura. Trabajamos con ellos para vislumbrar lo que es posible y, a continuación, proporcionar soluciones innovadoras para lograrlo. La formación y el apoyo contribuyen a avanzar cada procedimiento, garantizando que las ventajas de nuestras innovaciones se hagan realidad. Todo lo que hacemos tiene como objetivo ayudar a los profesionales de la odontología a ofrecer la mejor práctica odontológica posible, en beneficio de sus pacientes y consultas, tanto hoy como en el futuro.

## **1.5. Planteamiento del problema**

### **Campo de acción**

El campo de acción de la clínica Dental “ODONTO VIERA” se ubica en la localidad de Montero zona céntrica de Santa Cruz-Bolivia.

### **Objeto de estudio**

El sistema de información de Odontograma.

### **Formulación del problema científico.**

En la actualidad la clínica está recibiendo muchos clientes donde no cuentan con sistema de información de odontograma para su mejor administración de sus tratamientos dentales ya que actualmente se llenan manualmente en formulario con un panorama de los dientes marcando las zonas donde se debe realizar tratamiento.

Lo ideal para este problema es tener un sistema de información de odontrograma con un panorama digital de los dientes para mejor eficiencia, donde pueda seleccionar las zonas afectadas para su respectivo tratamiento y todo su historial clínico se guarde en una base de datos.

### **Matriz foda**

|  |  |
| --- | --- |
| FORTALEZAS | OPORTUNIDADES |
| Profesional de alta experiencia  Persistencia para alcanzar los objetivos.  Conocimiento de idiomas de inglés, castellano y quechua.  Buena organización con los tratamientos de los pacientes con mucha paciencia y detallismo.  Interés por aprender nuevas especialidades. | Existencia de pacientes debido al crecimiento poblacional.  Bajo costos de los implementos necesarios para realizar los tratamientos a los pacientes. |
| DEBILIDADES | AMENAZAS |
| Falta de financiamiento para los equipos de trabajo  Costo de tratamiento elevado debido a los altos precios de compra  Dificulta para realizar trabajo bajo presión. | Asentamiento de nuevos competidores en la zona.  Dependencia de la inversión de gobierno con puesto de salud gratuito.  Compras elevadas debido al aumento de precio de los implementos necesario para la clinica. |

### **Árbol de problema**

Perdida de información de Valiosa del historial de los tratamientos de los pacientes.

Pérdida de tiempo valioso al rellenar de nuevo el formulario del dental

Escases visita de pacientes

Los pacientes buscan una nueva clínica

Falta de un sistema de odontograma para el manejo administrativo de los tratamientos.

Desorden de los formularios de los pacientes.

## **1.6. OBJETIVOS**

### **Objetivo General**

Implementar un sistema de información a medida de odontograma para que tenga el mejor manejo administrativo de los tratamientos dentales a través de un panorama digital de los dientes de manera dinámica y eficiente, utilizando la metodología ágil XP, en la clínica dental clínica dental “ODONTO VIERA”.

### **Objetivos específicos**

* Diagnosticar las deficiencias en los procesos de gestión de la clínica odontología, con el fin de identificar los requerimientos que deberá satisfacer el nuevo Sistema de Información.
* Analizar los requisitos adquiridos del proyecto por etapas para respectivo diseño.
* Diseñar la arquitectura del sistema a desarrollar mediante representaciones de la interfaz y el detalle procedimental (algoritmo) que suele representarse con diagramas y textos, a partir de los resultados del diagnóstico de deficiencias y requerimientos, para luego pasar implementación de códigos.
* Implementar el sistema de información en base a las especificaciones del Diseño utilizando un lenguaje de alto nivel Visual Studio 2017 en C# y SQL Server 2017 como motor de la base de datos utilizando la metodología de tres capas, para su creación del sistema mediante códigos informáticos.
* Elaborar un plan de pruebas al sistema para verificar posibles errores y mejorarlos antes de la entrega del sistema.

## **1.7 Hipótesis o idea a defender**

Un sistema de información de odontograma para la clínica “ODONTO VIERA” que registre los tratamientos dentales.

## **1.8 Definición de variables**

### **Odontograma**

Un odontograma es un esquema utilizado por los odontólogos que permite registrar información sobre la boca de una persona. En dicho gráfico, el profesional detalla qué cantidad de piezas dentales permanentes tiene el paciente, cuáles han sido restauradas y otros datos de importancia.

### **Sistema de odontograma**

Atrás quedó el tiempo en el que los odontogramas se dibujaban a mano. En la actualidad para hacer uno se usan todo tipo de herramientas informáticas que facilitan enormemente el trabajo utilizando un sistema de información de odontograma para marcar digitalmente las zonas donde se van a realizar tratamientos y guardar un historial del paciente.

## **1.9 Justificación del proyecto o investigación**

### **Justificación teórica**

Para este proyecto está basado en toda la experiencia aprendida en los semestres que pasamos, tomando como referencia más importante en las materias de programación, base de datos, sistema de información, libros, tutoriales e investigaciones sobre el tema relacionado, en cual llegamos a realizar este informe como manual y poder entenderlo cómo se realizó el sistema.

### **Justificación social**

El sistema información brindará a los odontólogos del país una mejor administración en una forma clara y sencilla del historial clínico donde se podrá registrar todos los tratamientos dentales que el paciente se ha realizado y los que se tienen que solucionar con una mayor eficacia y eficiencia donde también el paciente podrá pedir un informe de su tratamiento.

### **Justificación económica**

Una vez el proyecto esté terminado, permitirá a los odontólogos de nuestro país en el área de odontograma minimizar los costos y así poder trabajar en un sistema de información.

### **Justificación técnica y tecnológica**

El presente proyecto sirve para realizar un control sobre sus registros de sus problemas dentales realizados y lo que se van a realizar, mediante el uso del software el cual guarda registros al momento de gestionar la información.

Durante el transcurso del desarrollo del sistema de información se aplicará toda la información recolectada y los conocimientos aprendidos en materias pasadas de Base de Datos I, II, Software I además de la investigación propia.

El sistema de información actual está desarrollado en plataformas de uso relativamente libre, la base de datos está desarrollada en SQL SERVER y el entorno de la programación para una mayor seguridad se desarrolló en lenguaje C#.

## **1.10 Delimitación o alcance**

El proyecto que le vamos a presentar abarca únicamente para el servicio más eficiente del uso del odontograma en los tratamientos de la clínica dental “ODONTO VIERA”

### **Delimitación temática o sustantiva (lo pueden dividir en módulos)**

Se implementará un módulo de odontograma y Atencion.

### **Delimitación temporal**

El presente proyecto tendrá una duración de 5 meses que comenzó a inicios del mes de abril en fecha 02/04/2019.

### **Delimitación geográfica**

Se desarrolló con la finalidad de facilitar el servicio y mejor manejo a través del uso de un sistema informacional de odontograma para clínica dental “ODONTO VIERA” ubicado en la calle arenales y mariscal santa cruz de la localidad de Montero –Santa Cruz – Bolivia.

# **1.11 Metodología de la investigación**

## **Tipo de investigación**

Nuestro presente proyecto utiliza la investigación descriptiva con el objetivo de establecer una descripción más completa del funcionamiento de una situación o elemento en concreto, para lograrlo se debe conocer las situaciones, costumbres y actividades que se realizan ya que el objetivo es esclarecer la situación problemática que presenta.

## **Métodos de investigación**

En este presente proyecto utilizamos la metodología de investigación cualitativa para describir las cualidades de un fenómeno o situación de la clínica, como técnica descriptiva para la recolección de información de datos que nos ayudan a explicar el comportamiento llegando a comprender del por qué, cómo o de qué manera subyacente se da un determinado comportamiento.

## **Metodología de desarrollo**

El presente trabajo se basa en la metodología de Programación Extrema (XP), para el desarrollo del producto se utiliza el Framework SCRUM.

## **Lenguaje de modelado**

El UML (Unified Modeling Language) tiene sus orígenes en la necesidad que se había generado en la industria para construir modelos orientados a objetos. Nace en el año 1994 por iniciativa de Grady Booch y Jim Rumbaugh para combinar dos famosos métodos: el de Booch y el OMT (Object Modeling Technique). Más tarde se les unió Ivar Jacobson, creador del método OOSE (Object-Oriented Software Engineering). En respuesta a una petición de OMG (Object Management Group), para definir un lenguaje y una notación estándar del lenguaje de construcción de modelos, en 1997 propusieron el UML como candidato. UML es ante todo un lenguaje, lenguaje que se centra en representación gráfica de un sistema. Es un lenguaje visual estándar empleado para la especificación, construcción y documentación de software orientado a objetos, por medio de diversos elementos y procesos que interactúan de alguna forma con el software.

## **Paradigmas o modelos de programación**

El sistema de información utiliza la programación orientada a objetos porque es una las tendencias más novedosas de la programación que se basa su resolución de problemas, en la creación de los llamados objetos, que no son más que unidades que contienen una serie de características y atributos a los cuales se les asignará una serie de datos para resolver el problema.

## **Herramientas de desarrollo**

El presente proyecto de sistema de información de odontograma utiliza como herramientas de desarrollo Visual Studio 2019 C#, SQL Server 2017, Framework entity y como herramienta de diseño y modelado Enterprise Architect.

## **Tecnica ie investigacion**

Para realizar el proyecto utilizamos la técnica de las entrevistas las cuales se realizó al odontólogo encargado de la clínica dental “ODONTO VIERA”.

## **Instrumentos de investigación**

En el presente proyecto se utilizó como instrumento de investigación cuestionario de entrevistas que se realizó al odontólogo encargado de la clínica dental.

## **1.12 Cronograma**

|  |  |
| --- | --- |
| CRONOGRAMA DE ACTIVIDADES | |
| 29 de marzo | Se asignó el Sistema en el cual se trabajaría de forma grupal |
| 15 de abril | Se fue al lugar en el cual se realizará el proyecto y se recaudó información para realizar el mismo |
| 1 de mayo | Se analizó toda la información recaudada de las consultas que se realizaron a la clínica dental “ODONTO VIERA” |
| 15 de mayo | Se realizó el diseño conceptual de la Base de Datos |
| 20 de mayo | Se implementó la Base de Datos a un Gesto de Base de Datos SQL Server |
| 30 de mayo | Se realizó la metodología de 3 capas(Capa Negocio, Capa Datos y Capa Presentación) |
| 10 de junio | Se terminó con la realización del sistema de información |
| 20 de junio | Se empezaron con las validaciones del sistema |
| 25 de junio | Se realizó pruebas a al software. |
| 30 de junio | Finalmente se completó el sistema y respectiva presentación |

# **CAPITULO II: MARCO TEORICO**

## **2.1 Sql server 2017**

SQL Server 2017 se considera una versión importante en la historia del ciclo de vida de SQL Server por varias razones. Desde mi punto de vista personal, SQL Server 2017 es una versión interesante. Después de escribir mucho sobre él y probar varias características de SQL Server 2017, me gustaría mostrarle algunas de sus características interesantes.

1. está ahora en la plataforma de o elección; SQL Server 2017 en varias distribuciones de Linux, que pueden ejecutarse en contenedores Docker, y SQL VM en Azure, junto con el buen servidor SQL en Windows.
2. expande su alcance para soportar la tecnología de base de datos Graph
3. Incluye ajuste automático de la base de datos.
4. Apoya el aprendizaje automático mediante la adopción de Python
5. Contiene nuevas vistas y funciones de gestión dinámica.
6. tiene nuevas funciones de cadena
7. Y más…

Vamos a descubrir el misterio detrás del desarrollo y la implementación más rápidos de SQL Server. El tamaño del archivo del ejecutable de SQL Server, sqlservr.exe, es de ~ 3 MB en SQL Server 2017, en comparación con ~ 64 MB en SQL Server 2008 R2. El equipo de SQL Server se dio cuenta de la importancia de desacoplar y modularizar el código de área de superficie en componentes mucho más pequeños. La arquitectura completa se divide en múltiples microservicios más pequeños para que las implementaciones y la integración se realicen en cada uno de los componentes respectivos, y que el proceso sea continuo.

A partir de SQL Server 2017, Microsoft ya no usa el antiguo modelo de lanzamiento bajo el cual se lanzaban las actualizaciones acumulativas (CU) cada dos meses y los paquetes de servicio (SP) que contenían todas las correcciones de las CU anteriores se lanzaban una vez al año. Este nuevo modelo permitió acortar el ciclo de lanzamiento y las características se agregaron a la Base de datos SQL y al Servidor SQL más rápido que nunca. Más recientemente, el equipo de ingeniería de SQL Server logró lanzar SQL Server 2017 junto con la nueva compatibilidad multiplataforma dentro de los 15 meses posteriores al lanzamiento de SQL Server 2016. Esto contrasta con los ciclos de envío heredados de tres a cinco años de el pasado.

Uno podría sentir por qué SQL Server2016 y 2017 son lanzamientos tan grandes en tan solo unos meses. Sin embargo, a partir de 2017, Microsoft se está moviendo hacia una cadencia de lanzamiento rápido con una funcionalidad mejorada y nuevas funciones que se implementan anualmente en lugar de tener que esperar dos o tres años entre versiones.

**Bases de datos de grafos**

Esta sección explora la integración de los componentes de la base de datos Graph en SQL Server 2017. El puntero al final de esta sección incluye un conjunto de datos de trabajo de bases de datos relacionales y gráficas, y modelos de datos. Además, explica cómo integrar bases de datos de gráficos con bases de datos relacionales y cómo convertir las tuplas relacionales en un conjunto de datos de gráficos.

Un concepto clave de las bases de datos de gráficos son los bordes y los nodos. Las relaciones permiten a las entidades vincularse directamente y, en muchos casos, recuperarse con una sola operación. En algunos casos, debido al diseño de modelado de datos estricto, el modelo de datos relacionales puede no ser un buen ajuste, ya que incurre en costosas operaciones de unión. En el pasado, la falta de alternativas viables y la falta de apoyo hicieron del modelo gráfico una alternativa difícil, pero ya no es así.

También veremos la integración del soporte de la base de datos de gráficos para ayudar a modelar relaciones de muchos a muchos como nodos y bordes, en lugar de con el modelo relacional tradicional. La huella de la tecnología de base de datos de gráficos en la industria de TI es claramente visible hoy en día, y por buenas razones, como agregar eficiencia a las recomendaciones sociales, análisis de redes de TI, detección de fraudes, recomendaciones de productos, etc.

**Funciones de cadena**

Una de las principales capacidades de las computadoras modernas es el procesamiento del lenguaje humano. Básicamente, las cadenas se transforman en el código que luego es procesado por una máquina. Las funciones de cadena incorporadas siempre han sido eficientes en el manejo de literales de cadena. Es posible encontrar y alterar valores de cadena usando varias opciones. Sin embargo, las funciones de cadena y la manipulación de cadena consumen la mayor parte del tiempo de ejecución de la consulta al descodificar las diversas partes de los literales de caracteres. SQL Server 2017 ofrece varias funciones nuevas de manipulación de cadenas, que han estado en la conversación por muchas de sus características que simplifican la vida de un desarrollador; Ya no es necesario escribir sentencias T-SQL largas con tablas temporales y lógica compleja, solo para manipular y agregar cadenas.

Estas son algunas de las nuevas funciones de manipulación de cadenas presentes en SQL Server 2017:

* CONCAT\_WS
* TRADUCIR
* RECORTAR
* STRING\_AGG

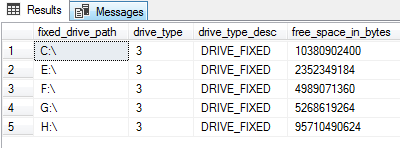
**DMVs y DMFs**

Con la llegada de los objetos del sistema, los metadatos de varias piezas de SQL Server se han expuesto para una mejor comprensión de todo el sistema. Estos objetos del sistema, en su mayor parte, son una colección acumulativa de datos, o la agregación o acumulación de los diversos valores de contador o estructuras de datos diferentes de SQL Server. Lo más importante es que los datos son en tiempo real, y obviamente de naturaleza dinámica. En la mayoría de los casos, los DMV se utilizan y definen la línea de base o los puntos de presión de varias métricas que determinan el rendimiento del sistema. Los DMF proporcionan estadísticas acumulativas de los parámetros solicitados.

Los nuevos DMV introducidos en SQL Server 2017 son los siguientes:

* sys.dm\_tran\_version\_store\_space\_usage
* sys.dm\_db\_stats\_histogram (Transact-SQL)
* sys.dm\_exec\_query\_statistics\_xml
* sys.dm\_os\_host\_info
* sys.dm\_os\_sys\_info
* Una nueva columna modified\_extent\_page\_count introducida en sys.dm\_db\_file\_space\_usage para rastrear los cambios diferenciales en cada archivo de base de datos de la base de datos.
* Identifique el espacio libre en disco utilizando el nuevo DMV sys.dm\_os\_enumerate\_fixed\_drives.

SELECT \* FROM sys.dm\_os\_enumerate\_fixed\_drives;



**Reconstrucción de índice en línea reanudable**

SQL Server 2017 es el primer RDBMS comercial que admite la funcionalidad de pausa y reanudación para las operaciones de mantenimiento de índices. La mayoría de los administradores consideran que la reconstrucción de índices en VLDB es una tarea desalentadora para administrar. Muchas soluciones de bases de datos críticas no permiten realizar operaciones de mantenimiento de bases de datos fuera de línea. En la mayoría de los casos, el diseño de la base de datos juega un papel importante en ese sentido. SQL Server 2017 proporciona tareas de mantenimiento de índices con gran flexibilidad como una solución alternativa para administrar las operaciones de mantenimiento. Hay situaciones en las que los administradores de bases de datos pueden necesitar liberar temporalmente los recursos del sistema. Por ejemplo, ¿qué sucede si una tarea prioritaria necesita recursos y la reconstrucción del índice de menor prioridad está eliminando los recursos? Preferimos que la operación de reconstrucción de índice se ejecute en alguna otra ventana de mantenimiento disponible, pero al mismo tiempo, ¿Y si la reconstrucción estaba a medio camino? Tampoco queremos perder ese estado. En tal caso, la operación se puede pausar y reanudar durante una ventana de mantenimiento. Esto se puede denominar como "operación de mantenimiento de índice por partes".

* Reanudar: reanudar una operación de reconstrucción de índice, después de un error.
* Pausa: pausa la operación de reconstrucción (y la reanuda en un momento posterior).
* Reconstruir: reconstruir índices grandes con un uso mínimo del espacio de registro.

**Ajuste automático de la base de datos**

Una tarea común de administración de la base de datos es supervisar las bases de datos para obtener un rendimiento óptimo de las consultas, crear y mantener los índices requeridos y eliminar índices raramente utilizados, no utilizados o costosos. SQL Server 2017 ahora puede ayudar a los administradores de bases de datos a realizar estas operaciones de rutina mediante la identificación de planes de ejecución de consultas problemáticos y la solución de problemas de rendimiento del plan SQL. El ajuste automático comienza con la supervisión continua de la base de datos y el aprendizaje sobre la carga de trabajo que sirve. El ajuste automático de la base de datos se basa en la Inteligencia Artificial; AI ahora ofrece una gran flexibilidad para administrar y ajustar el rendimiento de los sistemas de bases de datos.

El código original de SQL Server ha sido desarrollado por Sybase; a finales de 1980, Microsoft, Sybase y Ashton-Tate colaboraron para producir la primera versión del producto, SQL Server 4.2 para OS/2. Posteriormente, tanto Sybase como Microsoft ofrecieron productos de SQL Server. Sybase cambió después el nombre de su producto a Adaptive Server Enterprise.

Referencia sitio web (<https://www.sqlshack.com/whats-new-in-sql-server-2017/> )

Referencia sitio web (<https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017>)

## ***2.2 Visual studio***

Microsoft Visual Studio. Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) creado por Microsoft para sistemas operativos Windows como también para Mac y Linux. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

En el presente proyecto se utilizó Visual Studio 2019, bueno Microsoft acaba de anunciar el lanzamiento del nuevo Visual Studio 2019, su conocido entorno de desarrollo unificado llega renovado y cargado de características interesantes, y está disponible para su descarga gratuita tanto en Windows como en macOS.

Entre sus principales novedades destacan que **Visual Studio 2019** ahora incluye finalización del código asistida por **inteligencia artificial** gracias a **Visual Studio IntelliCode**, y también se ha integrado por completo la herramienta de colaboración en tiempo real Visual Studio Live Share.

**Interfaz gráfica renovada**

Lo primero que vemos al abrir Visual Studio 2019, es que ha cambiado ligeramente su interfaz, y ahora nos muestra un menú renovado para elegir la solución con la que vamos a trabajar y en el que, además, podemos clonar directamente un repositorio Git desde su fuente como también el menú de **creación de proyectos** también ha sido actualizado y por último, la interfaz de trabajo es ahora más compacta al desaparecer la barra superior con el título, y añade un buscador directamente en la barra de herramientas.

**Navegación entre errores y advertencias**

Esta nueva utilidad nos permite movernos entre las líneas donde se están generando errores y advertencias de compilación y nos lleva a cada línea concreta de la lista. Esto es similar a cuando en versiones anteriores pulsábamos sobre la línea del error, pero podemos saltar de una a otra sin necesidad de buscarlas en la lista.

**Portapapeles integrado**

En Visual Studio 2019 se ha incluido una utilidad que nos permite manejar un portapapeles para copiar y pegar contenido. Con esta utilidad, podemos acceder a las últimas cosas que hayamos copiado en vez de poder acceder solo a la última. Para acceder al portapapeles hay que pulsar Ctrl+Shift+V.

**Búsqueda en las variables locales**

Una herramienta muy interesante que introduce Visual Studio 2019 es la búsqueda en las variables locales. Con ella podemos buscar entre el nombre de las variables, y lo que es aún más interesante, buscar en el contenido de las variables sin tener que estas desplegándolas, pudiendo además indicar hasta qué profundidad de variables anidadas queremos llegar:

**Visual Studio Live Share integrado**

Esta herramienta para equipos de trabajo que apareció para Visual Studio 2017 como una extensión, ahora forma parte de manera nativa de Visual Studio 2019. Con ella, es posible trabajar en equipo en el mismo código, y facilita la programación en pareja o en equipo, sin estar todos presentes.

**Soporte para Kubernetes**

Al igual que con Visual Studio Live Share, existía soporte para Kubernetes a través de extensiones en Visual Studio 2017. Ahora, el soporte es nativo y viene integrado en el entorno.

Esta opción se instala automáticamente con la carga de trabajo Desarrollo de Azure

**Desarrollo web**

* Mejoras en los diálogos de proyecto para ASP.NET Core
* Se ha agregado compatibilidad para aplicaciones de ASP.NET Core que se ejecutan en una instancia de Azure Kubernetes Service.
* ASP.NET es compatible con la herramienta Uso de CPU del Generador de perfiles de rendimiento.
* Al publicar aplicaciones ASP.NET sobre Azure App Service se pueden asociar los recursos dependientes.

**Desarrollo móvil con Xamarin**

* Se ha reducido el tamaño de la carga de trabajo en torno a un 50%.
* Información detallada del proceso de compilación.
* Posibilidad de utilizar IntelliCode para XAML.
* Es posible obtener una vista previa sin necesitar compilar el proyecto.
* Mejoras en el rendimiento de compilación e implementación.
* Mejoras de Android Emulator.

**Novedades en C#**

* Se añade soporte para la versión preliminar de C# 8.0
* Soporte para .NET Core Preview 3.
* Integración con IntelliCode.
* Nuevas refactorizaciones disponibles.
* Realce de sintaxis en expresiones regulares

**Limpieza de código para C#**

Esta nueva versión viene con una herramienta de limpieza de código, que nos permite dejar solo lo necesario en nuestros proyectos.

Esta herramienta dispone de dos perfiles configurables con varias opciones, algunas de ellas muy interesantes, por ejemplo:

* Eliminar instrucciones using innecesarias.
* Reordenar las instrucciones using.
* Quitar variables no utilizadas.
* Quitar conversiones innecesarias.
* Ordenar modificadores de accesibilidad.

**Novedades en C++**

* Mejora de IntelliSense en proyectos CMake.
* Análisis de código ejecutado en segundo plano.
* Ir a ahora funciona en las directivas #include.
* Mejoras de rendimiento en la compilación.
* Compatibilidad básica con OpenMP SIMD.
* Integración con IntelliCode.
* Se ha retirado la plantilla de C++ administrado.
* Visual Studio Live Share ahora también admite C++.

**Novedades en F#**

* Se añade soporte para la versión preliminar de F# 4.6.
* Mejoras importantes en el rendimiento.
* Mejoras en el compilador.

**Novedades en Python**

* Se ha agregado un cuadro de diálogo Agregar para simplificar la creación y manejo de entornos virtuales y entornos de conda en el proyecto.
* Mejoras en el IntelliSense.
* Visual Studio Live Share ahora admite también Python.

**Novedades en JavaScript/TypeScript**

* Compatibilidad para habilitar la depuración de JavaScript dentro de pruebas unitarias en proyectos de Node.js.
* Novedades en el control de código fuente
* Posibilidad de gestionar PR (solicitud de extracción o Pull Request en inglés) desde Visual Studio mediante la extensión Pull Requests for Visual Studio.
* Guardado provisional de los cambios.
* Ahora se pueden asignar los elementos de trabajo de Azure DevOps desde Visual Studio.

**Otras muchas cosas nuevas e interesantes**

Es imposible hablar sobre todas las nuevas opciones sin hacer una entrada larguísima de leer, por eso he intentado seleccionar las cosas más interesantes de cada área. La lista completa es verdaderamente larga, y puedes encontrarla aquí: Notas de la versión de Visual Studio 2019.

Referencia sitio web(<https://www.campusmvp.es/recursos/post/que-hay-de-nuevo-en-visual-studio-2019.aspx>)

Referencia sitio web (<https://docs.microsoft.com/en-us/visualstudio/ide/whats-new-visual-studio-2019?view=vs-2019>)

## **2.3 Lenguaje de programación C#**

El lenguaje C# (en inglés es pronunciado como “C Sharp”, en español como “C Almohadilla”), es un lenguaje de programación diseñado por la conocida compañía Microsoft. Fue estandarizado en hace un tiempo por la ECMA e ISO dos de las organizaciones más importantes a la hora de crear estándares para los servicios o productos. El lenguaje de programación C# está orientado a objetos.

La programación orientada a objetos es una rama de la informática que usa como su propio nombre indica los objetos y las interacciones de estos para diseñar aplicaciones y programas informáticos. Cabe destacar que un objeto en programación es una entidad que combina el estado (son los datos del objeto), comportamiento o método (las que define qué operaciones puede hacer el objeto) e identidad (es el factor diferenciador de los otros objetos).

C# es considerado como una evolución y necesidad de ciertas circunstancias. Evolución por sus lenguajes antecesores que son el C y el C++ y necesidad a la hora en que la compañía tuvo problemas con la empresa creadora del lenguaje Java. Es por lo anterior que C# presenta los atributos positivos de C++, Java y Visual Basic y los mejora otorgando un lenguaje fuerte y actualizado para los tiempos actuales.

### **2.3.1 Historia de C#**

El lenguaje de programación C# fue creado por el danés Anders Hejlsberg que diseño también los lenguajes Turbo Pascal y Delphi. El C# (pronunciado en inglés “C Sharp” o en español “C sostenido”) es un lenguaje de programación orientado a objetos.

Con este nuevo lenguaje se quiso mejorar con respecto de los dos lenguajes anteriores de

los que deriva el C, y el C++.

Se hace énfasis en lo anterior debido que al ser .Net la plataforma por la cual se diseñó C# las características de dicha plataforma serán características propias del lenguaje de programación, por ende, estás son algunas características:

* **Sencillez:** En comparación a los otros lenguajes antecesores de este, C# elimina cierto objetos y atributos innecesarios para que la acción de programar sea más intuitiva.
* **Modernidad**: Aunque hemos mencionado que su creación esta también enfocada para dar solución a los temas actuales, también el lenguaje C# realiza de manera automática e intuitiva la incorporación de algunos objetos que con el paso de los años han sido necesarios a la hora de programar.
* **Seguridad:** Desde unas instrucciones para realizar acciones seguras y un mecanismo muy fuerte para la seguridad de los objetos.
* **Sistemas de tipos unificados**: Todos los datos que se obtienen al programar el lenguaje C# quedan guardadas en una base para que puedan ser utilizada posteriormente.
* **Extensibilidad**: Esta característica es muy positiva, debido a que puedes añadir tipos de datos básicos, operadores y modificadores a la hora de programar.
* **Versionable:** Dispone la característica de tener versiones, es decir, actualizarse y mejorar constantemente.
* **Compatible**: Tanto con sus antecesores como con Java y muchos otros lenguajes de programación, C# integra a todos estos para facilidad del programador.

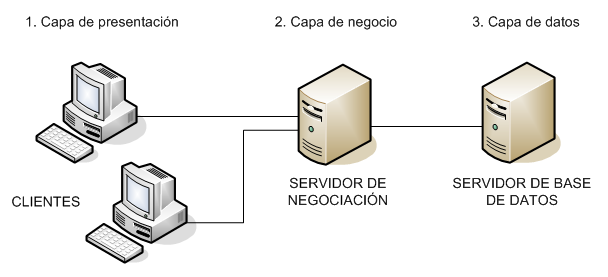
A pesar que el lenguaje C# forma parte de la plataforma .NET, que es una interfaz de programación de aplicaciones, C# es un lenguaje independiente que originariamente se creó para producir programas sobre esta plataforma .NET.

La plataforma C# se creó, porque el Visual Basic era uno de los lenguajes de programación que se encargaban de desarrollar estas aplicaciones, pero el Visual Basic es un lenguaje orientado a objetos algo pobre, porque se quiso que fuese desde su creación un lenguaje fácil de aprender para los programadores novatos. Por esto, surgió el C#, para suplir esta deficiencia del Visual Basic.

El Visual Basic no tiene algunas de las características necesarias como la herencia, los métodos virtuales, la sobrecarga de operadores, etc. Que se han conseguido con el C# y la plataforma .NET.

Referencia sitio web(http://biblioteca.utec.edu.sv/siab/virtual/tesis/)

## **2.4 Programación En 3 Capas**



La programación por capas se refiere a un estilo de programación que tiene como objetivo separar responsabilidades de la tal manera que cada capa cumpla una función específica y diferente a las demás.

Dentro de este estilo de programación se destaca “El desarrollo de software a tres capas” el cual es una técnica ampliamente usada en el desarrollo de sistemas de información que involucren conexiones a bases de datos.

Entre las ventajas que se pueden destacar sobre el desarrollo de software a tres capas ese tiene:

* La posibilidad de reutilizar código fácilmente
* La separación de roles en tres capas hace más sencillo reemplazar o modificar a una, sin afectar a las demás.
* Poder cambiar la presentación de la aplicación sin afectar a la lógica de ni a la Base de datos.
* La capacidad de poder cambiar el motor de Base de Datos sin grandes impactos al resto del proyecto.

La programación a tres capas es una técnica de desarrollo de software fundamentada en la programación orientada a objetos (POO), que divide los componentes de la aplicación en las capas de presentación, de lógica de negocio y de acceso a datos.

1. **Capa de presentación**

Es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario dado un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

1. **Capa negocio**

Es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica de negocio), pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

1. **Capa de datos**

Es donde residen los datos. Está formada por uno u más gestores de base de datos que realiza todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa negocio.

## **2.5 Programacion extrema xp**

La metodología XP o Programación Extrema es una metodología ágil y flexible utilizada para la gestión de proyectos. Extreme Programming se centra en potenciar las relaciones interpersonales del equipo de desarrollo como clave del éxito mediante el trabajo en equipo, el aprendizaje continuo y el buen clima de trabajo. Esta metodología pone el énfasis en la retroalimentación continua entre cliente y el equipo de desarrollo y es idónea para proyectos con requisitos imprecisos y muy cambiantes.

La programación extrema es una metodología de desarrollo ligero (o ágil) basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas. Este modelo de programación se basa en una serie de metodologías de desarrollo de software en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.   
         Una de las características principales de este método de programación, es que sus ingredientes son conocidos desde el principio de la informática. Los autores de XP han seleccionado aquellos que han considerado mejores y han profundizado en sus relaciones y en cómo se refuerzan los unos con los otros. El resultado de esta selección ha sido esta metodología única y compacta. Por esto, aunque no está basada en principios nuevos, sí que el resultado es una nueva manera de ver el desarrollo de software.   
         El objetivo que se perseguía en el momento de crear esta metodología era la búsqueda de un método que hiciera que los desarrollos fueran más sencillos. Aplicando el sentido común.

## **2.6 Entity Framework**

Entity Framework es un conjunto de tecnologías en ADO.NET que admite el desarrollo de aplicaciones de software orientadas a datos. Los arquitectos y desarrolladores de aplicaciones orientadas a datos han luchado con la necesidad de lograr dos objetivos muy diferentes. Deben modelar las entidades, las relaciones y la lógica de los problemas comerciales que están resolviendo, y también deben trabajar con los motores de datos utilizados para almacenar y recuperar los datos. Los datos pueden abarcar múltiples sistemas de almacenamiento, cada uno con sus propios protocolos; incluso las aplicaciones que funcionan con un solo sistema de almacenamiento deben equilibrar los requisitos del sistema de almacenamiento con los requisitos para escribir un código de aplicación eficiente y fácil de mantener.

Entity Framework permite a los desarrolladores trabajar con datos en forma de objetos y propiedades específicos del dominio, como las direcciones de clientes y clientes, sin tener que preocuparse por las tablas y columnas de la base de datos subyacentes donde se almacenan estos datos. Con el Entity Framework, los desarrolladores pueden trabajar a un nivel más alto de abstracción cuando tratan con datos, y pueden crear y mantener aplicaciones orientadas a datos con menos código que en las aplicaciones tradicionales. Debido a que Entity Framework es un componente de .NET Framework, las aplicaciones de Entity Framework pueden ejecutarse en cualquier computadora en la que esté instalado .NET Framework a partir de la versión 3.5 SP1.

**Dar vida a las modelos**

Un enfoque de diseño común y de larga data al construir una aplicación o servicio es la división de la aplicación o servicio en tres partes: un modelo de dominio, un modelo lógico y un modelo físico. El modelo de dominio define las entidades y relaciones en el sistema que se está modelando. El modelo lógico para una base de datos relacional normaliza las entidades y las relaciones en tablas con restricciones de clave externa. El modelo físico aborda las capacidades de un motor de datos en particular especificando los detalles de almacenamiento, como la partición y la indexación.

Los administradores de bases de datos refinan el modelo físico para mejorar el rendimiento, pero los programadores que escriben el código de la aplicación se limitan principalmente a trabajar con el modelo lógico escribiendo consultas SQL y llamando a procedimientos almacenados. Los modelos de dominio se usan generalmente como una herramienta para capturar y comunicar los requisitos de una aplicación, con frecuencia como diagramas inertes que se ven y discuten en las primeras etapas de un proyecto y luego se abandonan. Muchos equipos de desarrollo omiten la creación de un modelo conceptual y comienzan especificando tablas, columnas y claves en una base de datos relacional.

El Entity Framework le da vida a los modelos al permitir que los desarrolladores consulten entidades y relaciones en el modelo de dominio (llamado modelo conceptual en el Entity Framework) mientras confían en el Entity Framework para traducir esas operaciones a comandos específicos de la fuente de datos. Esto libera las aplicaciones de dependencias codificadas en una fuente de datos en particular.

**Mapear objetos a datos**

La programación orientada a objetos plantea un desafío para interactuar con los sistemas de almacenamiento de datos. Aunque la organización de clases a menudo refleja la organización de tablas de bases de datos relacionales, el ajuste no es perfecto. Múltiples tablas normalizadas con frecuencia corresponden a una sola clase, y las relaciones entre clases a menudo se representan de manera diferente a las relaciones entre tablas. Por ejemplo, para representar al cliente para un pedido de venta, una Orderclase puede usar una propiedad que contiene una referencia a una instancia de una Customerclase, mientras que una Orderfila de la tabla en una base de datos contiene una columna de clave externa (o conjunto de columnas) con un valor que corresponde a un valor de clave principal en la Customertabla. Una Customerclase podría tener una propiedad llamadaOrdersque contiene una colección de instancias de la Orderclase, mientras que la Customertabla en una base de datos no tiene una columna comparable. El Entity Framework proporciona a los desarrolladores la flexibilidad de representar las relaciones de esta manera, o para modelar las relaciones más de cerca a medida que se representan en la base de datos.

**Acceder y cambiar datos de la entidad**

Más que una solución de mapeo de objeto relacional, el Entity Framework se basa fundamentalmente en permitir que las aplicaciones accedan y cambien datos que se representan como entidades y relaciones en el modelo conceptual. Entity Framework utiliza la información del modelo y los archivos de mapeo para traducir las consultas de objetos contra los tipos de entidades representados en el modelo conceptual en consultas específicas de la fuente de datos. Los resultados de la consulta se materializan en objetos que gestiona Entity Framework. Entity Framework proporciona las siguientes formas de consultar un modelo conceptual y devolver objetos:

* LINQ a Entidades. Proporciona soporte de consulta integrada en el lenguaje (LINQ) para consultar tipos de entidades que se definen en un modelo conceptual. Para más información, vea LINQ to Entities.
* Entidad SQL. Un dialecto de SQL independiente del almacenamiento que trabaja directamente con las entidades en el modelo conceptual y que admite los conceptos del modelo de datos de entidad. Entity SQL se usa tanto con consultas de objetos como con consultas que se ejecutan utilizando el proveedor EntityClient.

**Proveedores de datos**

El EntityClientproveedor amplía el modelo de proveedor ADO.NET al acceder a los datos en términos de entidades conceptuales y relaciones. Ejecuta consultas que utilizan Entity SQL. Entity SQL proporciona el lenguaje de consulta subyacente que permite EntityClientcomunicarse con la base de datos. Para obtener más información, consulte EntityClient Provider para Entity Framework .

Entity Framework incluye un proveedor de datos de SqlClient actualizado que admite árboles de comandos canónicos. Para obtener más información, consulte SqlClient para Entity Framework.

Sitio web referencial:( <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview>)

## **2.7 Odontograma**

Un odontograma es un esquema utilizado por los odontólogos que permite registrar información sobre la boca de una persona. En dicho gráfico, el profesional detalla qué cantidad de piezas dentales permanentes tiene el paciente, cuáles han sido restauradas y otros datos de importancia.

El odontograma, de este modo, supone un registro de la historia clínica del individuo. Se trata, por lo tanto, de una herramienta de identificación. El odontólogo, al analizar el odontograma de un paciente, puede saber qué trabajos se realizaron en la boca de la persona en cuestión y establecer comparaciones entre el estado bucal actual y el registrado en la visita anterior.

Los odontogramas, por otra parte, pueden ser utilizados en la identificación de cuerpos no reconocidos. Cuando la policía halla un cadáver en mal estado y no logra determinar su identidad, puede investigar distintos odontogramas para tratar de reconocer al fallecido a través de sus dientes, que son piezas de gran resistencia y que no se descomponen, como sí ocurre con los órganos.

El esquema del odontograma puede tener diferentes formatos. Hay versiones que identifican los dientes con números, otros con letras mayúsculas e incluso algunos con pares numéricos. Todo depende de la preferencia del odontólogo para elegir uno u otro formato.

Ya sea para que el odontólogo tenga presentes los trabajos hechos en la boca de su paciente, para identificar a una persona o para facilitar el intercambio de datos médicos entre profesionales, el odontograma es una historia clínica de gran importancia que debe actualizarse con frecuencia.

### **¿Cuántos Dientes Tenemos?**

¿Cuántos dientes tenemos? ¿Alguna vez te lo has preguntado? La respuesta depende de algunos factores: el más importante es la edad. Los niños y los adultos tienen diferentes grupos de dientes. Entonces, ¿cuántos dientes tienen los niños?

***Dientes de leche***

La dentición de los niños comienza alrededor de los 6 meses de edad. El término técnico para estos primeros dientes es «dentición decidua» porque con el tiempo se caen, tal como las hojas caen de un árbol en otoño. La mayoría de las personas los conoce como dientes de leche, a pesar de que a veces también se los llama dientes de bebé o dientes primarios.

En total, los niños tienen 20 dientes de leche: 10 en la parte superior y 10 en la parte inferior de la boca. Estos dientes actúan como marcadores de posición para los dientes de adulto que crecen después de que los dientes de leche se caen. Tenga en cuenta que solo porque estos dientes se van a caer, no significa que no se les debería cuidar de la misma manera que a los dientes de adultos. Tener hábitos de alimentación y de cepillado saludables debería comenzar desde que le comiencen a salir los dientes a su hijo. Alrededor de los seis años de edad las muelas de leche se caen, las cuales se reemplazan con los dientes de adulto. Este proceso continuará hasta comienzos de la adolescencia.

***Dientes de adulto***

Los adultos tienen más dientes que los niños; la mayoría de los adultos tienen 32 dientes. Entre ellos, hay 8 incisivos, 4 caninos, 8 premolares y 12 molares (lo que incluye 4 muelas de juicio). La mayoría de las personas tienen todos los dientes de adulto cuando alcanzan la adolescencia. Es común que a los adultos se les extraigan las muelas de juicio porque no siempre hay mucho lugar para que crezcan de forma cómoda o sin provocar la desalineación de otros dientes.

Entonces, ¿cuántos dientes tenemos? Por lo general, los niños tienen 20 dientes, y los adultos tienen 32 (28 si se les extraen las muelas de juicio). Y cada uno de ellos necesita cuidados. Conozca más sobre el cuidado de sus dientes aquí, en los recursos de Cuidado oral de Colgate.

Sitio web (<https://www.colgate.com/es-py/oral-health/basics/mouth-and-teeth-anatomy/how-many-teeth-do-we-have-0113>)

# **CAPITULO III: MARCO APLICATIVO**

## **3.1 Requerimientos**

### **3.1.1 Requerimientos funcionales (product backlog)**

|  |  |
| --- | --- |
| **Identificación del documento** | |
| **Código:** | SD-WED-02 |
| **Titulo:** | Sistema de información de odontograma para la clinica dental “Odonto Viera" |
| **Fecha de Creación:** | 14 de junio del 2019 |
| **Elaborado por:** | Widen Valles Álvarez  Luis Rodrigo Abasto Torrez  Luis Fernando Quinteros |

|  |  |
| --- | --- |
| **Lista de Distribución** | |
| **Desarrollo:** | Widen Valles Alvarez  Luis Rodrigo Abasto Torrez  Luis Fernando Quinteros Santos |
| **Cliente.** | Clínica Dental “Odonto Viera” |

|  |  |
| --- | --- |
| **Revisión del Documento** | |
| **Revisado por:** | Ing. Magno Edwin Calizaya A. |
| **En fecha.** | 14/06/2019 |

|  |  |
| --- | --- |
| **Aprobación del Documento** | |
| **Aprobado por:** | Ing. Magno Edwin Calizaya A. |
| **En Fecha:** | 14/06/2019 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Control de Cambio** | | | |
| **Versión** | **Causa el Cambio** | **Responsable** | **Fecha** |
| **V01** | Requerimiento Inicial | Realizar Odontograma | 14/06/2019 |
| **V02** | Base de Datos | Odontograma | 14/06/2019 |
| **V03** | Requerimiento Final | Realizar Atencion | 14/06/2019 |

**Tabla 1.1 Product blacklog**

Fuente (Elaboración Propia)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nro.** | **Nombre del Requerimiento** | **Como probarlo**  **(Atributos de calidad)** | **Departamento Sección** | **Estimación (History Point)** |
| RF01 | Gestionar Paciente | Permite registrar, eliminar, modificar y buscar los datos del estudiante. El ID es generado por el sistema | Administración | 24 |
| RF02 | Gestionar Trabajador | Permite registrar, eliminar, modificar, buscar los datos de los trabajadores y asignarlo un cargo y rol para poder iniciar sección en el sistema. El ID es generado por el sistema. | Administración | 24 |
| RF03 | Gestionar Agenda | Permite registrar, eliminar, modificar, agregar odontólogo disponible y buscar los datos de la Agenda | Administración | 24 |
| RF04 | Gestionar Atencion | Permite registrar, agregar paciente, agregar agenda, agregar usuario, eliminar, modificar y buscar los datos de la Atencion. | Administración | 24 |
| RF05 | Gestionar Tratamiento | Permite registrar, eliminar, modificar y buscar los datos del tratamiento. El ID es generado por el sistema | Administración | 24 |
| RF06 | Gestionar Parte del diente | Permite registrar, eliminar, modificar y buscar los datos. El id es asignado respectivamente por el usuario según el número de diente. | Administración | 24 |
| RF07 | Gestionar  Odontograma | Permite registrar, eliminar, modificar, buscar los datos y asignar tratamientos, diente y la parte del diente que se va realizar el odontograma. | Administración | 24 |
| RF08 | Gestión  De Reportes | Permite generar reportes de datos de todos los módulos y reportes de del paciente que realizo el odontograma mas el costo de los tratamiento. | Administración | 24 |

### **3.1.2 Requerimientos no funcionales**

**a) Tiempo de aprendizaje**

La capacitación del personal, en el manejo de software y mantenimiento del sistema deberá considerar un tiempo considerable por lo menos 10 horas laborales de capacitación (Como mínimo 1 hora por día), hasta que los alineamientos de manejo del sistema y otros hayan sido debidamente adquiridos por el personal previamente asignado.

**b) Identificación del usuario propio de la aplicación**

El usuario ingresará al sistema con su cuenta y contraseña, que será validadas por el sistema, los permisos según el rol que tiene asignado.

Contraseña del usuario: Para registración de la contraseña de usuario deberán asegurarse que: La longitud de la contraseña debe ser de 6 caracteres mínimos y como máximo 15, además de cambiarla gradualmente.

**c) Confiabilidad**

**Tiempo de disponibilidad del sistema**

La aplicación puede estar disponible 12 horas de lunes a viernes en horarios 07:30 am a 19:30 pm durante todo el año para los empleados de la clínica dental.

Debido a que cualquier usuario con usuario y contraseña (registrado en sistema) puede tener acceso al sistema, es necesario hacer notar que, para cumplir con la mera de la disponibilidad del sistema, se deberá analizar cierto parámetro como: Rendimiento de servidor, Calidad del enlace y calidad de conexión de red local para el usuario final.

**Tiempo fuera de servicio**

El tiempo máximo de fuera de operaciones depende del funcionamiento del servidor como ser: Hardware del ordenador, Base de Datos y la infraestructura de la red. El mismo debe ser: Fallas comunes 10 minutos (Aprox.) o Fallas no comunes 1 hora (Aprox.).

**d) Performance**

**Acceso de usuarios al sistema:**

Los usuarios pueden acceder a los datos en tiempo real.

**Tiempo de respuesta**

El tiempo de respuesta al acceso del usuario debe ir de 7 segundos, la primera vez que ingresa al sistema, después menos de 3 segundos.

El tiempo de repuesta para una tracción promedio también debe ser de 7, la primera vez que se ingresa al sistema, después menos de 3 segundos

El tiempo de respuesta para una transacción promedio también debe ser de 7 segundos.

**Calidad de atención al usuario**

El sistema debe poder atender normalmente 2 o más usuarios al mismo tiempo.

**e) Restricciones de diseño**

**Estándares de diseño**

Versión de framework 4.7.2

**Estándares de arquitectura**

* Se debe usar para el desarrollo de software la arquitectura en 3 capas.
* Para la infraestructura de red, se debe implementar la arquitectura Cliente/Servidor**.**

**Motor de base de datos**

Se utiliza el motor de base de datos SQL Server 2017(Estándar Edition) de Microsoft.

**Cliente de escritorio**

La aplicación deberá ser accesible utilizando un ordenador Core i3 o superior con sistema Operativo Windows 7 o superior (exclusivamente de Microsoft).

**Servidor de datos**

El servidor será: Windows Server Estándar Edition.

**Lenguaje de programación**

La aplicación se desarrolló en Visual Studio 2019, utilizando Windows Form de Visual C#.net y derivados de SQL con T/SQL para el motor de base de datos SQL Server 2017

**f) Interfaces**

**Interfaz de usuario**

No debe existir presencia de imágenes distorsionadas o difíciles de entender. La presentación de mensaje de error o de infracción al usuario deberá ser lo más específico posible y comunicarse con el administrador del sistema.

**Interfaz del Hardware**

Los usuarios necesitan un dispositivo que mediante una interfaz de red (Dominio de red Local),

Active Directory, etc.) Les permita acceder al sistema, vía red Local.

Este dispositivo puede estar basado en cualquier arquitectura, pero deberá disponer al menos de un procesador con capacidad suficiente para ejecutar el sistema contable

**Interfaz de comunicaciones**

Existe una conexión entre los usuarios y el servidor donde está alojado la base de datos:

* Los usuarios cliente se conectaran al sistema contable mediante una red local. Esta conexión la realizarán desde su oficina en la misma infraestructura de la empresa.
* Para el Entorno de Red: la aplicación deberá tener la capacidad de funcionar en un entorno de red LAN.

**Interfaz de software**

Cualquier usuario que desee conectarse al sistema de escritorio necesitara de un sistema operativo Windows 7 o 8 para poder acceder a ella.

Por otra parte, el servidor de base de datos (SQL Server 2017 Enterprise Edition) se implementará bajo el sistema operativo Windows Server 2017.

## **3.2 Desarrollo del Software**

### **3.2.1 Especificación de Requerimientos de Software**

A diferencia de la metodología tradicional, XP utiliza las historias de usuario para la especificación de requerimiento, permitiendo disminución. XP presenta 4 valores que seguirlos y utilizarlos facilita la especificación de requerimientos:

* **La comunicación:** permite que el cliente y el programador lleguen a un acuerdo en la especialización de requerimientos evitando los malos entendidos.
* **La sencillez:** es lo que diferencia a XP con las demás metodologías tradicionales las cuales utilizan estándares para la especificación de requerimientos que hacen del sistema muy complejo. La sencillez evita la documentación extensa centrándose en lo básico, en lo que se utiliza en este momento y no en lo que se podría utilizar.
* **La realimentación:** permite que la especificación de requerimientos se comprenda mejor con el pasar del tiempo, permitiendo que los usuarios aprendan a escribir mejor las historias.
* **Las historias de usuario:** es una pequeña descripción del programa con el fin de estimar tiempo y costo, el programador pregunta al cliente, aumentando el detalle de cada historia.

### **3.2.1 Historias de Usuario**

Las historias de usuario permiten obtener los requerimientos del sistema a implementar los primeros requerimientos por parte del usuario, es importante no detallar las historias del usuario porque son utilizadas solo para dar una pequeña visión de lo que requiere obtener.

**Historia de Usuario para Gestionar Paciente**

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Titulo** | |
| 01 | Gestionar Paciente | |
| **Descripción** | | |
| Permite el registro de datos personales del paciente en el sistema | | |
| **Estimación(HP)** | **Prioridad** | **Dependiente de** |
| 24 | Alta | ------------------- |
| **Criterios de Aceptación** | | |
| * El código del Paciente se genera de manera automática, y debe ser único para cada paciente. * Un Paciente puede solicitar una consulta para luego realizarse su respectivo odontograma. * No se puede eliminar al cliente una vez que se realizó el odontograma. * Para realizar cualquier operación de modificar, guardar o buscar hay botones para los mismos. | | |
| **Responsable** | Widen Valles Álvarez  Luis Rodrigo Abasto Torrez  Luis Fernando Quinteros Santos | |
| **Adjunto** | | |
|  | | |

**Historia de usuario para Gestionar Empleado**

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Titulo** | |
| 02 | Gestionar Empleados | |
| **Descripción** | | |
| Permite el registro de datos personales del empleado en el sistema | | |
| **Estimación(HP)** | **Prioridad** | **Dependiente de** |
| 24 | Alta | ------------------- |
| **Criterios de Aceptación** | | |
| * El código del empleado se genera de manera automática, y debe ser único para cada empleado. * Un empleado puede realizar el servicio de cita y su respectivo odontograma de tratamiento. * No se puede eliminar al empleado una vez que haiga realizado una cita y odontograma. * Para realizar cualquier operación de modificar, guardar o buscar hay botones para los mismos. | | |
| **Responsable** | Widen Valles Álvarez  Luis Rodrigo Abasto Torrez  Luis Fernando Quinteros Santos | |
| **Adjunto** | | |
|  | | |

**Historia de usuario para gestionar Atencion**

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Titulo** | |
| 03 | Gestionar Atencion | |
| **Descripción** | | |
| Permite el registro de datos de la Atencion de los pacientes y odontólogo disponibles según su agenda | | |
| **Estimación(HP)** | **Prioridad** | **Dependiente de** |
| 24 | Alta | ------------------- |
| **Criterios de Aceptación** | | |
| * El código de la Atencion se genera de manera automática. * La Atencion se realiza para afirmar la fecha, hora que van a regresar el paciente para continuar su tratamiento. * No se puede eliminar al Paciente una vez que haiga realizado su respectiva Atencion. * Para realizar cualquier operación de modificar, guardar o buscar hay botones para los mismos. | | |
| **Responsable** | Widen Valles Álvarez  Luis Rodrigo Abasto Torres  Luis Fernando Quinteros Santos | |
| **Adjunto** | | |
|  | | |

**Historia de usuario para gestionar Tratamiento**

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Titulo** | |
| 04 | Gestionar Tratamiento | |
| **Descripción** | | |
| Permite el registra los tratamientos que se van realizar en el sistema | | |
| **Estimación(HP)** | **Prioridad** | **Dependiente de** |
| 24 | Alta | ------------------- |
| **Criterios de Aceptación** | | |
| * El código del Tratamiento se genera de manera automática, y debe ser único para cada tratamiento. * Un el tratamiento es creado para especificar en el odontograma que es lo que se va realizar y poder sacar su costo del tratamiento. * No se puede eliminar el tratamiento una vez que se realizó un odontograma. * Para realizar cualquier operación de modificar, guardar o buscar hay botones para los mismos. | | |
| **Responsable** | Widen Valles Álvarez  Luis Rodrigo Abasto Torrez  Luis Fernando Quinteros Santos | |
| **Adjunto** | | |
|  | | |

**Historia de usuario para gestionar Parte del Diente**

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Titulo** | |
| 05 | Gestionar Parte Diente | |
| **Descripción** | | |
| Permite registrar la parte del diente en el sistema. | | |
| **Estimación(HP)** | **Prioridad** | **Dependiente de** |
| 24 | Alta | ------------------- |
| **Criterios de Aceptación** | | |
| * El código de la parte del diente es creado por el usuario para las 5 partes, y debe ser único para cada parte del diente. * La parte del diente es creado para especificar en el odontograma cuál es la parte dañada del diente. * No se puede eliminar la Parte del Diente una vez se ha registrado en un odontograma. * Para realizar cualquier operación de modificar, guardar o buscar hay botones para los mismos. | | |
| **Responsable** | Widen Valles Álvarez  Luis Rodrigo Abasto Torrez  Luis Fernando Quinteros Santos | |
| **Adjunto** | | |
|  | | |

**Historia de usuario para Gestionar Diente**

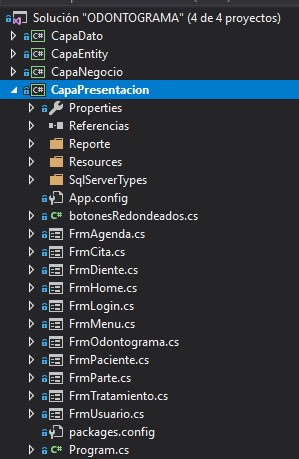
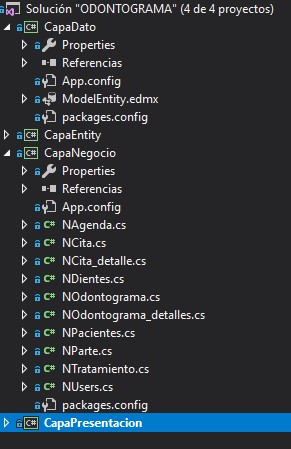
|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Titulo** | |
| 06 | Gestionar Diente | |
| **Descripción** | | |
| Permite registrar el diente en el sistema. | | |
| **Estimación(HP)** | **Prioridad** | **Dependiente de** |
| 24 | Alta | ------------------- |
| **Criterios de Aceptación** | | |
| * El código de la parte del diente es creado por el usuario según su estándar de numeración científica, y debe ser único para cada diente. * El diente es creado para especificar en el odontograma cuál es el diente dañado que se va realizar el tratamiento. * No se puede eliminar el Diente una vez se ha registrado en un odontograma. * Para realizar cualquier operación de modificar, guardar o buscar hay botones para los mismos. | | |
| **Responsable** | Widen Valles Álvarez  Luis Rodrigo Abasto Torrez  Luis Fernando Quinteros Santos | |
| **Adjunto** | | |
|  | | |

**Historia de usuario para Gestionar Odontograma**

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Titulo** | |
| 07 | Gestionar Odontograma | |
| **Descripción** | | |
| Permite registrar el odontograma con su tratamiento diagnosticado al paciente | | |
| **Estimación(HP)** | **Prioridad** | **Dependiente de** |
| 24 | Alta | ------------------- |
| **Criterios de Aceptación** | | |
| * El código del Odontograma es generado automáticamente por el sistema, y debe ser único para cada diente. * El Odontograma es creado para especificar el tratamiento que sea diagnosticado para su respectivo procedimiento que va realizar al paciente. * No se puede eliminar el odontograma una vez realizado su diagnóstico o su procedimiento al paciente. * Para realizar cualquier operación de modificar, guardar o buscar hay botones para los mismos. | | |
| **Responsable** | Widen Valles Álvarez  Luis Rodrigo Abasto Torrez  Luis Fernando Quinteros Santos | |
| **Adjunto** | | |
|  | | |

## **3.3 Diseño**

### **3.3.1 Arquitectura de la solución para el Modelo de Capas**

****

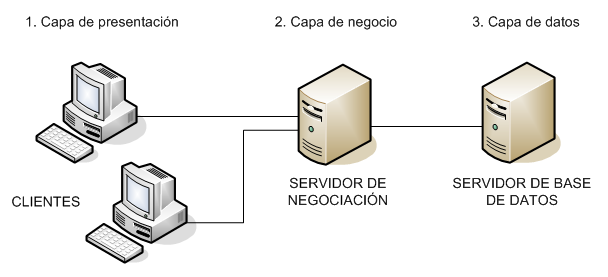
### **3.3.2 Descripción de la arquitectura**

A continuación, se hace una breve descripción de los distintos componentes que forman parte de la solución en general:

* **Biblioteca de datos:** Permite hacer la conexión al motor de base de datos de SQL Server, dentro de esta biblioteca se encuentra la clase llamada **Conexion.cs,** en esta clase están definidos los métodos principales para las operaciones al momento de manipular datos en la base de datos, estos métodos son: conectar, desconectar, consultar, etc.
* **Biblioteca de negocio:** Se encuentra cada una de las clases que componen el cuerpo del sistema, estas clases se caracterizan por tener la definición de atributos, propiedades y métodos que permiten dar forma a los objetos del sistema para cada uno de los requerimientos de negocios del sistema.
* **Biblioteca de presentación:** Dentro de la aplicación se encuentran todos los elementos de diseño para el sistema, como ser Formulario Principal, los formularios de los clientes, empleados, proveedores, productos, registro de ventas, registro de compras, almacén, ajustes, etc. Que son la base de diseño para poder invocar a los distintos formularios que forman parte del sistema.

### **3.3.3 Arquitectura de la solución basada en la tecnologia .Net**

La programación por capas es un estilo de programación en la que le objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa presentación al usuario.



La ventaja principal de este estilo, es que en caso de algún cambio sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además, permite distribuir el trabajo de creación de una aplicación por capas, de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles.

En el diseño de sistemas informáticos actual se suele usar la arquitectura Programación por capas.

En dicha arquitectura a cada capa se le confía una misión simple, lo que permite el diseño de arquitecturas escalable (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

### **3.3.4 Descripción de Capas**

**Capa de presentación**

Es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

**Capa de negocio**

Es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la para presentación para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar los datos en él.

**Capa de datos**

Es donde residen los datos. Está formada por uno o más gestores de base de datos que realiza todo el almacenamiento de datos y reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

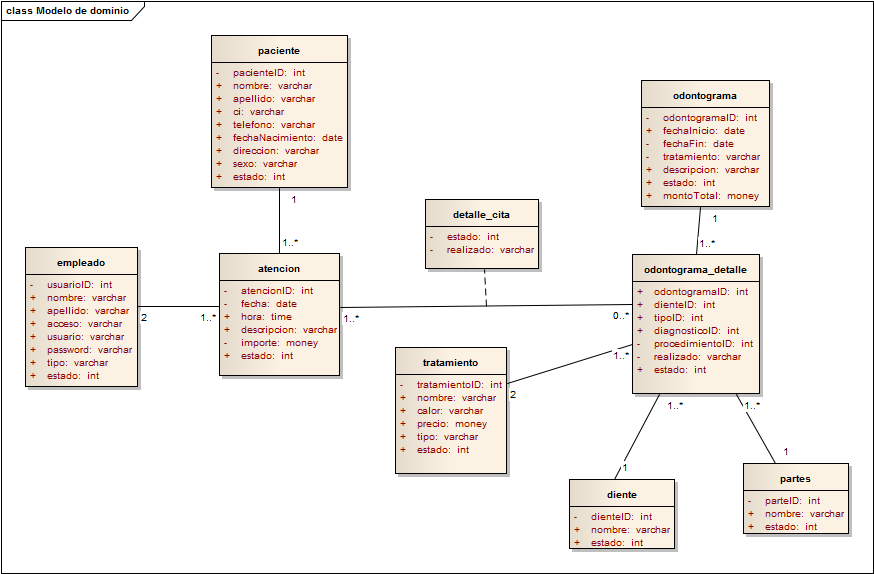
### **3.3.5 ADO.NET**

Los proveedores de acceso a datos ADO.NET (conocidos como “Managed Data Providers”), representan conjuntos específicos de clases que permiten conectarse e interactuar con una base de datos, cada uno utilizando un protocolo particular.

ADO.NET provee una arquitectura extensible, posibilitando que terceras partes creen sus propios proveedores de acceso nativo para aplicaciones .NET.

### **3.3.4 Diseño de la base de datos**

**Diagrama conceptual de la base de datos**

****

**Diseño Lógico de la Base de Datos**

**Paciente**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PK |  |  |  |  |  |  |  |
| pacienteID | Nombre | Apellidos | CI | Fecha Nacimiento | Telefono | dirreccion | sexo |

**Atencion**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PK | FK |  |  |  |  |  |
| atencionID | empleadoID | fecha | hora | descripción | importe | estado |

**Empleado**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PK |  |  |  |  |  |  |  |
| empleadoID | Nombre | Apellidos | Acceso | usuario | password | tipo | estado |

**Tratamiento**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PK |  |  |  |  |  |
| tratamientoID | Nombre | color | precio | tipo | estado |

**Odontograma**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PK |  |  |  |  |  |
| odontogramaD | fechaInicio | FechaFin | tratamiento | descripción | estado |

**Atencion\_Detalle**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| FK | FK | FK | FK | FK |  |  |
| citaID | odontogramaD | dienteID | tratamientoID | parteID | realizado | estado |

**Odontograma Detalle**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FK | FK | FK | FK |  |  |
| odontogramaD | dienteID | tratamientoID | parteID | realizado | estado |

**Diente**

|  |  |  |
| --- | --- | --- |
| PK |  |  |
| dienteID | Nombre | estado |

**Parte**

|  |  |  |
| --- | --- | --- |
| PK |  |  |
| parteID | Nombre | estado |

**Diseño Físico de la Base de Datos**

create database odontograma

go

use odontograma

go

create table paciente(

pacienteID int primary key identity(1,1),

nombre varchar(50),

apellido varchar(50),

ci varchar(20),

telefono varchar(20),

fechaNacimiento date,

direccion varchar(200),

sexo varchar(20),

estado int

)

go

create table users(

usuarioID int primary key identity(1,1),

nombre varchar(50),

apellido varchar(50),

tipo varchar(20),

usuario varchar (50),

password varchar(50),

estado int

)

go

create table cita(

citaID int primary key identity(1,1),

pacienteID int,

empleadoID int,

odontologoID int,

fecha date,

hora time,

importe decimal(20,2),

descripcion varchar(300),

tipo varchar(50),

estado int,

foreign key(pacienteID)references paciente(pacienteID),

foreign key(empleadoID)references users(usuarioID),

foreign key(odontologoID)references users(usuarioID)

)

go

create table odontograma(

odontogramaID int primary key identity(1,1),

fechaInicio date,

fechaFinal date,

descripcion varchar(300),

montoTotal decimal(20,2),

tratamiento varchar(50),

estado int,

)

go

create table diente(

dienteID int primary key,

nombre varchar(50),

estado int

)

go

create table tratamiento(

tratamientoID int primary key identity(1,1),

nombre varchar(50),

color varchar(50),

precio decimal(20,2),

tipo int,

estado int

)

go

create table parte(

parteID int primary key,

nombre varchar(50),

estado int

)

go

create table odontograma\_detalle(

odontogramaID int,

dienteID int,

parteID int,

diagnosticoID int,

procedimientoID int,

realizado varchar(10),

estado int,

primary key(odontogramaID,dienteID,parteID,diagnosticoID,procedimientoID),

foreign key(odontogramaID)references odontograma(odontogramaID),

foreign key(dienteID)references diente(dienteID),

foreign key(parteID)references parte(parteID),

foreign key(diagnosticoID)references tratamiento(tratamientoID),

foreign key(procedimientoID)references tratamiento(tratamientoID)

)

create table cita\_detalle(

citaID int,

odontogramaID int,

dienteID int,

parteID int,

diagnosticoID int,

procedimientoID int,

realizado varchar(10),

estado int,

primary key(citaID,odontogramaID,dienteID,parteID,diagnosticoID,procedimientoID),

foreign key(citaID)references cita(citaID),

foreign key(odontogramaID,dienteID,parteID,diagnosticoID,procedimientoID)references odontograma\_detalle(odontogramaID,dienteID,parteID,diagnosticoID,procedimientoID)

)

go

## **3.4 Implementación**

### **Aplicación de escritorio**

**Implementación de paciente: Clase Paciente**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDato;

using CapaEntity;

namespace CapaNegocio

{

public class NPacientes

{

public static long save(EPaciente Paciente)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<paciente> pacientes = new List<paciente>();

paciente Obj = new paciente();

pacientes = (from p in cn.paciente

where p.estado==1

where p.nombre == Paciente.nombre && p.apellido == Paciente.apellido

select p).ToList();

if (pacientes.Count > 0)

{

throw new Exception("El Paciente Ya Existe");

}

Obj.nombre = Paciente.nombre;

Obj.apellido = Paciente.apellido;

Obj.telefono = Paciente.telefono;

if (Obj.nombre == string.Empty && Obj.apellido == string.Empty)

{

throw new Exception("Ingrese Nombre y Apellido");

}

Obj.ci = Paciente.ci;

Obj.fechaNacimiento = Paciente.fechaNacimiento;

Obj.direccion = Paciente.direccion;

Obj.sexo = Paciente.sexo;

Obj.estado = 1;

cn.paciente.Add(Obj);

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.pacienteID;

}

else

{

throw new Exception("Error Al guardar El Registro");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static long update(EPaciente Paciente)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<paciente> pacientes = new List<paciente>();

paciente Obj = new paciente();

pacientes = (from p in cn.paciente

where p.estado == 1

where p.nombre == Paciente.nombre && p.apellido == Paciente.apellido

select p).ToList();

if (pacientes.Count > 1)

{

throw new Exception("El Paciente Ya Existe");

}

Obj = (from p in cn.paciente

where p.pacienteID == Paciente.pacienteID

select p).First();

Obj.nombre = Paciente.nombre;

Obj.apellido = Paciente.apellido;

Obj.telefono = Paciente.telefono;

Obj.ci = Paciente.ci;

Obj.fechaNacimiento = Paciente.fechaNacimiento;

Obj.direccion = Paciente.direccion;

Obj.sexo = Paciente.sexo;

Obj.estado = 1;

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.pacienteID;

}

else

{

throw new Exception("No hubo Ningun Cambio al Editar");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static string delete(EPaciente Paciente)

{

string rpta = "";

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

paciente Obj = new paciente();

Obj = cn.paciente.Find(Paciente.pacienteID);

rpta = Obj.estado == 1 ? "OK" : "No se Puede Eliminar el Registro";

Obj.estado = 0;

cn.SaveChanges();

}

catch (Exception ex)

{

rpta = (ex.Message);

}

return rpta;

}

public static List<EPaciente> mostrar(string nombre, string apellido, int pag)

{

try

{

List<EPaciente> Pacientes = new List<EPaciente>();

List<paciente> pacientes = new List<paciente>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

pacientes = (from p in cn.paciente

where p.estado == 1

where p.nombre.Contains(nombre)

where p.apellido.Contains(apellido)

orderby p.pacienteID descending

select p).ToList();

pag = pag \* 10;

var tabla = pacientes.Skip(pag).Take(10);

if (pacientes.Count < pag)

{

tabla = pacientes.Skip(pag).Take(10);

}

foreach (var item in tabla)

{

EPaciente Obj = new EPaciente();

Obj.pacienteID = item.pacienteID;

Obj.nombre = item.nombre;

Obj.apellido = item.apellido;

Obj.telefono = item.telefono;

Obj.ci = item.ci;

Obj.fechaNacimiento = item.fechaNacimiento;

Obj.direccion = item.direccion;

Obj.sexo = item.sexo;

Obj.estado = item.estado;

Pacientes.Add(Obj);

}

return Pacientes;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public EPaciente mostraDatos(int id)

{

try

{

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

paciente item = new paciente();

item = cn.paciente.Find(id);

EPaciente Obj = new EPaciente();

Obj.pacienteID = item.pacienteID;

Obj.nombre = item.nombre;

Obj.apellido = item.apellido;

Obj.telefono = item.telefono;

Obj.ci = item.ci;

Obj.fechaNacimiento = item.fechaNacimiento;

Obj.direccion = item.direccion;

Obj.sexo = item.sexo;

Obj.estado = item.estado;

return Obj;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public EPaciente mostraPacienteOdontograma(int ID)

{

try

{

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

paciente item = new paciente();

item = (from p in cn.paciente join c in cn.cita on p.pacienteID equals c.pacienteID

join ct in cn.cita\_detalle on c.citaID equals ct.citaID

join od in cn.odontograma\_detalle on ct.odontogramaID equals od.odontogramaID

join o in cn.odontograma on od.odontogramaID equals o.odontogramaID

where p.estado == 1 && o.odontogramaID == ID

select p).First();

EPaciente Obj = new EPaciente();

Obj.pacienteID = item.pacienteID;

Obj.nombre = item.nombre;

Obj.apellido = item.apellido;

Obj.telefono = item.telefono;

Obj.ci = item.ci;

Obj.fechaNacimiento = item.fechaNacimiento;

Obj.direccion = item.direccion;

Obj.sexo = item.sexo;

Obj.estado = item.estado;

return Obj;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static List<EPaciente> buscar(string nombre, string apellido)

{

try

{

List<EPaciente> Pacientes = new List<EPaciente>();

List<paciente> pacientes = new List<paciente>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

pacientes = (from p in cn.paciente

where p.estado == 1

where p.nombre.Contains(nombre) || p.apellido.Contains(apellido)

select p).ToList();

foreach (var item in pacientes)

{

EPaciente Obj = new EPaciente();

Obj.pacienteID = item.pacienteID;

Obj.nombre = item.nombre;

Obj.apellido = item.apellido;

Obj.telefono = item.telefono;

Obj.ci = item.ci;

Obj.fechaNacimiento = item.fechaNacimiento;

Obj.direccion = item.direccion;

Obj.sexo = item.sexo;

Obj.estado = item.estado;

Pacientes.Add(Obj);

}

return Pacientes;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static List<EPaciente> mostrarTodo()

{

try

{

List<EPaciente> Pacientes = new List<EPaciente>();

List<paciente> pacientes = new List<paciente>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

pacientes = (from p in cn.paciente

where p.estado == 1

select p).ToList();

foreach (var item in pacientes)

{

EPaciente Obj = new EPaciente();

Obj.pacienteID = item.pacienteID;

Obj.nombre = item.nombre;

Obj.apellido = item.apellido;

Obj.telefono = item.telefono;

Obj.ci = item.ci;

Obj.fechaNacimiento = item.fechaNacimiento;

Obj.direccion = item.direccion;

Obj.sexo = item.sexo;

Obj.estado = item.estado;

Pacientes.Add(Obj);

}

return Pacientes;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static int mostrarTotal(string nombre , string apellido)

{

try

{

List<paciente> pacientes = new List<paciente>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

pacientes = (from p in cn.paciente

where p.estado == 1

where p.nombre.Contains(nombre)

where p.apellido.Contains(apellido)

select p).ToList();

return pacientes.Count;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

**Implementación de empleado: Clase Empleado**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDato;

using CapaEntity;

namespace CapaNegocio

{

public class NUsers

{

public static long save(EUsers Usuario)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<users> user = new List<users>();

users Obj = new users();

user = (from u in cn.users

where u.nombre == Usuario.nombre && u.apellido == Usuario.apellido

select u).ToList();

if (user.Count > 0)

{

throw new Exception("El Nombre y Apellido Del Usuario Ya Existe");

}

user = (from u in cn.users

where u.usuario == Usuario.usuario

select u).ToList();

if (user.Count > 0)

{

throw new Exception("El Usuario Ya Existe");

}

Obj.nombre = Usuario.nombre;

Obj.apellido = Usuario.apellido;

Obj.tipo = Usuario.tipo;

if (Obj.nombre == string.Empty && Obj.apellido == string.Empty)

{

throw new Exception("Ingrese Nombre y Apellido");

}

Obj.usuario = Usuario.usuario;

Obj.password = Usuario.password;

Obj.estado = 1;

cn.users.Add(Obj);

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.usuarioID;

}

else

{

throw new Exception("Error Al guardar El Registro");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static long update(EUsers Usuario)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<users> usuarios = new List<users>();

users Obj = new users();

usuarios = (from u in cn.users

where u.nombre == Usuario.nombre && u.apellido == Usuario.apellido

select u).ToList();

if (usuarios.Count >= 2)

{

throw new Exception("El Usuario Ya Existe");

}

Obj = (from u in cn.users

where u.usuarioID == Usuario.usuarioID

select u).First();

Obj.nombre = Usuario.nombre;

Obj.apellido = Usuario.apellido;

Obj.tipo = Usuario.tipo;

Obj.usuario = Usuario.usuario;

Obj.password = Usuario.password;

Obj.estado = 1;

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.usuarioID;

}

else

{

throw new Exception("No Hubo Ningun Cambio al Editar");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static string delete(EUsers Usuario)

{

string rpta = "";

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

users Obj = new users();

Obj = cn.users.Find(Usuario.usuarioID);

rpta = Obj.estado == 1 ? "OK" : "No se Puede Eliminar el Registro";

Obj.estado = 0;

cn.SaveChanges();

}

catch (Exception ex)

{

rpta = (ex.Message);

}

return rpta;

}

public static List<EUsers> mostrar(string nombre ,string apellido, int pag)

{

try

{

List<EUsers> Usuarios = new List<EUsers>();

List<users> usuarios = new List<users>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

usuarios = (from u in cn.users

where u.estado == 1

where u.nombre.Contains(nombre)

where u.apellido.Contains(apellido)

orderby u.usuarioID descending

select u).ToList();

pag = pag \* 10;

var tabla = usuarios.Skip(pag).Take(10);

if (usuarios.Count < pag)

{

tabla = usuarios.Skip(pag).Take(10);

}

foreach (var item in tabla)

{

EUsers Obj = new EUsers();

Obj.usuarioID = item.usuarioID;

Obj.nombre = item.nombre;

Obj.apellido = item.apellido;

Obj.tipo = item.tipo;

Obj.usuario = item.usuario;

Obj.password = item.password;

Obj.estado = item.estado;

Usuarios.Add(Obj);

}

return Usuarios;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public EUsers login(EUsers Usuario)

{

try

{

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

users Obj = new users();

EUsers login = new EUsers();

Obj = (from u in cn.users

where u.usuario == Usuario.usuario && u.password == Usuario.password

select u).First();

login.usuarioID = Obj.usuarioID;

login.nombre = Obj.nombre;

login.apellido = Obj.apellido;

login.tipo = Obj.tipo;

login.usuario = Obj.usuario;

return login;

}

}

catch (Exception ex)

{

throw new Exception("El Usuario que Ingresaste no coinciden con ninguna Cuenta "+ex.Message);

}

}

public static int mostrarTotal(string nombre, string apellido)

{

try

{

List<users> usuarios = new List<users>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

usuarios = (from u in cn.users

where u.estado == 1

where u.nombre.Contains(nombre)

where u.apellido.Contains(apellido)

orderby u.usuarioID descending

select u).ToList();

return usuarios.Count;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public EUsers mostrarUserID(int ID)

{

try

{

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

users Obj = new users();

EUsers login = new EUsers();

Obj = (from u in cn.users

where u.usuarioID==ID

select u).First();

login.usuarioID = Obj.usuarioID;

login.nombre = Obj.nombre;

login.apellido = Obj.apellido;

login.tipo = Obj.tipo;

login.usuario = Obj.usuario;

login.password = Obj.password;

return login;

}

}

catch (Exception ex)

{

throw new Exception("El Usuario que Ingresaste no coinciden con ninguna Cuenta " + ex.Message);

}

}

public static List<EUsers> mostrarOdontologo(string nombre, string apellido, int pag)

{

try

{

List<EUsers> Usuarios = new List<EUsers>();

List<users> usuarios = new List<users>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

usuarios = (from u in cn.users

where u.estado == 1

where u.tipo == "Odontologo"

where u.nombre.Contains(nombre)

where u.apellido.Contains(apellido)

orderby u.usuarioID descending

select u).ToList();

pag = pag \* 10;

var tabla = usuarios.Skip(pag).Take(10);

if (usuarios.Count < pag)

{

tabla = usuarios.Skip(pag).Take(10);

}

foreach (var item in tabla)

{

EUsers Obj = new EUsers();

Obj.usuarioID = item.usuarioID;

Obj.nombre = item.nombre;

Obj.apellido = item.apellido;

Obj.tipo = item.tipo;

Obj.usuario = item.usuario;

Obj.password = item.password;

Obj.estado = item.estado;

Usuarios.Add(Obj);

}

return Usuarios;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

}

}

**Implementación de cita: Clase Atencion**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDato;

using CapaEntity;

namespace CapaNegocio

{

public class NCita

{

public static long save(ECita Atencion)

{

try

{

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

cita Obj = new cita();

Obj.pacienteID = Atencion.pacienteID;

Obj.empleadoID = Atencion.empleadoID;

Obj.odontologoID = Atencion.odontologoID;

Obj.fecha = Convert.ToDateTime(DateTime.Now.ToLongDateString());

Obj.hora = Atencion.hora;

Obj.importe = Atencion.importe;

Obj.descripcion = Atencion.descripcion;

Obj.estado = 1;

cn.cita.Add(Obj);

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.citaID;

}

else

{

throw new Exception("Error Al guardar El Registro");

}

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static long update(ECita Atencion)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<cita> citas = new List<cita>();

cita Obj = new cita();

if (citas.Count > 1)

{

throw new Exception("El cita Ya Existe");

}

Obj = (from p in cn.cita

where p.citaID == Atencion.citaID

select p).First();

Obj.pacienteID = Atencion.pacienteID;

Obj.empleadoID = Atencion.empleadoID;

Obj.odontologoID = Atencion.odontologoID;

Obj.fecha = Convert.ToDateTime(DateTime.Now.ToLongDateString());

Obj.hora = Atencion.hora;

Obj.importe = Atencion.importe;

Obj.descripcion = Atencion.descripcion;

Obj.estado = 1;

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.citaID;

}

else

{

throw new Exception("No hubo Ningun Cambio al Editar");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static string delete(ECita Atencion)

{

string rpta = "";

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

cita Obj = new cita();

Obj = cn.cita.Find(Atencion.citaID);

rpta = Obj.estado == 1 ? "OK" : "No se Puede Eliminar el Registro";

Obj.estado = 0;

cn.SaveChanges();

}

catch (Exception ex)

{

rpta = (ex.Message);

}

return rpta;

}

public static List<ECita> mostrar(int pag)

{

try

{

List<ECita> Citas = new List<ECita>();

List<cita> citas = new List<cita>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

citas = (from p in cn.cita

where p.estado == 1

orderby p.citaID descending

select p).ToList();

pag = pag \* 10;

var tabla = citas.Skip(pag).Take(10);

if (citas.Count < pag)

{

tabla = citas.Skip(pag).Take(10);

}

foreach (var item in tabla)

{

ECita Obj = new ECita();

Obj.citaID = item.citaID;

Obj.pacienteID = item.pacienteID;

Obj.empleadoID = item.empleadoID;

Obj.odontologoID = item.odontologoID;

Obj.fecha = item.fecha;

Obj.hora = item.hora;

Obj.importe = item.importe;

Obj.descripcion = item.descripcion;

Obj.Odontologo = item.odontologo.nombre + " " + item.odontologo.apellido;

Obj.Paciente = item.paciente.nombre + " " + item.paciente.apellido;

Obj.Empleado = item.empleado.nombre + " " + item.empleado.apellido;

Obj.estado = item.estado;

Citas.Add(Obj);

}

return Citas;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static List<ECita> mostrarReporteCitaRango(DateTime inicio, DateTime fin)

{

try

{

List<ECita> Citas = new List<ECita>();

List<cita> citas = new List<cita>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

citas = (from p in cn.cita

where p.estado == 1

where p.fecha >= inicio.Date && p.fecha <= fin.Date

select p).ToList();

foreach (var item in citas)

{

ECita Obj = new ECita();

Obj.citaID = item.citaID;

Obj.pacienteID = item.pacienteID;

Obj.empleadoID = item.empleadoID;

Obj.odontologoID = item.odontologoID;

Obj.fecha = item.fecha;

Obj.hora = item.hora;

Obj.importe = item.importe;

Obj.tipo = item.tipo;

Obj.descripcion = item.descripcion;

Obj.Odontologo = item.odontologo.nombre + " " + item.odontologo.apellido;

Obj.Paciente = item.paciente.nombre + " " + item.paciente.apellido;

Obj.Empleado = item.empleado.nombre + " " + item.empleado.apellido;

Obj.estado = item.estado;

Citas.Add(Obj);

}

return Citas;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static List<ECita> mostrarClienteID(int ID)

{

try

{

List<ECita> Citas = new List<ECita>();

List<cita> citas = new List<cita>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

citas = (from p in cn.cita

where p.estado == 1

where p.paciente.pacienteID==ID

select p).ToList();

foreach (var item in citas)

{

ECita Obj = new ECita();

Obj.citaID = item.citaID;

Obj.pacienteID = item.pacienteID;

Obj.empleadoID = item.empleadoID;

Obj.odontologoID = item.odontologoID;

Obj.fecha = item.fecha;

Obj.hora = item.hora;

Obj.importe = item.importe;

Obj.descripcion = item.descripcion;

Obj.Odontologo = item.odontologo.nombre + " " + item.odontologo.apellido;

Obj.Paciente = item.paciente.nombre + " " + item.paciente.apellido;

Obj.Empleado = item.empleado.nombre + " " + item.empleado.apellido;

Obj.estado = item.estado;

Citas.Add(Obj);

}

return Citas;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static List<ECita> mostrarCitaID(int ID)

{

try

{

List<ECita> Citas = new List<ECita>();

List<cita> citas = new List<cita>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

citas = (from p in cn.cita

where p.estado == 1

where p.citaID == ID

select p).ToList();

foreach (var item in citas)

{

ECita Obj = new ECita();

Obj.citaID = item.citaID;

Obj.pacienteID = item.pacienteID;

Obj.empleadoID = item.empleadoID;

Obj.odontologoID = item.odontologoID;

Obj.fecha = item.fecha;

Obj.hora = item.hora;

Obj.importe = item.importe;

Obj.descripcion = item.descripcion;

Obj.Odontologo = item.odontologo.nombre + " " + item.odontologo.apellido;

Obj.Paciente = item.paciente.nombre + " " + item.paciente.apellido;

Obj.Empleado = item.empleado.nombre + " " + item.empleado.apellido;

Obj.estado = item.estado;

Citas.Add(Obj);

}

return Citas;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static List<ECita> mostrarOdontogramaCita(int ID)

{

try

{

List<ECita> Citas = new List<ECita>();

List<cita> citas = new List<cita>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

citas = (from p in cn.cita join cd in cn.cita\_detalle on p.citaID equals cd.citaID

where p.estado == 1

where cd.odontogramaID==ID

select p).ToList();

foreach (var item in citas)

{

ECita Obj = new ECita();

Obj.citaID = item.citaID;

Obj.pacienteID = item.pacienteID;

Obj.empleadoID = item.empleadoID;

Obj.odontologoID = item.odontologoID;

Obj.fecha = item.fecha;

Obj.hora = item.hora;

Obj.importe = item.importe;

Obj.descripcion = item.descripcion;

Obj.Odontologo = item.odontologo.nombre + " " + item.odontologo.apellido;

Obj.Paciente = item.paciente.nombre + " " + item.paciente.apellido;

Obj.Empleado = item.empleado.nombre + " " + item.empleado.apellido;

Obj.estado = item.estado;

Citas.Add(Obj);

}

return Citas;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public ECita mostrarCitaCliente(int ID)

{

try

{

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

cita item = new cita();

item = (from c in cn.cita

where c.estado == 1 && c.pacienteID == ID

select c).First();

ECita Obj = new ECita();

Obj.citaID = item.citaID;

Obj.pacienteID = item.pacienteID;

Obj.empleadoID = item.empleadoID;

Obj.odontologoID = item.odontologoID;

Obj.fecha = item.fecha;

Obj.hora = item.hora;

Obj.importe = item.importe;

Obj.descripcion = item.descripcion;

Obj.Odontologo = item.odontologo.nombre + " " + item.odontologo.apellido;

Obj.Paciente = item.paciente.nombre + " " + item.paciente.apellido;

Obj.Empleado = item.empleado.nombre + " " + item.empleado.apellido;

Obj.estado = item.estado;

return Obj;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static int mostrarTotal()

{

try

{

List<cita> citas = new List<cita>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

citas = (from c in cn.cita

where c.estado == 1

orderby c.citaID descending

select c).ToList();

return citas.Count;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

}

}

**Implementación de diente: Clase Diente**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDato;

using CapaEntity;

namespace CapaNegocio

{

public class NDiente

{

public static long save(EDiente Diente)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<diente> dientes = new List<diente>();

diente Obj = new diente();

dientes = (from d in cn.diente

where d.dienteID == Diente.dienteID && d.nombre == Diente.nombre

select d).ToList();

if (dientes.Count > 0)

{

throw new Exception("El Diente Ya Existe");

}

Obj.dienteID = Diente.dienteID;

Obj.nombre = Diente.nombre;

Obj.estado = 1;

if (Obj.nombre == string.Empty)

{

throw new Exception("ingrese el Nombre del diente");

}

cn.diente.Add(Obj);

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.dienteID;

}

else

{

throw new Exception("error al guardar");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static long update(EDiente Diente)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<diente> dientes = new List<diente>();

diente Obj = new diente();

dientes = (from d in cn.diente

where d.nombre == Diente.nombre && d.dienteID == Diente.dienteID

select d).ToList();

if (dientes.Count > 0)

{

throw new Exception("El Diente Ya Existe");

}

Obj = (from p in cn.diente

where p.dienteID == Diente.dienteID

select p).First();

Obj.dienteID = Diente.dienteID;

Obj.nombre = Diente.nombre;

Obj.estado = 1;

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.dienteID;

}

else

{

throw new Exception("No Hubo Ningun Cambio al Editar");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static string delete(EDiente Diente)

{

string rpta = "";

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

diente Obj = new diente();

Obj = cn.diente.Find(Diente.dienteID);

rpta = Obj.estado == 1 ? "OK" : "No se Puede Eliminar el Registro";

Obj.estado = 0;

cn.SaveChanges();

}

catch (Exception ex)

{

rpta = (ex.Message);

}

return rpta;

}

public static List<EDiente> mostrar(string texto, int pag)

{

try

{

List<EDiente> Dientes = new List<EDiente>();

List<diente> dientes = new List<diente>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

dientes = (from d in cn.diente

where d.estado == 1

where d.nombre.Contains(texto)

orderby d.dienteID descending

select d).ToList();

pag = pag \* 10;

var tabla = dientes.Skip(pag).Take(10);

if (dientes.Count < pag)

{

tabla = dientes.Skip(pag).Take(10);

}

foreach (var item in tabla)

{

EDiente Obj = new EDiente();

Obj.dienteID = item.dienteID;

Obj.nombre = item.nombre;

Obj.estado = item.estado;

Obj.vector = item.vector;

Dientes.Add(Obj);

}

return Dientes;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static int mostrarTotal(string texto)

{

try

{

List<diente> dientes = new List<diente>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

dientes = (from d in cn.diente

where d.estado == 1

where d.nombre.Contains(texto)

orderby d.dienteID descending

select d).ToList();

return dientes.Count;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static string mostrarNro(int nro)

{

string rpta = "";

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

diente Obj = new diente();

Obj = cn.diente.Find(nro);

rpta = Obj.nombre;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

return rpta;

}

}

}

**Implementación de tratamiento: Clase Tratamiento**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDato;

using CapaEntity;

namespace CapaNegocio

{

public class NTratamiento

{

public static long save(ETratamiento tratamiento)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<tratamiento> tratamientos = new List<tratamiento>();

tratamiento Obj = new tratamiento();

tratamientos = (from d in cn.tratamiento

where d.nombre == tratamiento.nombre && d.color == tratamiento.color

select d).ToList();

if (tratamientos.Count > 1)

{

throw new Exception("El Tratamiento Ya Existe");

}

Obj.nombre = tratamiento.nombre;

Obj.color = tratamiento.color;

if (Obj.nombre == string.Empty && Obj.color == string.Empty)

{

throw new Exception("Ingrese Nombre y el Color");

}

Obj.tipo = tratamiento.tipo;

Obj.precio = tratamiento.precio;

Obj.estado = 1;

cn.tratamiento.Add(Obj);

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.tratamientoID;

}

else

{

throw new Exception("Error al guardar");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static long update(ETratamiento tratamiento)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<tratamiento> tratamientos = new List<tratamiento>();

tratamiento Obj = new tratamiento();

tratamientos = (from d in cn.tratamiento

where d.nombre == tratamiento.nombre

&& d.color == tratamiento.color

&& d.tipo==tratamiento.tipo

select d).ToList();

if (tratamientos.Count > 1)

{

throw new Exception("El el tratamiento Ya Existe");

}

Obj = (from d in cn.tratamiento

where d.tratamientoID == tratamiento.tratamientoID

select d).First();

Obj.nombre = tratamiento.nombre;

Obj.color = tratamiento.color;

Obj.tipo = tratamiento.tipo;

Obj.precio = tratamiento.precio;

Obj.estado = 1;

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.tratamientoID;

}

else

{

throw new Exception("No Hubo Ningun Cambio al Editar");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static string delete(ETratamiento tratamiento)

{

string rpta = "";

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

tratamiento Obj = new tratamiento();

Obj = cn.tratamiento.Find(tratamiento.tratamientoID);

rpta = Obj.estado == 1 ? "OK" : "No se Puede Eliminar el Registro";

Obj.estado = 0;

cn.SaveChanges();

}

catch (Exception ex)

{

rpta = (ex.Message);

}

return rpta;

}

public static List<ETratamiento> mostrar(string texto, int pag)

{

try

{

List<ETratamiento> ETratamientos = new List<ETratamiento>();

List<tratamiento> tratamientos = new List<tratamiento>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

tratamientos = (from d in cn.tratamiento

where d.estado == 1

where d.nombre.Contains(texto)

orderby d.tratamientoID descending

select d).ToList();

pag = pag \* 10;

var tabla = tratamientos.Skip(pag).Take(10);

if (tratamientos.Count < pag)

{

tabla = tratamientos.Skip(pag).Take(10);

}

foreach (var item in tabla)

{

ETratamiento Obj = new ETratamiento();

Obj.tratamientoID = item.tratamientoID;

Obj.nombre = item.nombre;

Obj.color = item.color;

Obj.tipo = item.tipo;

Obj.precio = item.precio;

Obj.estado = item.estado;

ETratamientos.Add(Obj);

}

return ETratamientos;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static List<ETratamiento> mostrarTipo(string texto,string tipo)

{

try

{

List<ETratamiento> ETratamientos = new List<ETratamiento>();

List<tratamiento> tratamientos = new List<tratamiento>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

tratamientos = (from d in cn.tratamiento

where d.estado == 1

where d.tipo==tipo

where d.nombre.Contains(texto)

select d).ToList();

foreach (var item in tratamientos)

{

ETratamiento Obj = new ETratamiento();

Obj.tratamientoID = item.tratamientoID;

Obj.nombre = item.nombre;

Obj.color = item.color;

Obj.tipo = item.tipo;

Obj.precio = item.precio;

Obj.estado = item.estado;

ETratamientos.Add(Obj);

}

return ETratamientos;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static int mostrarTotal(string texto)

{

try

{

List<tratamiento> tratamientos = new List<tratamiento>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

tratamientos = (from d in cn.tratamiento

where d.estado == 1

where d.nombre.Contains(texto)

select d).ToList();

return tratamientos.Count;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

}

}

**Implementación de parte: Clase Parte**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDato;

using CapaEntity;

namespace CapaNegocio

{

public class NParte

{

public static long save(EParte Parte)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<parte> partes = new List<parte>();

parte Obj = new parte();

partes = (from t in cn.parte

where t.nombre == Parte.nombre

select t).ToList();

if (partes.Count > 0)

{

throw new Exception("El Parte Ya Existe");

}

Obj.nombre = Parte.nombre;

cn.parte.Add(Obj);

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.parteID;

}

else

{

throw new Exception("Error al guardar");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static long update(EParte Parte)

{

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

List<parte> partes = new List<parte>();

parte Obj = new parte();

partes = (from t in cn.parte

where t.nombre == Parte.nombre

select t).ToList();

if (partes.Count > 0)

{

throw new Exception("El parte Ya Existe");

}

Obj = (from t in cn.parte

where t.parteID == Parte.parteID

select t).First();

Obj.nombre = Parte.nombre;

int result = cn.SaveChanges();

if (result > 0)

{

return Obj.parteID;

}

else

{

throw new Exception("No Hubo Ningun Cambio al Editar");

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static string delete(EParte Parte)

{

string rpta = "";

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

parte Obj = new parte();

Obj = cn.parte.Find(Parte.parteID);

cn.SaveChanges();

}

catch (Exception ex)

{

rpta = (ex.Message);

}

return rpta;

}

public static List<EParte> mostrar(string texto)

{

try

{

List<EParte> Partes = new List<EParte>();

List<parte> partes = new List<parte>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

partes = (from t in cn.parte

where t.estado == 1

where t.nombre.Contains(texto)

orderby t.parteID descending

select t).ToList();

foreach (var item in partes)

{

EParte Obj = new EParte();

Obj.parteID = item.parteID;

Obj.nombre = item.nombre;

Obj.estado = item.estado;

Partes.Add(Obj);

}

return Partes;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static string mostrarNombre(int ID)

{

try

{

parte partes = new parte();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

partes =cn.parte.Find(ID);

return partes.nombre;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

}

}

**Implementación de odontograma: Clase Odontograma**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using CapaDato;

using CapaEntity;

namespace CapaNegocio

{

public class NOdontograma

{

public static long save(EOdontograma Odontograma, DataTable dtDetalles, ECita cita,EPaciente paciente, EUsers odontologo, EUsers empleado)

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

var db = cn.Database.BeginTransaction();

try

{

odontograma Obj = new odontograma();

Obj.fechaInicio = Convert.ToDateTime(DateTime.Now.ToLongDateString());

Obj.fechaFinal = Convert.ToDateTime(DateTime.Now.ToLongDateString());

Obj.estado = 1;

Obj.tratamiento = Odontograma.tratamiento;

Obj.montoTotal = Odontograma.montoTotal;

cn.odontograma.Add(Obj);

int result = cn.SaveChanges();

foreach (DataRow row in dtDetalles.Rows)

{

odontograma\_detalle OD = new odontograma\_detalle();

OD.odontogramaID = Obj.odontogramaID;

OD.dienteID = Convert.ToInt32(row["dienteID"].ToString());

OD.diagnosticoID = Convert.ToInt32(row["diagnosticoID"].ToString());

OD.procedimientoID = Convert.ToInt32(row["procedimientoID"].ToString());

OD.parteID = Convert.ToInt32(row["parteID"].ToString());

OD.realizado= Convert.ToString(row["realizado"].ToString());

OD.estado = 1;

cn.odontograma\_detalle.Add(OD);

result = cn.SaveChanges() + result;

}

cita citaR = new cita();

citaR.fecha = Convert.ToDateTime(DateTime.Now.ToLongDateString());

citaR.hora = DateTime.Now.TimeOfDay;

citaR.importe = cita.importe;

citaR.descripcion = cita.descripcion;

citaR.estado = 1;

citaR.tipo = "TRATAMIENTO";

citaR.pacienteID = paciente.pacienteID;

citaR.odontologoID = odontologo.usuarioID;

citaR.empleadoID = empleado.usuarioID;

cn.cita.Add(citaR);

result = cn.SaveChanges() + result;

int contador = 0;

foreach (DataRow row in dtDetalles.Rows)

{

if (Convert.ToString(row["realizado"].ToString()) == "SI")

{

contador++;

cita\_detalle citaD = new cita\_detalle();

citaD.citaID = citaR.citaID;

citaD.odontogramaID = Obj.odontogramaID;

citaD.dienteID = Convert.ToInt32(row["dienteID"].ToString());

citaD.diagnosticoID = Convert.ToInt32(row["diagnosticoID"].ToString());

citaD.procedimientoID = Convert.ToInt32(row["procedimientoID"].ToString());

citaD.parteID = Convert.ToInt32(row["parteID"].ToString());

citaD.realizado = Convert.ToString(row["realizado"].ToString());

citaD.estado = 1;

cn.cita\_detalle.Add(citaD);

result = cn.SaveChanges() + result;

}

}

if (result >0 && contador!=0)

{

db.Commit();

return citaR.citaID;

}

else

{

db.Rollback();

throw new Exception("Error al guardar");

}

}

catch (Exception ex)

{

db.Rollback();

throw new Exception(ex.Message);

}

}

public static long update(EOdontograma Odontograma, DataTable dtDetalles, ECita cita, EPaciente paciente, EUsers odontologo, EUsers empleado)

{

int result =0;

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

var db = cn.Database.BeginTransaction();

try

{

odontograma Obj = new odontograma();

Obj = cn.odontograma.Find(Odontograma.odontogramaID);

Obj.fechaFinal = Convert.ToDateTime(DateTime.Now.ToLongDateString());

Obj.odontogramaID = Odontograma.odontogramaID;

Obj.tratamiento = Odontograma.tratamiento;

Obj.montoTotal = Odontograma.montoTotal;

Obj.estado = 1;

result = cn.SaveChanges();

List<odontograma\_detalle> ODT = new List<odontograma\_detalle>();

ODT = (from od in cn.odontograma\_detalle

where od.odontogramaID == Odontograma.odontogramaID

select od).ToList();

foreach(var item in ODT)

{

odontograma\_detalle ODTF = new odontograma\_detalle();

ODTF = (from dt in cn.odontograma\_detalle

where dt.odontogramaID == Odontograma.odontogramaID &&

dt.dienteID == item.dienteID && dt.procedimientoID == item.procedimientoID &&

dt.diagnosticoID == item.diagnosticoID &&

dt.parteID == item.parteID

select dt).First();

ODTF.estado = 0;

cn.SaveChanges();

}

foreach (DataRow row in dtDetalles.Rows)

{

odontograma\_detalle OD = new odontograma\_detalle();

OD.odontogramaID = Obj.odontogramaID;

OD.dienteID = Convert.ToInt32(row["dienteID"].ToString());

OD.diagnosticoID = Convert.ToInt32(row["diagnosticoID"].ToString());

OD.procedimientoID = Convert.ToInt32(row["procedimientoID"].ToString());

OD.parteID = Convert.ToInt32(row["parteID"].ToString());

List<odontograma\_detalle> ODL = new List<odontograma\_detalle>();

ODL = (from dt in cn.odontograma\_detalle

where dt.odontogramaID == OD.odontogramaID &&

dt.dienteID == OD.dienteID &&

dt.diagnosticoID == OD.diagnosticoID &&

dt.procedimientoID== OD.procedimientoID &&

dt.parteID == OD.parteID

select dt).ToList();

if (ODL.Count>0)

{

OD = (from od in cn.odontograma\_detalle

where od.odontogramaID == Obj.odontogramaID &&

od.diagnosticoID == OD.diagnosticoID &&

od.procedimientoID == OD.procedimientoID &&

od.dienteID == OD.dienteID &&

od.parteID == OD.parteID

select od).First();

OD.realizado = Convert.ToString(row["realizado"].ToString());

OD.estado = 1;

}

else

{

OD.realizado = Convert.ToString(row["realizado"].ToString());

OD.estado = 1;

cn.odontograma\_detalle.Add(OD);

}

result = cn.SaveChanges()+result;

}

cita citaR = new cita();

citaR.fecha = Convert.ToDateTime(DateTime.Now.ToLongDateString());

citaR.hora = DateTime.Now.TimeOfDay;

citaR.importe = cita.importe;

citaR.descripcion = cita.descripcion;

citaR.estado = 1;

citaR.tipo = "TRATAMIENTO";

citaR.pacienteID = paciente.pacienteID;

citaR.odontologoID = odontologo.usuarioID;

citaR.empleadoID = empleado.usuarioID;

cn.cita.Add(citaR);

result = cn.SaveChanges() + result;

int contador = 0;

foreach (DataRow row in dtDetalles.Rows)

{

if (Convert.ToString(row["realizado"].ToString()) == "SI")

{

cita\_detalle citaD = new cita\_detalle();

citaD.citaID = citaR.citaID;

citaD.odontogramaID = Obj.odontogramaID;

citaD.dienteID = Convert.ToInt32(row["dienteID"].ToString());

citaD.diagnosticoID = Convert.ToInt32(row["diagnosticoID"].ToString());

citaD.procedimientoID = Convert.ToInt32(row["procedimientoID"].ToString());

citaD.parteID = Convert.ToInt32(row["parteID"].ToString());

List<cita\_detalle> cd = new List<cita\_detalle>();

cd = (from dt in cn.cita\_detalle

where dt.estado == 1

where dt.odontogramaID == citaD.odontogramaID &&

dt.dienteID == citaD.dienteID &&

dt.diagnosticoID == citaD.diagnosticoID &&

dt.procedimientoID == citaD.procedimientoID &&

dt.parteID == citaD.parteID && dt.realizado == "SI"

select dt).ToList();

if (cd.Count == 0)

{

contador++;

citaD.realizado = Convert.ToString(row["realizado"].ToString());

citaD.estado = 1;

cn.cita\_detalle.Add(citaD);

result = cn.SaveChanges() + result;

}

}

}

if (result > 0 && contador!=0)

{

db.Commit();

return citaR.citaID;

}

else

{

db.Rollback();

throw new Exception("Error al guardar");

}

}

catch (Exception ex)

{

db.Rollback();

throw new Exception(ex.Message);

}

}

public static long updateCita(EOdontograma Odontograma, DataTable dtDetalles, ECita cita, EPaciente paciente, EUsers odontologo, EUsers empleado)

{

int result = 0;

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

var db = cn.Database.BeginTransaction();

try

{

odontograma Obj = new odontograma();

Obj = cn.odontograma.Find(Odontograma.odontogramaID);

Obj.fechaFinal = Convert.ToDateTime(DateTime.Now.ToLongDateString());

Obj.odontogramaID = Odontograma.odontogramaID;

Obj.tratamiento = Odontograma.tratamiento;

Obj.montoTotal = Odontograma.montoTotal;

Obj.estado = 1;

result = cn.SaveChanges();

List<odontograma\_detalle> ODT = new List<odontograma\_detalle>();

ODT = (from od in cn.odontograma\_detalle

where od.odontogramaID == Odontograma.odontogramaID

select od).ToList();

foreach (var item in ODT)

{

odontograma\_detalle ODTF = new odontograma\_detalle();

ODTF = (from dt in cn.odontograma\_detalle

where dt.odontogramaID == Odontograma.odontogramaID &&

dt.dienteID == item.dienteID && dt.procedimientoID == item.procedimientoID &&

dt.diagnosticoID == item.diagnosticoID &&

dt.parteID == item.parteID

select dt).First();

ODTF.estado = 0;

cn.SaveChanges();

}

foreach (DataRow row in dtDetalles.Rows)

{

odontograma\_detalle OD = new odontograma\_detalle();

OD.odontogramaID = Obj.odontogramaID;

OD.dienteID = Convert.ToInt32(row["dienteID"].ToString());

OD.diagnosticoID = Convert.ToInt32(row["diagnosticoID"].ToString());

OD.procedimientoID = Convert.ToInt32(row["procedimientoID"].ToString());

OD.parteID = Convert.ToInt32(row["parteID"].ToString());

List<odontograma\_detalle> ODL = new List<odontograma\_detalle>();

ODL = (from dt in cn.odontograma\_detalle

where dt.odontogramaID == OD.odontogramaID &&

dt.dienteID == OD.dienteID &&

dt.diagnosticoID == OD.diagnosticoID &&

dt.procedimientoID == OD.procedimientoID &&

dt.parteID == OD.parteID

select dt).ToList();

if (ODL.Count > 0)

{

OD = (from od in cn.odontograma\_detalle

where od.odontogramaID == Obj.odontogramaID &&

od.diagnosticoID == OD.diagnosticoID &&

od.procedimientoID == OD.procedimientoID &&

od.dienteID == OD.dienteID &&

od.parteID == OD.parteID

select od).First();

OD.realizado = Convert.ToString(row["realizado"].ToString());

OD.estado = 1;

}

else

{

OD.realizado = Convert.ToString(row["realizado"].ToString());

OD.estado = 1;

cn.odontograma\_detalle.Add(OD);

}

result = cn.SaveChanges() + result;

}

cita citaR = new cita();

citaR = cn.cita.Find(cita.citaID);

citaR.fecha = Convert.ToDateTime(DateTime.Now.ToLongDateString());

citaR.hora = DateTime.Now.TimeOfDay;

citaR.importe = cita.importe;

citaR.descripcion = cita.descripcion;

citaR.estado = 1;

citaR.tipo = "TRATAMIENTO";

citaR.pacienteID = paciente.pacienteID;

citaR.odontologoID = odontologo.usuarioID;

citaR.empleadoID = empleado.usuarioID;

result = cn.SaveChanges() + result;

List<cita\_detalle> CDT = new List<cita\_detalle>();

CDT = (from cd in cn.cita\_detalle

where cd.citaID == citaR.citaID

select cd).ToList();

foreach (var item in CDT)

{

cita\_detalle CDTF = new cita\_detalle();

CDTF = (from dt in cn.cita\_detalle

where dt.odontogramaID == Odontograma.odontogramaID &&

dt.dienteID == item.dienteID && dt.procedimientoID == item.procedimientoID &&

dt.diagnosticoID == item.diagnosticoID &&

dt.parteID == item.parteID && dt.citaID==citaR.citaID

select dt).First();

CDTF.estado = 0;

cn.SaveChanges();

}

int contador = 0;

foreach (DataRow row in dtDetalles.Rows)

{

if (Convert.ToString(row["realizado"].ToString()) == "SI")

{

cita\_detalle citaD = new cita\_detalle();

citaD.citaID = citaR.citaID;

citaD.odontogramaID = Obj.odontogramaID;

citaD.dienteID = Convert.ToInt32(row["dienteID"].ToString());

citaD.diagnosticoID = Convert.ToInt32(row["diagnosticoID"].ToString());

citaD.procedimientoID = Convert.ToInt32(row["procedimientoID"].ToString());

citaD.parteID = Convert.ToInt32(row["parteID"].ToString());

List<cita\_detalle> cd = new List<cita\_detalle>();

cd = (from dt in cn.cita\_detalle

where dt.estado == 1

where dt.odontogramaID == citaD.odontogramaID &&

dt.dienteID == citaD.dienteID &&

dt.diagnosticoID == citaD.diagnosticoID &&

dt.procedimientoID == citaD.procedimientoID &&

dt.parteID == citaD.parteID &&

dt.citaID==citaR.citaID

select dt).ToList();

if (cd.Count > 0)

{

contador++;

citaD = (from od in cn.cita\_detalle

where od.odontogramaID == Obj.odontogramaID &&

od.diagnosticoID == citaD.diagnosticoID &&

od.procedimientoID == citaD.procedimientoID &&

od.dienteID == citaD.dienteID &&

od.parteID == citaD.parteID &&

od.citaID== citaR.citaID

select od).First();

citaD.realizado = Convert.ToString(row["realizado"].ToString());

citaD.estado = 1;

result = cn.SaveChanges() + result;

}

else

{

List<cita\_detalle> cdt = new List<cita\_detalle>();

cdt = (from dt in cn.cita\_detalle

where dt.estado == 1

where dt.odontogramaID == citaD.odontogramaID &&

dt.dienteID == citaD.dienteID &&

dt.diagnosticoID == citaD.diagnosticoID &&

dt.procedimientoID == citaD.procedimientoID &&

dt.parteID == citaD.parteID && dt.realizado=="SI" && dt.estado==1

select dt).ToList();

if (cdt.Count == 0)

{

contador++;

citaD.realizado = Convert.ToString(row["realizado"].ToString());

citaD.estado = 1;

cn.cita\_detalle.Add(citaD);

result = cn.SaveChanges() + result;

}

}

}

}

if (result > 0 && contador != 0)

{

db.Commit();

return citaR.citaID;

}

else

{

db.Rollback();

throw new Exception("Error al guardar");

}

}

catch (Exception ex)

{

db.Rollback();

throw new Exception(ex.Message);

}

}

public static string delete(EOdontograma Odontograma)

{

string rpta = "";

try

{

CapaDato.dbodontogramaEntity cn = new dbodontogramaEntity();

odontograma Obj = new odontograma();

Obj = cn.odontograma.Find(Odontograma.odontogramaID);

rpta = Obj.estado == 1 ? "OK" : "No se Puede Eliminar el Registro";

Obj.estado = 0;

cn.SaveChanges();

}

catch (Exception ex)

{

rpta = (ex.Message);

}

return rpta;

}

public static List<EOdontograma> mostrar(string nombre,string apellido,int pag)

{

try

{

List<EOdontograma> O = new List<EOdontograma>();

List<odontograma> OD = new List<odontograma>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

OD = (from o in cn.odontograma

join od in cn.odontograma\_detalle

on o.odontogramaID equals od.odontogramaID

join cd in cn.cita\_detalle

on od.odontogramaID equals cd.odontogramaID

join c in cn.cita

on cd.citaID equals c.citaID

join pc in cn.paciente on c.pacienteID equals pc.pacienteID

where o.estado == 1

where pc.nombre.Contains(nombre)

where pc.apellido.Contains(apellido)

orderby o.odontogramaID descending

select o).Distinct().ToList();

pag = pag \* 10;

var tabla = OD.Skip(pag).Take(10);

if (OD.Count < pag)

{

tabla = OD.Skip(pag).Take(10);

}

foreach (var item in tabla)

{

EOdontograma Obj = new EOdontograma();

Obj.odontogramaID = item.odontogramaID;

Obj.fechaInicio = item.fechaInicio;

Obj.fechaFinal = item.fechaFinal;

Obj.tratamiento = item.tratamiento;

Obj.montoTotal = item.montoTotal;

Obj.estado = item.estado;

O.Add(Obj);

}

return O;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static int mostrarTotal(string nombre, string apellido)

{

try

{

List<odontograma> OD = new List<odontograma>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

OD = (from o in cn.odontograma

join od in cn.odontograma\_detalle

on o.odontogramaID equals od.odontogramaID

join cd in cn.cita\_detalle

on od.odontogramaID equals cd.odontogramaID

join c in cn.cita

on cd.citaID equals c.citaID

join pc in cn.paciente on c.pacienteID equals pc.pacienteID

where o.estado == 1

where pc.nombre.Contains(nombre)

where pc.apellido.Contains(apellido)

orderby o.odontogramaID descending

select o).Distinct().ToList();

return OD.Count;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static List<EOdontograma> mostrarCita(int ID)

{

try

{

List<EOdontograma> Odontogramas = new List<EOdontograma>();

List<odontograma> odontogramas = new List<odontograma>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

odontogramas = (from o in cn.odontograma

join ct in cn.cita\_detalle on o.odontogramaID equals ct.odontogramaID

where o.estado == 1 && ct.cita.citaID == ID

select o).ToList();

foreach (var item in odontogramas)

{

EOdontograma Obj = new EOdontograma();

Obj.odontogramaID = item.odontogramaID;

Obj.fechaInicio = item.fechaInicio;

Obj.fechaFinal = item.fechaFinal;

Obj.estado = item.estado;

Obj.tratamiento = item.tratamiento;

Obj.montoTotal = item.montoTotal;

Obj.estado = item.estado;

Odontogramas.Add(Obj);

}

return Odontogramas;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static int mostrarCitaID(int ID)

{

try

{

odontograma odontogramas = new odontograma();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

odontogramas = (from o in cn.odontograma

join ct in cn.cita\_detalle on o.odontogramaID equals ct.odontogramaID

where o.estado == 1 && ct.cita.citaID == ID

select o).First();

return odontogramas.odontogramaID;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public EOdontograma mostrarID(int ID)

{

try

{

EOdontograma Obj = new EOdontograma();

odontograma item = new odontograma();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

item = (from o in cn.odontograma

where o.estado == 1 && o.odontogramaID==ID

select o).First();

Obj.odontogramaID = item.odontogramaID;

Obj.fechaInicio = item.fechaInicio;

Obj.fechaFinal = item.fechaFinal;

Obj.estado = item.estado;

Obj.tratamiento = item.tratamiento;

Obj.montoTotal = item.montoTotal;

Obj.estado = item.estado;

return Obj;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public static List<EOdontograma> mostrarCliente(int pacienteID)

{

try

{

List<EOdontograma> O = new List<EOdontograma>();

List<odontograma> OD = new List<odontograma>();

using (dbodontogramaEntity cn = new dbodontogramaEntity())

{

OD = (from o in cn.odontograma join od in cn.odontograma\_detalle

on o.odontogramaID equals od.odontogramaID

join cd in cn.cita\_detalle

on od.odontogramaID equals cd.odontogramaID

join c in cn.cita

on cd.citaID equals c.citaID

join pc in cn.paciente on c.pacienteID equals pc.pacienteID

where o.estado == 1

where pc.pacienteID == pacienteID

select o).Distinct().ToList();

foreach (var item in OD)

{

EOdontograma Obj = new EOdontograma();

Obj.odontogramaID = item.odontogramaID;

Obj.fechaInicio = item.fechaInicio;

Obj.fechaFinal = item.fechaFinal;

Obj.tratamiento = item.tratamiento;

Obj.montoTotal = item.montoTotal;

Obj.estado = item.estado;

O.Add(Obj);

}

return O;

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

}

}

## **3.5 Pruebas**

### **Pruebas del software**

**Pruebas de software Paciente**

|  |
| --- |
| TEST CASE |
| Identificador: **001**  Título: **Probar el registro de los pacientes correctamente.**  Descripción: **Seleccionar el botón Guardar y el sistema guarda los datos, así como también debe auto-generar el identificador del paciente.** |
| Pasos |
| 1. **Ejecutar la Aplicación** 2. **Ingresar al menú de los pacientes y registrar nuevo paciente.** 3. **Digite los datos del paciente.** 4. **Guardar datos digitados del paciente.** 5. **Modificar y eliminar los datos del paciente.** 6. **Consultar los datos de a través de las búsquedas por parámetros.** |
| Resultado Esperado |
| **Deberá verificar si guarda, modifica, elimina y realiza la búsqueda del paciente.** |
| Comentarios/Observaciones |
| **Ninguna** |

**Pruebas de software Empleado**

|  |
| --- |
| TEST CASE |
| Identificador: **002**  Título: **Probar el registro de los empleados correctamente.**  Descripción: **Seleccionar el botón Guardar y el sistema guarda los datos, así como también debe auto-generar el identificador de los empleados.** |
| Pasos |
| 1. **Ejecutar la Aplicación** 2. **Ingresar al menú principal, luego la opción empleado y registrar nuevo empleado.** 3. **Digite los datos del empleado.** 4. **Guardar datos digitados del empleado.** 5. **Modificar y eliminar los datos del empleado.** 6. **Consultar los datos de a través de las búsquedas por parámetros.** |
| Resultado Esperado |
| **Deberá verificar si guarda, modifica, elimina y realiza la búsqueda del empleado.** |
| Comentarios/Observaciones |
| **Ninguna** |

**Pruebas de software Tratamiento**

|  |
| --- |
| TEST CASE |
| Identificador: **003**  Título: **Probar el registro de los tratamientos correctamente.**  Descripción: **Seleccionar el botón Guardar y el sistema guarda los datos, así como también debe auto-generar el identificador de los tratamientos.** |
| Pasos |
| 1. **Ejecutar la Aplicación** 2. **Ingresar al menú principal, luego la opción tratamiento y registrar nuevo.** 3. **Digite los datos requerido del tratamiento.** 4. **Guardar datos digitados del tratamiento.** 5. **Modificar y eliminar los datos del tratamiento.** 6. **Consultar los datos de a través de las búsquedas por parámetros.** |
| Resultado Esperado |
| **Deberá verificar si guarda, modifica, elimina y realiza la búsqueda de los tratamientos.** |
| Comentarios/Observaciones |
| **Ninguna** |

**Pruebas de software Diente**

|  |
| --- |
| TEST CASE |
| Identificador: **004**  Título: **Probar el registro de los dientes correctamente.**  Descripción: **Seleccionar el botón Guardar y el sistema guarda los datos, el identificador será proporcionado por el administrador según a su numeración científica de los dientes.** |
| Pasos |
| 1. **Ejecutar la Aplicación** 2. **Ingresar al menú principal, luego la opción dientes y registrar nuevo.** 3. **Digite los datos requerido del diente.** 4. **Guardar datos digitados del diente.** 5. **Modificar y eliminar los datos del diente.** 6. **Consultar los datos de a través de las búsquedas por parámetros.** |
| Resultado Esperado |
| **Deberá verificar si guarda, modifica, elimina y realiza la búsqueda de los dientes.** |
| Comentarios/Observaciones |
| **Ninguna** |

**Pruebas de software Parte de los Dientes**

|  |
| --- |
| TEST CASE |
| Identificador: **005**  Título: **Probar el registro de las partes de los dientes correctamente.**  Descripción: **Seleccionar el botón Guardar y el sistema guarda los datos, así como también debe auto-generar el identificador de las partes de los dientes.** |
| Pasos |
| 1. **Ejecutar la Aplicación** 2. **Ingresar al menú principal, luego la opción parte y registrar nuevo.** 3. **Digite los datos requerido.** 4. **Guardar datos digitados.** 5. **Modificar y eliminar los datos de las partes de los dientes.** 6. **Consultar los datos de a través de las búsquedas por parámetros.** |
| Resultado Esperado |
| **Deberá verificar si guarda, modifica, elimina y realiza la búsqueda de las partes.** |
| Comentarios/Observaciones |
| **Ninguna** |

**Pruebas de Software Odontograma**

|  |
| --- |
| TEST CASE |
| Identificador: **006**  Título: **Probar el registro del Odontograma correctamente.**  Descripción: **Seleccionar el botón Guardar y el sistema guarda los datos, así como también debe auto-generar el identificador del Odontograma.** |
| Pasos |
| 1. **Ejecutar la Aplicación** 2. **Ingresar al menú principal, luego la opción Odontograma y registrar nuevo.** 3. **Digite los datos requerido.** 4. **Seleccione Cliente, odontólogo, tratamiento, dientes.** 5. **Guardar datos digitados.** 6. **Modificar y eliminar los datos del odontograma.** 7. **Consultar los datos de a través de las búsquedas por parámetros.** |
| Resultado Esperado |
| **Deberá verificar si guarda, modifica, elimina y realiza la búsqueda del odontograma.** |
| Comentarios/Observaciones |
| **Ninguna** |

**Pruebas de Software Atencion**

|  |
| --- |
| TEST CASE |
| Identificador: **007**  Título: **Probar el registro de la cita correctamente.**  Descripción: **Seleccionar el botón Guardar y el sistema guarda los datos, así como también debe auto-generar el identificador de la Atencion.** |
| Pasos |
| 1. **Ejecutar la Aplicación** 2. **Ingresar al menú principal, luego la opción Atencion y registrar nuevo.** 3. **Digite los datos requerido.** 4. **Seleccione Cliente, odontólogo.** 5. **Guardar datos digitados.** 6. **Modificar y eliminar los datos de la cita.** 7. **Consultar los datos de a través de las búsquedas por parámetros.** |
| Resultado Esperado |
| **Deberá verificar si guarda, modifica, elimina y realiza la búsqueda de la cita.** |
| Comentarios/Observaciones |
| **Ninguna** |

## **3.6 Recomendaciones**

Implementar el sistema desarrollado en la sección de Atencion, para que los usuarios puedan desempeñar sus funciones en un ambiente de trabajo automatizado y organizado.

Elaborar un plan de adiestramiento dirigido al personal de la clinica que hará uso del sistema, con el fin de aprender a manejarlo y poder sacarle el máximo provecho a cada una de sus funcionalidades.

## **3.7 Conclusiones**

Se observa que la utilización de la metodología ágil XP en el presente proyecto ha proporcionado un buen resultado, ya que, a diferencia de las convencionales, estas son las flexibles antes los cambios y requerimientos inesperados.

Según al antecedente del presente proyecto “Sistema Informático de Odontograma para la clínica Odonto Viera”, se confirma que el implementar un sistema permite automatizar, reducir y mejorar los tiempos en los procesos sin perder información importante para la empresa.

## **3.8 Referencia bibliográfica**

<http://www.odontologia-online.com/estudieantes/trabajos/gnb/gnb02/gnb02.html>

<https://www.sqlshack.com/whats-new-in-sql-server-2017/>

<https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017>

<https://www.campusmvp.es/recursos/post/que-hay-de-nuevo-en-visual-studio-2019.aspx>

<https://docs.microsoft.com/en-us/visualstudio/ide/whats-new-visual-studio-2019?view=vs-2019>

<http://biblioteca.utec.edu.sv/siab/virtual/tesis/>

<https://www.colgate.com/es-py/oral-health/basics/mouth-and-teeth-anatomy/how-many-teeth-do-we-have-0113>

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview>

## **3.9 Bibligrafia**

Dos Santos, J. (1995). Diagnóstico y tratamiento de la sintomatología craneomandibular.

Actualidades Médico Odontológicas Latinoamérica, Venezuela.

Batres, E. (1982). Prácticas ergonómicas en odontología. Continental, México.

Breñón, E. N. (1982). Odontología legal y práctica forense. Purinzón, Buenos Aires.

Abreu, O., Pousa, M.S. y González, E. (2004). Relación entre la postura de la cabeza y las mordidas cruzadas posteriores unilaterales. Revista latinoamericana de ortodoncia y odontopediatría [en línea].

Accesible en: <http://www.odontologia-online.com/estudieantes/trabajos/gnb/gnb02/gnb02.html>

Moya Pueyo, V. et al. (1994). Odontología legal y forense. Masson, España.

Roberts. (1998). Analgesia local en odontología. El Manual Moderno, México, 2a. ed.

## **3.10 Anexos**

