AITEC | INSTITUTO SUPERIOR UNIVERSITARIO ALMIRANTE ILLINGWORTH

Desarrollo de Software

GUIA DE ESTUDIO

Sistemas Cliente-Servidor con .net

El presente material fue evaluado por pares académicos experimentados en el área.

Edición: primera, agosto 2024
Autor: Ing. Ernesto Robles Peñaherrera

¿Cómo citar esta obra?

Robles, E. (2024). *Guía de estudio: Sistemas Cliente-Servidor con .net.* Instituto Superior Tecnológico Almirante Illingworth. https://investiga.aitec.edu.ec/guias-de-estudio-2024/

ISBN: 978-9942-7122-4-0
Edición con fines académicos no lucrativos.
Distribuido digitalmente y elaborado en Ecuador
Director del equipo editorial: MSc. Rolando Álvarez.
Coordinadora editorial: Mtr. Jennifer Castillo Ortiz.

Sello Editorial:
Instituto Superior Tecnológico Almirante Illingworth
Departamento de Investigación e Innovación
Av. José Gómez Gault, Kilómetro 8,5 vía a Daule
Guayaquil-Ecuador
Telf: (+593) 4 370 3300 Extensión 118
https://aitec.edu.ec/

Tabla de contenido

PRESENTACIÓN

Al momento de desarrollar un software, algunos programadores se basan en una arquitectura de software, los cuales facilitan el diseño y la construcción del mismo (Casanovas, 2004), estos permiten al conjunto de desarrolladores seguir una misma línea de trabajo, enmarcando las formas de cómo se comportará el sistema en su funcionamiento e interacción con los usuarios.

Para los fines de este módulo se trabajará en una programación por tres capas: la capa de presentación o frontera, la capa de lógica de negocio o control, y la capa de datos; las cuales permite "avanzar de manera más segura en el desarrollo, al ser dividida la aplicación general en varios módulos y capas que pueden ser tratados de manera independiente y hasta en forma paralela" (Valle & Granados, 2007), esto conlleva a que si existiesen cambios dentro del sistema, dichas actualizaciones se las realizará en menor tiempo y menos engorroso.

Al finalizar este módulo, usted será capaz de diseñar aplicación en arquitecturas de software cuyo modelo cliente-servidor será abordado bajo un enfoque de programación en tres capas; visto en tres unidades:

- Modelos y Arquitecturas de diseño de software, es una introducción a la forma de como delimitar y organizar la programación
- Programación por capas, aplicación de dicha estructura y funciones, permite la división y jerarquización de los accesos de cada nivel
- Reportes y ensamblado, una vez terminado el desarrollo de la aplicación de realizaran los respectivos reportes y ensamblado de la aplicación.

MATERIA PRERREQUISITO

No aplica

MATERIA CORREQUISITO

No aplica

RESULTADOS DE APRENDIZAJE.

- Infiere en los principios de Arquitectura de Software
- Diferencia entre los elementos que sustentan un cliente y un servidor
- Aplicación y desarrollo de sesiones, aplicando el uso de la herramienta de programación visual basic .net
- Desarrolla aplicación aplicando la metodología de Modelo por Capas

UNIDAD 1: ARQUITECTURA DE SOFTWARE

1.1. Concepto

Una arquitectura de software permite la organización en la construcción de un software, alineándose bajo una interacción de distintos componentes y que estos puedan ser aplicados y modificados de forma eficaz y rápida; el estándar IEEE Std 1471-2000 indica que "La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución" (Reynoso, 2004), por lo tanto, la misma provee de una visión de cómo se irán desarrollando el sistema.

1.2. Diseño

Para el diseño de la arquitectura del software podemos partir de:

- ✓ Escenarios que sean significativos para el proyecto
- ✓ Plataforma en la que actuará el software
  - o Sistema operativo
  - o SMBD
  - o Otros sistemas existentes
  - o Etc.
- ✓ A través de otras arquitecturas previas o patrones de diseño previamente definidos por algún estándar

Cada uno de estos aspectos pueden ser descritos por diagramas UML, tales como:

- ✓ Casos de uso
- ✓ De clases y componentes
- ✓ Subsistemas
- ✓ Colaboraciones

1.3. Tipos de arquitecturas.

Las arquitecturas de software han evolucionado en el tiempo, mejorando su contexto de interacción entre los componentes que conforman el patrón del diseño del software.

Figura 1. *Evolución de las arquitecturas de software*

Nota: Elaboración propia a partir de *(Universidad Simón Bolivar, 2007)*

Existen algunos tipos de arquitectura, dependiendo el tipo de patrón que sugiere la arquitectura tenemos:

- ✓ Cliente-Servidor
- ✓ Blackboard
- ✓ Modelos entre capas
- ✓ Interprete
- ✓ Orientado a servicios

Cada uno de los modelos expuestos conllevan a un desarrollo de software, que permite encontrara fallos de forma rápida y que los cambios no requieran demasiado para su aplicación, (Casanovas, 2004) define que dichas arquitecturas tienen la responsabilidad de:

- ✓ Definir los módulos principales
- ✓ Definir las responsabilidades que tendrá cada uno de estos módulos
- ✓ Definir la interacción que existirá entre dichos módulos:
- ✓ Control y flujo de datos
- ✓ Secuenciación de la información
- ✓ Protocolos de interacción y comunicación
- ✓ Ubicación en el hardware

Cada uno de las responsabilidades expuesta conlleva a una visión de desarrollo con distintos tipos de modelos.

1.4. Modelos de desarrollo de software

El modelo de desarrollo de software será una representación lógica de la abstracción de los proceso que puedan tener ya sea un negocio desde cualquier enfoque empresarial pequeña, mediana, gran empresa o de economías popular y solidaria; la idea fundamental es comprender el o los procesos que se vayan a automatizar; por ejemplo cuando se construye un edificio o cualquier otra obra de construcción requiere de un modelo previo a escala para tener una guía de cómo se llevara a cabo la obra, corregir errores o llevar u orden especifico al momento de realizar dicha construcción.

Los mimos están apoyados según:

- Modelos con diferentes niveles de abstracción, escritos en lenguajes bien definidos.
- Definiciones de cómo un modelo se transforma en otro modelo más específico.
- Herramientas de software que den soporte a la creación de modelos y su posterior transformación.

Por lo antes expuesto, un modelo de desarrollo de software pude considerar distinto modelos en su desarrollo ya sea estos desde la abstracción, requerimientos, representación, procesos, y en algunos caso generar otro modelo partiendo un anterior.

1.4.1. Tipos de Modelos de desarrollo de software

Para determinar el tipo de modelo, en algunas ocasiones podemos ayudarnos contestando las siguientes interrogantes (Pons y otros, 2010):

- ¿En qué parte del proceso de desarrollo de software es usado el modelo?
- ¿El modelo es abstracto o es detallado?
- ¿Qué sistema es descrito por el modelo?
- ¿Es un modelo del negocio o es un modelo de software?
- ¿Qué aspectos del sistema son descritos por el modelo?
- ¿Es un modelo de la estructura o es un modelo del comportamiento?
- ¿Es un modelo orientado a una tecnología específica o es un modelo independiente de la tecnología?

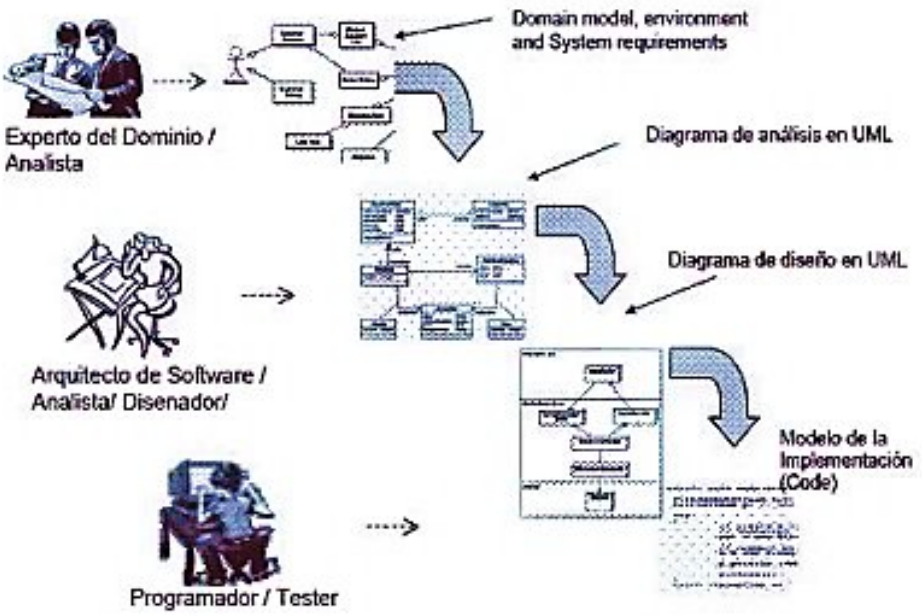
A partir de su respuesta podemos tener:

Figura 1. *Tipos Modelos de desarrollo de software*

Nota: Elaboración propia a partir de (Pons y otros, 2010)

En cada uno delo modelos expuestos en la Figura 2, podemos indicar que comportamiento estará dado en cómo se aborde cada aspecto del modelo, por ejemplo en un modelo en proceso de desarrollo en primera instancia se establecen los requisitos esenciales del sistema indistintamente de cómo se vaya a implementar; luego, indica si será el sistema orientado a objetos, por componentes u otro, por lo tanto, se conocerá que sistema operativo o plataforma se utilizará para el desarrollo; así como lo muestra la Figura 3 .

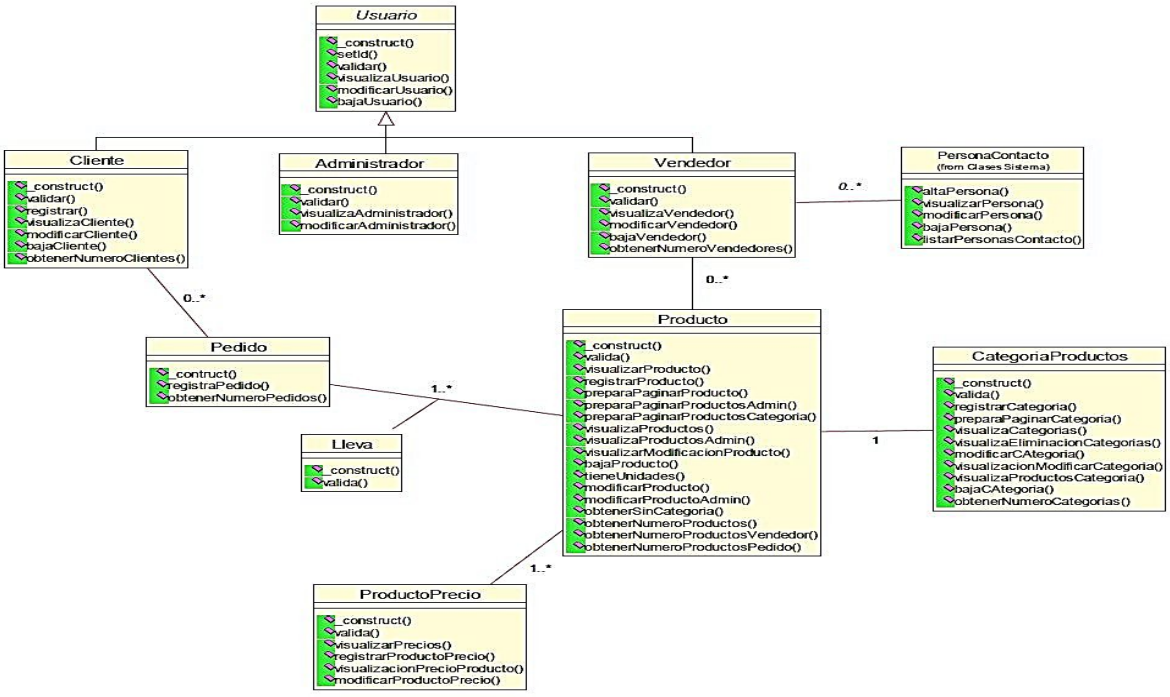
Figura 2. *Diferentes modelos a lo largo del proceso*



Nota: Elaboración (Pons y otros, 2010)

Por otro lado, en los modelos abstractos, podemos utilizar diagramas UML para su visualización ejemplo tener un jerarquía de clases por un diagrama de clases y de esta manera distinguir la implementación de cada proceso abstraído, ver Figura 4

Figura 3. *Jerarquía de Clases de un sistema dado*

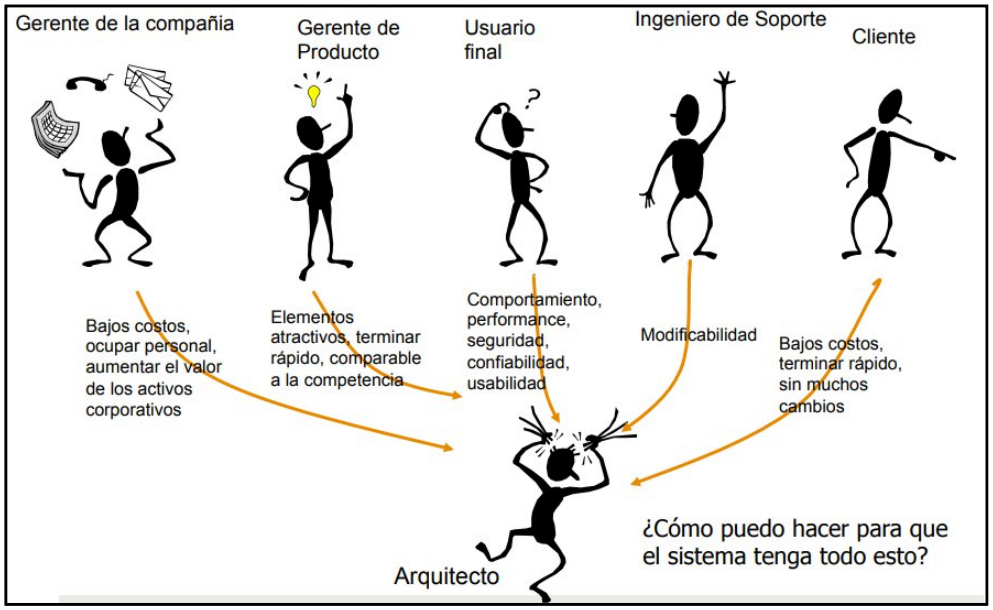


Nota: Elaboración propia

Si observamos la figura UML de la Figura 4, la misma se la puede desarrollar en herramienta de desarrollo que puede ser pago como por ejemplo UModel https://www.altova.com/es/umodel/download que permite interactuar con algunos lenguajes de programación

Es importante destacar que el modelo que se escoja para el desarrollo del software, dependerá de las necesidades del cliente; el cual, para una mayor legibilidad, puede expresarse con diagramas UML de distintos índoles dado que, "UML es una notación de modelado visual, que utiliza diagramas para mostrar distintos aspectos de un sistema" (Fontela, 2011), de ésta forma expresaremos cada aspecto del desarrollo de nuestro software según sea el caso del tipo modelo de software que vamos a aplicar.

Figura 4. *Influencia en los interesados del desarrollo del software.*



Nota: Elaboración propia a partir de *(Barraza, Modelado y Diseño de Arquitectura de Software, 2013)*

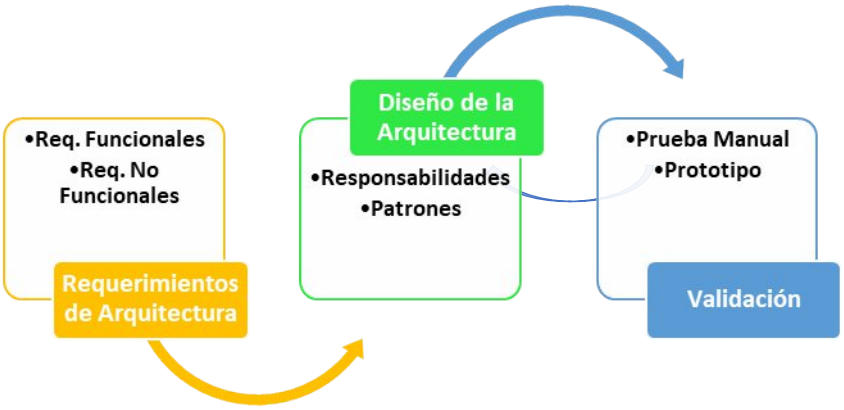
Al momento de establecer el alcance de la arquitectura de software, el mismo estará influenciado por algunos actores que inciden en la composición y elementos del diseño; así como también, el modelo a elegir en el desarrollo del software, ver Figura 2. Por eso es importante minimizar el conflicto que entre todos los interesados, involucrando a todos y cada uno, con una participación activa, comprendiendo:

- Las restricciones reales del sistema
- Prioridades del sistema
- Decisiones de alcance y entrega

1.5. Proceso de modelado de Arquitectura de Software

Existen métodos y guías para la definición de la arquitectura, muchos de los cuales se focalizan en los requisitos funcionales u otros elementos al momento de realizar un levantamiento de información, es posible crear una arquitectura basada en las necesidades de atributos de calidad

Figura 5. *Proceso de modelado de arquitectura de software*



Nota: Elaboración propia

1.5.1. Requerimientos de la arquitectura de software

De la Figura 6, se debe determinar los requerimientos del software, posterior se infiere en las responsabilidades de los componentes del sistema y por ultimo determinar si existe algún cambio por realizar u optimizar en la arquitectura del software.

Caso de estudio tomado de la tesis del Tnlgo. Jeffrey Palacios (Jeffrey, 2018) :

Consultorio de Fisioterapia "FisioSport"

El consultorio de Fisioterapia "FisioSport" tiene como fin ofrecer servicios de tratamiento y recuperación de pacientes que han sufrido de lesiones, esguinces, contracturas y fracturas a través de la fisioterapia, el registro de estos pacientes se lo redacta en un cuaderno, el cual provoca que se desgaste por la manipulación y da lugar a que los datos anotados se pierdan, así mismo se lleva el proceso de seguimiento de pacientes exponiendo que la información se pierda con el desgaste de dicho cuaderno, al ser muchos pacientes este problema crece, si tomamos en cuenta que existen pacientes que requieren de medicación esto conlleva a que se trunque el proceso de recuperación de un paciente. O lo que es más grave que se somete a la repetición de algún medicamento lo que sería grave en la evolución de un paciente, esto hace visible que no haya una forma organizada de almacenar la información de cada paciente, esto hace que el almacenamiento de la información no sea eficiente.

El consultorio al tener varios años en funcionamiento, resulta que la cantidad de papeles donde se registran los datos de los pacientes, en volumen es muy extensa, el almacenamiento de dichos documentos ha dado lugar a que se pierdan, se deterioren y no se logra obtener un estimado de historias clínicas de los pacientes, lo que hace que el almacenamiento no sea una tarea que posee al momento este consultorio.

Al finalizar el día de trabajo, para obtener la cantidad de pacientes atendidos, se procede a realizar el conteo manual y es ahí cuando al observar los registros se encuentran

inconsistencias de legibilidad o anotados en desorden, lo que indica una desorganización de cómo se lleva este proceso tan importante en un consultorio de fisioterapia.

Para dar inicio a nuestro proceso de modelado estableceremos los requerimientos funcionales y no funcionales del software; por lo tanto:

¿Qué sería un requerimiento de software?

Según el glosario de ingenieros eléctricos y electrónicos como "Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo" (Hyderabad, 1998); es decir, del caso expuesto estableceremos los aspectos que el usuario requiere y como el software contribuirá a la resolución de dichas necesidades.

Antes de establecer el requerimiento funcional y no funcional del caso de estudio, es importante determinar los elementos que principales que conlleva a un problema por resolver, tales como:

- No existe un seguimiento correcto del tratamiento que se le da al paciente
- Verificar en cada consulta la medicación que le ha brindado al paciente
- Manejo engorroso del historial clínico del paciente
- Estadístico de atención realizadas en el consultorio en el día

Por lo antes expuesto, ahora procederemos a determinar los requerimientos funcionales de la propuesta:

- Se registrarán a los pacientes nuevos; y a su vez con su respectivo historial clínico
- Se registrarán las recetas con su respectivas posología del paciente
- Se asignarán citas posteriores en caso de que lo requiera el paciente
- El sistema permitirá solo a usuarios autorizados en sus acceso
- …

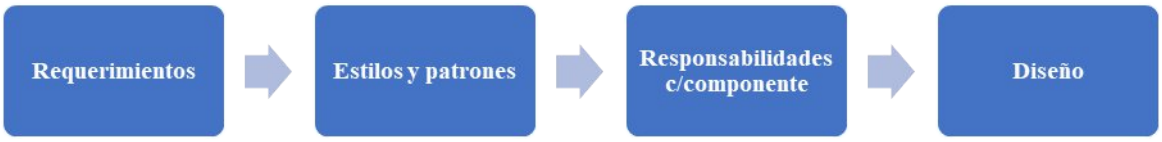
Ahora se redactarán los requerimientos no funcionales del caso de estudio:

- El sistema se implementará sobre el equipo actualmente utilizado
- Su gestión de datos será con el framework developer 4.5.2
- Se requieren como proveedor de base de datos al conector .net de mysql 6.10.8
- Se respaldarán los datos, por automatización de tareas
- …

1.5.2. Diseño de la arquitectura de software

Una vez que se conocen los requerimientos funcionales y no funcionales del contexto que se vaya a automatizar, ahora tendrán que definirse los componentes que comprenderán la arquitectura del software:

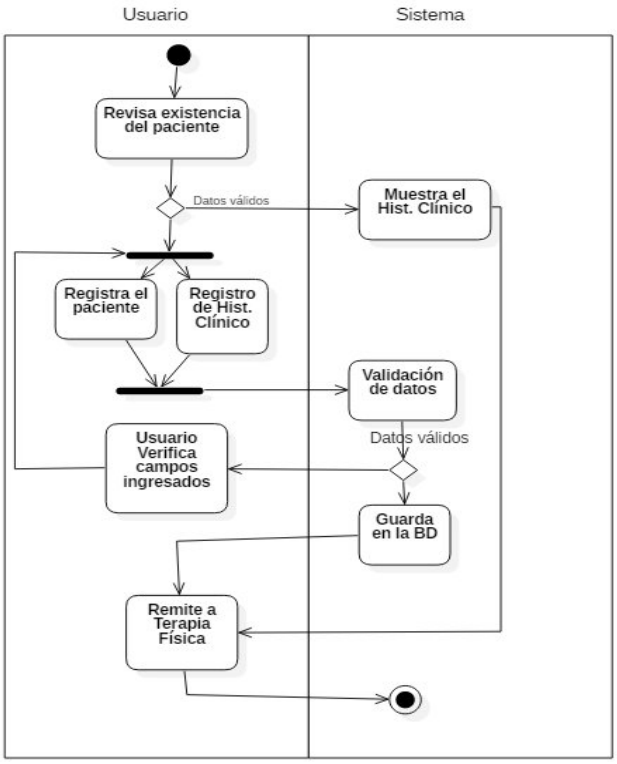
Figura 6. *Diseño de la arquitectura de un software*



Nota: Elaboración a partir de (Barraza, Modelado y Diseño de Arquitectura de Software, 2013)

Para delimitar de forma correcta la abstracción del patrón del que vamos a utilizar, que para nuestro estudio será el modelado cliente-servidor de tres capas; utilizaremos en primera instancia un diagrama de actividad de los requerimientos planteados en el proyecto a resolver identificando el flujo de los procesos que se automatizarán.

Figura 7. *Diagrama de actividad de registro de pacientes e historial cínico*

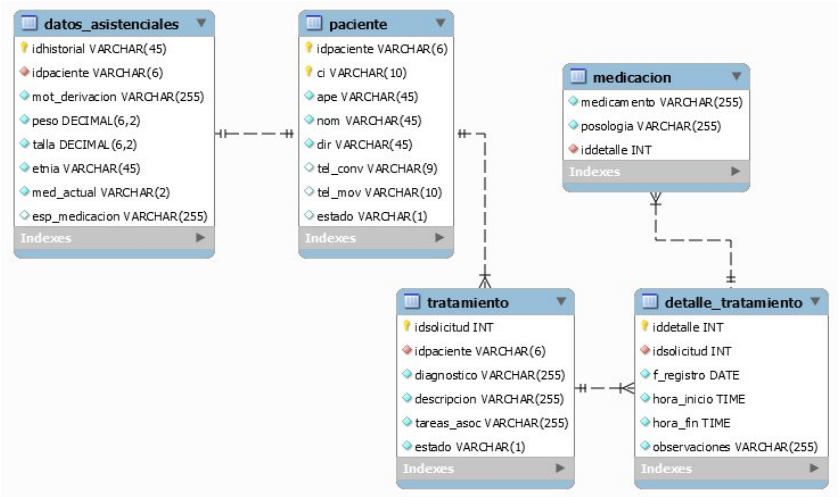


Nota: Elaboración propia

En la Figura 8, podemos verificar el flujo del proceso del registro de los pacientes cuando lleguen a la clínica; una vez que se establezcan los diagramas de actividad podemos llevarnos una mejor perspectiva del desarrollo de la aplicación; para que de ésta forma al abstraer las realidades del modelo puedan ser representados para el timo de modelado que vamos aplicar.

A continuación, podemos observar en la Figura 9 el modelo de datos para la aplicación que se está resolviendo como caso de estudio:

Figura 8. *Modelado de datos del caso de estudio Fisioport*



Nota: Elaboración propia

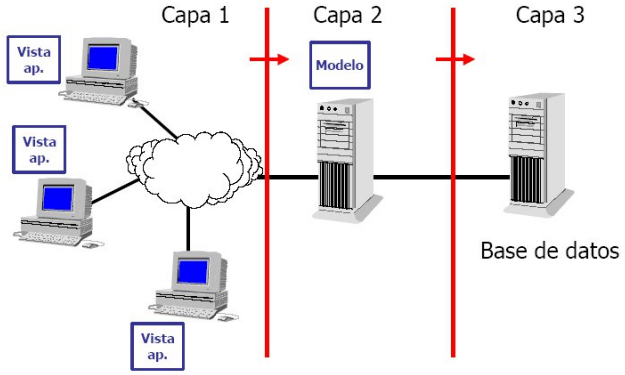
Es importante destacar que el modelo fue pensado en base a la información que se contaba con la tesis del estudiante; es decir, que lo planteado en la Figura se trató que se aproxime a la realidad de un fisioterapeuta, descargue aquí el modelo: https://n9.cl/se39e7

Para terminar el diseño de la arquitectura de software se lo abordará en la unidad siguiente.

1.6. Modelo cliente-servidor

Es una forma distribuida de las distintas tareas que puede tener una aplicación por parte de un usuario específico denominado cliente y proveedor de servicios denominado servidor, (Marin, 2012) establece que un modelo cliente-servidor "la capacidad de proceso está repartida entre los clientes y los servidores,… la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema"; por lo antes expuestos, el modelo está basado en dos capas; sin embargo se trabará en tres capas así como lo muestra la Figura 10, enfocados al uso de la programación orientada a objetos.

Figura 9. Modelo Cliente-Servidor en tres capas *(Vendrell, 2011)*



Nota: Elaboración propia

Es importante recalcar que existen diferentes formas para el desarrollo de una programación con servidores orientadas a objetos, además de varios frameworks, librerías u otros que permiten un mejor desarrollo en este tipo de programación; además de ser independiente del origen de los datos ver Figura 10; a pesar de lo expuesto no necesariamente debe estar en distintas maquinas sino más bien su implementación puede estar en una misma máquina.

En estos tipos de modelos, para resolver inconvenientes que puedan suscitarse sistemas de tipo heterogéneo y/o distribuido, para lo cual proveedores ofrecen interfaces de programación, además protocolos los cuales se denominan middleware.

Este servicio permite la interacción entre distintas plataformas, (Herrera, 2004) menciona que un middleware "es un servicio de propósito general que se ubica entre plataformas y aplicaciones", al ser un servicio maneja distintos protocoles que permiten ejecutar una aplicación en distintas plataformas; por lo tanto, permite la interoperabilidad uy portabilidad de las aplicaciones.

ACTIVIDADES PARA DESARROLLAR EN LA UNIDAD 1

1. Establezca más requerimiento funcionales y no funcionales, para el caso de estudio planteado desde la página 9
2. Diseñe los diagramas de actividad, faltantes según los requerimientos funcionales que hayan establecido
3. Dado el siguiente link de consultas:
   https://docs.microsoft.com/es-es/dotnet/framework/data/adonet/sql-server-connection-pooling
   Conteste las siguientes interrogantes:
   - ✓ ¿Cuál es la función de agrupar conexiones?
   - ✓ Cuándo de abre un grupo de conexiones, ¿Qué objetos se agregan al grupo?
   - ✓ ¿Qué sucede cuando se tiene más de 4 minutos de inactividad en el pool (grupo) de conexiones?
   - ✓ ¿En qué momento se crea un pool de conexiones

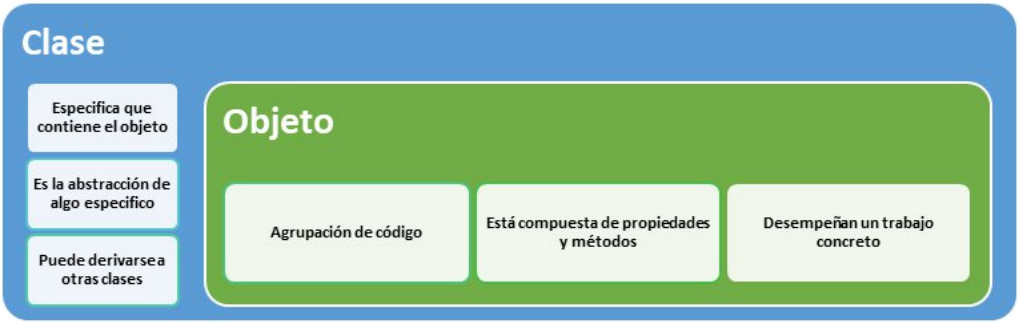
UNIDAD 2: MODELO SISTEMA CLIENTE – SERVIDOR DE TRES CAPAS

2.1 Programación orientada a objetos.

Antes de iniciar con el modelo de la unidad, se abordarán algunos aspectos que permitan un mejor entendimiento en la creación de dos capas especificas del modelado cliente-servidor, que este caso es el de negocio y datos. Por lo antes expuestos, la programación orientada a objetos se caracteriza por crear una estructura de código que está compuesto según (Blanco, 2002) por "un conjunto de procedimientos e información que ejecutan una serie de procesos destinados a resolver un grupo de tareas con un denominador común", de los cuales serán a accesibles siguiendo ciertas reglas para su utilización.

Dependiendo del desarrollador dichos procedimientos e información se crearán objetos que provendrán de una clase específica, así como lo muestra en la figura 11, la relación existente entre un objeto y una clase, sería que la clase vendría a ser como una plantilla o molde de los cuales se crean cada objeto.

Figura 10. *Relación entre un objeto y una clase.*

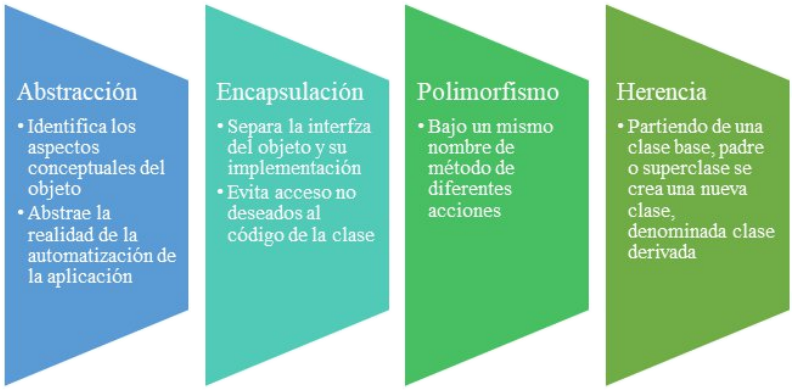


Nota: Elaboración a partir de (Blanco, 2002)

Al momento de crear un objeto, lo que estamos realizando es la instancia de una clase, es decir, que a partir de la clase se pueden crear un sin número de objetos y de ésta forma utilizar las propiedades o métodos que se hayan establecido en la clase, según la conveniencia de la aplicación.

Al momento de hablar de POO u OOP terminología en inglés, las mismas cumplen con ciertas características que son: Abstracción, Encapsulación, Polimorfismo y Herencia, de los cuales cada se caracteriza, según lo expuesto en la Figura 12.

Figura 11. *Características de la programación orientada a objetos.*



Nota: Elaboración a partir de (Blanco, 2002)

Es importante que para que este catálogo de objetos que se vayan a utilizar, se establezca una correcta clasificación de la estructura de código; para que de esta forma se tenga un mejor control de las clases de que se vayan a utilizar, dado que un objeto puede estar formado por otro objeto según las tareas que realizará cada uno. Al momento que se diseñan de forma correcta los mismos pueden ser reutilizados para un mismo proyecto y otra aplicación.

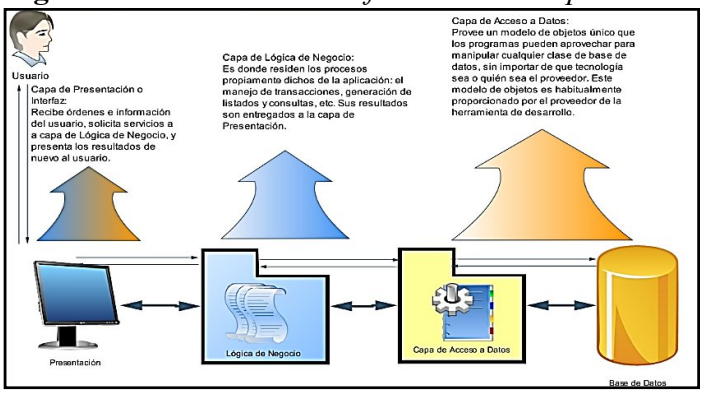
2.2 Arquitectura Lógica en tres capas

La programación en tres capas es una técnica, que partiendo del contexto de una programación orientada a objetos se organizan en tres capas:

- Presentación o frontera,
- Lógica de negocio o control, y
- De datos

Al ser una programación que separa responsabilidades, y cada con ciertas funciones específicas (Suarez, 2017); es decir, que cada uno realizaran distintas actividades o tareas entre sí, así como lo muestra la Figura 13, donde se especifica las acciones que realiza cada una y de esta forma organizaremos mejor nuestra programación al establecer roles en el diseño de la misma.

Figura 12. *Desarrollo de software a tres capas.*



Nota: Elaborado por (Suarez, 2017)

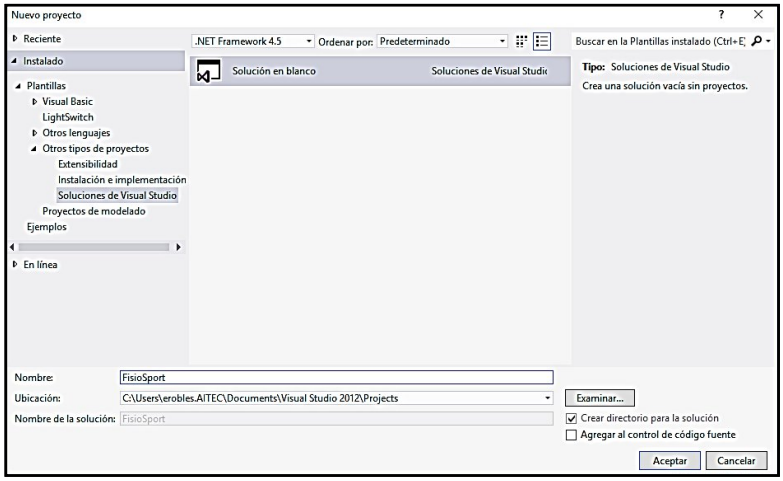
Como se puede apreciar en la Figura 13, la función principal de la ***capa de presentación o frontera***, facilita la interacción que tendrá el usuario con la aplicación; por lo tanto, es aquella interfaz que debe ser amigable y que cumpla con aspectos tales como lo menciona (Vargas Del Valle & Maltés Granados , 2007):

- Debe solicitar información consistente
- No utilizar más campos de lo necesario, claro y conciso
- Satisface los requerimientos del usuario
- Para cualquier validez de los datos, se debe comunicar con la capa de negocio

Para nuestro estudio vamos a crear una nueva solución en un proyecto denominado **FisioSport** del caso de estudio planteado en la unidad uno.

1. Diríjase a nuevo proyecto, en la categoría *otros tipos de proyectos*, esto puede variar un poco en relación a la ubicación, según la instalación y el programa que haya predeterminado para visual studio; en la Figura 14 se podrá observar que debe ubicar la opción *Solución en blanco*, escriba el nombre que corresponde el proyecto en nuestro caso FisioSport.

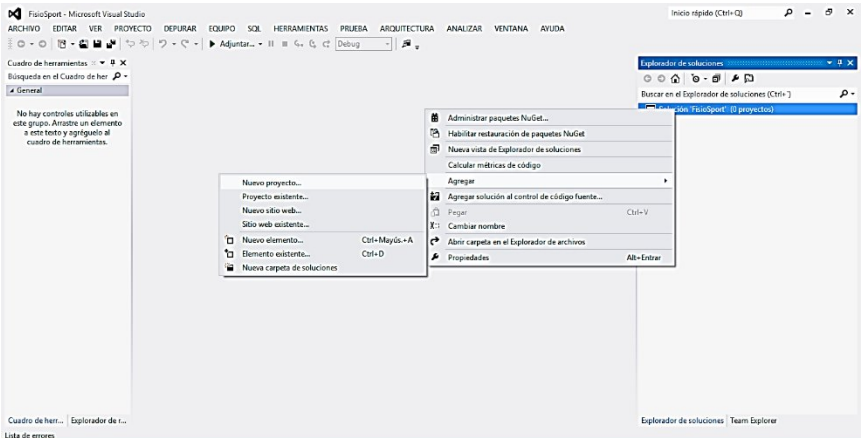
Figura 13. *Cuadro de dialogo para la creación de un nuevo proyecto*



Nota: Elaboración propia

2. Ahora agregaremos tres proyecto en nuestra solución; diríjase al explorador de soluciones y a través del mouse clic derecho y escoja la opción de agregar un *nuevo proyecto*; así como lo muestra la Figura 15, los proyectos los denominaremos según la metodología de estudio de esta asignara: CapaDeDatps, CapaDeNegocio, CapaDePresentacion.

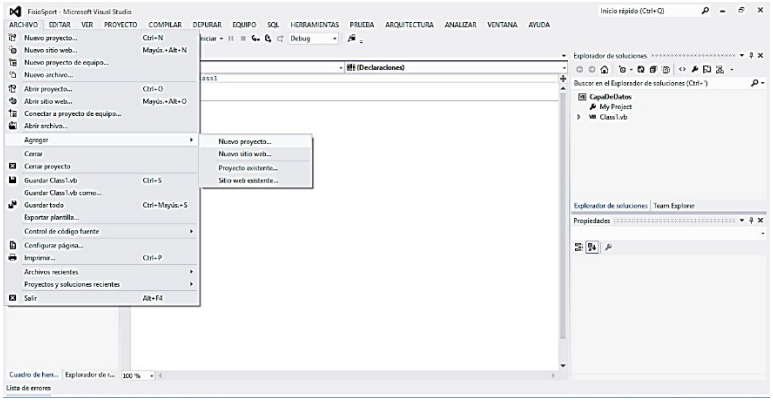
Figura 14. *Opción de agregar nuevos proyectos a la solución en blanco*



Nota: Elaboración propia

Para los proyectos de capa de datos y negocio utilizaremos la opción de biblioteca de clases y el de presentación como *aplicación de window forms*; otra forma de agregar un nuevo proyecto es como lo visualizamos en la Figura 16.

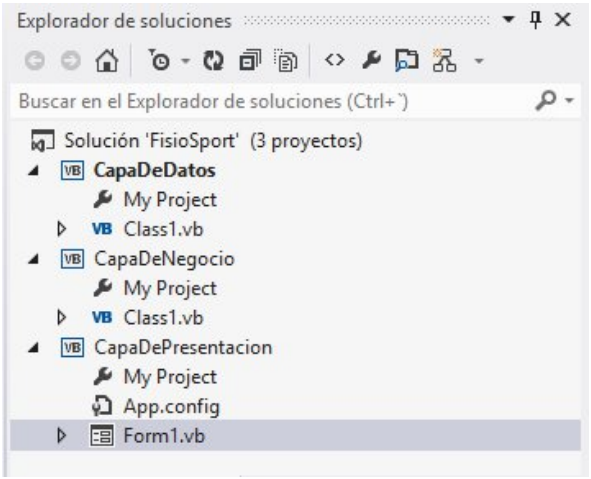
Figura 15. *Opción de agregar un nuevo proyecto en la solución de visual studio*



Nota: Elaboración propia

Quedando el explorador de soluciones con los tres tipos de proyectos agregados, ver la Figura 17.

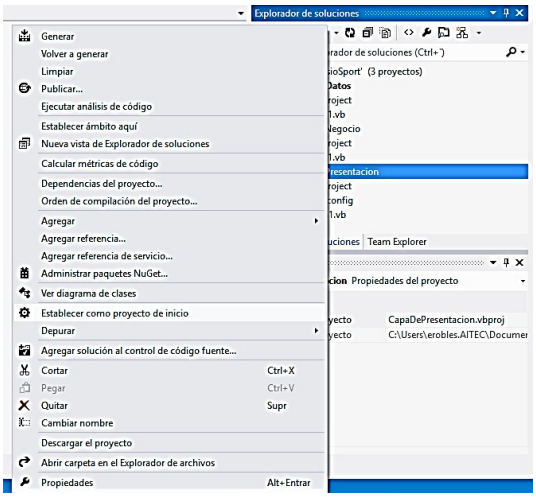
Figura 16. *Explorador de soluciones con los tres proyectos agregados*



Nota: Elaboración propia

3. Y por último estableceremos como proyecto de inicio al proyecto que hace referencia a la capa de presentación, ver Figura 18

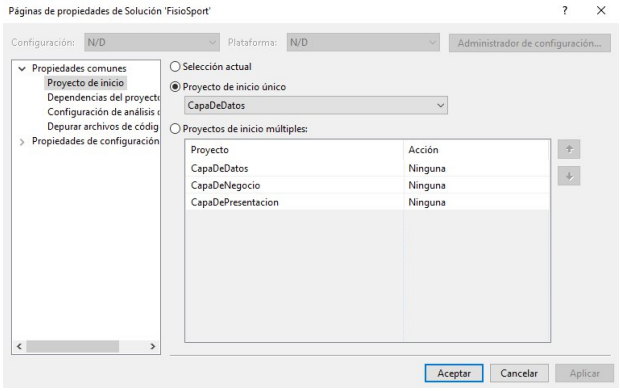
Figura 17. *Permite que en la solución creada se establezca el proyecto con el que inicia la aplicación*



Nota: Elaboración propia

Otra forma de realizar dicha acción se tendría que ir al explorador de soluciones, clic derecho y propiedades de la solución, ver Figura 19.

Figura 18. *Propiedades de una solución*



Nota: Elaboración propia

En el ámbito de la ***capa de negocio o control***, básicamente establecen las normas que deben seguirse para la ejecución correcta de la aplicación, tales como (Vargas Del Valle & Maltés Granados , 2007):

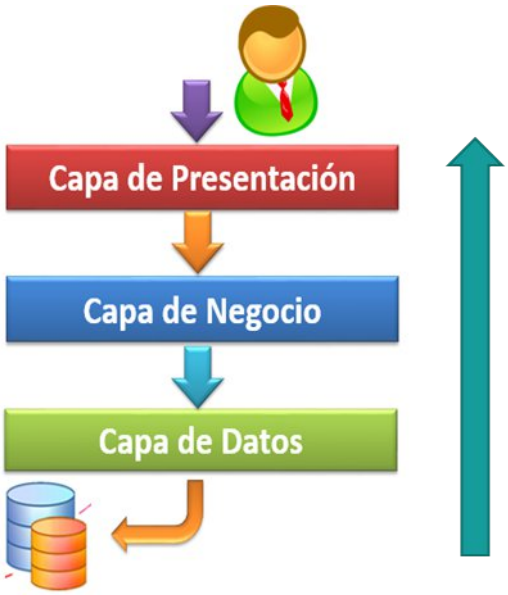
- Estructura de los datos y objetos, para la manipulación de los datos existentes
- Procesa la información ingresada en la capa de presentación
- Crea otros objetos según la lógica de la aplicación
- Se comunica con la capa de datos, para obtener información existente en la base de datos

Por ultimo tenemos a la ***capa de datos o persistencia***, la cual realiza se encarga de las distintas transacciones de la base de datos hacia la aplicación y viceversa, considere que interactúa con la capa de negocio.

2.3 Instanciación de clase

Al usar esta metodología el proyecto podrá ser llevado de forma ordenada bajo un esquema de relaciones entre cada capa y como serán retornado los datos, ver Figura 20.

Figura 19. *Metodología de programación en tres capas*

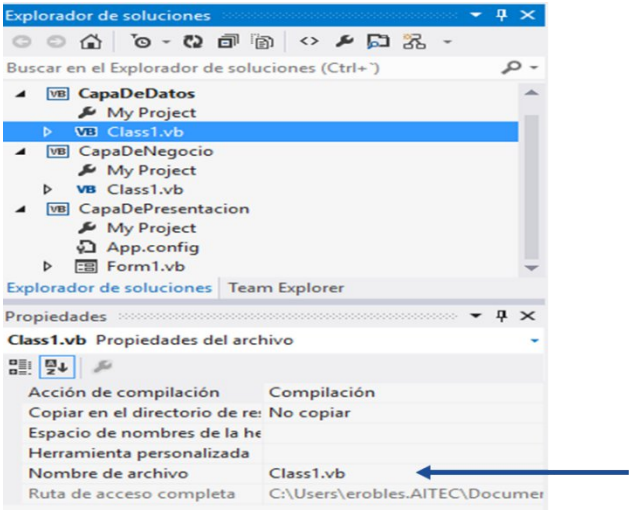


Nota: Elaboración propia

Para nuestro estudio, comenzaremos con la sección de la capa de datos; el cual permitirá abordar la temática planteada en esta unidad.

1. Cambiaremos el nombre del archivo de la clase, lo denominaremos *conexión*; podremos hacer la ventana propiedades, previamente seleccionando la clase, ver la Figura 21.

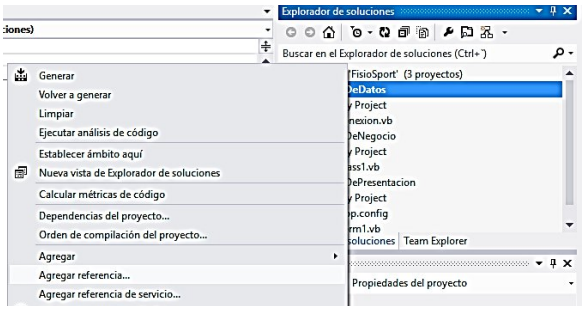
Figura 20. *Ventana propiedades*



Nota: Elaboración propia

2. Agregue al proyecto la referencia del proveedor de la base de datos, recuerde que lo puede realizar haciendo clic derecho al proyecto y *agregar referencia* o también la dicha opción la encontramos en la barra de menú *proyecto*.

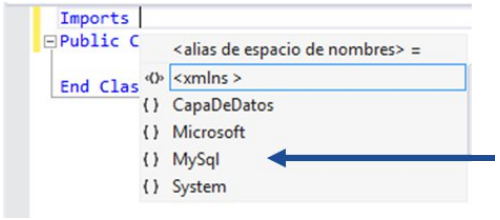
Figura 21. *Opción de como agregar una referencia a un proyecto*



Nota: Elaboración propia

3. Ahora en la clase, importemos la clases MySqlClient, ver Figura 23.

Figura 22. *Formas de importa clases dentro de otras clases*



Nota: Elaboración propia

Quedará de la siguiente forma `Imports MySql.Data.MySqlClient`

4. Cree una función que devuelva una conexión del proveedor de la bases de datos, ver Figura 24

Figura 23. *Código de conexión a una base de datos*

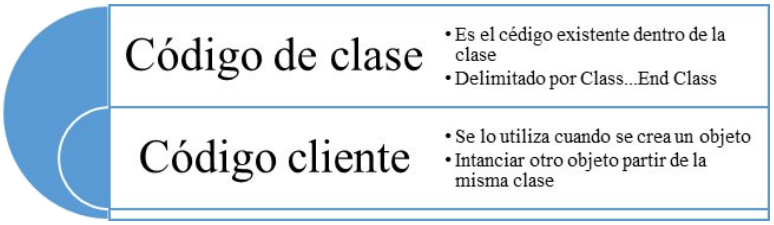
```
Imports MySql.Data.MySqlClient
Public Class conexion
    Private _conexion As MySqlConnection
    Public Function conectar() As MySqlConnection
        _conexion = New MySqlConnection("server=localhost;user id=eralro;port=3308;password=admin;database=fisiosport")
        Return _conexion
    End Function
End Class
```

Nota: Elaboración propia

Es importante que recuerde que los datos que se proveen en la conexión dependerán de lo establecido por el desarrollador.

Cabe recalcar que cuando estamos editando una clase, debemos establecer dos conceptos importantes para la creación de un objeto de basado en una clase que es (Blanco, 2002):

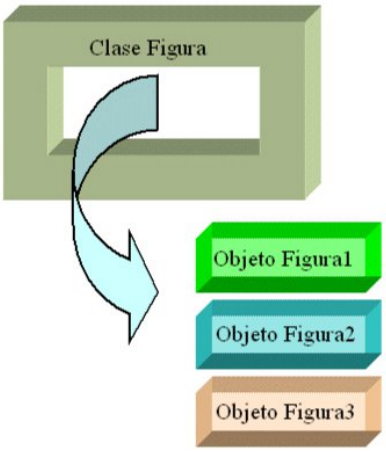
Figura 24. *Tipos de código de una clase*



Nota: Elaboración propia

Al momento que de crear un objeto basado en una clase es a lo que denominamos instanciación de objetos basados en la clase, es como tener una plantilla y a partir de la misma crear otros aspectos en el programa, ver Figura 25.

Figura 25. *Instanciación de un objeto a partir de una clase*



Nota: Elaboración propia

De esta manera podemos hacer uso de las propiedades y/o métodos que declaren en l clase; inclusive podemos hacer uso de los objetos de tipo control y hacer uso de sus propiedades y métodos a los que ellos tienen referencia.

Figura 26. *Formas de instanciar un objeto a partir de una clase*

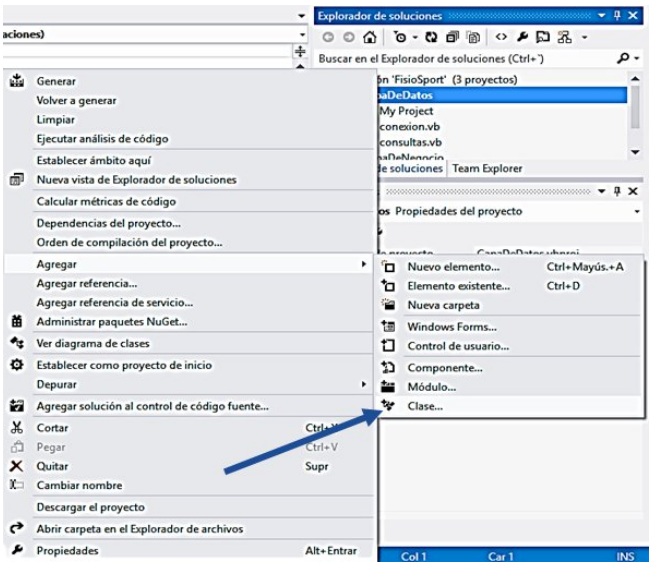
```
Module General

    Sub Main()
       ' declarar primero la variable
       ' y después instanciar el objeto
       Dim loEmpleado1 As Empleado
       loEmpleado1 = New Empleado()

       ' declaración e instanciación simultánea
       Dim loEmpleado2 As New Empleado()

       ' declaración y posterior instanciación en
       ' la misma línea de código
       Dim loEmpleado3 As Empleado = New Empleado()
    End Sub

End Module
```

Nota: Elaboración propia

Como podemos observar en la Figura 27, se puede apreciar distintas formas de instanciar un objeto y en cada uno de los ejemplos se puede notar el uso importante la de palabra reservada *New* de esta forma podemos hacer uso de los miembros de la clase que estemos instanciando sin ningún inconveniente.

A continuación, crearemos una clase de consultas en la capa de datos; por el momento crearemos una función que obtendrá los datos del tratamiento que se le ha dado a un paciente específico; para crear la nueva clase observe la Figura 28, el cual indica la opción para la nueva clase; considere que debe estar dentro de la capa de datos.

Figura 27. *Agregar una nueva clase para un proyecto dado*



Nota: Elaboración propia

A la clase creada se la denominará *consultas*, el cual, contendrán las consultas requeridas a la bases de tatos; agregue el código que se visualiza en la Figura 29, que tiene detallado la programación para requerir el tratamiento de un paciente.

Figura 28. *Programación que extrae los datos de un procedimiento almacenado en MySql*

```
Imports MySql.Data.MySqlClient
Public Class consultas
    Dim cn As New conexion
    Dim datosT As MySqlDataAdapter
    Dim datosC As MySqlCommand
    Dim tablaDatos As DataTable
    Public Function obtieneTratamiento(id As String) As DataTable
        Try
            datosC = New MySqlCommand("Tratamiento", cn.conectar)
            datosC.CommandType = CommandType.StoredProcedure
            datosC.Parameters.AddWithValue("@id", id)
            datosT = New MySqlDataAdapter(datosC)
            tablaDatos = New DataTable
            datosT.Fill(tablaDatos)
            Return tablaDatos
        Catch ex As Exception
            Throw ex
        Finally
            datosC.Dispose()
            datosT.Dispose()
            tablaDatos.Dispose()
            cn.conectar.Dispose()
        End Try
    End Function
End Class
```

Recuerde que debe importar la referencia agregada

Instancia la clase conexión creada

Considerar que nuestro modelo existe un procedimiento almacenado con dicho nombre

Nota: Elaboración propia

Como se puede observar en la Figura 29, devolverá una tabla con la información del procedimiento tratamiento, el mismo estará parametrizado con un *id* que corresponde al código del paciente.

2.4 Miembros de una clase

Al momento de mencionar los miembros de una clase, estamos indicando básicamente las propiedades y métodos que el mismo podría tener; sin embargo, estos aspectos dependerán del programador de la clase; es decir, que una clase puede manipular la información por un identificador o variable dentro de la clase, o podrías hacer uso de una propiedad las cuales pueden ser de tipo:

- Lectura, retorna información de la clase hacia donde es llamado
- Escritura, almacena la información con el uso de una variable para la propiedad

Las propiedades deben ser ámbito público y estar declarado como un procedimiento de tipo *property*, como se puede observar en la Figura 30.

Figura 29. *Declaración de un propiedad, en una clase.*

```
Public Class Empleado

    ' declarar una variable de propiedad
    ' para la propiedad Nombre
    Private msNombre As String

    ' declarar el procedimiento Property
    ' para la propiedad Nombre
    Public Property Nombre() As String
        ' bloque Get para devolver
        ' el valor de la propiedad
        Get
            Return msNombre
        End Get

        ' bloque Set para asignar
        ' valor a la propiedad
        Set(ByVal Value As String)
            msNombre = Value
        End Set
    End Property

End Class
```

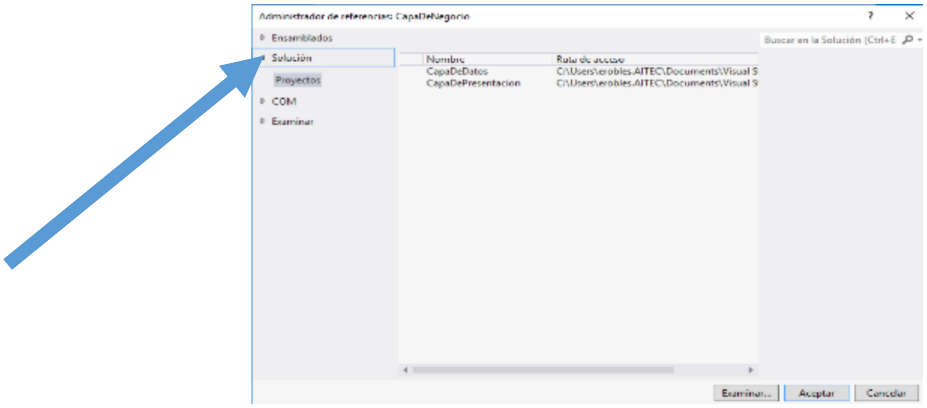
Nota: Elaboración a partir de *(*Blanco, 2002)

Podemos destacar que cuando se requiere que una clase devuelva un valor a través de sus propiedades según el bloque expuesto en la Figura 30, el bloque *Get* retornará un valor con el uso de la palabra reservada *return* y el bloque *set* se lo utilizara para guardar la información dentro de la clase, ambos utilizan una variable privada solo para la clase, y la misma será la intermediaria para dar lectura o almacenamiento dentro de la clase.

Se pueden crear propiedades independiente de lectura y escritura de forma o ambas, ver Figura 32; el trabajo de cada será como se lo menciona y se hará uso delas palabras reservadas *Write* y *ReadOnly*.

Para comprender mejor esta apartado continuaremos con el desarrollo de nuestro ejercicio; en la capa negocio, la clase existente la denominaremos tratamiento; según nuestro el modelo de programación se agregará como referencia a la capa de datos; la misma forma como agregamos una referencia cualquiera a un proyecto, pero en este caso será de la misma solución, ver la Figura 29.

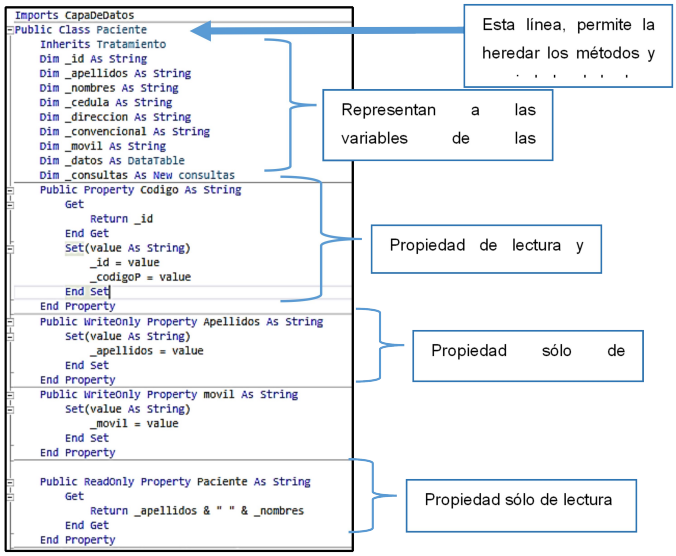
Figura 30. *Agregar a un proyecto la referencia de otro proyecto*



Nota: Elaboración propia

Posteriormente agregaremos una clase denominada *Paciente* en la capa Negocio el cual contendrá propiedades que administren la información del paciente para nuestro modelo o desarrollo del ejercicio, ver Figura 32 y además heredaremos a la clase *tratamiento*.

Figura 31. *Declaración de propiedades de la clase paciente*



Nota: Elaboración propia

Continúe con la declaración de propiedades, para esta clase según las variables declaradas, con la ayuda de su docente; además agregue la siguiente función que permitirá mostrar el paciente y sus tratamiento, ver Figura 33.

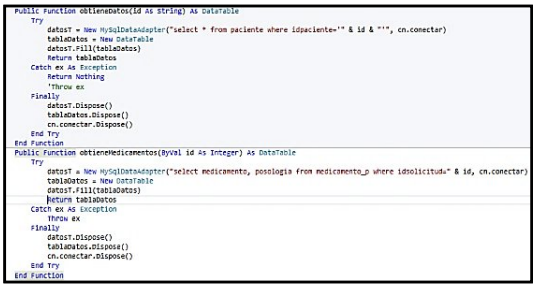
Figura 32. *Búsqueda del paciente solicitado*

```
Public Function buscar() As Boolean
    _datos = New DataTable
    _datos = _consultas.obtieneDatos(_id)
    If _datos.Rows.Count <> 0 Then
        _apellidos = _datos.Rows(0).Item(2).ToString
        _nombres = _datos.Rows(0).Item(3).ToString
        Return True
    Else
        Return False
    End If
End Function
```

Nota: Elaboración propia

En su programación le saldrá un error, debido a que no tiene declarado la en la clase consulta de la capa de datos, el método *obtieneDatos*, para tal acción vea la Figura 34 el cual hace referencia a otros métodos que utilizaremos en nuestra aplicación.

Figura 33. *Métodos para la obtención de los datos de un paciente y sus medicamentos*



Nota: Elaboración propia

Agregue una clase denominada *Medicamentos*, y posteriormente adaptaremos la clase tratamiento con otras propiedades que se detallan en la Figura 36; que de forma adicional hereda la clase medicamentos. Antes de ir a la clase tratamiento veamos las propiedades y métodos que tendrá la clase medicamentos, ver Figura 35, el cual contendrá el código requerido para lo que estamos realizando.

Figura 34. *Código que obtiene los medicamentos de un paciente dado*

```
Imports CapaDeDatos
Public Class Medicamentos
    Dim _medicamentos As DataTable
    Dim _consulta As New consultas
    Protected _idSolicitudMed As Integer
    Public ReadOnly Property Medicamento As DataTable
        Get
            _medicamentos = New DataTable
            _medicamentos = _consulta.obtieneMedicamentos(_idSolicitudMed)
            Return _medicamentos
        End Get
    End Property
End Class
```

Nota: Elaboración propia

Como se puede observar en la Figura 36, en la clase tratamiento tenemos una variable protegida, con la finalidad de obtener la información del código del paciente; recuerde que el ámbito de variables se caracterizan por el nivel de acceso que pueden tener (Microsoft, 2018), mas información dar clic aquí https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/declared-elements/access-levels donde podremos observar el nivel de acceso que se le otorga a una variable, método o cualquier otro objetos declarado dentro de una clase, modulo, etc.

Figura 35. *Código que obtiene la información de un tratamiento*

```
Imports CapaDeDatos
Public Class Tratamiento
    Dim _consulta As New consultas
    Dim datos As DataTable
    Dim _diagnostico As String
    Dim _descripcion As String
    Dim _tarea As String
    Dim _fecha As Date
    Dim _horaInicio As DateTime
    Dim _horaFin As DateTime
    Dim _observaciones As String
    Dim _tratamiento As DataTable
    Protected _codigoP As String
    Public Property Diagnostico As String
        Get
            Return _diagnostico
        End Get
        Set(value As String)
            _diagnostico = value
        End Set
    End Property
    Public Property Descripcion As String
        Get
            Return _descripcion
        End Get
        Set(value As String)
            _descripcion = value
        End Set
    End Property
```

Recuerde que debe importar la referencia agregada

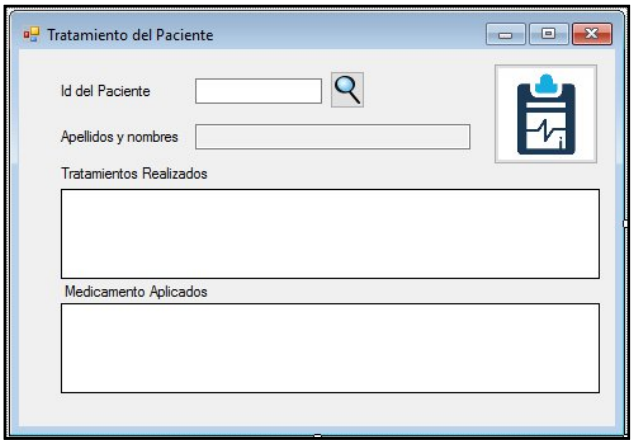
Variable protegida

Nota: Elaboración propia

Finalmente, para aplicar las clases y métodos que hemos desarrollado en los anteriores aparatos de este módulo; diseñaremos en la capa de presentación un formulario que alimente y visualice la información requerida para nuestro ejemplo planteado.

Guíese por la lustración 37, para el diseño de su formulario los nombres de los controles y demás, usted en conjunto con su profesor establezcan los mismos.

Figura 36. *Formulario que muestra la información de un paciente dado*



Nota: Elaboración propia

Una vez diseñado el formulario en la capa presentación, verifique el código que contendrá, considerando que hay dos eventos a programar:

- El botón que mostrara el paciente y su tratamiento
- Y la grilla que mostrara el medicamento según el tratamiento realizado

Vea la Figura 38, y guíese con lo indicado.

Figura 37. *Código que visualiza la información del caso de estudio planteado*

```
Imports CapaDeNegocio
Public Class frmtratamiento
    Dim paciente As New Paciente
    Private Sub btnBuscar_Click(sender As Object, e As EventArgs) Handles btnBuscar.Click
        paciente.Codigo = txtid.Text
        If paciente.buscar Then
            txtpaciente.Text = paciente.Paciente
            dtgTratamiento.DataSource = paciente.Tratamiento
            dtgTratamiento.Columns(0).Visible = False
            dtgTratamiento.Columns(1).Visible = False
        Else
            MsgBox("Código no existe...")
        End If

    End Sub

    Private Sub dtgTratamiento_CellContentClick(sender As Object, e As DataGridViewCellEventArgs)
        paciente.Solicitud = dtgTratamiento.Item(0, e.RowIndex).Value
        dtgmedicamentos.DataSource = Nothing
        dtgmedicamentos.DataSource = paciente.Medicamento
    End Sub
End Class
```

Nota: Elaboración propia

Recuerde que en la capa de presentación, debe importar a la capa negocio, según el modelo planteado en nuestro apartado.

ACTIVIDADES PARA DESARROLLAR EN LA UNIDAD 2

1. Dado el siguiente enlace:https://docs.microsoft.com/es-es/dotnet/visual-basic/programming-guide/concepts/object-oriented-programming
    - Determine lo siguiente:
        - ¿Cuál es la diferencia entre el uso de una interfaz y aplicar herencia?
        - ¿Cómo se aplicaría la encapsulación dentro de un proyecto dado?
        - ¿Cuáles son los ámbitos que puede tener un reemplazo de miembros en la herencia?

2. Redacte como funciona un modelo de N-capas; link recomendado: https://s3.amazonaws.com/academia.edu.documents/40842810/a07v7n2.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1555020930&Signature=9ymQS18GOM8hRDGF6DburAHTujU%3D&response-content-disposition=inline%3B%20filename%3DProgramacion_por_capas.pdf

3. Diseñe un menú para el caso de estudio propuesto y agregue ventanas tales como:
    - Inserción de un nuevo paciente
    - Diseñe fechas de citas posteriores
    - Llamado al formulario creado en el módulo, Tratamiento
    - Establezca las validaciones entrantes y saliente según sea el caso del proyecto planteado

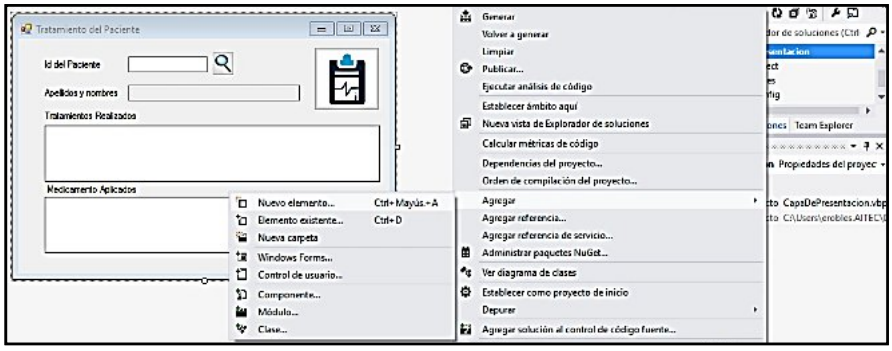
UNIDAD 3: REPORTES Y EMPAQUETADO

3.1. Conjunto de datos

Para el inicio de esta unidad tenemos que considerar algunos elementos en el diseño del reporte previamente, en este caso el conjunto de datos; hay que comprender que el mismo permite la administración de los datos, denominado CRUD, sin que esté conectado a las bases de datos; debido a que los mismos se encuentran en memoria (Microsoft, 2018).

Para nuestro proyecto, lo agregaremos en la capa de presentación, pata esto observe la Figura 39, el cual indica como agregar el conjunto de datos.

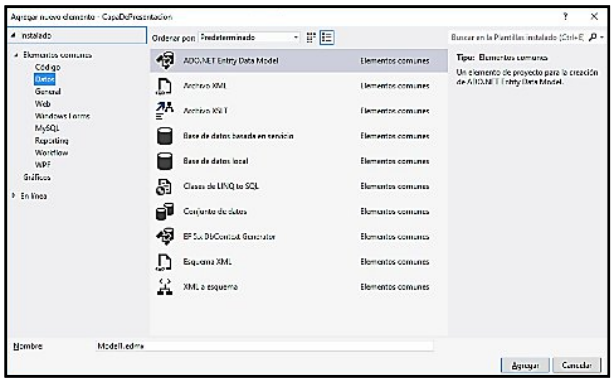
Figura 38. *Agrega un nuevo elemento, conjunto de datos*



Nota: Elaboración propia

De allí saldrá una ventana de dialogo de inserción de elementos, nos dirigimos a la categoría datos de allí escogemos *conjunto de datos*, la denominaremos *CDatos*, observe la Figura 40.

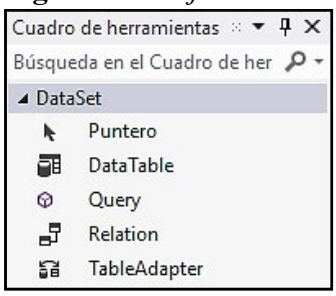
Figura 39. *Cuadro de diálogo que agrega elementos a proyectos en visual estudio*



Nota: Elaboración propia

Una vez agregado el conjunto de datos, se abre un espacio de diseñador de dataset, el cual podemos agregar los objetos que mantendrá en memoria la información que se requiera, los cuales están en el cuadro de herramienta, ver Figura 41.

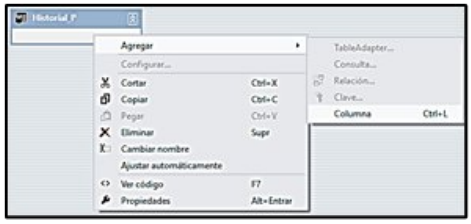
Figura 40. *Objetos del diseñador de dataset*



Nota: Elaboración propia

Para nuestro caso se agregarán DataTable y agregaremos los campos que serán requeridos para nuestro reporte; es importante que los campos agregados dependerán de lo que planteamos en nuestro proyecto. Además, se debe colocar el nombre de cada campo según lo establecido en la tabla o vista del modelo que dispongamos; es nuestro ejemplo vamos a imprimir el historial médico de un paciente dado, utilice la vista de nuestro modelo estudiado, el cual se denomina *historial_paciente*; para ir insertando los campos al datatable, podemos observar la Figura 42, es importante que el nombre de cada campo sea igual al descrito en la vista del ejercicio propuesto.

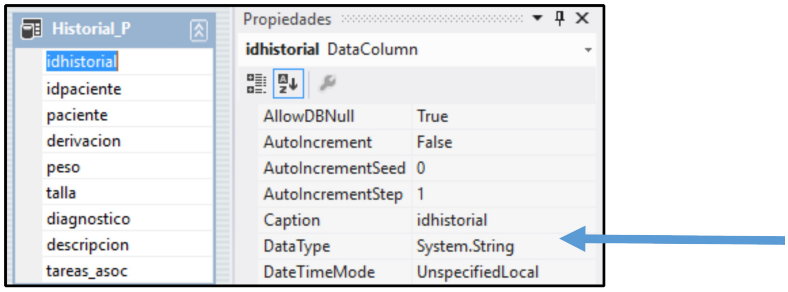
Figura 42. *Pasos para agregar un campo a un DataTable, en el espacio de un conjunto de datos en visual basic*



Nota: Elaboración propia

Una vez que agregue los campos de la vista de la base de datos; verifique el tipo de dato de cada uno y ajústelo según el tipo correspondiente, ver Figura 43; debe seleccionar el cambio que vaya a modificar el tipo de datos y en la sección de propiedades, aparecerá lo requerido, las otras opciones pueden dejarlo por defecto.

Figura 43. *Modificación de los tipos de datos de un campo insertado, dentro de un DataTable*



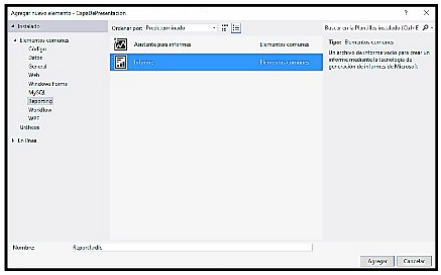
Nota: Elaboración propia

3.2. Diseño de Informes

Al momento de diseñar un informes, básicamente lo que agregamos es un archivo de informes .rdlc, utilizado en el control report viewer de visual studio, para nuestro caso visual basic (Ghanayem, Informes, elementos de informe y definiciones de informe, 2018).

Posteriormente diseñaremos nuestro informe con el uso de informes, ver la Figura 44; sin embargo, para organizar nuestra información agregaremos una carpeta, donde almacenaremos dichos diseño de informes, recuerde que todos estos aspectos están siendo en la capa de presentación.

Figura 44. *Ventana que agrega un nuevo informe a un proyecto dado*



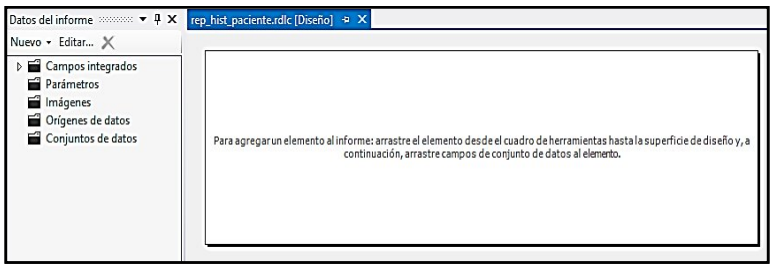
Nota: Elaboración propia

El reporte lo denominaremos, rep_hist_paciente.rdcl, ver la Figura 45, el cual se observa el espacio que permitirá realizar el diseño de nuestro informe, los cuales son usado por el control ReportViewer (Ghanayem, Informes, elementos de informe y definiciones de informe (Generador de informes y SSRS), 2018), un informe puede contener los siguientes elementos:

- Gráficos
- Mapas
- Tablas
- Medidores
- Parámetros
- Matrices
- Imágenes
- Rectángulos
- Listas

Considere que un informe puede ser reutilizable, es decir agregar un informe dentro de otro previamente que sea existente.

Figura 45. *Cuadro de herramientas y su diseñador de informes en visual basic*



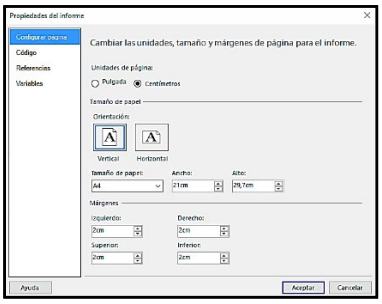
Nota: Elaboración propia

Un informe puede estar dividido en tres secciones:

- Encabezado
- Pie de pagina
- Y la parte central donde se colocan los elementos del informe

Cada uno tendrá las características de mostrar lo que el programador establezca, para el informe que se vaya a enviar. En la barra de herramienta podemos encontrar más aspectos que permitan mejorar la presentación de nuestro informe, los iconos mostrados permiten agregar encabezado, pie de página e ir a la configuración general de informe, ver Figura 46.

Figura 41. *Cuadro de dialogo de propiedades del informe*

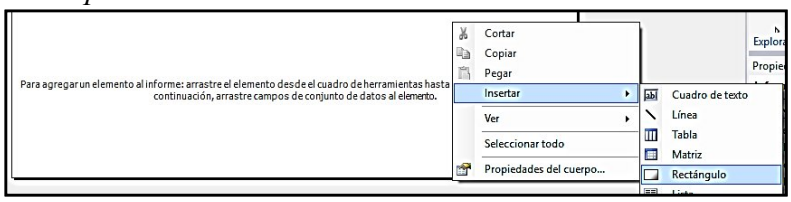


Nota: Elaboración propia

El cuadro de dialogo de la Figura 46, también lo podemos configurar con el uso de la ventana propiedades de visual studio, seleccionando previamente la categoría informe. Vamos a diseñar nuestro informe, siguiendo estos pasos:

1. Una vez agregados el informe, ver Figura 44, agregaremos un encabezado según lo expuesto en esta apartado, también lo puede realizar haciendo clic derecho en el centro del diseño del reporte y escoger la opción encabezado, ver Figura 47.

Figura 42. *Forma de agregar un encabezado, pie de página u otro control en el diseño del reporte*



Nota: Elaboración propia

2. Diseñe este encabezado, utilizando las herramientas propuesta por el diseñador del informe, el cual se agrega en la barra de herramientas, además lo puede hacer a través del uso de propiedades del elemento agregado; para nuestro ejemplo agregaremos una imagen y un cuadro de texto, ver Figura 48.

Figura 43. *Diseño del encabezado del reporte del paciente*

Nota: Elaboración propia

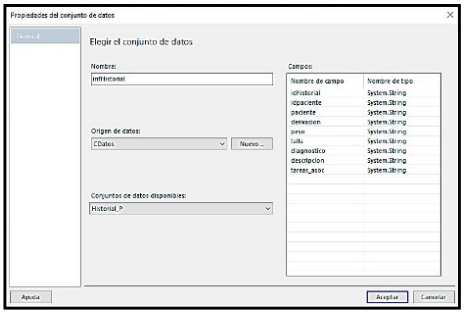
3. En la sección datos del i informe, agregaremos el conjunto de datos que agregamos el inicio de este apartado, ver Figura 49.

Figura 44. *Opción de como agregar un conjunto de datos a nuestro informe*

Nota: Elaboración propia

4. Se escribe un nombre para el Dataset, y se van escogiendo las opciones inferiores, agregando el conjunto de datos y en la última parte el DataTable creado, si hubiesen más Datatable aparecerán en dicho listado, ver Figura 50.

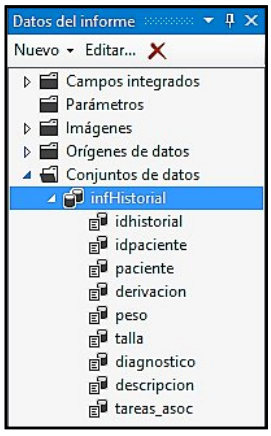
Figura 45. *Cuadro dialogo de las propiedades de un conjunto de datos en el informe*



Nota: Elaboración propia

5. Una vez agregado, aparecerá todos los campos creados en DataTable del conjunto de datos agregado, ver Figura 51.

Figura 46. *Campo del conjunto de datos agregados*



Nota: Elaboración propia

6. Por último, ir agregando los campos requeridos según nuestro diseño del informe, ver Figura 52, el cual, puede hacer uso del control cuadro de texto como etiqueta para los campos agregados.

Figura 47. *Diseño del informe del caso de estudio propuesto.*

Nota: Elaboración propia

7. Agreguemos en la capa de datos, dentro de la clase *consultas*, una consulta a la vista del modelo estudiado, ver Figura 53.

Figura 48. *Código de consulta para el informe realizado*

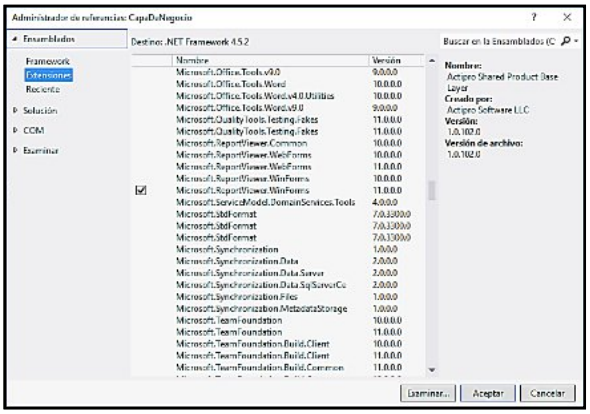
```
Public Function infPaciente() As DataTable
    Try
        datosT = New MySqlDataAdapter("select * from historial_paciente", cn.conectar)
        tablaDatos = New DataTable
        datosT.Fill(tablaDatos)
        Return tablaDatos
    Catch ex As Exception
        Throw ex
    Finally
        datosT.Dispose()
        tablaDatos.Dispose()
        cn.conectar.Dispose()
    End Try
End Function
```

Nota: Elaboración propia

8. Agreguemos una referencia de servicio, ver Figura 54; dentro de la capa de negocio; para que de esta forma importar la clase

```
Imports Microsoft.Reporting.WinForms
```

Figura 49. *Agregar referencias de servicio a un proyecto dado*



Nota: Elaboración propia

9. Ahora, diseñaremos una clase denominada Informes que permitirá cargar cada informes según el requerimiento del usuario, para esto observemos que en la Figura 55, se tiene el código necesario para la automatización de los llamados

repetitivos que pueda tener un informe.

Figura 50 *Clase Informe, que contiene el código para llenar un control report viewer*

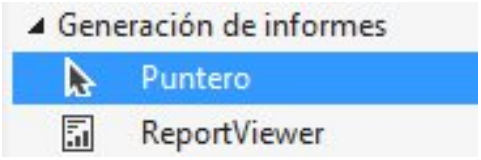
Nota: Elaboración propia

10. Por último, agregaremos un nuevo formulario; además se inserta un control report viewer y se lo deja acoplado al formulario.

3.3. Control report viewer

El report viewer es un control que permite visualizar informes que se ejecutan en el .Net Framework, ubicado en el conjunto de los controles del cuadro de herramientas, dentro de la categoría *Generación de Informe*

Figura 51. *Control Report Viewer*



Nota: Elaboración propia

Al momento de insertar al formulario, se lo realiza de igual forma que los otros controles, es decir se puede ajustar al formulario y redimensionarlo según nuestra necesidades. Dicho control también tiene la posibilidad de exportar:

- Excel,
- PDF, y
- Word

Así como también, otras formas de visualización, como lo podemos observar en la Figura 57, configuración de página, búsqueda de texto.

Figura 52. *Informe ejemplo del caso de estudio tratado*

Nota: Elaboración propia

Continuando con nuestro proyecto, una vez insertado el report viewer, se procede a invocar al reporte según lo establecido en todo el desarrollo del caso de estudio propuesto, programando en un botón, para que llame al formulario que tiene el report viewer, el cual al momento de cargarse deberá contener la codificación expuesta en la Figura 58.

Figura 53. *Código que permite llenar el control report viewer con el informe requerido por el caso de estudio*

```
Imports CapaDeNegocio
Public Class frmHistorialPaciente
    Dim reporte As New Informe
    Private Sub frmHistorialPaciente_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        reporte.nombre = "infHistorial"
        With rvwInforme
            .Reset()
            .LocalReport.DataSources.Clear()
            .LocalReport.ReportPath = reporte.rutaInforme
            .LocalReport.DataSources.Add(reporte.Paciente)
            .RefreshReport()
        End With
    End Sub
End Class
```

Nota: Elaboración propia

Como podemos observar en la Figura 58, se importa la capa de negocio donde tenemos la clase *Informe* que procesa la petición del usuario y retorna un ReportSorce y diseño de informe para que el report viewer visualice el reporte deseado.

3.4. Empaquetamiento de una aplicación

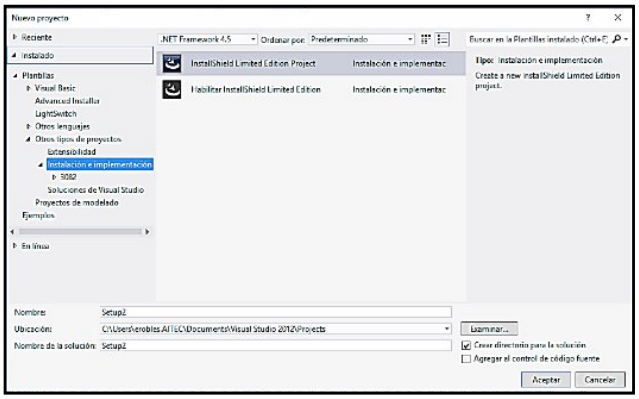
Al momento de finalizar una aplicación, a nivel de escritorio, se requiere crear un ensamblado para la instalación en los equipos donde vaya a funcionar dicha aplicación, para nuestro estudio, el que va al servidor y los que se instala en nuestros clientes.

Cuando mencionamos, que vamos a –empaquetar una aplicación–, básicamente no estamos refiriendo a las forma de cómo se va a "vender tu aplicación Plataforma universal de Windows UWP o distribuirla a otros usuarios" (Schofield, 2019), en esta apartado vamos a detallar el proceso de creación de un paquete aplicación para la instalación de nuestro proyecto utilizando InstallShield, siendo esta una herramienta para crear instaladores o paquetes de software.

3.4.1 Creación de paquetes de software

Comenzaremos agregando a nuestra solución, un nuevo proyecto, nos dirigiremos a la categoría *instalación e implementación*, vea la Figura 59, donde se observa la opciones de la categoría escogida, de allí utilizaremos la opción de *InstallShield;* el cual, nos enviará a un sitio web para su descarga.

Figura 54. *Creación de un paquete de instalación con InstallShield*



Nota: Elaboración propia

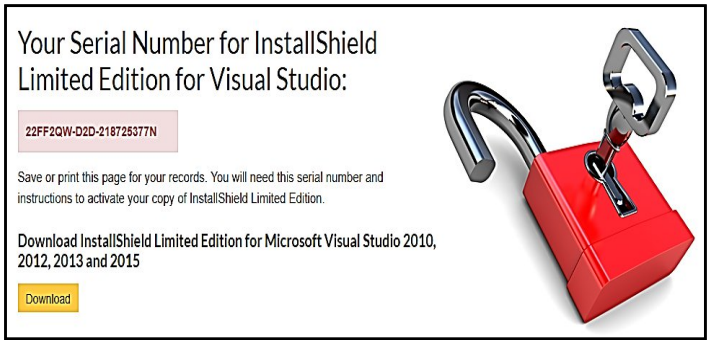
Este sitio de descarga, va a permitir bajar el aplicativo, ver Figura 60, y a su vez será instalado a través de wizard. Es importante resaltar que para la descarga del mismo tendrá que llenar un formulario.

Figura 55. *Sitio web que reenvía a la descarga de la aplicación.*

Nota: Elaboración propia

Antes iniciar la descarga se lo redireccionará a otro sitio donde le dará a conocer una contraseña con validará el software descargado, vea la Figura 61.

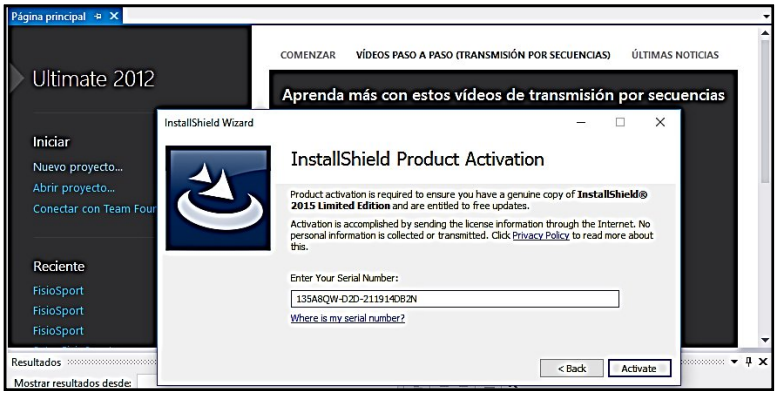
Figura 56. *Página de InstallShield, que envía código de convalidación previo registro del formulario.*



Nota: Elaboración propia

Se recomienda cerrar visual studio, aperture nuevamente y realice los pasos establecidos en la Figura 59, al momento de realizarlo le solicitará el código indicado al momento de la descarga, vea la Figura 62.

Figura 57.*Solicitud de activación de InstallShield*



Nota: Elaboración propia

Cuando ingrese la clave, usted podrá ver un espacio para trabajar para el inicio de su empaquetamiento de la aplicación.

3.4.2 Proceso de construcción del empaquetado

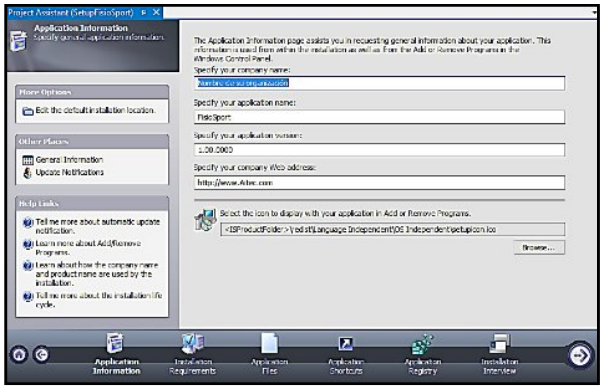
Al momento de que iniciamos nuestro proyecto de empaquetamiento la pantalla que apares es un asistente para la creación del mismo, donde se dividirá en 6 secciones:

- Información de la aplicación
- Requerimiento de la instalación
- Archivos que componen la aplica
- Creación de accesos rápidos
- Registro de activación
- Licencia y diálogos del empaquetado

3.4.3 Información de la aplicación

Este es el primer espacio del asistente de la aplicación, en esta apartado daremos a conocer información acerca de nuestra aplicación, especificaciones de índole informativo del programa, vea la Figura 63.

Figura 58. *Información básica de la aplicación que se va a empaquetar*



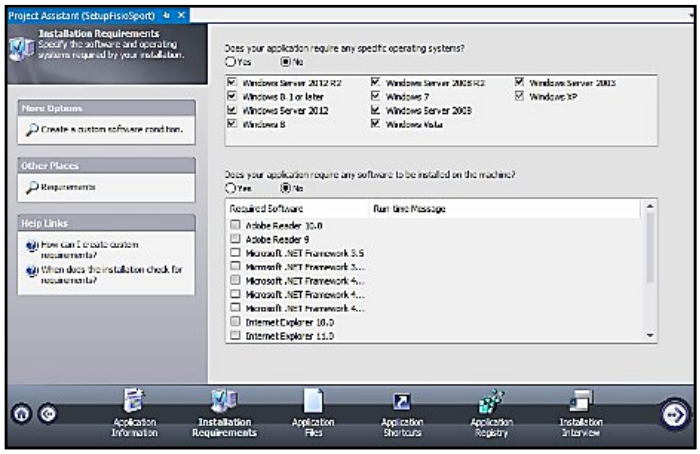
Nota: Elaboración propia

En el panel izquierdo podremos detallara más especificaciones del software, tales como: serializaciones de modificaciones y/o actualizaciones, especificaciones de derechos de autor, entre otras. Además encontramos en el panel izquierdo, temas de ayuda en relación a la sección que se encuentre.

3.4.2.1. Requerimientos de instalación

En esta sección, se establecerán los requerimiento no funcionales del software, es decir todos aquellos elementos que permitan la ejecución de nuestro programa; framewrok, base de datos, Plugins, Api, etc. Vea la Figura 64.

Figura 59. *Sección de los requerimientos previos para nuestra solución*



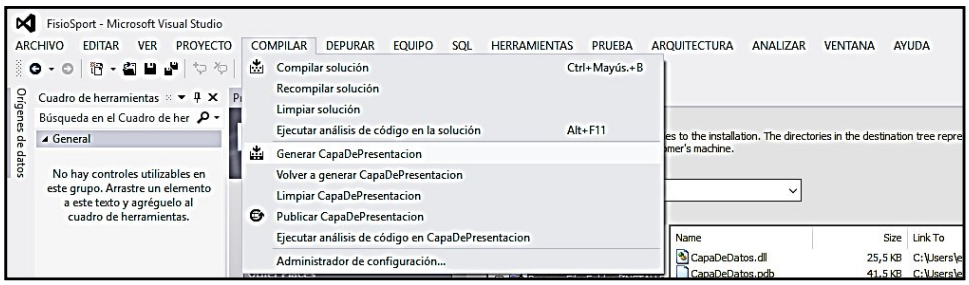
Nota: Elaboración propia

Además, tiene la opción de crear nuestras propias condiciones de requerimiento, es decir algún otro programa específico que necesite de instalación.

3.4.2.2. Archivos que componen la aplicación

Aquí se deben establecer todos los archivos que permitan la ejecución de nuestra aplicación, para los cual en primera instancia debe haber sido generado el aplicativo antes de iniciar el proceso de instalación, con la barra de menú *Compilar* y genere la aplicación, ocurriendo dos aspectos en la compilación del proyecto, uno de ellos es la verificación de existencia de algún error, y otro es crear el ejecutable bajo la compilación realizada, vea la Figura 65.

Figura 60. *Opción de compilación y generación del programa de nuestro caso de estudio*

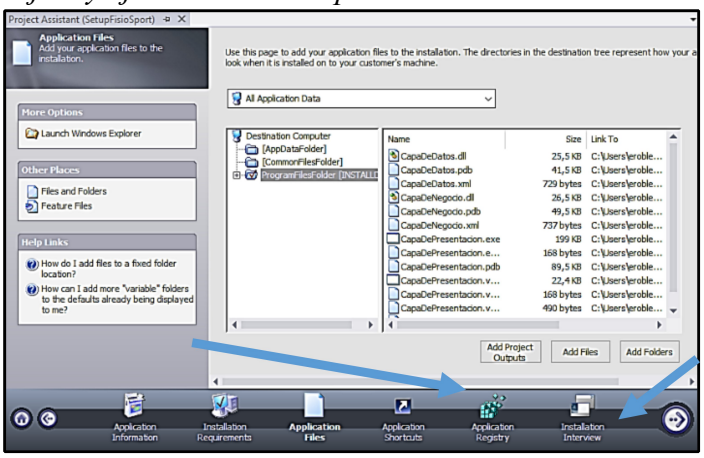


Nota: Elaboración propia

Como se puede notar que el proyecto que se genera es el de la capa de presentación, debido a que allí están los formularios o ventanas del proyecto; además agregar la carpeta que tiene los archivos del informe para que de ésta forma cuando la aplicación busque

para para su respectiva impresión se puedan cargar sin ningún inconveniente. Vea la Figura 66.

Figura 61. *Sección que se agregan los archivos de la aplicación compilada, el código objeto y ejecutable de la aplicación.*



Nota: Elaboración propia

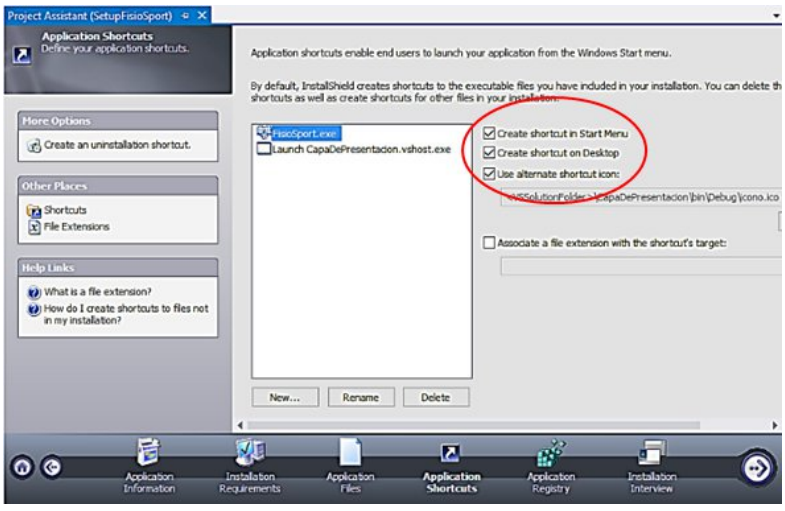
Usted podrá seleccionar los archivos bajos dos contextos:

- Archivos de proyectos de salida.- Esta opción te envía todos los archivos de salida que tiene la aplicación.
- Archivos.- Esta opción permite agregar archivos externos a la aplicación, en nuestro caso los de tipo rdcdl, es decir los archivos de informes.
- Carpetas.- Además de agregar carpeta de forma completa

3.4.2.3. Creación de accesos rápidos

En esta sección se establecerán los accesos rápidos de la aplicación y en que ámbitos del sistema operativo se van a crear, ya sea desde el menú de Windows, escritorio o según lo que se especifique como ubicación, vea la Figura 67.

Figura 62. *Creación de accesos rápidos de una aplicación*



Nota: Elaboración propia

Como lo puede notar en la Figura 67, se puede en ese momento establecer un icono a la aplicación, InstallShield, acoge el ejecutable se la sección anterior. Así como también, crear un acceso a un desinstalador.

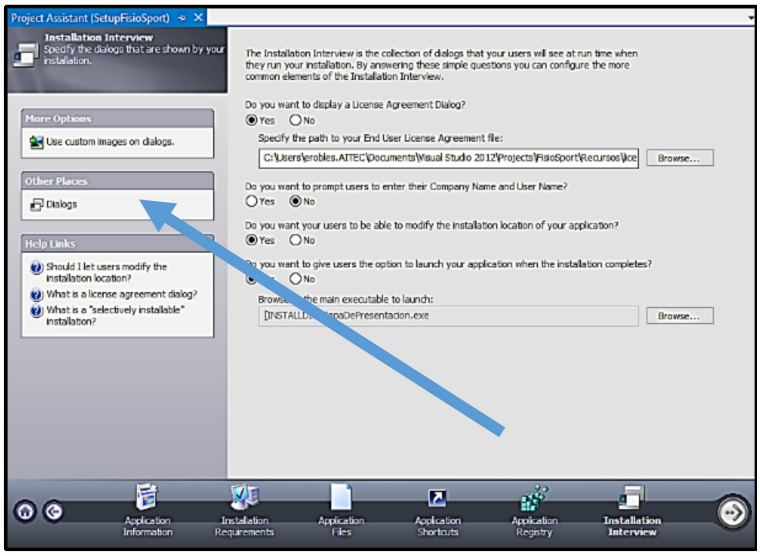
3.4.2.4. Registro de la aplicación

Al momento que llegamos a esta sección, podemos establecer información incluida en el registro de una computadora los cuales incluye los perfiles de usuario, el hardware y el software instalados en la computadora y la configuración de la propiedad, "El Registro hospeda información del sistema operativo, así como de las aplicaciones hospedadas en el equipo" (Dollard, 2015), la finalidad de un registro es que no se realicen aspectos inadecuados con el software y mantener un nivel de seguridad, para mayor información puede acceder aquí https://docs.microsoft.com/es-es/dotnet/visual-basic/developing-apps/programming/computer-resources/reading-from-and-writing-to-the-registry

3.4.2.5. Licencia y diálogos del empaquetado

Para finalizar nuestro empaquetado, se utiliza esta sección para los cuadros dialogo, licencia u otros aspectos finales al momento de que se inicializa la aplicación; inclusive algunos autores en este espacio establecen splash de su autoría, logos de la empresa, texto de licencia, entre otros a través del panel izquierdo de la ventana, ver la Figura 68.

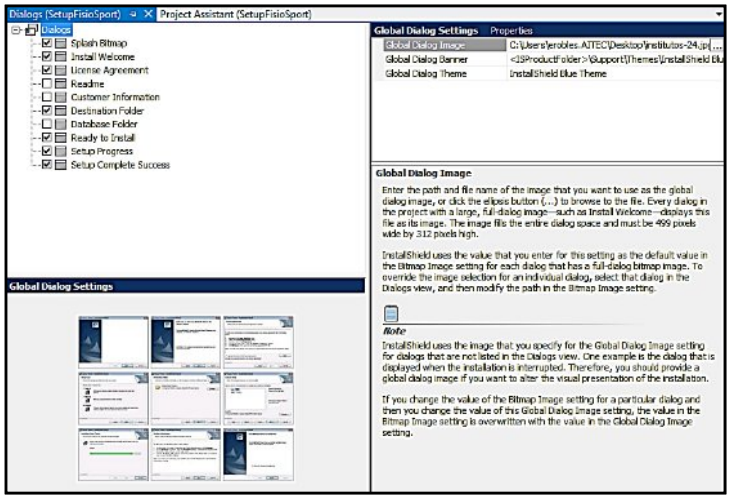
Figura 63. *Sección de cuadros diálogos finales de la aplicación*



Nota: Elaboración propia

Como puede notar en la Figura anterior, podemos dirigirnos a la poción de dialogos y de esta manera quitar o agregar los cuadros de diálogos para el setup según nuestro criterio, ver Figura 69.

Figura 64. *Opciones de dialogo de un empaquetado*



Nota: Elaboración propia

Como puede observar en la Figura anterior, usted puede agregar o quitar diálogos en el setup de la aplicación y en el lado derecho modificar sus propiedades según requerimiento del usuario, ejemplo: cambio logo, splash, licencia, escritos en los diálogos, etc.

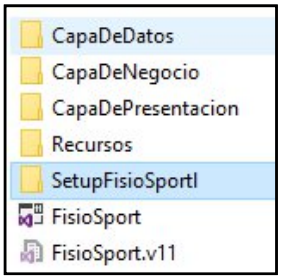
Para finalizar nuestro proyecto de empaquetamiento, solo tendremos que generar el mismo y de ésta forma realizar la compilación y creación del empaquetado, ver Figura 70.

Figura 65. *Generación del proyecto, para la creación del Setup de la aplicación*

Nota: Elaboración propia

Después de generar, se compilará el proyecto y creara el setup con alas especificaciones realizadas en todo el proyecto. El setup lo podrá encontrar dentro de la solución con un nombre de directorio igual al del proyecto, vea la Figura 71.

Figura 66. *Directorio de los proyectos de nuestra solución*



Nota: Elaboración propia

Dentro de la carpeta mostrada en la Figura podrá encontrar el setup de su aplicación en \SetupFisioSportl\SetupFisioSport\Express\DVD-5\DiskImages\DISK1 y allí se encontrará lo requerido, ver Figura 72.

Figura 67. *Contenido del aplicativo del empaquetado de la aplicación,*

Nota: Elaboración propia

ACTIVIDADES PARA DESARROLLAR EN LA UNIDAD 3

1. **Con el enlace:**

Saisang , C., & Dollard, K. (Julio de 2015). Leer y escribir en el Registro en Visual Basic. Obtenido de Microsoft Docs: https://docs.microsoft.com/es-es/dotnet/visual-basic/developing-apps/programming/computer-resources/reading-from-and-writing-to-the-registry

Conteste:

Especifique de forma puntual, ¿Cuáles son las funciones que tienen los parámetros dentro de un informe?

Al momento de crear parámetros, ¿De qué tipo podemos encontrar?

¿Cuáles son las opciones que debería considerar para la selección de un parámetro?

¿Qué tipo de aspectos podemos encontrar al momento de que un informes es publicado?

2. Diseñe un reporte del caso de estudio planteado, que permita determinar el número de veces que ha asistido a una consulta médica dentro de un mes.
3. Diseñe un reporte del caso de estudio planteado, mostrar la información anterior a través de un gráfico de barras.
4. Diseñe un reporte del caso de estudio planteado, que subcategorice el tipo de medicamento aplicado según las visitas realizadas.

BIBLIOGRAFÍA

- **Bibliografía básica (biblioteca virtual)**

Pérez Rodríguez, M. D. (Coord.). (2012). Desarrollo de elementos software para gestión de sistemas: (2 ed.). Editorial ICB. https://elibro.net/en/lc/aitec/titulos/106375

Otálora-Luna, J. E. Callejas-Cuervo, M. & Alarcón-Aldana, A. C. (2018). Metamodelo de medición de esfuerzo en proyectos de desarrollo de software: (ed.). Editorial UPTC. https://elibro.net/en/lc/aitec/titulos/132832

Huércano Ruíz, F. & Villar Cueli, J. (2016). Desarrollo de componentes software para servicios de comunicaciones: UF1288: (ed.). IC Editorial. https://elibro.net/en/lc/aitec/titulos/44520

Guillamón Morales, A. (2013). Manual desarrollo de elementos software para gestión de sistemas: (ed.). Editorial CEP, S.L. https://elibro.net/en/lc/aitec/titulos/50603

- **Bibliografía complementaria**

Barraza, F. (Agosto de 2013). *Modelado y Diseño de Arquitectura de Software.* Departamento de Electrónica y Ciencias de la Computación: http://decc.javerianacali.edu.co/wiki/lib/exe/fetch.php?media=materias:s2_conceptosdemodelado.pdf

Barraza, F. (s.f.). *Departamento de Electrónica y Ciencias de la Computación.* Modelado y Diseño de una Arquitectura de Software: http://decc.javerianacali.edu.co/wiki/lib/exe/fetch.php?media=materias:s2_conceptosdemodelado.pdf

Blanco, L. (2002). *Programación en Visual Basic .Net.* Madrid: Eidos.

Casanovas, J. (Septiembre de 2004). *Usabilidad y arquitectura del software*. DesarrolloWeb.com: https://desarrolloweb.com/articulos/1622.php

Dollard, K. (Julio de 2015). *Leer y escribir en el Registro Visual Basic*. Microsoft Docs: https://docs.microsoft.com/es-es/dotnet/visual-basic/developing-apps/programming/computer-resources/reading-from-and-writing-to-the-registry

Fontela, C. (2011). *UML: Modelado de software para profesionales* (Primera ed.). Buenos Aires, Argentina: Alfaomega.

Ghanayem, M. (mayo de 2018). *Informes, elementos de informe y definiciones de informe*. Microsoft Docs: https://docs.microsoft.com/es-es/sql/reporting-services/report-design/reports-report-parts-and-report-definitions-report-builder-and-ssrs?view=sql-server-2017

Ghanayem, M. (Mayo de 2018). *Informes, elementos de informe y definiciones de informe (Generador de informes y SSRS)*. Microsoft Docs: https://docs.microsoft.com/es-es/sql/reporting-services/report-design/reports-report-parts-and-report-definitions-report-builder-and-ssrs?view=sql-server-2017

Herrera, Y. R. (Noviembre de 2004). *Componentes DCOM*. Monografias: https://www.monografias.com/trabajos16/componentes/componentes.shtml

Hyderabad, I. (1998). *Software without Borders I IBM, 54, 55, 67, 108, 143 IDEs, See interactive development environments IEEE Standard for Application and Management of the Systems Engineering Process (IEEE Std 1220-1998), 137 IEEE/ANSI Guide to Software Requirements Specific*. IEEE Standards Association logo: https://standards.ieee.org/reading/ieee/std_public/description/se/610.12-1990_desc.html

Jeffrey, P. R. (Diciembre de 2018). Diseño de una aplicación para el seguimiento clínico y control de pacientes en el consultorio de fisioterapia "Fisiosport". *Instituto Tecnológico Superior "Almirante Illingworth*. Guayaquil.

Marin, E. (Octubre de 2012). *El Modelo Cliente/Servidor*. Any system is as secure as the weakest of its parts. – ISSAF: https://www.linuxito.com/docs/el-modelo-cliente-servidor.pdf

Microsoft. (Noviembre de 2018). *Crear y configurar conjuntos de datos en Visual Studio*. Microsoft Docs: https://docs.microsoft.com/es-es/visualstudio/data-tools/create-and-configure-datasets-in-visual-studio?view=vs-2019

Microsoft. (9 de Septiembre de 2018). *Niveles de acceso en Visual Basic*. Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/declared-elements/access-levels

Pons, C., Giandini, R., & Pérez, G. (2010). *Desarrollo de Software dirigido por Modelos* (Primera ed.). Buenos Aires, Argentina: Edulp. http://sedici.unlp.edu.ar/bitstream/handle/10915/26667/Documento_completo__.pdf?sequence=1&isAllowed=y

Reynoso, C. B. (Marzo de 2004). *Introducción a la Arquitectura de Software.* Carlos Reynoso: Investigación, Publicaciones y Cursos de Antropología, Ciencia Cognitiva y Complejidad: http://carlosreynoso.com.ar/archivos/arquitectura/Arquitectura-software.pdf

Schofield, M. (Marzo de 2019). *Empaquetar una aplicación para UWP con Visual Studio*. Microsoft Docs: https://docs.microsoft.com/es-es/windows/uwp/packaging/packaging-uwp-apps

Shaw, M., & Garlan, D. (Junio de 2005). *Formulations and Formalisms in Software.* Springer Link: https://link.springer.com/chapter/10.1007/BFb0015251

Suarez, N. (2017). *Desarrollo de software a tres capas* . Ambiente de aprendizaje - AmViNet: https://infonessuper.jimdo.com/app/download/9156989369/Software+A+Tres+Capas.pdf?t=1506948313&mobile=1.

Universidad Simón Bolivar. (Octubre de 2007). *Arquitectura de Software.* Departamento de Computación y Tecnología de la Información: https://ldc.usb.ve/~mgoncalves/IS2/sd07/clase7.pdf

Valle, R. J., & Granados, J. P. (2007). *Programación en Capas.* http://www.di-mare.com/adolfo/cursos/2007-2/pp-3capas.pdf

Vargas Del Valle, R., & Maltés Granados , J. (Febrero de 2007). *Programación en capas.* Revista Acta Académica: http://www.di-mare.com/adolfo/cursos/2007-2/pp-3capas.pdf

Vendrell, M. C. (Junio de 2011). *Sistemas cliente-servidor vs Sistemas multi-capa.* El Tamiz: https://eltamiz.com/elcedazo/2010/06/24/sistemas-cliente-servidor-vs-sistemas-multi-capa/

AITEC INSTITUTO SUPERIOR UNIVERSITARIO ALMIRANTE ILLINGWORTH

GUIA DE ESTUDIO

Sistemas Cliente- Servidor con .net

ISBN: 978-9942-7122-4-0

9 789942 712240