UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE

INGENIERO EN SISTEMAS COMPUTACIONALES

TEMA:

"DESARROLLO DEL SISTEMA ACADÉMICO APLICANDO LA HERRAMIENTA ASP.NET CORE 2 PARA LA ESCUELA EUFRASIA PELLETIER"

AUTOR:

DIEGO ARMANDO QUELAL ENRÍQUEZ

DIRECTOR:

ING. MSC, DIEGO JAVIER TREJO ESPAÑA.

IBARRA-ECUADOR

2019

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE.

IDENTIFICACIÓN DE LA OBRA.

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información.

DATOS DEL CONTACTO	
Cédula de identidad	100423800-0
Apellidos y Nombres	Quelal Enríquez Diego Armando
Dirección	Otavalo – González Suárez
E-mail	daquelale@utn.edu.ec
Teléfono móvil	0997840930
DATOS DE LA OBRA	
Título	"DESARROLLO DEL SISTEMA ACADÉMICO APLICANDO LA HERRAMIENTA ASP.NET CORE 2 PARA LA ESCUELA EUFRASIA PELLETIER"
Autor	Quelal Enríquez Diego Armando
Fecha	06 de marzo de 2019

Programa	Pregrado
Título	Ingeniero en Sistemas Computacionales
Director	ING. MSC, DIEGO JAVIER TREJO ESPAÑA

AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD.

Yo, Quelal Enríquez Diego Armando, con cédula de identidad Nro. 100423800-0, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad de material y como apoyo a la educación, investigación y extensión, en concordancia con la ley de Educación Superior Articulo 144.

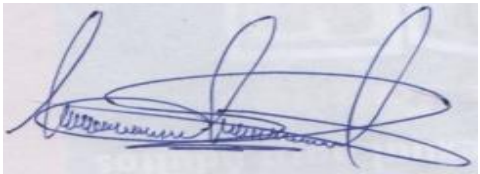
CONSTANCIA

Yo, Quelal Enríquez Diego Armando declaro bajo juramento que el trabajo aquí escrito es de mi autoría; y que este no ha sido previamente presentado para ningún grado o calificación profesional y que he consultado las referencias bibliográficas que se presentan en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Universidad Técnica del Norte, según lo establecido por las leyes de propiedad intelectual, reglamentos y normatividad vigente de la Universidad Técnica del Norte.

En la ciudad de Ibarra, a los 6 días del mes marzo de 2019.

EL AUTOR



……..

Quelal Enríquez Diego Armando

CI: 100423800-0

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE.

Yo, Quelal Enríquez Diego Armando, con cédula de identidad Nro. 100423800-0, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador artículos 4, 5 y 6, en calidad de autor del trabajo de grado con el tema: “Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier”. Que ha sido desarrollado con propósito de obtener el título de Ingeniero en Sistemas Computacionales de la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia suscribo en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

……

Quelal Enríquez Diego Armando

CI: 100423800-0

Ibarra, 6 de marzo de 2019.

CERTIFICACIÓN DIRECTOR

Ibarra, 27 de febrero de 2019

CERTIFICACIÓN DIRECTOR DEL TRABAJO DE TITULACIÓN

Por medio de la presente yo Diego Javier Trejo España certifico que el señor Diego Armando Quelal Enríquez con CI Nro.100423800-0 ha trabajado en el desarrollo del trabajo de grado "**DESARROLLO DEL SISTEMA ACADÉMICO APLICANDO LA HERRAMIENTA ASP.NET CORE 2 PARA LA ESCUELA EUFRASIA PELLETIER**", previo a la obtención del título de ingeniero en Sistemas Computacionales realizándolo en su totalidad con interés profesional y responsabilidad.

Es todo cuanto puedo certificar en honor a la verdad.

Ing.Msc, Diego Trejo
DIRECTOR DE TRABAJO DE GRADO

CARTA DE AUSPICIO

ESCUELA DE EDUCACIÓN BÁSICA
"EUFRASIA PELLETIER"

"Procuren tener los mismos sentimientos y pensamientos de Jesús buen pastor que va tras la oveja perdida"

Oficio N°15EPC-2017

Quito, 31 de enero 2018

ASUNTO: AUTORIZACIÓN

Ingeniero
Pedro Granda
COORDINADOR DE LA CARRERA DE INGENERIA EN SISTEMAS COMPUTACIONALES DE LA UTN
Presente.-

Reciba un cordial saludo de quienes conformamos la Escuela de Educación Básica Eufrasia Pelletier" .

A petición del Sr. **DIEGO ARMANDO QUELAL ENRIQUEZ** con Cédula de Identidad N° 1004338000, Estudiante De la Carrera de Ingeniería en Sistemas Computacionales, se autoriza al mencionado señor para que desarolle su trabajo de grado denominado **DESARROLLO DE SISTEMA ACADÉMICO CON ASP CORE EN LA ESCUELA DE EDUCACIÓN BÁSICA EUFRASIA PELELTIER" .** Confíamos que sera un gran aporte para nuestra Institución en beneficio de nuestros niños/as y adolescentes.

Nos despedimos de Ud. Confiando que Jesús Buen Pastor bendiga su trabajo y familia.

ESCUELA DE EDUCACIÓN BÁSICA "EUFRASIA PELLETIER" COORDINACIÓN EDUCATIVA Dir: Huacho E2-63 y José Peralta Telf:. 2652 861

Atentamente,

Carlos Teca
COORDINADOR EDUCATIVO

RELIGIOSAS DEL BUEN PASTOR
Contactos: eufrasia.pelletier@gmail.com / Telf.: 265286117
Dirección: Huacho E2-63 y José Peralta
QUITO – ECUADOR

CETIFICACIÓN DE TERMINACIÓN DEL PROYECTO

ESCUELA DE EDUCACIÓN BÁSICA
"EUFRASIA PELLETIER"

"Procuren tener los mismos sentimiento y pensamientos de Jesús buen pastor que va tras la oveja perdida" S.M.E.

CERTIFICACIÓN

Quito, 22 de enero de 2019

Señores
UNIVERSIDAD TÉCNICA DEL NORTE
Presente.-

De mi consideración. -

Siendo auspiciante del proyecto de tesis del estudiante **DIEGO ARMANDO QUELAL ENRÍQUEZ** con CI: **100423800-0**, para desarrollar e implementar su trabajo de grado

con el tema **"DESARROLLO DEL SISTEMA ACADÉMICO APLICANDO LA HERRAMIENTA ASP.NET CORE 2 PARA LA ESCUELA EUFRASIA PELLETIER"**, me es grato informar que

Se han superado con satisfacción las pruebas técnicas y la revisión de cumplimiento de los requerimientos funcionales, por lo que se recibe el proyecto como culminado y realizado por parte del estudiante DIEGO ARMANDO QUELAL ENRÍQUEZ. Una vez que hemos recibido la capacitación y documentación respectiva, nos comprometemos a continuar utilizando el mencionado aplicativo en beneficio de nuestra institución.

El estudiante DIEGO ARMANDO QUELAL ENRÍQUEZ, puede hacer uso de este documento

para los fines pertinentes en la Universidad Técnica del Norte.

Atentamente,

Lcdo. Carlos Teca
COORDINADOR EDUCATIVO

ESCUELA DE EDUCACIÓN BÁSICA
"EUFRASIA PELLETIER"
COORDINACIÓN EDUCATIVA
Dir: Huacho E2-63 y José Peralta
Telf.. 2652 861

RELIGIOSAS DEL BUEN PASTOR
Contactos: eufrasia.pelletier@gmail.com / Telf.: 2652861
Dirección: Huacho E2-63 y José Peralta
QUITO – ECUADOR

ACTA ENTREGA RECEPCIÓN

ESCUELA DE EDUCACIÓN BÁSICA
"EUFRASIA PELLETIER"

"Procuren tener los mismos sentimiento y pensamientos de Jesús buen pastor que va tras la oveja perdida" S.M.E.

ACTA DE ENTREGA – RECEPCIÓN

Una vez realizadas las pruebas correspondientes al sistema académico para la escuela Eufrasia Pelletier, se procede a hacer la entrega del mencionado proyecto, que en la actualidad se encuentra en producción publicado en la siguiente dirección:

http://sistemaacademico.apphb.com/ acceso público.

http://sistemaacademico.apphb.com/ acceso interno.

INFORMACIÓN DEL SISTEMA

SISTEMA DE INFORMACIÓN	SISTEMA ACADÉMICO APLICANDO LA HERRAMIENTA ASP.NET CORE 2 PARA LA ESCUELA EUFRASIA PELLETIER		
FECHA:	**VERSIÓN**	**RESPONSABLE**	**DESCRIPCIÓN**
22/Ene /2019	1.0	**Programador:** Diego Quelal	"DESARROLLO DEL SISTEMA ACADÉMICO APLICANDO LA HERRAMIENTA ASP.NET CORE 2 PARA LA ESCUELA EUFRASIA PELLETIER"

FIRMAS DE ACEPTACIÓN.

	QUIEN ENTREGA	**QUIEN RECIBE**
NOMBRE	Diego Quelal	Lcdo. Carlos Teca
DEPENDENCIA		Coordinación Académica
CARGO	TESISTA	Coordinador Educativo
FIRMA		

RELIGIOSAS DEL BUEN PASTOR
Contactos: eufrasia.pelletier@gmail.com / Telf.: 2652861
Dirección: Huacho E2-63 y José Peralta
QUITO – ECUADOR

DEDICATORIA

Este trabajo va dirigido con todo mi amor a Dios, creador de todo lo que nos rodea, el único y gran amigo que nunca falla pues es el testigo fiel de lo que ha significado mi vida universitaria; Él sabe de todos aquellos esfuerzos que realicé y las veces que quise dejar todo, pero Él me mantuvo firme y lleno de fuerza para este gran logro. No puedo dejar de nombrar a mis hijos, esposa y madre figuras fundamentales en cada actividad que realizo son la razón de mi existencia y el motivo por el que quiero ser mejor cada día, los amo Yordani, Yaretzi y Pame.

Diego Armando Quelal Enríquez

AGRADECIMIENTO

Después de un largo tiempo de trabajo conjunto, por fin ha llegado el gran día de culminar una de las etapas más importantes de mi vida, luego de tan grande sacrificio no me queda más que agradecer de manera infinita a mi tutor y asesores: Msc. Diego Trejo, Msc. Víctor Caranqui y Msc. Silvia Arciniega quienes me acompañaron durante todo el desarrollo de esta investigación aportando con todo su talento y conocimiento; a cada uno de mis docentes quienes compartieron conmigo grandes saberes científicos y éticos. A mi querida Universidad quien me acogió en sus aulas para hacer de mí una profesional de calidad.

Por último, un agradecimiento especial a mi familia quien ha caminado conmigo apoyándome en todo cuanto ha estado a su alcance.

Diego Armando Quelal Enríquez

RESUMEN

Con la evolución de la tecnología y la construcción de software las empresas que venden plataformas de desarrollo, de igual forma han ido innovando y poniendo sus esfuerzos en crear nuevas herramientas para estar actualizadas con era moderna y en el mercado competitivo.

Este es el caso de Microsoft que en 2017 lanza una nueva herramienta de software libre y multiplataforma llamada ASP.NET CORE 2 que soporta los sistemas operativos macOS, Windows y Linux bajo los servidores web Nginx, Apache, IIS y Windows Service.

Generalmente Microsoft únicamente creaba software en su gran mayoría de tipo propietario, en decir con licencia de uso con su respectivo costo, pero con esta herramienta que lanzo en 2017, creó una revolución en cuestión de competitividad con otras herramientas de desarrollo que muchos las preferían por el licenciamiento y por su código fuente libre pero hoy Microsoft también está bajo este paradigma con uno de sus mejores lenguajes de programación C#.

En la sección de antecedentes se muestra una visión general del presente proyecto, en el que se tratará a detalle el problema, los objetivos, el alcance y la justificación.

Capítulo I, Detalla las ventajas de ASP.NET CORE 2 su uso, las características más importantes, también trata SQL server su uso y características y por último detalla lo que es AppHArbor.

El Capítulo II, Muestra de forma rápida el proceso de construcción e implementación de la solución web con la metodología de desarrollo ágil XP (Extreme Programming).

El Capítulo III, Analiza y valida el impacto que genera la implementación de la solución web en la institución.

ABSTRACT

With the evolution of technology and the construction of software companies that sell development platforms, they have also been innovating and putting their efforts in creating new tools to be updated with modern era and in the competitive market.

This is the case of Microsoft that in 2017 launches a new free and multiplatform software tool called ASP.NET CORE 2 that supports the macOS, Windows and Linux operating systems under the Nginx, Apache, IIS and Windows Service web servers.

Generally Microsoft only created software in its great majority of proprietary type, in saying with license of use with its respective cost, but with this tool that launched in 2017, it created a revolution in competitiveness with other development tools that many preferred them over licensing and its free source code but today Microsoft is also under this paradigm with one of its best C # programming languages.

In the section of antecedents a general vision of the present project is shown, in which the problem, the objectives, the scope and the justification will be treated in detail.

Chapter I, Details the advantages of ASP.NET CORE 2 its use, the most important features, also treats SQL server its use and characteristics and finally details what is AppHArbor.

Chapter II, Quickly shows the process of construction and implementation of the web solution with the agile development methodology XP (Extreme Programming).

Chapter III, Analyzes and validates the impact generated by the implementation of the web solution in the institution.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Antecedentes

En los primeros tiempos de la computación cliente-servidor “cada aplicación tenía su propio programa cliente que servía como interfaz de usuario que tenía que ser instalado por separado en cada ordenador personal de cada usuario” (Téllez, 2014). El cliente realizaba peticiones a otro programa (servidor) que daba respuesta. Una mejora en el servidor, como parte de la aplicación, requería normalmente una mejora de los clientes instalados en cada ordenador personal, añadiendo un coste de soporte técnico y disminuyendo la productividad.

A diferencia de lo anterior, las aplicaciones Web generan dinámicamente una serie de páginas en un formato estándar como ASP, ASPX, ASP CORE, HTML o XHTML, soportados por los navegadores web comunes (Ollero Sánchez, 2014). Se utilizan lenguajes interpretados en el lado del cliente, directamente o a través de plugins tales como JavaScript, Java, Flash, etcétera; para añadir elementos dinámicos a la interfaz de usuario. Generalmente cada página web en particular se envía al cliente como un documento estático, pero la secuencia de páginas ofrece al usuario una experiencia interactiva. Durante la sesión, el navegador web interpreta y muestra en pantalla las páginas, actuando como cliente para cualquier aplicación web (Valencia Pavón, 2014).

Además, la Escuela Eufrasia Pelletier es nueva y desea estar a la vanguardia de los servicios académicos con el uso de la tecnología y pretende implementar un Sistema Académico para la institución con una herramienta como Microsoft ASP.NET CORE 2 que promete mucho ya que es multiplataforma y de software libre bajo Internet Information Server Express para ejecutar las Aplicaciones.

Situación actual

Actualmente, la institución no cuenta con ningún sistema académico por lo que cumplir con los requerimientos que demandan los docentes, estudiantes, padres de familia y el personal administrativo resultan muy complicados, además de tener fallas técnicas causando retrasos en los informes mensuales, reportes de datos estadísticos, los mismos que se los realiza de forma manual o en Excel, provocando retrasos y pérdida de tiempo, desconocimiento del grado de aceptación por parte del estudiante o padre de familia hacia el docente y la institución.

La Herramienta ASP.NET CORE 2 es una herramienta nueva que no ha sido investigada por nuestra universidad ni por otras, la misma garantiza efectividad de procesos tecnológicos, lo que conlleva a inquirir sobre la misma, para adquirir todos sus beneficios.

Desde el punto de vista tecnológico, la educación va de la mano con el acceso a la información y con la gestión apropiada de la misma. Es por eso que cada paso que se dé dentro del progreso académico se vea reflejado en adelantos en la parte tecnológica, como apoyo ante el mayor afluente de datos que se podría manejar.

El proceso de desarrollo del sistema de gestión académica web para para la Escuela Eufrasia Pelletier, es con la finalidad de optimizar los procesos manuales que se realizan dentro de la institución que corresponde a la descripción de alumnos y personal docente, gestión de notas, matriculación y demás actividades concernientes a la administración de la información de los estudiantes.

Se aprovechará las facilidades que brinda en la actualidad las aplicaciones web, así como también la potencia de ASP.NET CORE 2, adaptado al sistema y a las nuevas políticas gubernamentales.

De acuerdo con la metodología que se utilizará para el desarrollo de sistemas el cual recurre a los requerimientos obtenidos en XP.

La Escuela Eufrasia Pelletier no cuenta con un Sistema Académico para gestionar sus procesos e información; tampoco cuenta con Infraestructura para alojamiento de aplicaciones por lo que se pretende el desarrollo de este proyecto.

Prospectiva

Con el uso de la herramienta ASP.NET CORE 2 de Visual Studio 2017 se va a poder generar una aplicación web aprovechando en gran medida el potencial de dicha herramienta, además mejorará los servicios a todos los involucrados en la institución educativa y dará solución a sus problemas ya que el sistema tendrá un conjunto de opciones prácticas que ayudará en la gestión académica, brindando así un mejor servicio a la comunidad institucional ya que se gestionará sus procesos e información de forma automática mediante esta nueva tecnología, con la que estará desarrollado el Sistema Académico para las personas dentro y fuera de la institución.

Definición del problema

¿Por qué no hay Sistema Académico con ASP.NET CORE 2 en Eufrasia Pelletier?

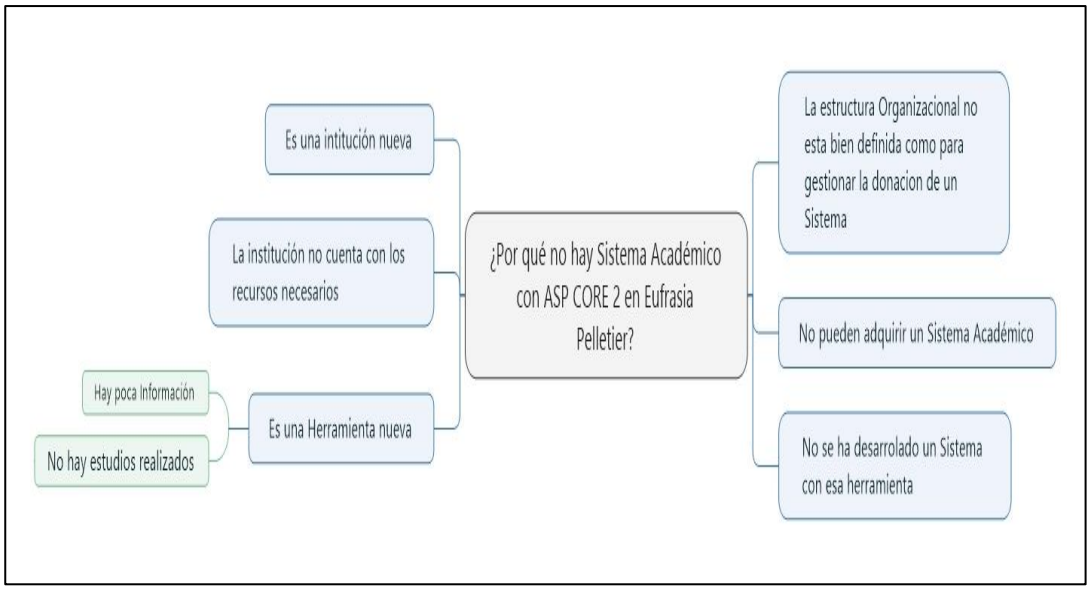


Figura 1. Árbol de problemas
Fuente: Propia

Objetivo general

- Desarrollar una solución web para la gestión académica de la Escuela Eufrasia Pelletier, aplicando la herramienta multiplataforma y de código abierto ASP.NET CORE 2.

Objetivos específicos

- Investigar la arquitectura, los beneficios y el uso de la Herramienta ASP.NET CORE 2.
- Diseñar una propuesta para la aplicación del Sistema de Gestión Académica.
- Ejecutar el proceso de puesta en marcha del sistema en la escuela Eufrasia Pelletier.
- Analizar Impactos de manera prospectiva en relación con la implantación del Sistema.

Alcance

El sistema podrá gestionar y administrar la información de los estudiantes y docentes, se podrán realizar matriculas, pases de nivel, ingresar notas, visualizar el rendimiento académico incluso el porcentaje de asistencia por estudiante estará en la capacidad de generar informes automáticos no solo por alumno sino también por curso, Todos estos datos se los podrá observar desde la web en cualquier parte del mundo con tan solo hacer un clic ya estará alojado en un hosting gratuito, beneficiando de esta manera directamente a la Escuela Eufrasia Pelletier.

Justificación

Impacto tecnológico

(Microsoft, Introduction to ASP.NET Core, 2017) “ASP.NET Core es un nuevo framework de código abierto y multiplataforma para la creación de aplicaciones modernas conectadas a Internet, como aplicaciones web y APIs Web está diseñado para proporcionar un framework de desarrollo optimizado para las aplicaciones que se implementan tanto en la nube como en servidores dedicados en las instalaciones del cliente además se pueden desarrollar y ejecutar aplicaciones ASP.NET Core en Windows, Mac y Linux por lo puede ejecutarse sobre el framework .NET completo o sobre .NET Core.” Por lo expuesto el aporte que se quiere es dar a conocer esta herramienta.

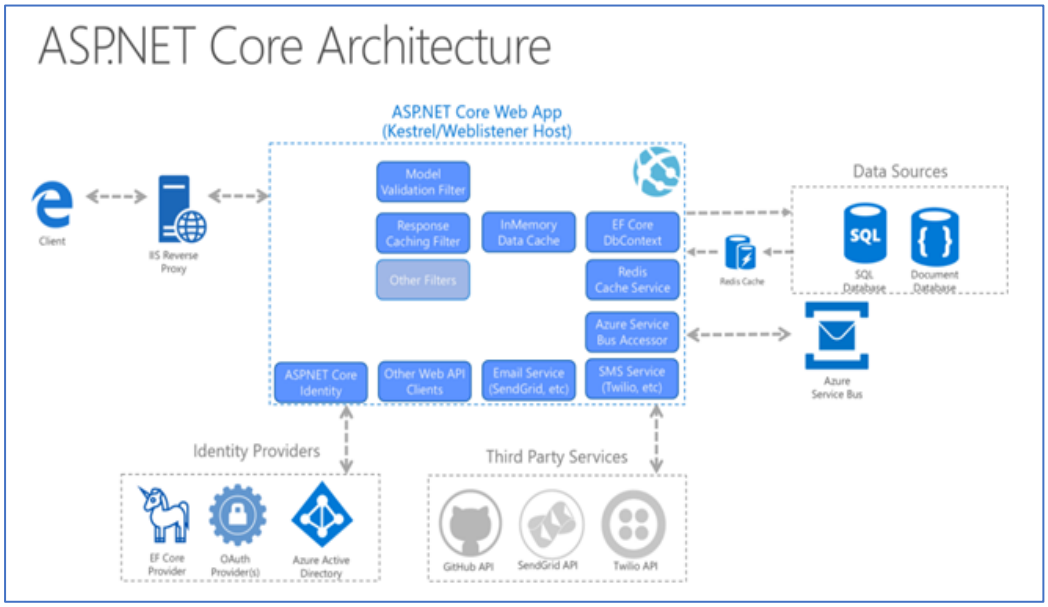


Figura 2: Arquitectura de ASP.NET CORE
Fuente: Web Applications with ASP.NET Core

Impacto Social

Como la Escuela Eufrasia Pelletier, no cuenta con ningún Sistema Académico se pretende aprovechar al máximo los beneficios de la herramienta antes descrita, para mitigar los problemas que existen en la institución y dejar un pequeño estudio práctico de esta novedosa herramienta para uso de la comunidad.

Impacto Ambiental

La implantación del Sistema Académico ayudará al medio ambiente ya que no utilizará hojas de papel normal de impresión, si no que se utilizará hojas hecha a base de caña de azúcar para preservar el medio, además como el Sistema permitirá la consulta de notas online se reducirá el consumo de papel.

Impacto Económico

Con el Sistema Académico la escuela Eufrasia Pelletier tendrá un ahorro de dinero, ya que se entiende, que una solución de este tipo gestiona los procesos, recursos humanos e información de padres de familia y estudiantes, por lo que la gestión resulta más eficiente conllevando a una reducción tanto en tiempo como dinero, por la adecuada gestión del personal que labora en la institución.

CAPÍTULO I

1. MARCO TEÓRICO

1.1. ASP.NET CORE 2

Para iniciar con el estudio diremos que la mayoría de la información recabada se ha tomado de los blogs, foros y paginas oficiales de Microsoft, empresa que está financiando directamente el proyecto de ASP.NET Core, que es un marco de código abierto, que permite su uso en multiplataformas, además, que ha mostrado ser de alto rendimiento para la construcción de aplicaciones modernas, basadas en la nube, conectadas a Internet además de crear aplicaciones y servicios web, también aplicaciones de IoT[1] y backends[2] móviles, por consiguiente el uso de la herramienta de desarrollo puede ser utilizada y ejecutada en los sistemas operativos Windows, MacOS y Linux. Así pues, la herramienta en estudio permite implementar y desarrollar en la en la nube de Microsoft o en las instalaciones multiplataforma que se ofrece en la página oficial (Microsoft, Introduction to ASP.NET Core, 2017).

Las aplicaciones, Web APIs[3] o bibliotecas de clases, que se generen con ASP.net CORE 2 pueden ser ejecutadas dentro del entorno de .NET Core o .NET Framework bajo el servidor Internet Information Server Express (IIS Express) o Internet Information Server (IIS) respectivamente además de Nginx y Apache (Anderson, Microsoft, 2018).

A continuación de muestra una gráfica de cómo trabaja la herramienta en las distintas plataformas soportadas.

1 "La internet de las cosas (IoT, por sus siglas en inglés) es un sistema de dispositivos de computación interrelacionados, máquinas mecánicas y digitales, objetos, animales o personas que tienen identificadores únicos y la capacidad de transferir datos a través de una red, sin requerir de interacciones humano a humano o humano a computadora" (Rouse, 2017).

2 "Es la labor de ingeniería que compone el acceso a bases de datos y generación de plantillas del lado del servidor" (Freddie, 2014).

3 "Es un marco que facilita la creación de servicios HTTP disponibles para una amplia variedad de clientes, entre los que se incluyen exploradores y dispositivos móviles" (Microsoft, Msdn, 2018)

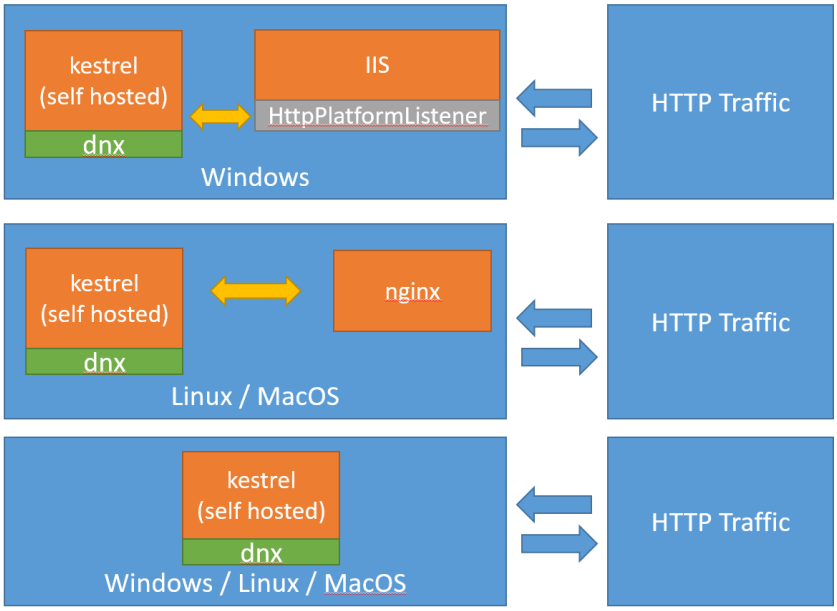


Figura 3: Plataformas que soporta ASP.NET CORE
Fuente: (Posadas, 2017)

Sin duda, su aparición ha supuesto el cambio más grande para las tecnologías de desarrollo web de Microsoft desde sus inicios, y tanto es así que, de hecho, es un producto totalmente nuevo, escrito desde cero que muestra un enfoque muy distinto al antiguo ASP.Net con formularios web que es una tecnología de Microsoft del tipo "lado del servidor" para páginas web generadas dinámicamente (Danysoft, 2017).

1.1.1 BENEFICIOS

Multiplataforma: ASP.NET Core brinda funciones específicas para la implementación de las distintas tipologías de la app que requiere además de poder volver a utilizar las mismas líneas de programación en las distintas plataformas que permite dicha herramienta. Actualmente, es las aplicaciones hechas en .NET CORE se las puede ejecutar sobre tres sistemas operativos: Windows, Linux y macOS. Permite implementar soluciones y APIs sin necesidad de realizar alguna modificación en los sistemas operativos que la herramienta lo soporta. Para ver una lista de todos los sistemas operativos compatibles.

Código abierto: Esta herramienta de desarrollo es una de las muchas que está financiada por .NET Foundation[4] por lo tanto su código fuente se ofrece en GitHub. Como se

4 Organización independiente, que se fundó el 31 de marzo de 2014, por la misma empresa Microsoft, que se enfocó en mejorar el desarrollo de software para código abierto y la colaboración en torno a .NET Framework (Wikipedia, 2017)

trata de un propósito de código abierto, ASP .NET Core ayuda a que el asunto de construcción sea mucho más limpio y que esté bajo vigilancia de una comunidad impulsadora y que está comprometida con la herramienta.

Desarrollo flexible: Basándose en que existen dos formas esenciales para el desarrollo de aplicaciones, implementación autocontenida y desarrollo basado en marco. En la implementación basada en marco, únicamente se instala la solución y las dependencias de otros desarrolladores, y la solución necesita de una instalación de ASP.NET Core completa en el sistema. Cuando se realiza una ejecución autocontenida, la versión de ASP.NET Core que se tiene para compilar nuestra solución asimismo se implementa a la par con la solución y las dependencias de otros desarrolladores, además permite ejecutar en conjunto con otras intenciones.

Modular: ASP.NET Core es modular, porque se permite publicar mediante NuGet en paquetes de prototipo ensamblado de forma más reducida. Sustituyendo un ensamblado de mayor proporción que tiene la mayoría de las funciones básicas, ASP.NET Core se compone como pequeños paquetes que se centran en las distintas particularidades de este. Este tipo de características admite un modelo de creación de aplicaciones más rápido permitiendo mejorar su solución para integrar únicamente los paquetes de NuGet que se utilizará teniendo así una mejor seguridad, mejor rendimiento, poco mantenimiento y reducidos costos del modelo ajustado a sus necesidades (Wenzel, 2017).

El framework de desarrollo ASP.NET Core, está constituido por algunos elementos, de los que se integran los compiladores administrados, las bibliotecas de clases, numerosos modelos de aplicaciones y, base el entorno en tiempo de ejecución como ASP.NET Core (Wenzel, 2017).

ASP.NET CORE trabaja en un poderoso lenguaje de programación y todas las facilidades que este presenta desde hace años de su aparición se trata del C#.net con todas sus bibliotecas y código de desarrollo moderno de aplicaciones.

Servidores Web: Las aplicaciones, Web APIs[5] o bibliotecas de clases, que se generen con ASP.net CORE 2 pueden ser ejecutadas dentro del entorno de .NET Core o .NET Framework

[5] "Es un marco que facilita la creación de servicios HTTP disponibles para una amplia variedad de clientes, entre los que se incluyen exploradores y dispositivos móviles" (Microsoft, Msdn, 2018)

bajo el servidor Internet Information Server Express (IIS Express) o Internet Information Server (IIS) respectivamente además de Nginx y Apache (Anderson, Microsoft, 2018).

1.2. C#

Este famoso lenguaje de programación llamado C# (C Sharp) que 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++.

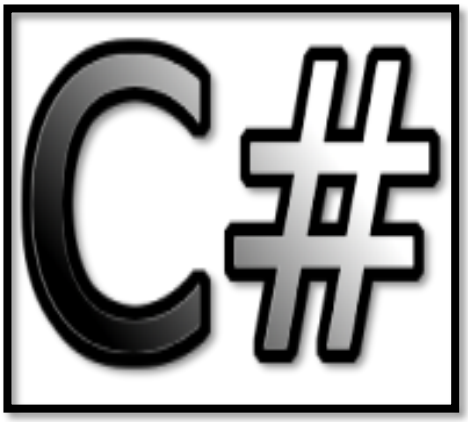


Figura 4: C#
Fuente: Propia

Con el C# se pretendió que incorporase las ventajas o mejoras que tiene el lenguaje JAVA. Así se consiguió que tuviese las ventajas del C, del C++, pero además la productividad que posee el lenguaje JAVA y se le denomino C#.

Algunas de las características del lenguaje de programación C# son: Su código se puede tratar íntegramente como un objeto. Su sintaxis es muy similar a la del JAVA. Es un lenguaje orientado a objetos y a componentes. Armoniza la productividad del Visual Basic con el poder y la flexibilidad del C++. Ahorramos tiempo en la programación ya que tiene una librería de clases muy completa y bien diseñada.

A pesar de 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. Esta plataforma se creó, entre otras razones, 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. C# es el nuevo lenguaje de propósito general orientado a objetos creado por Microsoft para su nueva plataforma .NET (Informatica, 2015).

C# como se mencionó se define como un lenguaje de programación orientado a objetos sencillo, vanguardia y estándar. Es más, en un comienzo del desarrollo de software la capacidad para crear módulos de programas era muy fácil ya que que se podía usar en ambientes distribuidos por lo que se puede decir que tiene:

- Portabilidad del código fuente
- Fácil migración del programador al nuevo lenguaje, especialmente para programadores familiarizados con C y C++.
- Soporte para internacionalización
- Adecuación para escribir aplicaciones de cualquier tamaño: desde las más grandes y sofisticadas como sistemas operativos hasta las más pequeñas funciones.
- Aplicaciones económicas en cuanto a memoria y procesado (Ecured, 2016).

1.3. APPHARBOR

AppHarbor es una plataforma .NET totalmente alojada como un servicio. AppHarbor puede implementar y escalar cualquier aplicación .NET estándar a la nube utilizado por miles de desarrolladores y empresas para alojar desde blogs personales hasta aplicaciones web de alto tráfico permite implementar y escalar instantáneamente aplicaciones .NET con su herramienta de control de versiones favorita. Instalar complementos es igual de fácil (Appharbor, Appharbor, 2018).

Es una plataforma .NET como servicio. Los desarrolladores empujan el código a AppHarbor usando Git o Mercurial. AppHarbor luego construye el código y ejecuta todas las pruebas unitarias. Si todo sale bien, el código se implementa en la plataforma de nube escalable de AppHarbor (Appharbor, Appharbor, 2018).

Además, les permite a los desarrolladores dedicar su tiempo a generar ideas y desarrollar aplicaciones, no a parchar servidores, preocuparse por la implementación, alterar los archivos de configuración o escalar. En AppHarbor, creemos que el desarrollo y la implementación de aplicaciones .NET deben ser simples y divertidos. AppHarbor ha sido diseñado para vincularse con los flujos de trabajo diarios de los desarrolladores. La implementación de una nueva versión en AppHarbor es tan simple como verificar el código cuando una nueva característica está lista (Appharbor, Appharbor, 2018).

AppHarbor ejecutará la mayoría de las aplicaciones ASP.NET sin modificaciones. Mover una aplicación a nuestra plataforma no requiere modificaciones al código específicas

de AppHarbor ni instalación de complementos u otro software personalizado. Creemos que los desarrolladores deben poder mover aplicaciones y datos a diferentes proveedores de la nube con la menor fricción posible, cree en dar a los desarrolladores acceso a un ecosistema abierto de complementos y servicios de terceros, no solo a los que decidimos proporcionarnos y fue fundada en septiembre de 2010 por Rune Sørensen, Troels Thomsen y Michael Friis (Appharbor, Appharbor, 2018).

AppHarbor se ofrece bajo la arquitectura PaaS, y nos permite correr aplicaciones ASP.NET en la nube. De forma sencilla podremos crear nuestras aplicaciones e integrarlas al repositorio de control de código fuente correspondiente. Uno de los aspectos característicos que tiene en su funcionamiento son los Add-ons. Por ejemplo, si queremos agregar un motor de base de datos a nuestra aplicación, el mismo será un Add-on. Hay una gran variedad de opciones de Add-ons, y dependiendo de las necesidades que tengamos habrá distintos planes comenzando desde opciones gratuitas hasta algunas pagas con características destacadas (BERSANO, 2014).

1.4. SQL SERVER

Microsoft SQL Server es un sistema de manejo de bases de datos relacionales que le permitirá programar en entornos híbridos, ya sea de forma local o en la nube de Microsoft. En combinación con Microsoft Azure, los elementos incorporados a SQL Server le proporcionan una fácil creación de soluciones ante problemas con las revisiones, los desastres y las copias de seguridad. Podrá, además, transferir bases de datos de una forma muy sencilla e intuitiva entre su entorno local y la nube. Es considerada como una de las bases de datos más seguras del mundo, por no decir la mejor, y su sistema de almacenamiento permite un rendimiento en las consultas muy superior al habitual. Todos los procesos de análisis, consulta, limpieza, formateo de datos y acceso se realizan a una velocidad que le sorprenderá (Makesoft, 2017).

La edición Enterprise le permitirá aprovechar absolutamente todas las ventajas de la herramienta. Tendrá acceso a todas las funcionalidades del centro de datos, pudiendo satisfacer así los requisitos de cualquier base de datos. Si adquiere la edición Business Intelligence, su organización podrá compilar e implementar soluciones corporativas seguras, escalables y administrables. Con la edición Standard podrá cubrir todas las necesidades que surjan al trabajar en un servidor con menos prestaciones. Tendrá acceso a las funciones Business Intelligence y a una administración de datos básica, que requieran de pocos recursos TI. También dispondrá de una edición gratuita, llamada Express. Con ella podrá

desarrollar e impulsar aplicaciones de web y de escritorio, así como otras sencillas de servidor (Makesoft, 2017).

1.4.1. FUNCIONES DE MICROSOFT SQL SERVER

Entre otras, podrá encontrar las siguientes:

- Respaldos y recuperaciones: se pueden configurar de antemano, lo que provoca que se ejecuten y lleven a cabo de forma sencilla (Makesoft, 2017)..
- Compresión: compresión extrema de tablas e índices (Makesoft, 2017)..
- Alta disponibilidad: el tiempo de inactividad causado por actualizaciones y revisiones desaparece o se minimiza, a fin de garantizar una disponibilidad casi permanente, exigida por el modelo de negocio global que impera a día de hoy. Dichas disponibilidades y estados de mantenimiento de las bases de datos se podrán comprobar de forma sencilla con un panel muy visual (Makesoft, 2017)..
- Programar tareas: con antelación se pueden programar tareas, que se ejecutarán automáticamente (Makesoft, 2017).

1.4.2. CARACTERÍSTICAS PRINCIPALES DE MICROSOFT SQL SERVER

- Rendimiento mejorado gracias a la funcionalidad in-memory para procesamiento de transacciones y mejoras del almacenamiento de datos (Makesoft, 2017)..
- Certificación SAP, lo que garantiza éxito en el trabajo con cargas muy pesadas.
- Tiempo de disponibilidad muy elevado y posibilidad de hacerlo en un entorno híbrido con máquinas virtuales de Microsoft Azure (Makesoft, 2017)..
- Cifrado de datos transparente, auditorías, administración de claves extensibles y copias de seguridad cifradas para proteger así los datos en las cargas de trabajo críticas (Makesoft, 2017)..
- Máxima flexibilidad para que pueda trabajar tanto en la nube como en un entorno local. El paso de uno a otro se da de una manera realmente sencilla, para que pueda aprovechar todas las ventajas de ambos ámbitos de trabajo (Makesoft, 2017).

1.5. METODOLOGÍA XP

Concebida pensando en el desarrollo de proyectos de software pequeños y medianos, es considerada una metodología de desarrollo de software ligera basada en la simplicidad, la retroalimentación y la reutilización de código. Esta metodología maneja la documentación

moderadamente, por lo que no es aconsejable su uso en proyectos en los que el cliente requiera documentación sumamente detallada (Enríquez, Repositorio UTN, 2017).

De acuerdo a Ramos, Noriega, Laínez y Durango (2015) menciona que:

La metodología XP surgió a partir de ideas de Kent Beck y Ward Cunningham y que fue utilizado por primera vez en un proyecto piloto en marzo de 1996, del cual el propio Beck formaba parte. Lo de Extreme del nombre de la metodología se debe al hecho de que ésta emplea al extremo, las buenas prácticas de la Ingeniería de Software.

Los principales objetivos que la metodología XP está orientada a satisfacer son:

- Aumentar la productividad del software.
- Potenciar el trabajo en grupo.
- Satisfacer de una forma rápido y con calidad las necesidades del cliente superando sus expectativas.

De acuerdo a Hernández Rodríguez (2014) XP utiliza una técnica denominada Historias de Usuario que son usadas para especificar los requisitos del software, son tarjetas en las cuales el cliente describe de forma rápida las características que el sistema debe poseer ya sean estos requisitos funcionales o no funcionales (Enríquez, Repositorio UTN, 2017).

XP propone ciclos del proyecto de software muy cortos y rápidos, realizando pruebas de unidad inmediatas y una integración continua, logrando así crear tantas pequeñas versiones o prototipos como sea posible, las cuales son probadas antes de continuar (Gómez Palomo & Moraleda Gil, 2014); gracias a que el sistema se va desarrollando incrementalmente el equipo de trabajo realiza mantenimiento del software y crea nuevas funcionalidades continuamente. A continuación, se puede observar un esquema básico de XP.

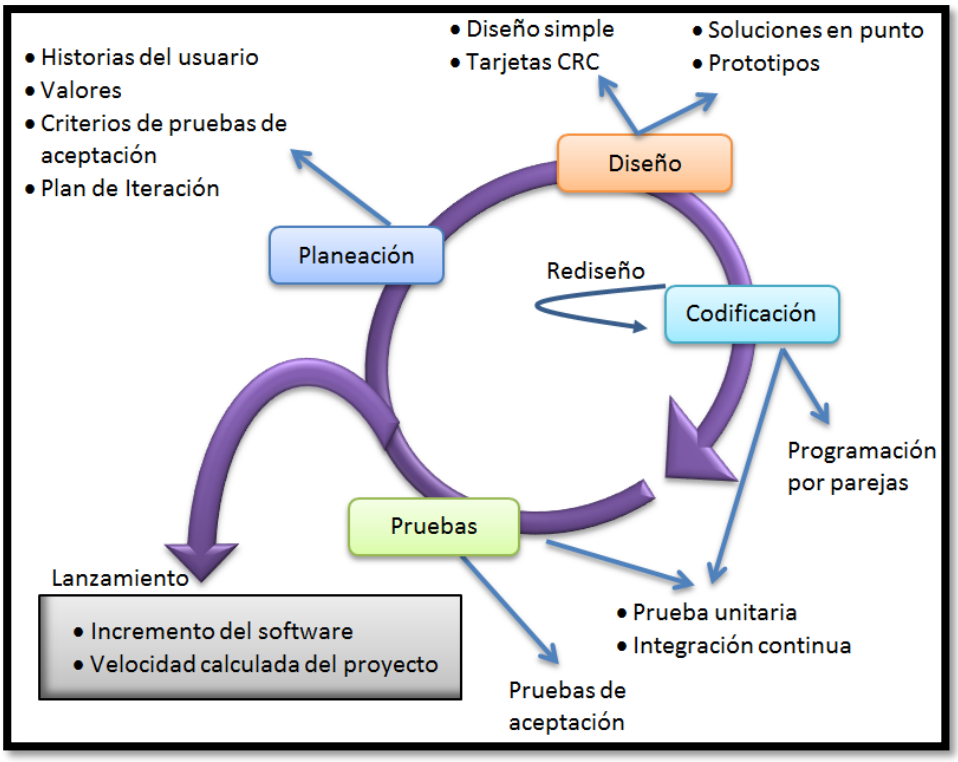


Figura 5: Extreme Programming (XP)
Fuente: https://www.codejobs.biz/www/lib/files/images/4e7e132bb7844ef.png

Debido a que la metodología promueve los valores entre el equipo de trabajo durante el desarrollo de los proyectos se puede considerar como una metodología filosófica. XP se define “por medio de valores, principios y prácticas. Los valores describen los objetivos de largo plazo y definen criterios para obtener el éxito. Los valores son: Feedback, Comunicación, Simplicidad, Coraje y Respeto” (Ramos, et al., 2015, p. 214).

El feedback o retroalimentación involucra tanto al cliente como al propio sistema; dentro del equipo de trabajo se integra al cliente ya que en muy poco tiempo se evalúa el diseño y se cambia si no cumple los requisitos establecidos. XP busca que la comunicación entre el cliente y el equipo de trabajo sea lo más directa posible logrando así que todos los pormenores del proyecto se cumplan con agilidad; sin embargo, por muy buena que sea la comunicación no es suficiente para lograr que el cliente pueda generar una retroalimentación rápidamente (Enríquez, Repositorio UTN, 2017).

La simplicidad hace alusión a que se debe implementar únicamente aquello que es suficiente para satisfacer las necesidades cambiantes del cliente, mediante la reutilización de código para conseguir que el software se mantenga sencillo, procurando dejar a un lado las preocupaciones acerca de lo que el cliente pueda pedir el día de mañana. Lo más simple es

lo mejor, funciona mejor, es más rápido, se puede adaptar con mayor facilidad y puede resultar más barato (Enríquez, Repositorio UTN, 2017).

El equipo debe tener coraje cuando se presenta el caso en que se debe realizar modificaciones a algo que ya está funcionando correctamente y estos cambios puedan producir fallos en el sistema. El equipo debe ser capaz de lograr que el software evolucione con seguridad y agilidad; afrontando la recodificación continúa del sistema, aceptando modificaciones de requerimientos, escuchando las peticiones del cliente, realizando pruebas o desechando código obsoleto (Enríquez, Repositorio UTN, 2017).

El respeto es un valor fundamental en todo momento; oír, comprender y respetar el punto de vista de los demás integrantes del equipo es vital para que el proyecto se desarrolle con la mayor armonía posible. Los valores descritos son valores vagos, necesitan ser transformados en principios concretos que puedan utilizarse, orientados a la elección de alternativas, cada principio incorpora los valores (Enríquez, Repositorio UTN, 2017).

1.5.1. VENTAJAS:

Extreme Programming ofrece una serie de ventajas al momento de ponerla en práctica:

- Disminuye la tasa de errores, se puede corregir errores antes de agregar nuevas funcionalidades al sistema.
- Lanzamiento de nuevas versiones cada cierto tiempo.
- Se obtiene software que tiene valor con mayor rapidez
- El proceso de integración es continuo, logrando integrar todo el trabajo con mayor facilidad.
- Se logra satisfacer las necesidades del usuario con mayor exactitud.
- Se obtiene código más simple y más fácil de entender.
- Es mucho más fácil realizar modificaciones a los requerimientos gracias al refactoring.
- Se logra un equipo de trabajo más contento y motivado gracias a que no se permite el trabajo en exceso y existe mayor integración entre todo el equipo de trabajo.

1.5.2. DESVENTAJAS:

- Es recomendable su uso en proyectos que pueden ser ejecutados en corto plazo.
- No es aconsejable para empresas que producen software masivo.
- Los usuarios pueden no desear frecuentes versiones del software.
- Altas comisiones en caso de presentar fallos.
- Requiere de un estricto ajuste a los principios.
- No siempre puede resultar más fácil que el desarrollo tradicional.

1.5.3. FASES DE LA METODOLOGÍA XP

1.5.3.1. FASE 1: PLANIFICACIÓN DEL PROYECTO

En esta primera fase se realiza la recopilación de todos los requerimientos del proyecto mediante una interacción con el usuario, se planifica entre el grupo de trabajo qué es lo que se deberá hacer para lograr cumplir con los objetivos finales. En esta fase se generan los siguientes documentos:

Historias de usuario: El cliente detalla en lenguaje no técnico las necesidades del sistema, no se debe hablar de algoritmos ni de diseños de bases de datos, etc. Las historias de usuario tienen la misma finalidad que los casos de uso. El tiempo de desarrollo ideal para una historia de usuario es entre 1 y 3 semanas.

Iteraciones: Cada iteración dura aproximadamente tres semanas. Al inicio de cada iteración se debe seleccionar las historias de usuario que serán implementadas y las historias que no pasaron el test de aceptación cuando se finalizó la iteración anterior.

1.5.3.2 FASE 2: DISEÑO

En esta fase se crean los diseños simples es decir se diseñan prototipos no funcionales del sistema, los documentos a entregar son:

Diseños simples o Wireframes: Se debe procurar hacer todo lo menos complicado posible para conseguir diseños simples y sencillos, que sean fáciles de entender para lograr desarrollarlo en menos tiempo y con menor esfuerzo.

1.5.3.3 FASE 3: DESARROLLO

El cliente es parte esencial del equipo de desarrollo, su presencia es indispensable para llevar a cabo cada fase, a la hora de codificar una historia de usuario el cliente no puede faltar ya que, es él, quien crea la historia. La codificación debe hacerse atendiendo a estándares ya creados, programar bajo estándares mantiene el código consistente facilitando su comprensión y escalabilidad. La optimización del código siempre se debe dejar para el final primero se debe cumplir con la funcionalidad (Enríquez, Repositorio UTN, 2017).

1.5.3.4 FASE 4: PRUEBAS

El uso de test para probar el código que se va implementando es un pilar fundamental de la metodología; es por ello que se debe realizar las pruebas necesarias para verificar la correcta funcionalidad del sistema. El código será implementado únicamente cuando haya superado las pruebas correspondientes (Enríquez, Repositorio UTN, 2017).

1.6 ARQUITECTURA

En la siguiente grafica se presenta a detalle la Arquitectura que maneja ASP.NET CORE donde se puede apreciar que se permite obtener los datos desde un archivo de base de datos de una base de datos misma, para luego interactuar con servicios provenientes de distintas fuentes con las debidas seguridades con la lógica de negocio y finalmente mostrarlas en un navegador web (Chiaretta & Lattanzi, 2017).

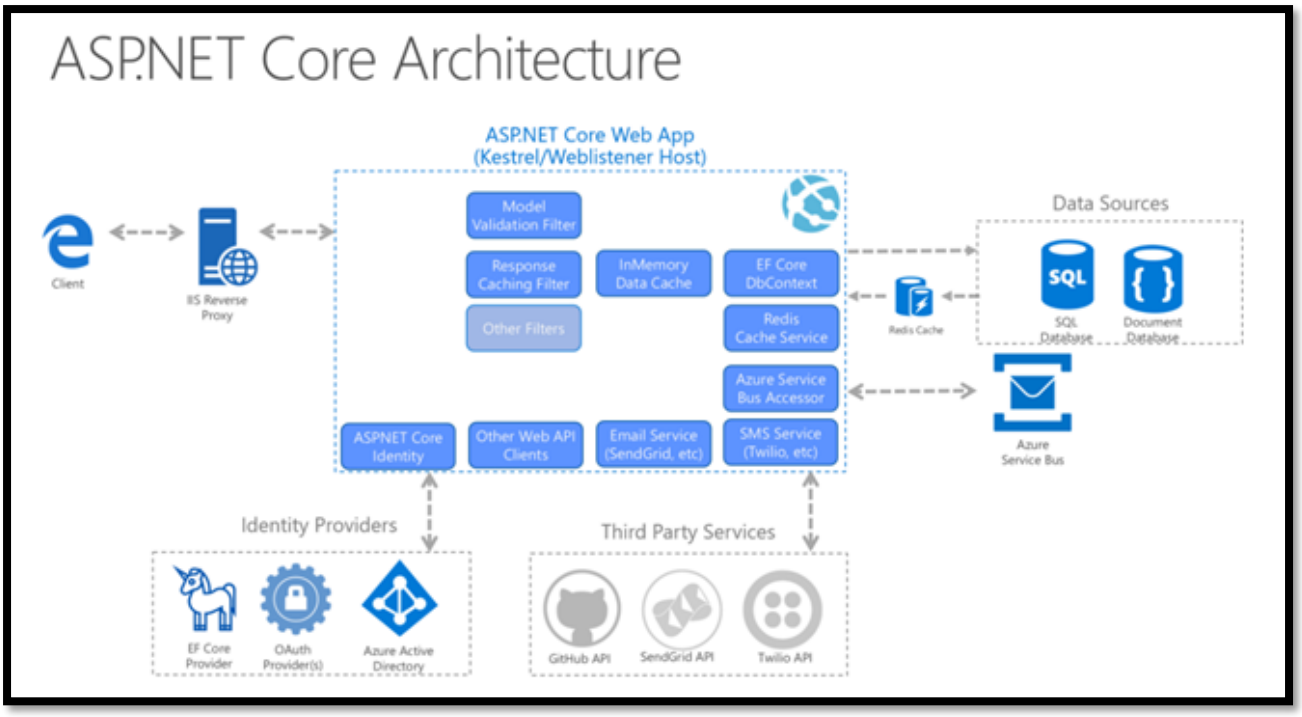


Figura 6: Arquitectura de ASP.NET CORE
Fuente: https://blogs.msdn.microsoft.com/dotnet/2017/08/09/web-apps-aspnetcore-architecture-guidance/

1.7 MODELOS, VISTAS Y CONTROLADORES

ASP.NET MVC no es diferente de la API web, por lo que el código que verá aquí será muy similar, excepto por el resultado (Chiaretta & Lattanzi, 2017).

Debido a que ya instaló y registró ASP.NET MVC para las API, puede saltar directamente al código. En la misma carpeta (Controlador) donde creó UserController, puede crear otro controlador con el objetivo de servir la página principal del sitio web. En este caso, llámalo HomeController (Chiaretta & Lattanzi, 2017).

```
using Microsoft.AspNetCore.Mvc;

namespace Syncfusion.Asp.Net.Core.Succinctly.Mvc.Controllers.MVC
{
    public class HomeController : Controller
    {
        [HttpGet]
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Figura 7: Controlador
Fuente: Propia

ASP.NET CORE es idéntico a ASP MVC por lo tanto trataremos de este este tema y diremos que Microsoft lanzó un nuevo marco web llamado ASP.NET MVC. Se basó en el patrón Modelo-Vista-Controlador (MVC) para mantener una separación clara entre la lógica de negocios y la lógica de presentación, permitiendo un control completo sobre el marcado HTML. Además de esto, se lanzó como una biblioteca separada, una no incluida en el marco. Gracias a este modelo de lanzamiento, y al no confiar 100% en el IDE para el diseño de la interfaz de usuario, fue posible actualizarlo fácilmente, manteniéndolo más en línea con el acelerado mundo del desarrollo web (Chiaretta & Lattanzi, 2017).

Sin embargo, ASP.NET MVC, aunque resolvió el problema del ciclo de lanzamiento lento y eliminó la abstracción del marcado HTML, aún sufría una dependencia de .NET Framework y System.Web, que aún lo hacía estrictamente acoplado con IIS y Windows.
Este código es similar al controlador API, las únicas diferencias son el atributo de ruta que falta en el controlador y el método de vista utilizado para devolver la acción. La primera no es necesaria porque no tiene que anular la configuración de enrutamiento predeterminada, y la segunda le indica al controlador que debe mostrar una vista y no JSON (Chiaretta & Lattanzi, 2017).

El patrón de arquitectura del controlador de vista de modelos (MVC) separa una aplicación en tres grupos de componentes principales: modelos, vistas y controladores. Este patrón permite lograr la separación de intereses. Con este patrón, las solicitudes del usuario se enrutan a un controlador que se encarga de trabajar con el modelo para realizar las acciones del usuario o recuperar los resultados de consultas. El controlador elige la vista para

mostrar al usuario y proporciona cualquier dato de modelo que sea necesario (Smith, Microsoft Docs, 2018).

En el siguiente diagrama se muestran los tres componentes principales y cuál hace referencia a los demás:

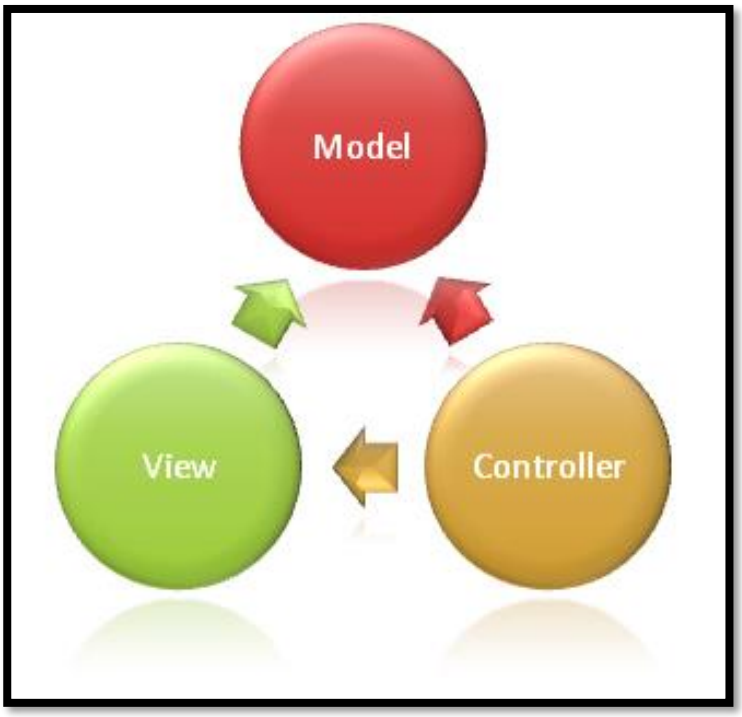


Figura 8: Controlador
Fuente: https://docs.microsoft.com/es-es/aspnet/core/mvc/overview?view=aspnetcore-2.1

Con esta delineación de responsabilidades es más sencillo escalar la aplicación, porque resulta más fácil codificar, depurar y probar algo (modelo, vista o controlador) que tenga un solo trabajo (y siga el principio de responsabilidad única). Es más difícil actualizar, probar y depurar código que tenga dependencias repartidas entre dos o más de estas tres áreas. Por ejemplo, la lógica de la interfaz de usuario tiende a cambiar con mayor frecuencia que la lógica de negocios. Si el código de presentación y la lógica de negocios se combinan en un solo objeto, un objeto que contenga lógica de negocios deberá modificarse cada vez que cambie la interfaz de usuario. A menudo esto genera errores y es necesario volver a probar la lógica de negocio después de cada cambio mínimo en la interfaz de usuario. La vista y el controlador dependen del modelo. Sin embargo, el modelo no depende de la vista ni del controlador. Esta es una de las ventajas principales de la separación. Esta separación permite compilar y probar el modelo con independencia de la presentación visual (Smith, Microsoft Docs, 2018).

1.7.3 MODELOS

El modelo en una aplicación de MVC representa el estado de la aplicación y cualquier lógica de negocios u operaciones que esta deba realizar. La lógica de negocios debe encapsularse en el modelo, junto con cualquier lógica de implementación para conservar el estado de la aplicación. Las vistas fuertemente tipadas normalmente usan tipos ViewModel diseñados para que contengan los datos para mostrar en esa vista. El controlador crea y rellena estas instancias de ViewModel desde el modelo. Hay muchas maneras de organizar el modelo en una aplicación que usa el patrón de arquitectura de MVC. Obtenga más información sobre algunos tipos diferentes de tipos de modelo (Smith, Microsoft Docs, 2018).

1.7.4 VISTAS

Las vistas se encargan de presentar el contenido a través de la interfaz de usuario. Usan el motor de vistas de Razor para incrustar código .NET en formato HTML. Debería haber la mínima lógica entre las vistas y cualquier lógica en ellas debe estar relacionada con la presentación de contenido. Si ve que necesita realizar una gran cantidad de lógica en los archivos de vistas para mostrar datos de un modelo complejo, considere la opción de usar un componente de vista, ViewModel, o una plantilla de vista para simplificar la vista (Smith, Microsoft Docs, 2018)..

1.7.5 CONTROLADORES

Los controladores son los componentes que controlan la interacción del usuario, trabajan con el modelo y, en última instancia, seleccionan una vista para representarla. En una aplicación de MVC, la vista solo muestra información; el controlador controla y responde a la interacción y los datos que introducen los usuarios. En el patrón de MVC, el controlador es el punto de entrada inicial que se encarga de seleccionar con qué tipos de modelo trabajar y qué vistas representar (de ahí su nombre, ya que controla el modo en que la aplicación responde a una determinada solicitud). Los controladores no deberían complicarse con demasiadas responsabilidades. Para evitar que la lógica del controlador se vuelva demasiado compleja, use el principio de responsabilidad única para sacar la lógica de negocios fuera el controlador y llevarla al modelo de dominio (Smith, Microsoft Docs, 2018)..

1.8 MIDDLEWARE, INYECCIÓN DE DEPENDENCIAS

Antes de ver cómo usar la inyección de dependencias dentro de las aplicaciones Core de ASP.NET, veamos qué es y por qué es importante usarlo. Para que sean fáciles de

mantener, los sistemas suelen estar compuestos de muchas clases, cada una de ellas con responsabilidades muy específicas. Por ejemplo, si desea crear un sistema que envíe correos electrónicos, es posible que tenga el punto de entrada principal del sistema y una clase responsable del formato del texto y luego una responsable del envío del correo electrónico.
El problema con este enfoque es que, si las referencias a estas clases adicionales se mantienen directamente dentro del punto de entrada, resulta imposible cambiar la implementación de la clase auxiliar sin tocar la clase principal (Chiaretta & Lattanzi, 2017).

Aquí es donde entra en juego la inyección de dependencia, generalmente conocida como DI. En lugar de crear una instancia directa de las clases de nivel inferior, los módulos de alto nivel reciben las instancias del exterior, generalmente como parámetros de sus constructores.
Descrito más formalmente por Robert C. Martin, los sistemas construidos de esta manera se adhieren a uno de los cinco principios SÓLIDOS, el principio de inversión de dependencia:
Los módulos de alto nivel no deben depender de módulos de bajo nivel. Ambos deben depender de las abstracciones. Las abstracciones no deben depender de los detalles. Los detalles deben depender de las abstracciones (Chiaretta & Lattanzi, 2017).

—Robert C. "Tío Bob" Martin

Mientras que crear e inyectar objetos manualmente puede funcionar en sistemas pequeños, esos objetos pueden volverse inmanejables cuando los sistemas crecen y se necesitan cientos o miles de clases. Para resolver este problema, se requiere otra clase: una fábrica que asume la creación de todos los objetos en el sistema, inyectando las dependencias correctas. Esta clase se llama contenedor. El contenedor, denominado contenedor de inversión de control (IoC), mantiene una lista de todas las interfaces necesarias y la clase concreta que las implementa. Cuando se le pide una instancia de cualquier clase, examina las dependencias que necesita y las pasa según la lista que mantiene. De esta manera, se pueden crear fácilmente gráficos de objetos muy complejos con unas pocas líneas de código.
Además de administrar las dependencias, estos contenedores IoC también administran la vida útil de los objetos que crean. Saben si pueden reutilizar una instancia o necesitan crear una nueva. Esta es una introducción muy breve de un tema muy importante y complicado relacionado con la calidad del software. Se han escrito innumerables artículos y libros sobre la inyección de dependencia y los principios SÓLIDOS en general. Un buen punto de partida son los artículos de Robert C. "Uncle Bob" Martin o de Martin Fowler (Chiaretta & Lattanzi, 2017).

1.8.1 CONFIGURACIÓN DE LA INYECCIÓN DE DEPENDENCIA EN ASP.NET CORE

Ahora que comprende la importancia de usar DI en las aplicaciones, puede preguntarse cómo configurarlo. En realidad, es fácil. Todo sucede en el método ConfigureServices. El parámetro que acepta el método es del tipo IServiceCollection. Esta es la lista utilizada por el contenedor para realizar un seguimiento de todas las dependencias que necesita la aplicación, por lo que es a esta colección a la que agrega sus clases. Hay dos tipos de dependencias que se pueden agregar a la lista de servicios (Chiaretta & Lattanzi, 2017).

Primero, los marcos necesitan los que funcionan para funcionar, y generalmente se configuran utilizando métodos de extensión como AddServiceName. Por ejemplo, si desea usar ASP.NET Core MVC, debe escribir services.AddMvc () para que todos los controladores y filtros se agreguen automáticamente a la lista. Además, si desea utilizar Entity Framework, debe agregar DBContext con services.AddDbContext <ExampleDbContext> (...). Luego están las dependencias específicas de su aplicación; deben agregarse individualmente especificando la clase concreta y la interfaz que implementa. Ya que los está agregando, también puede especificar la vida útil del servicio. Hay tres tipos de ciclos de vida disponibles en el contenedor IoC Core de ASP.NET, y cada uno de ellos debe agregarse utilizando un método diferente (Chiaretta & Lattanzi, 2017).

El primero es transitorio. Este ciclo de vida se usa para servicios livianos que no tienen ningún estado y son rápidos de instanciar. Se agregan utilizando los servicios del método.AddTransient <IClock, Clock> () y se crea una nueva instancia de la clase cada vez que se necesita. El segundo ciclo de vida es Scoped. Normalmente se usa para servicios que contienen un estado que es válido solo para la solicitud actual, como repositorios y clases de acceso a datos. Los servicios registrados como ámbito se crearán al comienzo de la solicitud y la misma instancia se reutilizará cada vez que la clase sea necesaria dentro de la misma solicitud. Se agregan utilizando el método services.AddScoped <IRepository, Repository> (). El último ciclo de vida se llama Singleton y, como su nombre lo indica, los servicios registrados de esta manera actuarán como singletons. Se crean la primera vez que se necesitan y se reutilizan en el resto de la aplicación. Dichos servicios generalmente tienen un estado de aplicación como un caché en memoria o problemas similares. Se agregan a través del método services.AddSingleton <IApplicationCache, ApplicationCache> () (Chiaretta & Lattanzi, 2017).

1.8.2 USO DE INYECCIÓN DE DEPENDENCIAS

Veamos ahora un ejemplo de cómo se utiliza DI-IoC en una aplicación MVC de ASP.NET. ASP.NET MVC se trata en detalle en un capítulo posterior, así que no se preocupe si algo no le resulta familiar; solo concéntrese en cómo se configuran y reutilizan las dependencias (Chiaretta & Lattanzi, 2017).

Para usar la inyección de dependencia, necesitas cuatro clases:

- La clase que necesita usar un servicio externo, también llamada clase del consumidor. En nuestro ejemplo, es un controlador MVC ASP.NET (Chiaretta & Lattanzi, 2017).
- La interfaz que define lo que hace el servicio externo, que en nuestro ejemplo es simplemente dar la hora del día (Chiaretta & Lattanzi, 2017).
- La clase que realmente implementa la interfaz (Chiaretta & Lattanzi, 2017).
- El archivo Startup.cs donde se guardará la configuración (Chiaretta & Lattanzi, 2017).

1.9 BASE DE DATOS

Entity Framework Core puede tener acceso a muchas bases de datos diferentes a través de bibliotecas de complementos denominadas proveedores de bases de datos. Los proveedores de EF Core se componen de una serie de orígenes. No todos los proveedores se mantienen como parte del proyecto Entity Framework Core. Al considerar un proveedor, evalúe la calidad, las licencias, el soporte técnico, etc. a fin de asegurarse de que satisface los requisitos. Además, asegúrese de revisar la documentación de cada proveedor para obtener información detallada de compatibilidad de versiones (Miller, 2018).

Paquete NuGet	Motores de base de datos compatibles	Mantenedor o proveedor	Notas o requisitos
Microsoft.EntityFrameworkCore.SqlServer	De SQL Server 2008 en adelante	Proyecto EF Core(Microsoft)	
Microsoft.EntityFrameworkCore.Sqlite	De SQLite 3.7 en adelante	Proyecto EF Core(Microsoft)	
Microsoft.EntityFrameworkCore.InMemory	Base de datos en memoria de EF Core	Proyecto EF Core(Microsoft)	Solo para pruebas
Npgsql.EntityFrameworkCore.PostgreSQL	PostgreSQL	Equipo de desarrollo de Npgsql	
Pomelo.EntityFrameworkCore.MySql	MySQL, MariaDB	Proyecto Pomelo Foundation	
Pomelo.EntityFrameworkCore.MyCat	Servidor MyCAT	Proyecto Pomelo Foundation	Versión preliminar, hasta EF Core 1.1
EntityFrameworkCore.SqlServerCompact40	SQL Server Compact 4.0	Erik Ejlskov Jensen	.NET Framework
EntityFrameworkCore.SqlServerCompact35	SQL Server Compact 3,5	Erik Ejlskov Jensen	.NET Framework
MySql.Data.EntityFrameworkCore	MySQL	Proyecto MySQL(Oracle)	Versión preliminar
FirebirdSql.EntityFrameworkCore.Firebird	Firebird 2.5 y 3.x	Jiří Činčura	De EF Core 2.0 en adelante
EntityFrameworkCore.FirebirdSql	Firebird 2.5 y 3.x	Rafael Almeida	De EF Core 2.0 en adelante
IBM.EntityFrameworkCore	Db2, Informix	IBM	Versión de Windows
IBM.EntityFrameworkCore-lnx	Db2, Informix	IBM	Versión de Linux
IBM.EntityFrameworkCore-osx	Db2, Informix	IBM	Versión de macOS
Devart.Data.Oracle.EFCore	De Oracle 9.2.0.4 en adelante	DevArt	Pagado
Devart.Data.PostgreSql.EFCore	De PostgreSQL 8.0 en adelante	DevArt	Pagado
Devart.Data.SQLite.EFCore	De SQLite 3 en adelante	DevArt	Pagado
Devart.Data.MySql.EFCore	De MySQL 5 en adelante	DevArt	Pagado
EntityFrameworkCore.Jet	Archivos de Microsoft Access	Bubi	EF Core 2.0, .NET Framework

Figura 9: Proveedores de Bases de Datos
Fuente: https://docs.microsoft.com/es-es/ef/core/providers/

1.10 CONSTRUCCIÓN DEL PROCESO

El presente proyecto se desarrolló en ASP.NET CORE MVC por lo tanto en este tema se abordará el proceso de construcción de una aplicación en .NET CORE MVC.

ASP.NET Core es un marco multiplataforma de código abierto para compilar aplicaciones web modernas optimizadas para la nube. Las aplicaciones ASP.NET Core son ligeras y modulares, con compatibilidad integrada para la inserción de dependencias, lo que permite una mayor capacidad de prueba y mantenimiento. Al combinarlo con MVC, que admite la creación de API web modernas además de aplicaciones basadas en vistas, ASP.NET Core es un marco eficaz con el que compilar aplicaciones web empresariales (Smith, Microsoft Docs, 2018).

1.10.1 ASIGNACIÓN DE SOLICITUDES A RESPUESTAS

En su núcleo, las aplicaciones ASP.NET Core asignan las solicitudes entrantes a las respuestas salientes. En un nivel bajo, esto se realiza mediante software intermedio y las aplicaciones y microservicios sencillos de ASP.NET Core pueden constar únicamente de software intermedio personalizado. Cuando se usa ASP.NET Core MVC, se puede trabajar en un cierto nivel superior y pensar en términos de rutas, controladores y acciones. Cada solicitud entrante se compara con la tabla de enrutamiento de la aplicación y, si se encuentra una ruta coincidente, se llama al método de acción asociado (perteneciente a un controlador) para controlar la solicitud. Si no se encuentra ninguna ruta que coincida, se llama a un controlador de errores (en este caso, se devuelve un resultado de NotFound). Las aplicaciones ASP.NET Core MVC pueden usar rutas convencionales, rutas de atributo o las dos. Las rutas convencionales se definen en el código, especificando convenciones de enrutamiento con una sintaxis similar a la del ejemplo siguiente (Smith, Microsoft Docs, 2018):

```
app.UseMvc(routes =>;
{
    routes.MapRoute("default","{controller=Home}/{action=Index}/{id?}");
});
```

Figura 10: Rutas en ASP.NET CORE
Fuente: https://docs.microsoft.com/es-es/dotnet/standard/modern-web-apps-azure-architecture/develop-asp-net-core-mvc-apps

En este ejemplo, se agregó una ruta con el nombre "default" a la tabla de enrutamiento. Define una plantilla de ruta con marcadores de posición para controlador, acción e identificador. Los marcadores de posición de acción y controlador tienen el valor predeterminado especificado ("Home" e "Index", respectivamente), y el marcador de posición del identificador es opcional (en virtud de la aplicación de "?"). La convención que se define aquí indica que la primera parte de una solicitud se debe corresponder al nombre del controlador, la segunda parte a la acción y después, si es necesario, una tercera parte representará un parámetro de identificador. Las rutas convencionales normalmente se definen en un solo lugar para la aplicación, por ejemplo, en el método Configure de la clase Startup (Chiaretta & Lattanzi, 2017).

Las rutas de atributo se aplican directamente a los controladores y acciones, en lugar de especificarse globalmente. Esto tiene la ventaja de que son mucho más sencillas de detectar cuando se examina un método concreto, pero significa que la información de enrutamiento no se mantiene en un lugar de la aplicación. Con las rutas de atributo, se pueden especificar fácilmente varias rutas para una acción determinada, así como combinar rutas entre controladores y acciones (Smith, Microsoft Docs, 2018). Por ejemplo:

```
[Route("Home")]
public class HomeController : Controller
{
    [Route("")] // Combines to define the route template "Home"
    [Route("Index")] // Combines to define route template "Home/Index"
    [Route("/")] // Does not combine, defines the route template ""
    public IActionResult Index() {}
}
```

Figura 11: Rutas en ASP.NET CORE dentro de un controlador
Fuente: https://docs.microsoft.com/es-es/dotnet/standard/modern-web-apps-azure-architecture/develop-asp-net-core-mvc-apps

Las rutas se pueden especificar en atributos [HttpGet] y similares, evitando la necesidad de agregar atributos [Route] independientes. Las rutas de atributo también pueden

usar tokens para reducir la necesidad de repetir los nombres de acciones o controladores (Smith, Microsoft Docs, 2018), como se muestra a continuación:

```
[Route("[controller\]")]
public class ProductsController : Controller
{
    [Route("")] // Matches 'Products'
    [Route("Index")] // Matches 'Products/Index'
    public IActionResult Index() {}
}
```

Figura 12: Rutas en ASP.NET CORE dentro de un controlador
Fuente: https://docs.microsoft.com/es-es/dotnet/standard/modern-web-apps-azure-architecture/develop-asp-net-core-mvc-apps

Una vez que una determinada solicitud se ha asociado a una ruta, pero antes de llamar al método de acción, ASP.NET Core MVC realizará el enlace del modelo y la validación del modelo en la solicitud. El enlace del modelo es responsable de convertir los datos HTTP entrantes en los tipos de .NET especificados como parámetros del método de acción que se va a llamar. Por ejemplo, si el método de acción espera un parámetro de identificador de tipo int, el enlace de modelos intentará proporcionar este parámetro de un valor que se proporciona como parte de la solicitud. Para ello, el enlace de modelos busca valores en un formulario enviado, valores en la propia misma ruta y valores de cadena de consulta. Suponiendo que se encuentra un valor de identificador, se convertirá en un entero antes de pasarlo al método de acción. Después de enlazar el modelo, pero antes de llamar al método de acción, se produce la validación del modelo. La validación del modelo usa atributos opcionales en el tipo de modelo y puede ayudar a garantizar que el objeto de modelo proporcionado cumple determinados requisitos de datos. Se pueden especificar determinados valores como obligatorios, o limitarlos a una longitud o un intervalo numérico determinados, etc. Si se especifican atributos de validación, pero el modelo no cumple sus requisitos, la propiedad ModelState.IsValid será false y el conjunto de reglas de validación con errores estará disponible para enviarlo al cliente que realiza la solicitud. Si se usa la validación del modelo, siempre se debe comprobar que el modelo es válido antes de ejecutar cualquier comando de modificación del estado, para asegurarse de que la aplicación no resulta dañada por datos no válidos. Se puede usar un filtro para evitar la necesidad de agregar código para esto en todas las acciones. Los filtros de ASP.NET Core MVC ofrecen

una manera de interceptar grupos de solicitudes, para poder aplicar directivas comunes e intereses transversales por cada destino. Los filtros se pueden aplicar a acciones individuales, controladores completos o de forma global para una aplicación (Smith, Microsoft Docs, 2018).

Para las API web, ASP.NET Core MVC admite la negociación de contenido, lo que permite a las solicitudes especificar cómo se debe aplicar formato a las respuestas. Según los encabezados proporcionados en la solicitud, las acciones que devuelven datos darán formato a la respuesta en XML, JSON o en otro formato compatible. Esta característica permite usar la misma API en varios clientes con requisitos de formato de datos diferentes (Smith, Microsoft Docs, 2018).

- Entonces el proceso en si se simplifica en crear un modelo son las respectivas anotaciones.
- Realizar las conexiones necesarias a BDDs o Archivo en el archivo Settings.JSON
- Generar el nuevo elemento Scaffold y automáticamente se crearán los controladores y vistas.
- Finalmente aplicar la migración o trabajar en lógica de negocio o interfaz.

1.11 CONFIGURACIÓN DE CONEXIONES

Para instalar Entity Framework Core, instale el paquete para los proveedores de base de datos de Entity Framework Core Core a los que desea dirigirse. Para obtener una lista de los proveedores disponibles, consulte Proveedores de bases de datos en la Figura 7.

Una vez instalado el proveedor de Base de datos proseguimos a configurar el archivo appsettings.json de la siguiente forma (Chiaretta & Lattanzi, 2017):

Figura 13: Configuración de Conexiones
Fuente: https://docs.microsoft.com/es-es/dotnet/standard/modern-web-apps-azure-architecture/develop-asp-net-core-mvc-apps

Modificamos la cadena de conexión para nuestra base de datos y empezamos a trabajar en nuestro proyecto (Chiaretta & Lattanzi, 2017).

1.12 SEGURIDADES

La protección de las aplicaciones web es un tema amplio, con muchas consideraciones. En su nivel más básico, la seguridad implica asegurarse de saber de quién procede una solicitud determinada y, después, asegurarse de que la solicitud solo tiene acceso a los recursos que debe. La autenticación es el proceso de comparación de las credenciales proporcionadas con una solicitud con las de un almacén de datos de confianza, para ver si la solicitud se debe tratar como procedente de una entidad conocida. La autorización es el proceso de restringir el acceso a determinados recursos en función de la identidad del usuario. Un tercer interés de seguridad es proteger las solicitudes contra el espionaje por parte de terceros, para lo que al menos se debe asegurar de que la aplicación usa SSL(Smith, Microsoft Docs, 2018).

1.13 AUTENTICACIÓN

ASP.NET Core Identity es un sistema de pertenencia que se puede usar para admitir la funcionalidad de inicio de sesión para la aplicación. Tiene compatibilidad con cuentas de usuario locales y también con proveedores de inicio de sesión externo como Microsoft Account, Twitter, Facebook, Google y muchos más. Además de ASP.NET Core Identity, la aplicación puede usar la autenticación de Windows o un proveedor de identidades de terceros como Identity Server. ASP.NET Core Identity se incluye en las nuevas plantillas de proyecto si se selecciona la opción Cuentas de usuario individuales. Esta plantilla incluye compatibilidad para el registro, inicio de sesión, inicios de sesión externos, contraseñas olvidadas y funcionalidad adicional (Smith, Microsoft Docs, 2018).

1.13.1 AUTORIZACIÓN

La forma más sencilla de autorización implica restringir el acceso a los usuarios anónimos. Esto se puede lograr mediante la aplicación del atributo [Authorize] a determinados controladores o acciones. Si se usan roles, el atributo se puede ampliar más para restringir el acceso a los usuarios que pertenecen a roles concretos (Smith, Microsoft Docs, 2018).

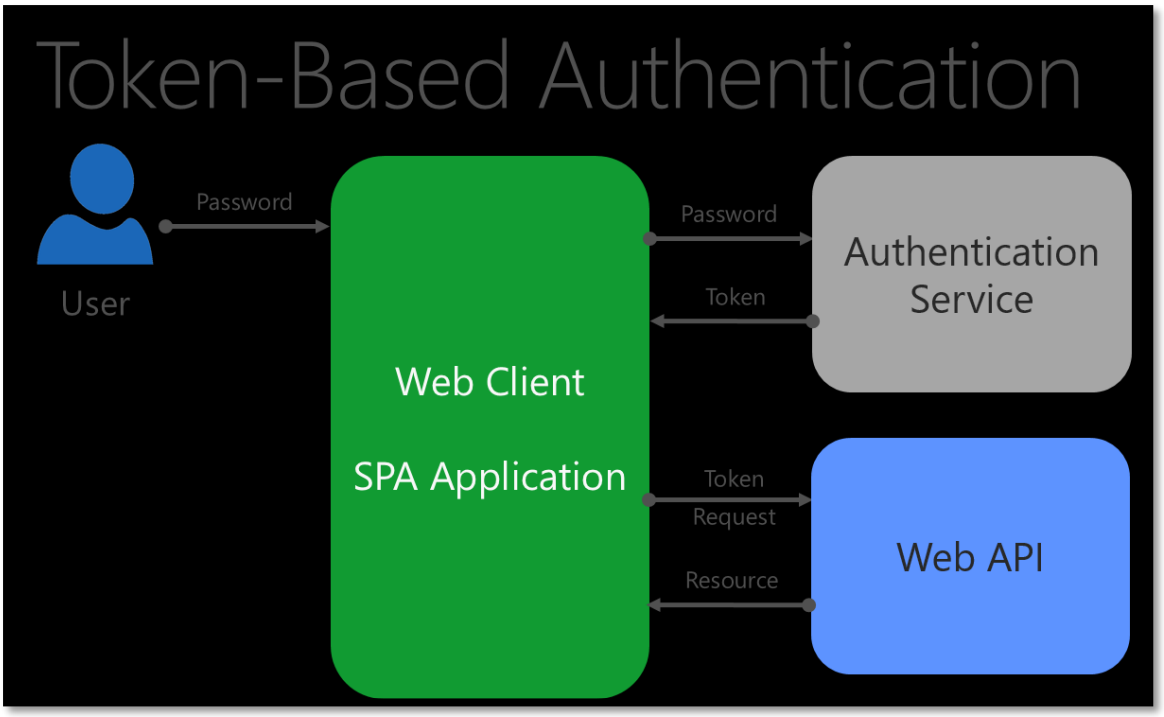


Figura 14: Seguridad en ASP.NET CORE
Fuente: https://docs.microsoft.com/es-es/dotnet/standard/modern-web-apps-azure-architecture/develop-asp-net-core-mvc-apps

1.14 VALIDACIONES

En esta sección, agregará la lógica de validación al modelo y se asegurará de que las reglas de validación se apliquen cada vez que un usuario cree o edite.

Uno de los principios de diseño de MVC es DRY ("No se repita"). ASP.NET Core MVC lo alienta a especificar la funcionalidad o el comportamiento solo una vez, y luego reflejarlo en todas partes en una aplicación. Esto reduce la cantidad de código que necesita escribir y hace que el código que usted escribe sea menos propenso a errores, más fácil de probar y más fácil de mantener. El soporte de validación proporcionado por MVC y Entity Framework Core Code First es un buen ejemplo del principio DRY en acción. Puede especificar de forma declarativa las reglas de validación en un lugar (en la clase modelo) y las reglas se aplican en todas partes en la aplicación (Anderson, Microsoft Docs, 2017).

1.14.1 REGLAS DE VALIDACIÓN

DataAnnotations proporciona un conjunto integrado de atributos de validación que usted aplica declarativamente a cualquier clase o propiedad (Anderson, Microsoft Docs,

2017). (También contiene atributos de formato como DataType esa ayuda con el formato y no proporciona ninguna validación) como se presenta en el siguiente ejemplo.

```
public class Movie
{
    public int ID { get; set; }

    [StringLength(60, MinimumLength = 3)]
    [Required]
    public string Title { get; set; }

    [Display(Name = "Release Date")]
    [DataType(DataType.Date)]
    public DateTime ReleaseDate { get; set; }

    [Range(1, 100)]
    [DataType(DataType.Currency)]
    [Column(TypeName = "decimal(18, 2)")]
    public decimal Price { get; set; }

    [RegularExpression(@"^[A-Z]+[a-zA-Z""'\s-]*$")]
    [Required]
    [StringLength(30)]
    public string Genre { get; set; }

    [RegularExpression(@"^[A-Z]+[a-zA-Z0-9""'\s-]*$")]
    [StringLength(5)]
    [Required]
    public string Rating { get; set; }
}
```

Figura 15: Anotaciones ASP.NET CORE
Fuente: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/validation?view=aspnetcore-2.1

Los atributos de validación especifican el comportamiento que desea imponer en las propiedades del modelo a las que se aplican. Los atributos Required y MinimumLength indican que una propiedad debe tener un valor; pero nada impide que un usuario ingrese espacios en blanco para satisfacer esta validación. El RegularExpression atributo que se utiliza para limitar los caracteres que se pueden ingresar. En el código anterior, Genrey Rating solo debe usar letras (mayúscula de la primera letra, espacios en blanco, no se permiten números ni caracteres especiales). El Range atributo que restringe un valor dentro de un rango especificado. El StringLength atributo que permite establecer la longitud máxima de una propiedad de cadena y, opcionalmente, su longitud mínima. Los tipos de valor (tales

como decimal, int, float, DateTime) son inherentemente necesarios y no necesitan el atributo [Required] (Anderson, Microsoft Docs, 2017).

En la figura 10 se muestra el error de validación:

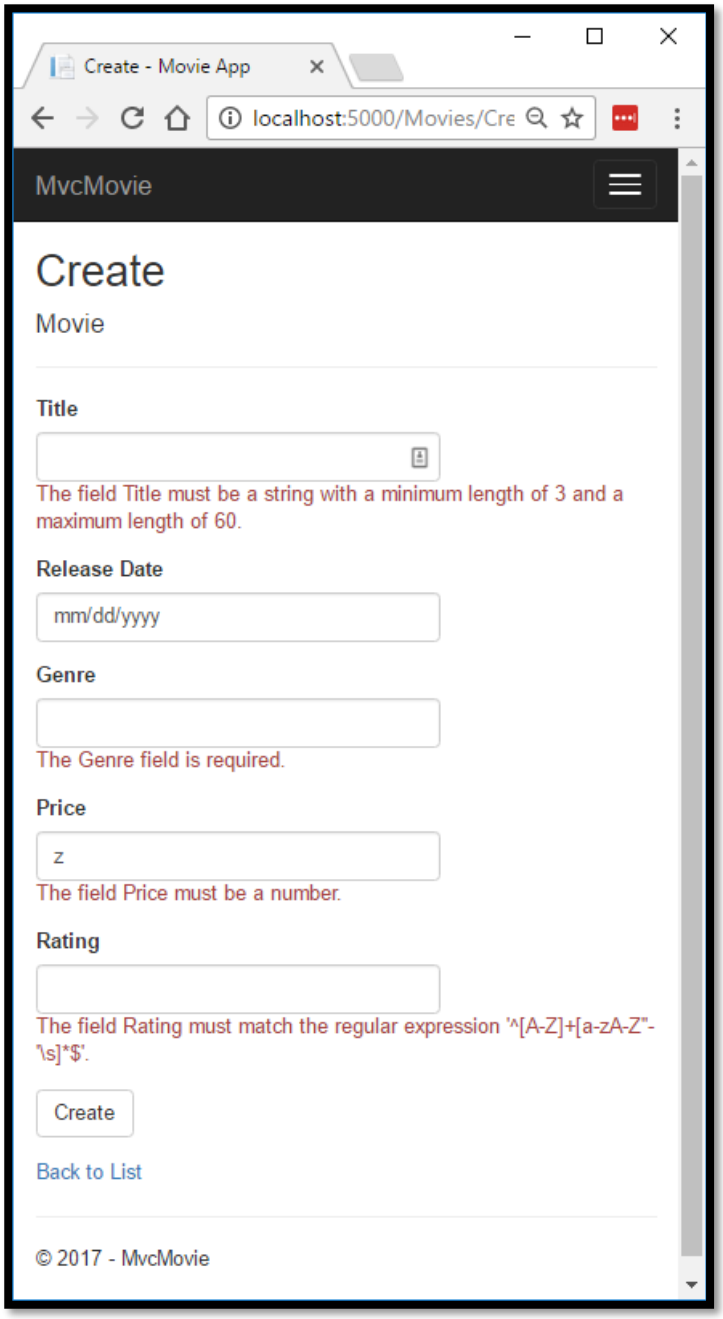


Figura 16: Error de validación.
Fuente: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/validation?view=aspnetcore-2.1

Observe cómo el formulario ha generado automáticamente un mensaje de error de validación apropiado en cada campo que contiene un valor no válido. Los errores se aplican

tanto en el lado del cliente (usando JavaScript y jQuery) como en el lado del servidor (en caso de que un usuario tenga JavaScript deshabilitado) (Anderson, Microsoft Docs, 2017).

El hecho de que ASP.NET Core imponga automáticamente las reglas de validación ayuda a que su aplicación sea más robusta. También garantiza que no se olvide de validar algo y dejar inadvertidamente que ingresen datos erróneos en la base de datos (Anderson, Microsoft Docs, 2017).

1.15 SINTAXIS RAZOR

Razor no es en realidad un lenguaje de programación, sino simplemente un motor de vistas. Básicamente, hay que analizar qué significan las siglas MVC: corresponden al término Modelo-Vista-Controlador, forma en que se denomina a uno de los patrones de arquitectura más populares. En MVC se pretende separar completamente la información que gestiona nuestra aplicación (el Modelo) de la forma en que se presenta (la Vista), desacoplando ambas capas mediante una capa intermedia (el Controlador) que contendrá toda la lógica para trasladar los datos a la presentación visual que deseamos realizar como se muestra en la figura 11. ASP.NET CORE MVC respeta firmemente estos principios, hasta el punto de existir varias sintaxis con las que expresar de qué modo queremos crear nuestra vista, siendo 100% reutilizables tanto los controladores como los modelos que tengamos por debajo. De este modo han ido naciendo distintos motores de vistas, como Razor, que nos permiten crear dichas vistas de la forma en que nos sintamos más cómodos (Holguera, 2010).

1.15.1 OBJETIVOS DE RAZOR

Son varios los objetivos que Microsoft se ha marcado en la creación de este motor de vistas, entre los que destacaríamos:

Compacto, expresivo y fluido: buscan reducir la cantidad de código que necesitamos para crear las vistas, evitando que tengamos que denotar de una forma especial cada línea de código procedural. El compilador será lo suficientemente inteligente como para inferir, en muchas ocasiones, qué tipo de código estamos escribiendo.

- **Fácil de aprender:** aunque esto es siempre relativo, puesto que depende del bagaje previo del desarrollador y de sus capacidades, con sólo este artículo y los dos próximos dejaremos asentados conceptos lo suficientemente amplios como para hacer frente al 80% de la funcionalidad que podamos necesitar de Razor.
- **Funciona en cualquier editor de texto**, luego no vamos a tener que estar anclados a Visual Studio, WebMatrix o cualquier otra herramienta para crear nuestros ficheros Razor.

Obviamente, el soporte al lenguaje es superior en una herramienta como Visual Studio, donde tendremos toda la potencia de IntelliSense a nuestro alcance, que, si editamos simplemente con el Block de Notas, pero la decisión última será nuestra.

- **Testeable:** podremos crear tests unitarios de las vistas (Holguera, 2010).

```
@{ var nombre = "Pedro"; }

<html>
<body>
<h2>Hola @nombre, son las @DateTime.Now</h2>
   </body>
</html>
```

Figura 17: Sintaxis Razor.
Fuente: https://desarrolloweb.com/articulos/la-sintaxis-razor.html

1.16 INTEFAZ ANGULAR 4

En ASP.NET CORE vine incluido una plantilla que trabaja con angular actualizada y proporciona un punto de inicio conveniente para las aplicaciones Core de ASP.NET que usan Angular y la CLI Angular para implementar una interfaz de usuario (UI) enriquecida del lado del cliente. La plantilla es equivalente a crear un proyecto Core de ASP.NET para que actúe como un backend de API y un proyecto de CLI angular para que actúe como una interfaz de usuario. La plantilla ofrece la comodidad de alojar ambos tipos de proyectos en un solo proyecto de aplicación. En consecuencia, el proyecto de la aplicación se puede construir y publicar como una sola unidad (Sanderson, 2018).

1.16.1 CREAR UNA NUEVA APLICACIÓN

Si tiene instalado ASP.NET Core 2.1, no es necesario instalar la plantilla de proyecto Angular.

Cree un nuevo proyecto desde un símbolo del sistema utilizando el comando dotnet new angularen en un directorio vacío. Por ejemplo, los siguientes comandos crean la aplicación en un directorio my-new-app y cambian a ese directorio:

```
dotnet new angular -o my-new-app
cd my-new-app
```

La plantilla de proyecto crea una aplicación Core de ASP.NET y una aplicación Angular. La aplicación ASP.NET Core está diseñada para ser utilizada para el acceso a datos, la autorización y otras inquietudes del servidor. La aplicación Angular, que reside en el subdirectorio ClientApp, está destinada a ser utilizada para todas las inquietudes de la interfaz de usuario (Sanderson, 2018)..

1.16.2 AÑADIR PÁGINAS, IMÁGENES, ESTILOS, MÓDULOS, ETC.

El directorio ClientApp contiene una aplicación estándar de Angular CLI. Consulte la documentación oficial de Angular para más información.

Existen ligeras diferencias entre la aplicación Angular creada por esta plantilla y la creada por Angular CLI (vía ng new); Sin embargo, las capacidades de la aplicación no han cambiado. La aplicación creada por la plantilla contiene un diseño basado en Bootstrap y un ejemplo de enrutamiento básico (Sanderson, 2018).

1.16.3 EJECUTAR COMANDOS NG

En un símbolo del sistema, cambie al subdirectorio ClientApp:

```
Dupdo
cd ClientApp
```

Si tiene la ngherramienta instalada globalmente, puede ejecutar cualquiera de sus comandos. Por ejemplo, puede ejecutar ng lint, ng cualquiera de los otros comandos de la CLI angular. Sin embargo, ng serve, no es necesario que se ejecute, ya que su aplicación ASP.NET Core se ocupa de servir las partes del lado del servidor y del lado del cliente de su aplicación. Internamente, utiliza ng serve en el desarrollo (Sanderson, 2018).

Si no tiene ng instalada, ejecute npm run ng. Por ejemplo, puede ejecutar npm run ng linto npm run ng test (Sanderson, 2018).

CAPÍTULO II

2. DESARROLLO DE LA APLICACIÓN

2.2. DIAGNOSTICO A LA INSTITUCIÓN

El Centro Integral de la Niñez y Adolescencia, CENIT, es una organización sin fines de lucro con 24 años de experiencia. Desarrollamos estrategias y programas de atención integral para acoger a las niñas/os y adolescentes en situación de riesgo y vulnerabilidad, con el objetivo de erradicar el trabajo infantil, promover el ejercicio de derechos, equidad de género, vivencia de valores y mejorar sus condiciones de vida (Quelal, 2019).

1.2. PROGRAMAS

1.2.1. RESCATE DE CALLE

Abordaje de niñas/os y adolescentes en situación de trabajo infantil y/o vulnerabilidad (Quelal, 2019).

1.2.1. VOLUNTARIADO

CENIT, recibe la colaboración de voluntarios ecuatorianos y extranjeros que brindan su apoyo integrándose de manera proactiva en las diferentes áreas y departamentos (Quelal, 2019).

1.2.2. RESCATE DE CALLE

Abordaje de niñas/os y adolescentes en situación de trabajo infantil y/o vulnerabilidad (Quelal, 2019).

1.2.3. BIOSICOSOCIAL

Intervención y seguimiento en psicología, trabajo social y salud las niñas/os, adolescentes tienen diariamente servicio de comedor (Quelal, 2019).

1.2.4. MI FAMI

Formación Humana: Capacitación a las familias en Buen Trato-Equidad de Género-Espiritualidad Talleres ocupacionales: Tarjetería, Bisutería, Corte – Confección (Quelal, 2019).

1.2.5. CENTRO DE APOYO CEA

Tareas dirigidas, refuerzo escolar y recreación para niñas/os trabajadores o en situación de riesgo de 2° a 7° Año de Educación Básica de las escuelas fiscales del sector (Quelal, 2019).

1.2.6. ESCUELA POPULAR EUFRASIA PELLETIER

Educación Básica de 2do a 7mo Años en 3 Niveles para niñas/os y adolescentes con rezago escolar. Octavo, Noveno y Décimo Año EB para adolescentes y jóvenes de 15 a 21 años con Talleres complementarios de Corte – Confección y Bordado, Panadería y Manualidades tema que lo abordamos en este proyecto (Quelal, 2019).

2.3. RECOPILACIÓN DE INFORMACIÓN Y NORMATIVA INSTITUCIONAL

Como se trata de una institución educativa no regiremos específicamente a la ley de educación vigente en nuestro país.

Año lectivo. El año lectivo se debe desarrollar en un régimen escolar de dos (2) quimestres en todas las instituciones educativas públicas, fiscomisionales y particulares, y debe tener una duración mínima de doscientos (200) días de asistencia obligatoria de los estudiantes para el cumplimiento de actividades educativas, contados desde el primer día de clases hasta la finalización de los exámenes del segundo quimestre (Constituyente Asamblea Nacional, 2011).

El año lectivo en las instituciones educativas públicas, fiscomisionales y particulares debe empezar la primera semana de mayo en el régimen de Costa y la primera semana de septiembre en el régimen de Sierra, salvo situaciones de emergencia oficialmente declaradas por el Nivel Central de la Autoridad Educativa Nacional (Constituyente Asamblea Nacional, 2011).

Son imputables al año lectivo, como actividades educativas de régimen escolar, las siguientes: clases, evaluaciones y programas educativos reconocidos por el Nivel Central de

la Autoridad Educativa Nacional. El resto de las actividades educativas deben constar en el cronograma de actividades del establecimiento y no pueden exceder del cinco por ciento (5 %) de los doscientos (200) días fijados como obligatorios para el año lectivo (Constituyente Asamblea Nacional, 2011).

Hora pedagógica. La hora pedagógica es la unidad de tiempo mínima en la que docentes y estudiantes desarrollan actividades de aprendizaje destinadas a cumplir con lo prescrito en el currículo. Este período debe ser de por lo menos cuarenta (40) minutos desde el subnivel de Básica Elemental en adelante (Constituyente Asamblea Nacional, 2011).

Expediente académico. Les corresponde a las instituciones educativas llevar el archivo de registro de matrículas y promociones debidamente legalizadas. Esta documentación constituye el expediente académico del estudiante que, en versión original, debe ser entregado al representante legal del estudiante en caso de cambio de plantel (Constituyente Asamblea Nacional, 2011).

Requisitos. Para la concesión de matrícula excepcional, los interesados deben presentar, al Nivel Distrital, la solicitud con los siguientes documentos:

- Certificados de matrícula y promoción de los años de estudio realizados;
- Aceptación de la institución educativa en la que continuará sus estudios; y,
- Informes y convenios, si los hubiere, en el caso de estudiantes que provengan del exterior (Constituyente Asamblea Nacional, 2011).

De la inscripción. La inscripción es la fase en la cual el representante legal del estudiante solicita a integración de su representado al sistema educativo público; el período de inscripción se realizará conforme a los procedimientos y cronogramas establecidos por la Autoridad Educativa Nacional (Constituyente Asamblea Nacional, 2011).

Promoción. Los estudiantes en el nivel de Educación Inicial y en el subnivel de Preparatoria serán promovidos automáticamente al grado siguiente.

Sin embargo, los estudiantes de Preparatoria que antes del inicio del subnivel de Básica Elemental no hubieren alcanzado el nivel de desarrollo necesario para el óptimo aprovechamiento del siguiente grado deberán desarrollar, antes del inicio del siguiente año

lectivo y con apoyo de su familia, una serie de actividades determinadas por el docente (Constituyente Asamblea Nacional, 2011).

Escala de calificaciones. Las calificaciones hacen referencia al cumplimiento de los objetivos de aprendizaje establecidos en el currículo y en los estándares de aprendizaje nacionales. Las calificaciones se asentarán según la siguiente escala (Constituyente Asamblea Nacional, 2011):

Escala cualitativa	Escala cuantitativa
Supera los aprendizajes requeridos.	10
Domina los aprendizajes requeridos.	9
Alcanza los aprendizajes requeridos.	7-8
Esta próximo a alcanzar los aprendizajes requeridos.	5-6
No alcanza los aprendizajes requeridos.	<= 4 DEROGADO

Tabla 1: Escalas de Calificación
Fuente: (Constituyente Asamblea Nacional, 2011)

Requisitos para la promoción. La calificación mínima requerida para la promoción, en cualquier establecimiento educativo del país, es de siete sobre diez (7/10) (Constituyente Asamblea Nacional, 2011).

Informes de aprendizaje. Las instituciones educativas deben emitir en un formato oficial definido por el Nivel Central de la Autoridad Educativa Nacional informes parciales, quimestrales y anuales de aprendizaje, que expresen cualitativa y cuantitativamente el alcance de los aprendizajes logrados por el estudiante en cada una de las asignaturas, y en los que se deben incluir recomendaciones para promover el aprendizaje del estudiante. Los informes se clasifican de la siguiente manera (Constituyente Asamblea Nacional, 2011).

Informe parcial de aprendizaje. Es un informe que expresa cualitativa y cuantitativamente el alcance de los aprendizajes logrados por el estudiante en cada una de las asignaturas, y formula recomendaciones y planes de mejoramiento académico que deben seguirse durante un período determinado, tal como se prevé en el Proyecto Educativo Institucional (Constituyente Asamblea Nacional, 2011).

Informe quimestral de aprendizaje. Es un informe que contiene el promedio de las calificaciones parciales y el examen quimestral. Expresa cualitativa y cuantitativamente el alcance de los aprendizajes logrados por el estudiante en cada una de las asignaturas, y

formula recomendaciones y planes de mejoramiento académico que deben seguirse. La nota del examen quimestral no puede ser mayor al veinte por ciento (20 %) de la nota total del quimestre correspondiente a cada asignatura, y el porcentaje restante debe corresponder a las notas parciales obtenidas durante ese período. (Constituyente Asamblea Nacional, 2011).

Informe anual de aprendizaje. Es un informe que contiene el promedio de las dos (2) calificaciones quimestrales, expresa cualitativa y cuantitativamente el alcance de los aprendizajes logrados por el estudiante en cada una de las asignaturas, formula recomendaciones y planes de mejoramiento académico que deben seguirse, y determina resultados de aprobación y reprobación (Constituyente Asamblea Nacional, 2011).

Examen de recuperación o de la mejora del promedio. El examen de recuperación tiene como objetivo dar la oportunidad de mejorar los promedios y se ofrece a cualquier estudiante que hubiere aprobado la asignatura con un promedio inferior a diez (10). Para el efecto, quince (15) días después de publicadas las calificaciones, los estudiantes podrán rendir por una sola vez una prueba recuperatoria acumulativa, cuyo resultado debe reemplazar al promedio quimestral más bajo, y debe servir solo para el mejoramiento de un promedio quimestral. Si la nota fuere más baja que la obtenida en los promedios quimestrales, deberá ser desechada (Constituyente Asamblea Nacional, 2011).

Examen supletorio. Si un estudiante hubiere obtenido un puntaje promedio anual de cinco (5) a seis comas nueve (6,9) sobre diez como nota final de cualquier asignatura, podrá rendir un examen supletorio acumulativo, que será una prueba de base estructurada. El examen supletorio se rendirá en un plazo de quince (15) días posterior a la publicación de las calificaciones finales. La institución educativa deberá ofrecer clases de refuerzo durante los quince (15) días previos a la administración del examen supletorio, con el fin de preparar a los estudiantes que deban presentarse a este examen. Para aprobar una asignatura a través del examen supletorio, se debe obtener una nota mínima de siete sobre diez (7/10), sin aproximaciones. El promedio final de una asignatura aprobada por medio de un examen supletorio siempre será siete sobre diez (7/10) (Constituyente Asamblea Nacional, 2011). Examen remedial. Si un estudiante hubiere obtenido un puntaje promedio anual menor a cinco sobre diez (5/10) como nota final de cualquier asignatura o no aprobare el examen supletorio, el docente de la asignatura correspondiente deberá elaborar un cronograma de actividades académicas que cada estudiante tendrá que cumplir en casa con ayuda de su familia, para que quince (15) días antes de la fecha de inicio de clases, rinda por una sola vez un examen remedial acumulativo, que será una prueba de base estructurada (Constituyente Asamblea Nacional, 2011)..

Para aprobar una asignatura a través del examen remedial, se debe obtener una nota mínima de siete sobre diez (7/10), sin aproximaciones. El promedio final de una asignatura aprobada por medio de un examen remedial siempre será siete sobre diez (7/10). Si un estudiante reprobare exámenes remediales en dos o más asignaturas, deberá repetir el grado o curso (Constituyente Asamblea Nacional, 2011).

Examen de gracia. En el caso de que un estudiante reprobare un examen remedial de una sola asignatura, podrá asistir al grado o curso siguiente de manera temporal, hasta rendir un examen de gracia un mes después del inicio de clases. De aprobar el examen, podrá continuar en ese grado o curso, pero en caso de reprobarlo, deberá repetir el grado o curso anterior (Constituyente Asamblea Nacional, 2011).

2.4. RECOLECCIÓN DE REQUERIMIENTOS

- Registrar a los estudiantes con sus respectivos representantes o familiares.
- Registrar representas, familiares o apoderados de los estudiantes.
- Registrar toda la información concerniente al Personal Administrativo y de servicio de la Institución.
- Registrar una matrícula e imprimirla por ciclos académicos y por cursos.
- Registrar horarios e imprimirlo por cursos y/o docentes.
- Registrar las notas de los estudiantes matriculados con su respectivo curso, parcial y/o quimestre e imprimir promociones, y consulta de reportes.
- Habilitar el ingreso de notas en cada parcial supletorio, remedial y gracia.
- Registrar los cursos y sus asignaturas.
- Creación de años escolares

2.5. PLANTEAMIENTO DE REQUISITOS FUNCIONALES Y NO FUNCIONALES

2.5.1. REQUISITOS FUNCIONALES

- Registrar a los estudiantes con sus respectivos representantes o familiares.
- Registrar representas, familiares o apoderados de los estudiantes.
- Registrar toda la información concerniente al Personal Administrativo y de servicio de la Institución.
- Registrar una matrícula e imprimirla por ciclos académicos y por cursos.
- Registrar horarios e imprimirlo por cursos y/o docentes.

- Registrar las notas de los estudiantes matriculados con su respectivo curso, parcial y/o quimestre e imprimir promociones, y consulta de reportes.
- Habilitar el ingreso de notas en cada parcial supletorio, remedial y gracia.
- Registrar los cursos y sus asignaturas.
- Creación de años escolares
- Asignar en cada asignatura un docente.

2.5.2. REQUISITOS NO FUNCIONALES

- Permitir su despliegue en AppHarbor.
- Funcionar con Windows x64 y x86.
- Funcionar en un entorno Web y en Red Local.
- Funcionar con 500MB de RAM.
- Funcionar con 1 Ghz de Procesamiento.
- El tamaño de la aplicación debe ser menor a 34MB.
- Interfaz Estética.
- Que esté disponible siempre.
- Que sea visible en dispositivos móviles.
- Que muestre en la página de inicio la información de la institución.

2.6. PROCESO DE DESARROLLO SEGÚN LAS FASES DE LA METODOLOGÍA

2.6.1. DEFINICIÓN DE ROLES Y RESPONSABILIDADES

Con la asignación de roles se pretende determinar los cargos y funciones de cada persona involucrada en el proyecto a desarrollar.

Tabla 2: Definición de roles.

Nombre Rol	Descripción	Responsabilidades

Programador	Es el encargado de: ➢ Escribir toda la programación del proyecto. ➢ Diseña el sistema. ➢ Implementa las distintas pruebas y las evalúa.	Define las tareas a realizar y las estimaciones de tiempo para cada una. Implementa las diferentes historias de usuario. Asiste a reuniones para realizar planificaciones y demostraciones de la funcionalidad del sistema. Brinda capacitación al usuario acerca del sistema.
Usuario	Es la persona encargada y capacitada para el manejo del Sistema Académico.	Pide la creación del sistema. Facilita la creación de las historias de usuario. Define las prioridades de las diferentes historias de usuario para la implementación. Realiza revisiones periódicas para verificar el cumplimiento de los requerimientos.
Encargado de pruebas	Persona encargada de la interacción con el cliente para el desarrollo de las diferentes pruebas de funcionalidad del sistema.	Realiza las pruebas de funcionalidad del sistema.
		Elabora los resultados de las pruebas para entregar al cliente.
Tutor	Es la persona encargada de brindar asesoría durante el desarrollo del sistema en sus diferentes fases.	Brinda apoyo durante todo el proceso de desarrollo.

Fuente: (Enríquez, Repositorio UTN, 2017)

2.6.2 DEFINICIÓN DE LOS INTEGRANTES DEL EQUIPO DE TRABAJO

Según la metodología XP hay que desarrollar el trabajo en equipos como se detalla a continuación.

Tabla 3: Equipo de trabajo para el desarrollo del sistema.

Nombre	Descripción	Rol
Ing. MSc. Diego Trejo	Encargado de las revisiones periódicas para verificar que se lleve a cabo el desarrollo del proyecto de acuerdo a la planificación establecida.	Director de Trabajo de Grado
Ing. Pedro Granda	Encargado de facilitar las diferentes guías para la culminación y entrega del proyecto.	Tutor de Trabajo de Grado
Sr. Diego Quelal	Encargado del desarrollo y entrega del sistema académico.	Programador (Tesista)

Fuente: (Enríquez, Repositorio UTN, 2017)

2.6.3 HISTORIAS DE USUARIO Y DISEÑO

Seguidamente se muestran las historias de usuario que se establecieron para el desarrollo del sistema.

2.6.3.1 HISTORIA DE USUARIO 1: ADMINISTRACIÓN DE INGRESO AL SISTEMA

Tabla 4: Historia de usuario 1

HISTORIA DE USUARIO
Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier.

Número: 1	**Usuario:** Administrador
Nombre de historia: Administración de ingreso al sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Estimación (horas): 5	**Iteraciones asignadas:** 1
Programador responsable: Diego Quelal	
Descripción: Para ingresar al sistema el usuario deberá ingresar con su respectivo usuario y contraseña y el sistema detectará que rol tiene que pueden ser Administrador, Docente y Estudiante.	
Observaciones: El sistema detecta tres roles Administrador, Estudiante y Docente.	
Fecha: 02 de octubre del 2018	
Firma:	

Fuente: Propia

Tabla 5: Tarea 1 – Historia de usuario 1

	Tarea
Número tarea: 1	**Número historia:** 1
Nombre tarea: Desarrollo de la vista Login.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 02 de octubre del 2018	**Fecha fin:** 02 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Permite desplegar la vista de usuario y contraseña para el acceso al Sistema.	

Fuente: Propia

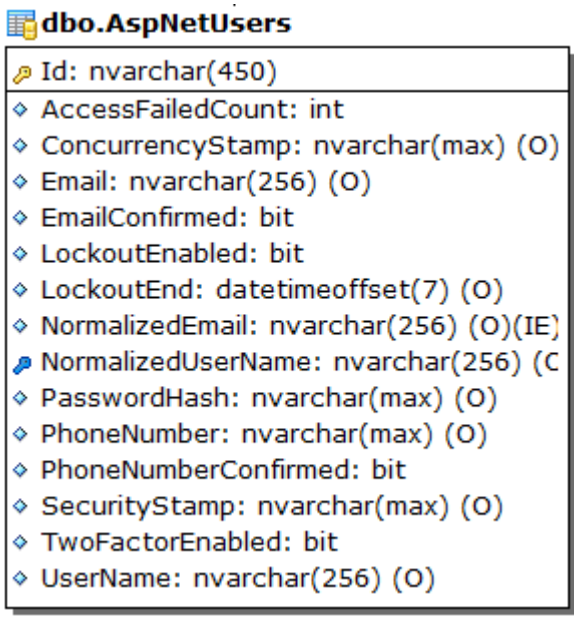


Figura 12: Modelo de datos – Usuarios.

Fuente: Propia

Figura 18: Formulario de ingreso al sistema.
Fuente: Propia

Tabla 6: Tarea 1 – Historia de usuario 2

Tarea	
Número tarea: 2	**Número historia:** 1
Nombre tarea: Cambio de contraseña del usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1

Fecha inicio: 02 de octubre del 2018	**Fecha fin:** 02 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Diseño de la vista y su funcionalidad para el cambio de clave.	

Fuente: Propia

2.6.3.2 HISTORIA DE USUARIO 2: PÁGINA DE INICIO

Tabla 7: Historia de usuario 2

HISTORIA DE USUARIO	
Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier.	
Número: 2	**Usuario:** Administrador
Nombre de historia: Página de Inicio	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Estimación (horas): 1	**Iteraciones asignadas:** 1
Programador responsable: Diego Quelal	
Descripción: Aquí se mostrará la información principal de cómo trabaja la institución y sus Respectivos programas.	
Observaciones: No tiene Modelo de datos es únicamente vista sin persistencia.	
Fecha: 03 de octubre del 2018	
Firma:	

Fuente: Propia

Tabla 8: Tarea 1 – Historia de usuario 2

Tarea	
Número tarea: 1	**Número historia:** 2

Nombre tarea: Modificar la vista home de la aplicación creada.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 02 de octubre del 2018	**Fecha fin:** 02 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Modificar la vista home de la aplicación creada.	

Fuente: Propia

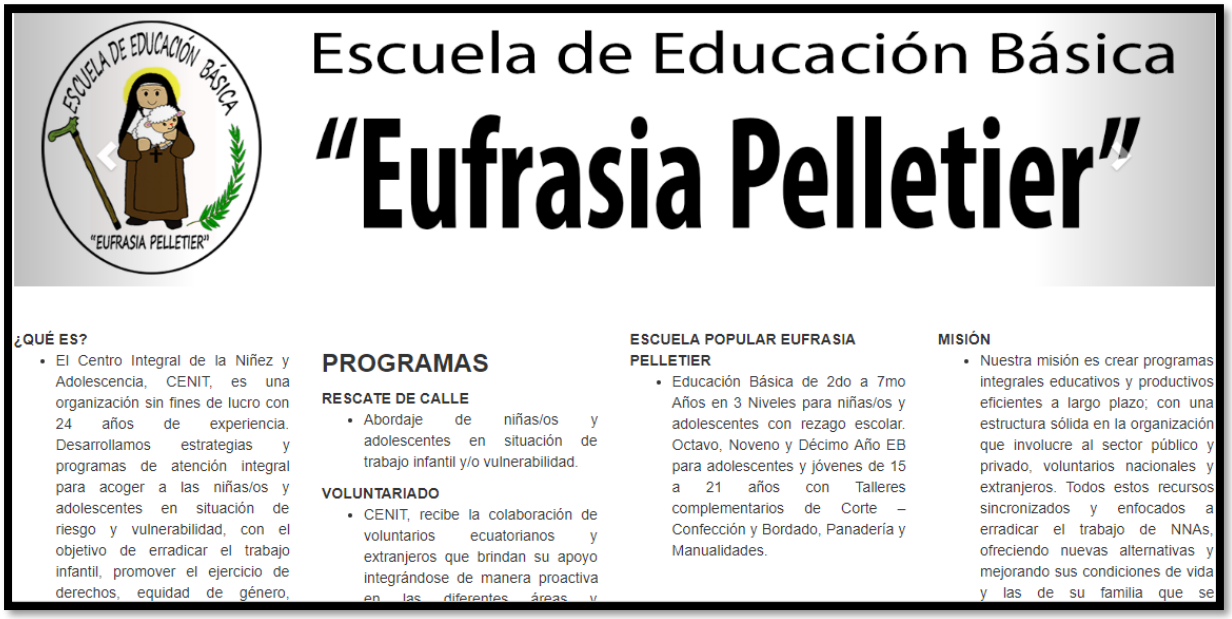


Figura 19: Página de inicio.

2.6.3.3 HISTORIA DE USUARIO 3: PERSONAL

Tabla 9: Historia de usuario 3

HISTORIA DE USUARIO	
Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier.	
Número: 3	**Usuario:** Administrador
Nombre de historia: Personal	

Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Estimación (horas): 8	**Iteraciones asignadas:** 1
Programador responsable: Diego Quelal	
Descripción: Se analizará, y se construirá un modelo de base de datos, generar las vistas e implementar toda la funcionalidad que se necesita del personal.	
Observaciones: Aquí se registra toda la información necesaria del personal datos personales, are de trabajo, rol, autodefinición, referencia laboral, título jornada de trabajo y si está activo o no.	
Fecha: 03 de octubre del 2018	
Firma:	

Fuente: Propia

Tabla 10: Tarea 1 – Historia de usuario 3

Tarea	
Número tarea: 1	**Número historia:** 3
Nombre tarea: Crear un Modelo de datos para crear la vista Personal.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 09 de octubre del 2018	**Fecha fin:** 09 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Crear un Modelo de datos para crear la vista Personal.	

Fuente: Propia

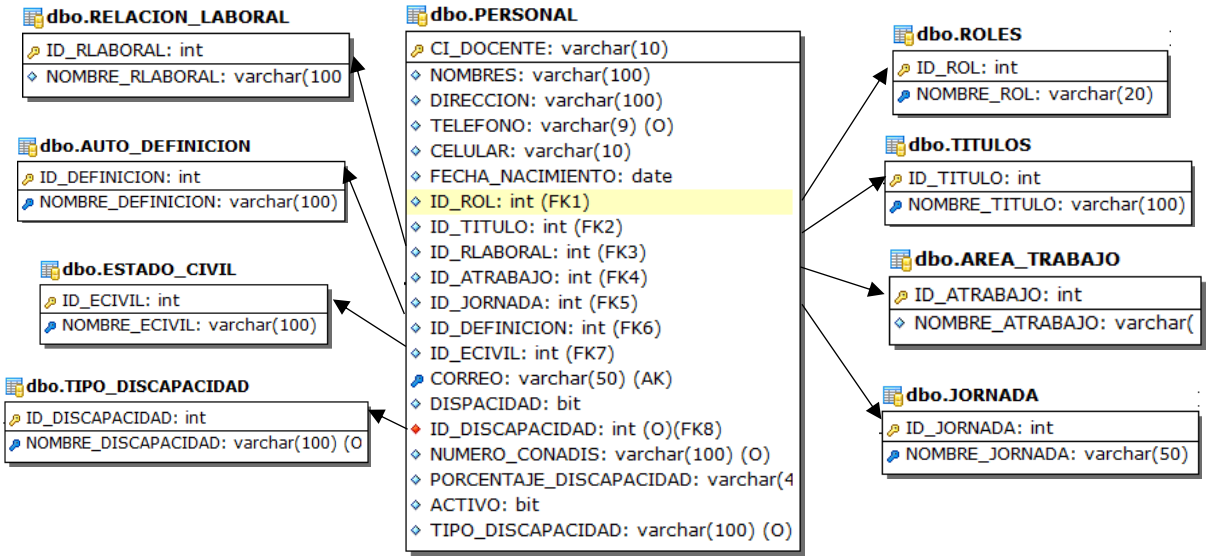


Figura 20: Modelo de datos – Registro personal.
Fuente: Propia

Tabla 11: Tarea 2 – Historia de usuario 3

Tarea	
Número tarea: 2	**Número historia:** 3
Nombre tarea: Creación del formulario para personal.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10 de octubre del 2018	**Fecha fin:** 10 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Se crea el modelo la vista y el controlador mediante scaffold.	

Fuente: Propia

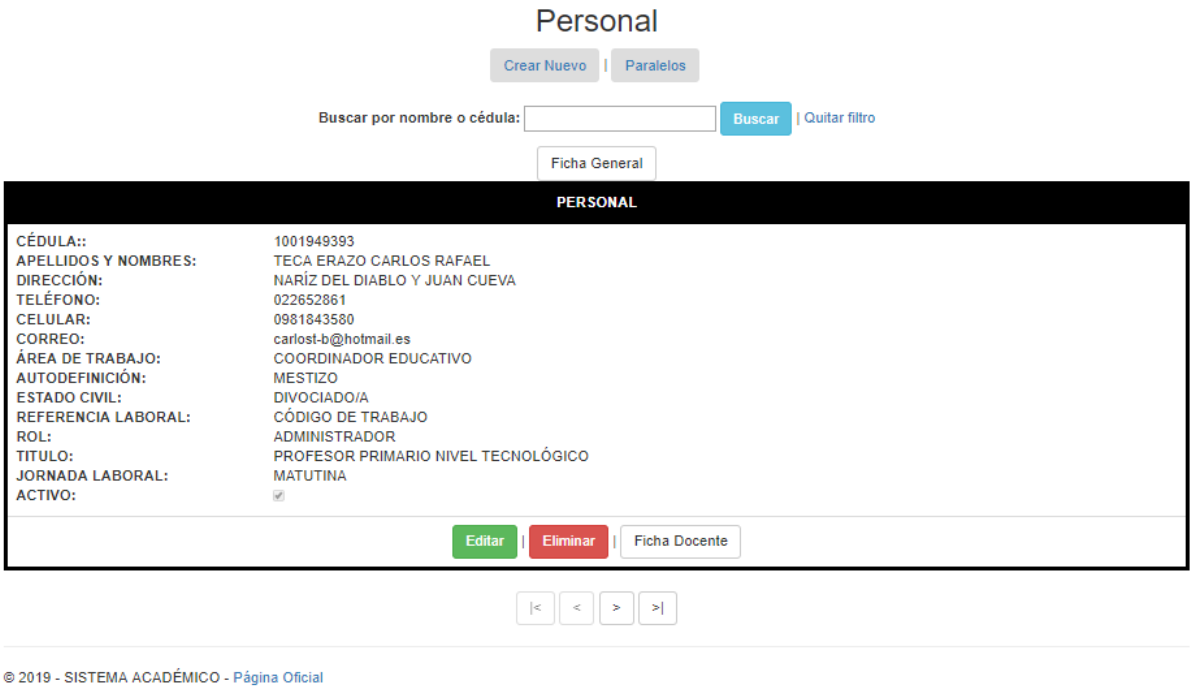


Figura 21: Personal.
Fuente: Propia

2.6.3.4 HISTORIA DE USUARIO 4: DESARROLLO DEL MODULO PARALELOS

Tabla 12: Historia de usuario 3

HISTORIA DE USUARIO	
Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier.	
Número: 4	**Usuario:** Administrador
Nombre de historia: Desarrollo del módulo paralelos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Estimación (horas): 8	**Iteraciones asignadas:** 1
Programador responsable: Diego Quelal	

Descripción: Se analizará, y se construirá un modelo de base de datos, generar las vistas e implementar toda la funcionalidad que se necesita de los paralelos.
Observaciones: Aquí se registra toda la información necesaria de los paralelos con sus respectivas asignaturas y docentes y si está activo o no.
Fecha: 09 de octubre del 2018
Firma:

Fuente: Propia

Tabla 13: Tarea 1 – Historia de usuario 3

Tarea	
Número tarea: 1	**Número historia:** 4
Nombre tarea: Análisis y desarrollo de un modelo de bases de datos para paralelos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 09 de octubre del 2018	**Fecha fin:** 09 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Crear un Modelo de datos para crear la vista Paralelos.	

Fuente: Propia

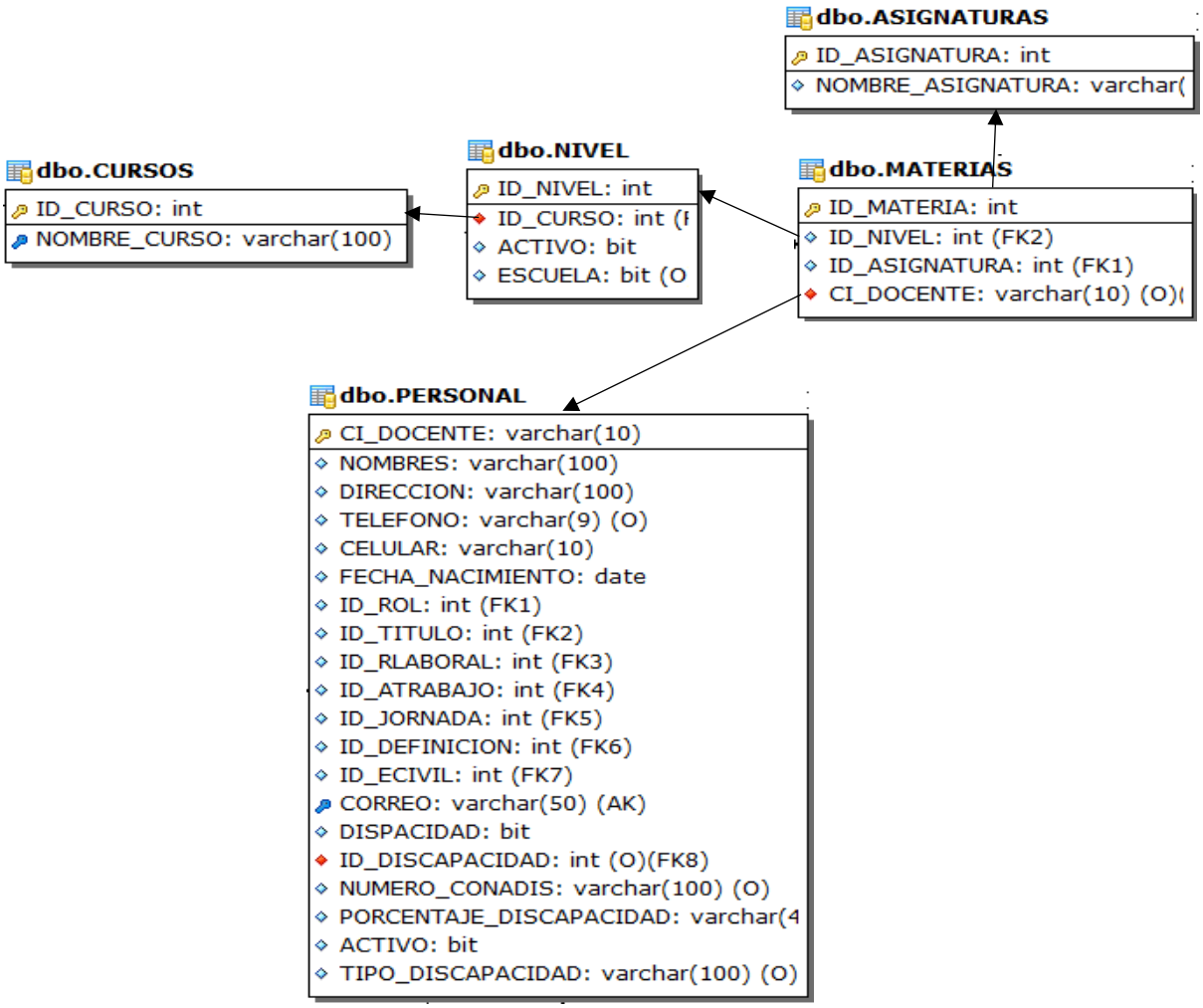


Figura 22: Modelo de datos – Paralelos.
Fuente: Propia

Tabla 14: Tarea 2 – Historia de usuario 3

Tarea	
Número tarea: 2	**Número historia:** 4
Nombre tarea: Creación del formulario para paralelos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10 de octubre del 2018	**Fecha fin:** 10 de octubre del 2018
Programador responsable: Diego Quelal	

Descripción:
Se crea el modelo la vista y el controlador mediante scaffold.

Fuente: Propia

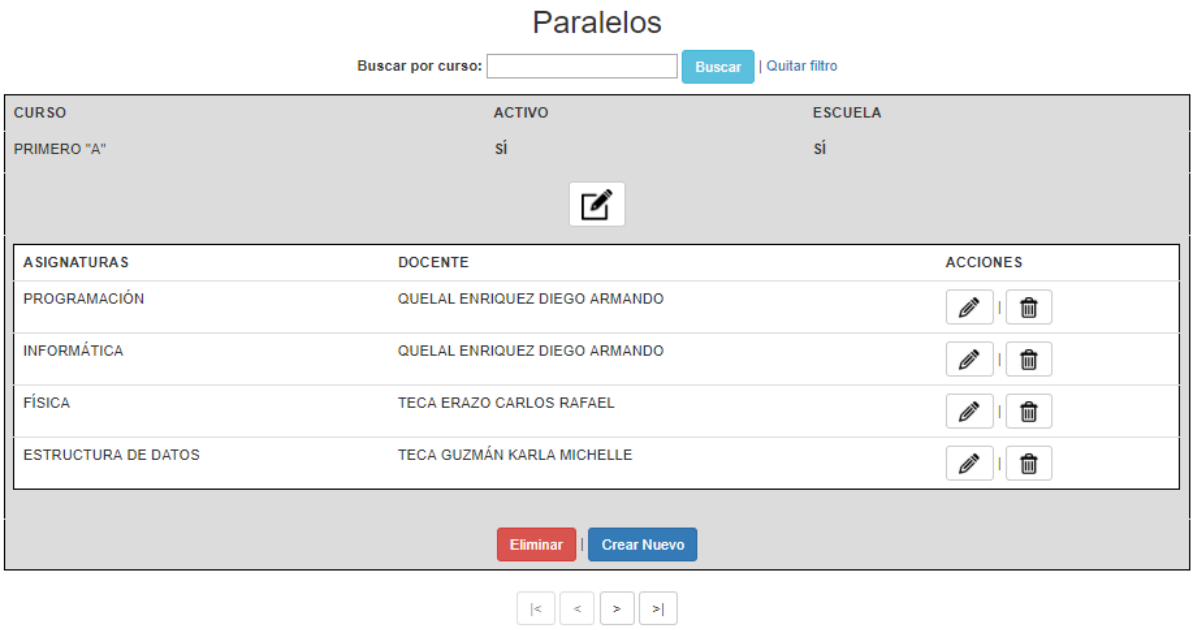


Figura 23: Vista paralelos.
Fuente: Propia

2.6.3.5 HISTORIA DE USUARIO 5: DESARROLLO DEL MODULO ESTUDIANTES

Tabla 15: Historia de usuario 5

HISTORIA DE USUARIO	
Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier.	
Número: 5	**Usuario:** Administrador
Nombre de historia: Desarrollo del módulo estudiantes.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Estimación (horas): 8	**Iteraciones asignadas:** 3
Programador responsable: Diego Quelal	

Descripción: Se analizará, y se construirá un modelo de base de datos, generar las vistas e implementar toda la funcionalidad que se necesita de los estudiantes.
Observaciones: Aquí se registra toda la información necesaria de los estudiantes datos personales, representantes o familiares, situación familiar y si está activo o no.
Fecha: 10 de octubre del 2018
Firma:

Fuente: Propia

Tabla 16: Tarea 1 – Historia de usuario 4

Tarea	
Número tarea: 1	**Número historia:** 5
Nombre tarea: Análisis y desarrollo de un modelo de bases de datos para estudiantes.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10 de octubre del 2018	**Fecha fin:** 10 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Crear un Modelo de datos para crear la vista estudiante.	

Fuente: Propia

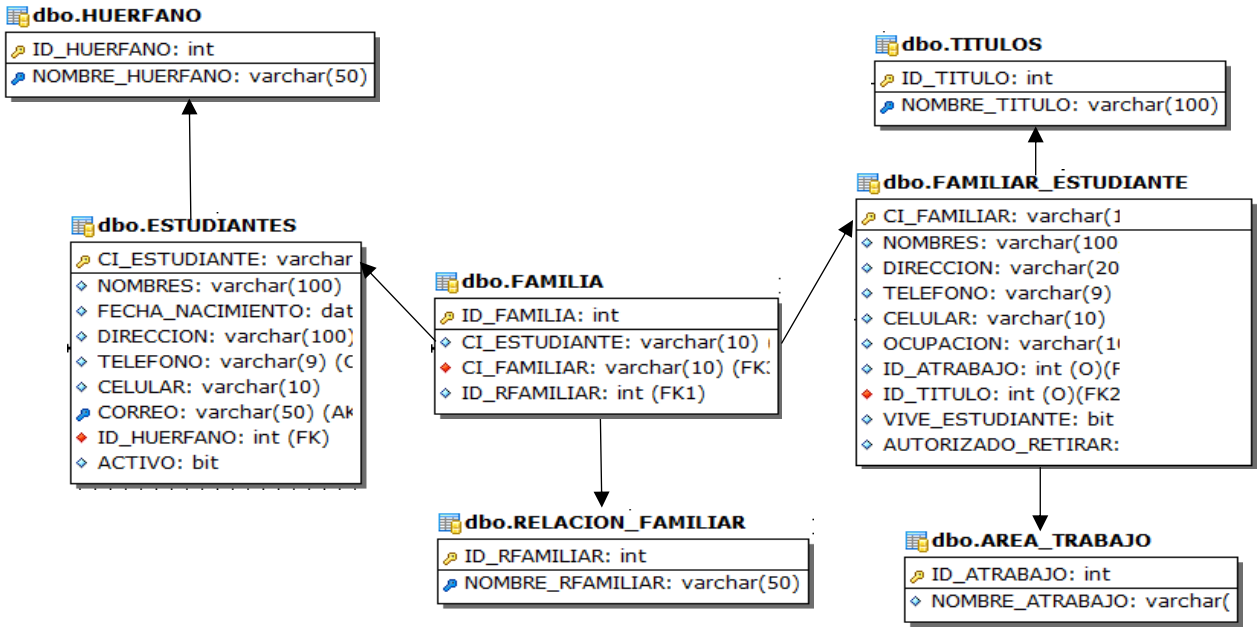


Figura 24: Modelo de datos – Estudiantes.
Fuente: Propia

Tabla 17: Tarea 2 – Historia de usuario 4

Tarea	
Número tarea: 2	**Número historia:** 5
Nombre tarea: Creación de la vista para estudiantes.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 11 de octubre del 2018	**Fecha fin:** 11 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Se crea el modelo la vista y el controlador mediante scaffold.	

Fuente: Propia

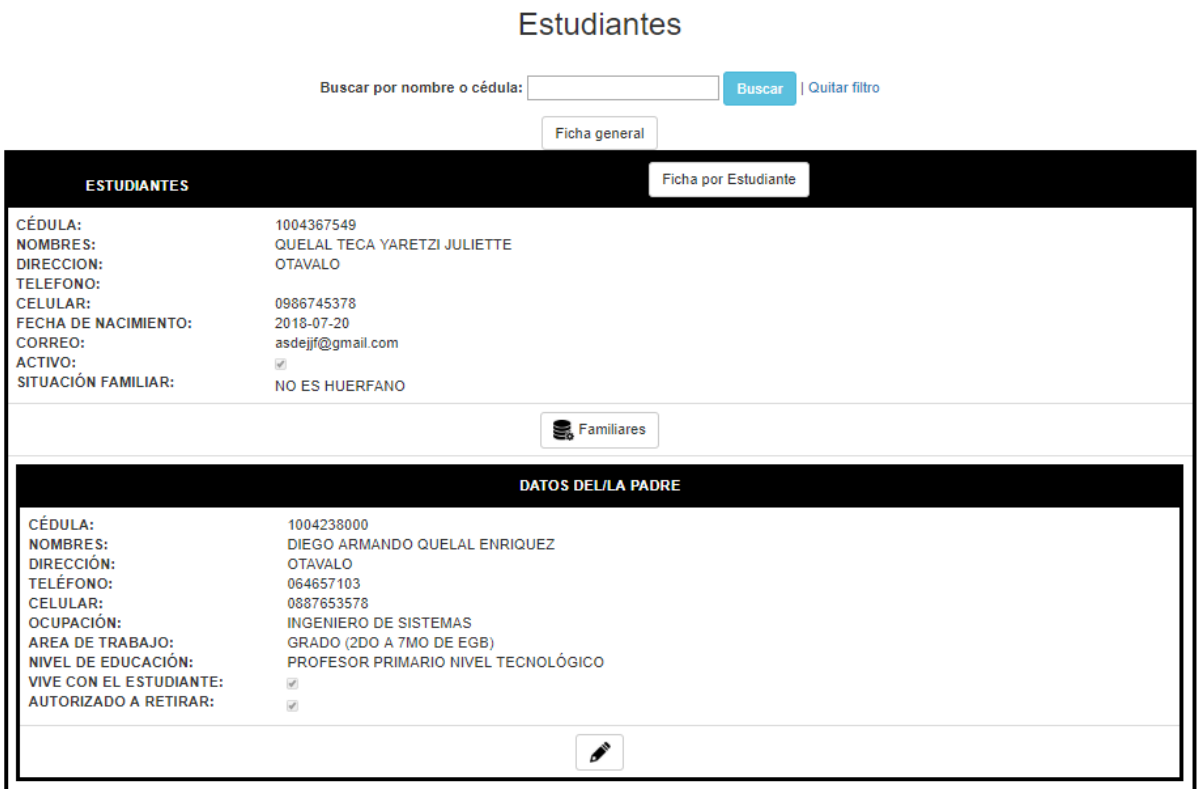


Figura 25: Vista paralelos.
Fuente: Propia

2.6.3.6 HISTORIA DE USUARIO 6: DESARROLLO DEL MODULO MATRÍCULAS

Tabla 18: Historia de usuario 6

HISTORIA DE USUARIO	
Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier.	
Número: 6	**Usuario:** Administrador
Nombre de historia: Desarrollo del módulo matrículas.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Estimación (horas): 8	**Iteraciones asignadas:** 1
Programador responsable: Diego Quelal	

Descripción: Se analizará, y se construirá un modelo de base de datos, generar las vistas e implementar toda la funcionalidad que se necesita de matrículas.
Observaciones: Aquí se selecciona un estudiante período académico y curso al que se va matricular.
Fecha: 11 de octubre del 2018
Firma:

Fuente: Propia

Tabla 19: Tarea 1 – Historia de usuario 6

Tarea	
Número tarea: 1	**Número historia:** 6
Nombre tarea: Análisis y desarrollo de un modelo de bases de datos para matrículas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 11 de octubre del 2018	**Fecha fin:** 11 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Crear un Modelo de datos para crear la vista matriculas.	

Fuente: Propia

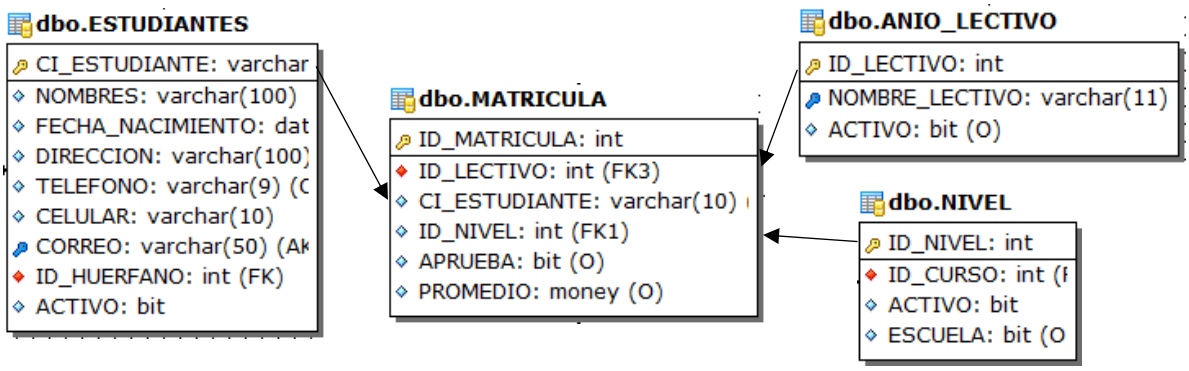


Figura 26: Modelo de datos – matriculas.
Fuente: Propia

Tabla 20: Tarea 2 – Historia de usuario 6

Tarea	
Número tarea: 2	**Número historia:** 6
Nombre tarea: Creación de la vista para matrículas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 12 de octubre del 2018	**Fecha fin:** 12 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Se crea el modelo la vista y el controlador mediante scaffold.	

Fuente: Propia

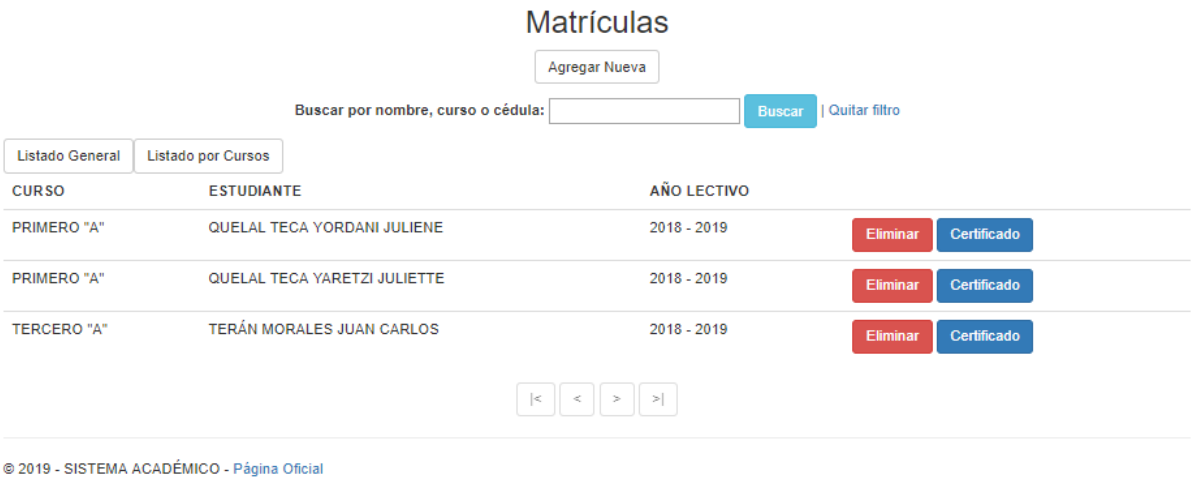


Figura 27: Vista paralelos.
Fuente: Propia

2.6.3.7 HISTORIA DE USUARIO 7: DESARROLLO DEL MODULO PADRES DE FAMILIA

Tabla 21: Historia de usuario 7

HISTORIA DE USUARIO	
Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier.	
Número: 7	**Usuario:** Administrador
Nombre de historia: Desarrollo del módulo matrículas.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Estimación (horas): 8	**Iteraciones asignadas:** 1
Programador responsable: Diego Quelal	
Descripción: Se analizará, y se construirá un modelo de base de datos, generar las vistas e implementar toda la funcionalidad que se necesita de padres de familia.	

Observaciones: Aquí se registra los datos personales, ocupación, instrucción, área de trabajo, autorizado a retirar y si vive con el estudiante.
Fecha: 12 de octubre del 2018
Firma:

Fuente: Propia

Tabla 22: Tarea 1 – Historia de usuario 7

Tarea	
Número tarea: 1	**Número historia:** 7
Nombre tarea: Análisis y desarrollo de un modelo de bases de datos para padres de familia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 12 de octubre del 2018	**Fecha fin:** 12 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Crear un Modelo de datos para crear la vista padres de familia.	

Fuente: Propia

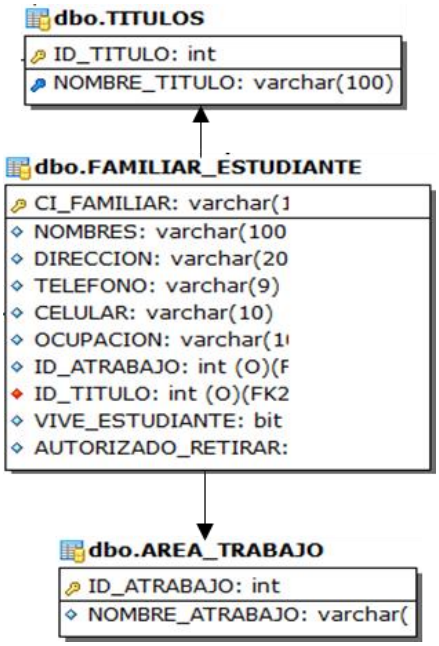


Figura 28: Modelo de datos – padres de familia.
Fuente: Propia

Tabla 23: Tarea 2 – Historia de usuario 7

Tarea	
Número tarea: 2	**Número historia:** 7
Nombre tarea: Creación de la vista para padres de familia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 13 de octubre del 2018	**Fecha fin:** 13 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Se crea el modelo la vista y el controlador mediante scaffold.	

Fuente: Propia

Figura 29: Vista paralelos.
Fuente: Propia

2.6.3.8 HISTORIA DE USUARIO 8: DESARROLLO DEL MODULO HORARIOS

Tabla 24: Historia de usuario 8

HISTORIA DE USUARIO	
Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier.	
Número: 8	**Usuario:** Administrador
Nombre de historia: Desarrollo del módulo horarios.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Estimación (horas): 8	**Iteraciones asignadas:** 1
Programador responsable: Diego Quelal	
Descripción: Se analizará, y se construirá un modelo de base de datos, generar las vistas e implementar toda la funcionalidad que se necesita de horarios.	

Observaciones: Aquí se registra los horarios de cada asignatura con su respectivo docente y curso.
Fecha: 13 de octubre del 2018
Firma:

Fuente: Propia

Tabla 25: Tarea 1 – Historia de usuario 8

Tarea	
Número tarea: 1	**Número historia:** 8
Nombre tarea: Análisis y desarrollo de un modelo de bases de datos para padres de familia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 13 de octubre del 2018	**Fecha fin:** 13 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Crear un Modelo de datos para crear la vista de horarios.	

Fuente: Propia

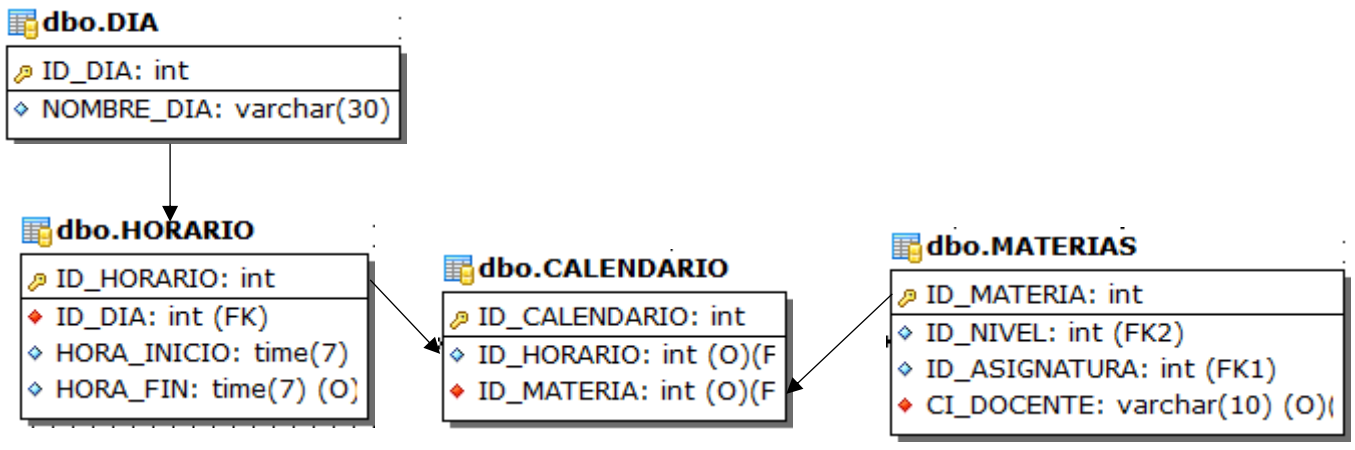


Figura 30: Modelo de datos – horarios.
Fuente: Propia

Tabla 26: Tarea 2 – Historia de usuario 8

Tarea	
Número tarea: 2	**Número historia:** 8
Nombre tarea: Creación de la vista para horarios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 14 de octubre del 2018	**Fecha fin:** 14 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Se crea el modelo la vista y el controlador mediante scaffold.	

Fuente: Propia

Figura 31: Vista horarios.
Fuente: Propia

2.6.3.9 HISTORIA DE USUARIO 9: DESARROLLO DEL MODULO NOTAS

Tabla 27: Historia de usuario 9

HISTORIA DE USUARIO	
Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier.	
Número: 9	**Usuario:** Administrador
Nombre de historia: Desarrollo del módulo notas.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Estimación (horas): 8	**Iteraciones asignadas:** 1
Programador responsable: Diego Quelal	
Descripción: Se analizará, y se construirá un modelo de base de datos, generar las vistas e implementar toda la funcionalidad que se necesita de notas.	
Observaciones: Aquí se registra los aportes de los estudiantes de todas las parciales y exámenes.	
Fecha: 14 de octubre del 2018	
Firma:	

Fuente: Propia

Tabla 28: Tarea 1 – Historia de usuario 9

Tarea	
Número tarea: 1	**Número historia:** 9
Nombre tarea: Análisis y desarrollo de un modelo de bases de datos para notas.	

Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 14 de octubre del 2018	**Fecha fin:** 14 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Crear un Modelo de datos para crear la vista de notas.	

Fuente: Propia

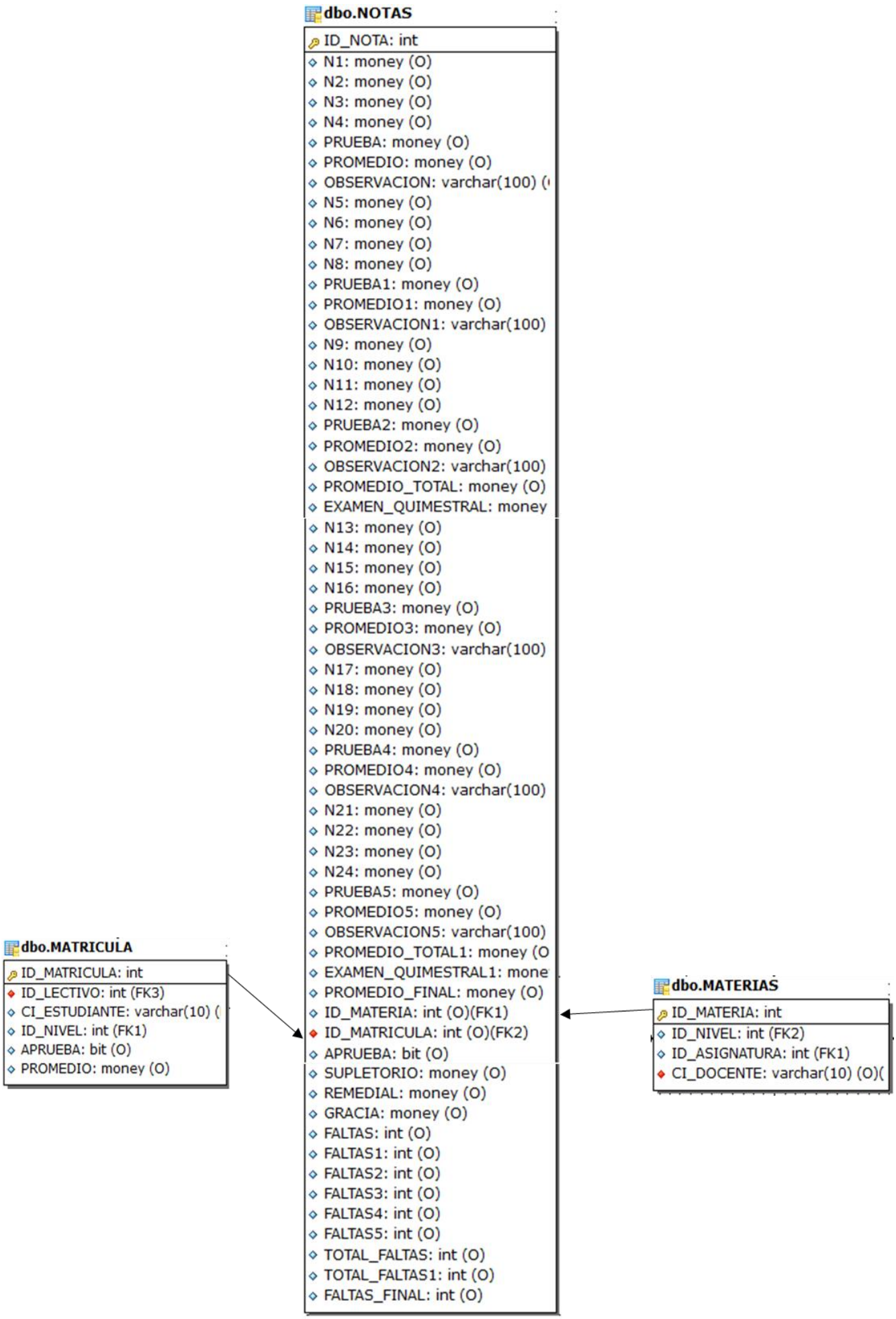

dbo.NOTAS
ID_NOTA: int
N1: money (O)
N2: money (O)
N3: money (O)
N4: money (O)
PRUEBA: money (O)
PROMEDIO: money (O)
OBSERVACION: varchar(100) (
N5: money (O)
N6: money (O)
N7: money (O)
N8: money (O)
PRUEBA1: money (O)
PROMEDIO1: money (O)
OBSERVACION1: varchar(100)
N9: money (O)
N10: money (O)
N11: money (O)
N12: money (O)
PRUEBA2: money (O)
PROMEDIO2: money (O)
OBSERVACION2: varchar(100)
PROMEDIO_TOTAL: money (O)
EXAMEN_QUIMESTRAL: money
N13: money (O)
N14: money (O)
N15: money (O)
N16: money (O)
PRUEBA3: money (O)
PROMEDIO3: money (O)
OBSERVACION3: varchar(100)
N17: money (O)
N18: money (O)
N19: money (O)
N20: money (O)
PRUEBA4: money (O)
PROMEDIO4: money (O)
OBSERVACION4: varchar(100)
N21: money (O)
N22: money (O)
N23: money (O)
N24: money (O)
PRUEBA5: money (O)
PROMEDIO5: money (O)
OBSERVACION5: varchar(100)
PROMEDIO_TOTAL1: money (O
EXAMEN_QUIMESTRAL1: mone
PROMEDIO_FINAL: money (O)
ID_MATERIA: int (O)(FK1)
ID_MATRICULA: int (O)(FK2)
APRUEBA: bit (O)
SUPLETORIO: money (O)
REMEDIAL: money (O)
GRACIA: money (O)
FALTAS: int (O)
FALTAS1: int (O)
FALTAS2: int (O)
FALTAS3: int (O)
FALTAS4: int (O)
FALTAS5: int (O)
TOTAL_FALTAS: int (O)
TOTAL_FALTAS1: int (O)
FALTAS_FINAL: int (O)
dbo.MATRICULA
ID_MATRICULA: int
ID_LECTIVO: int (FK3)
CI_ESTUDIANTE: varchar(10) (
ID_NIVEL: int (FK1)
APRUEBA: bit (O)
PROMEDIO: money (O)
dbo.MATERIAS
ID_MATERIA: int
ID_NIVEL: int (FK2)
ID_ASIGNATURA: int (FK1)
CI_DOCENTE: varchar(10) (O)(

Figura 32: Modelo de datos – notas.
Fuente: Propia

Tabla 29: Tarea 2 – Historia de usuario 9

Tarea	
Número tarea: 2	**Número historia:** 9
Nombre tarea: Creación de la vista para notas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 15 de octubre del 2018	**Fecha fin:** 15 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Se crea el modelo la vista y el controlador mediante scaffold.	

Fuente: Propia

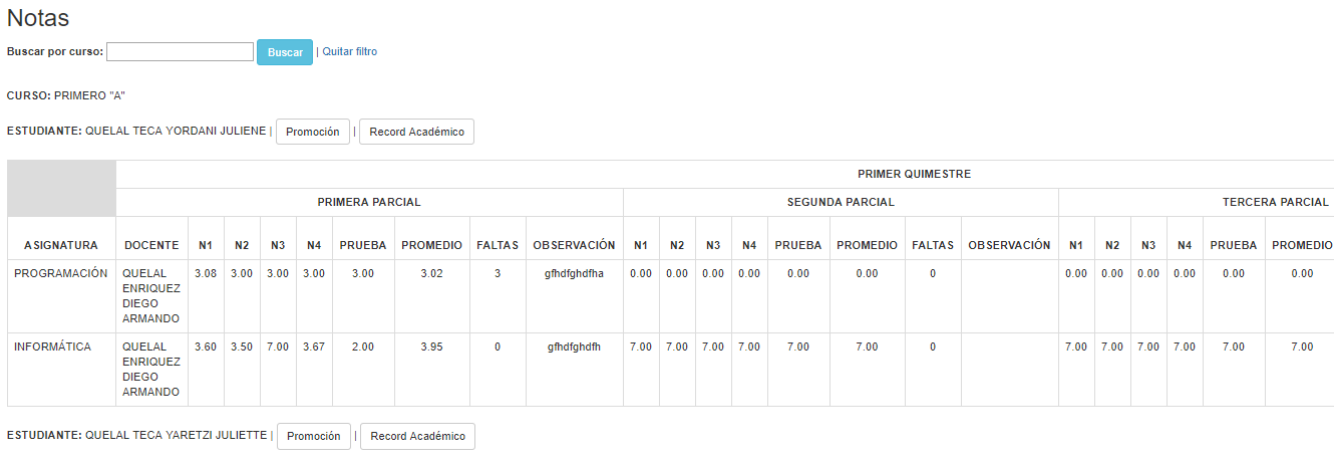


Figura 33: Vista notas.
Fuente: Propia

2.6.3.10 HISTORIA DE USUARIO 10: DESARROLLO DEL MODULO ACTIVAR NOTAS

Tabla 30: Historia de usuario 10

HISTORIA DE USUARIO	
Desarrollo del sistema académico aplicando la herramienta ASP.NET CORE 2 para la escuela Eufrasia Pelletier.	
Número: 10	**Usuario:** Administrador
Nombre de historia: Desarrollo del módulo activar notas.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Estimación (horas): 8	**Iteraciones asignadas:** 1
Programador responsable: Diego Quelal	
Descripción: Se analizará, y se construirá un modelo de base de datos, generar las vistas e implementar toda la funcionalidad que se necesita de activar notas.	
Observaciones: Aquí activarán las notas que estarán visibles para su registro.	
Fecha: 15 de octubre del 2018	
Firma:	

Fuente: Propia

Tabla 31: Tarea 1 – Historia de usuario 10

Tarea	
Número tarea: 1	**Número historia:** 10
Nombre tarea: Análisis y desarrollo de un modelo de bases de datos para activar notas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1

Fecha inicio: 15 de octubre del 2018	**Fecha fin:** 15 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Crear un Modelo de datos para crear la vista de activar notas.	

Fuente: Propia

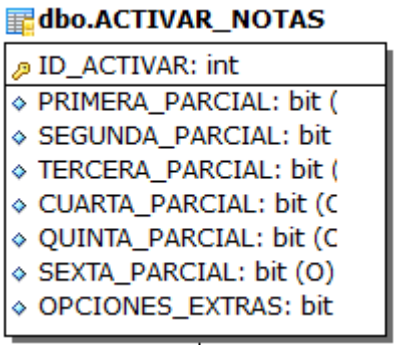


Figura 34: Modelo de datos – activar notas.
Fuente: Propia

Tabla 32: Tarea 2 – Historia de usuario 10

Tarea	
Número tarea: 2	**Número historia:** 10
Nombre tarea: Creación de la vista para notas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 16 de octubre del 2018	**Fecha fin:** 16 de octubre del 2018
Programador responsable: Diego Quelal	
Descripción: Se crea el modelo la vista y el controlador mediante scaffold.	

Fuente: Propia

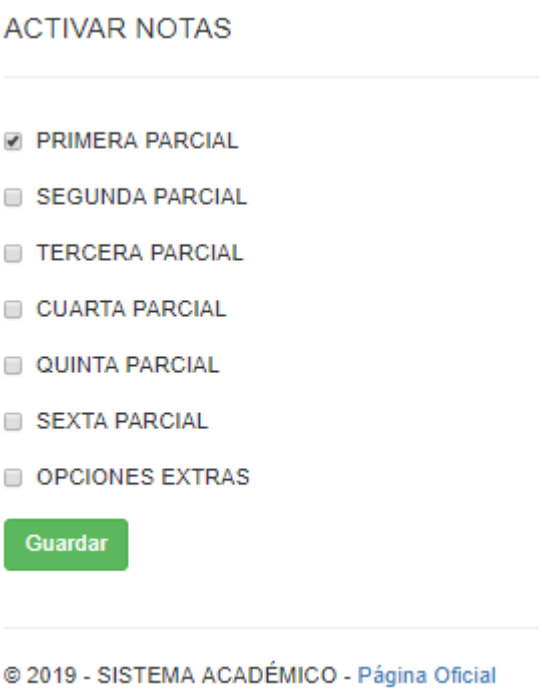


Figura 35: Vista activar notas.
Fuente: Propia

2.6.4 DIAGRAMA DE CASO DE USO

Según la metodología aplicada se proponen los siguientes casos de uso.

2.6.4.1 CASO DE USO ADMINISTRACIÓN DEL SISTEMA

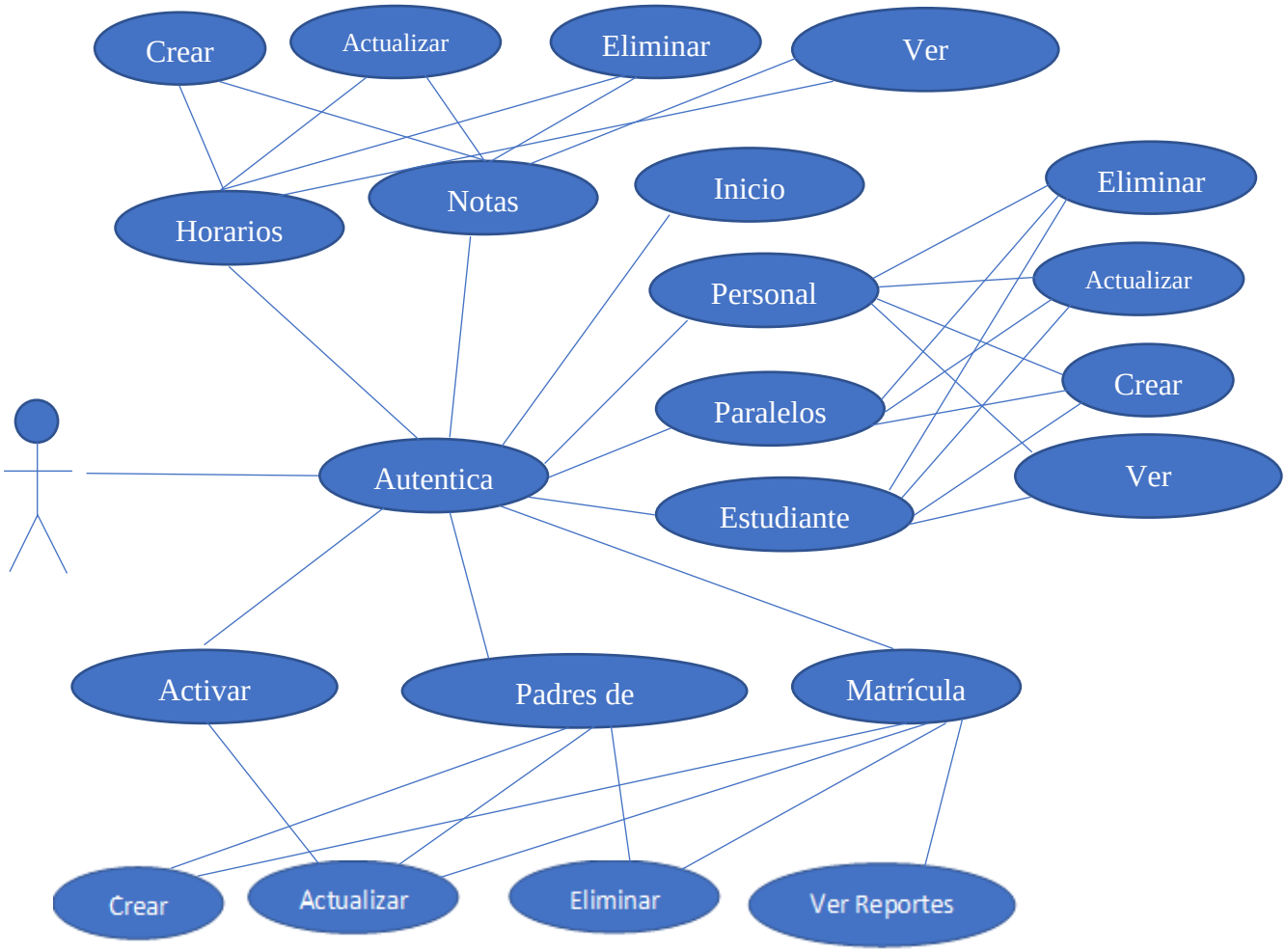


Figura 36: Caso de uso usuario administrador.
Fuente: Propia.

Tabla 33: Descripción caso de uso usuario administrador

Caso de uso	Usuario Administrador.
Descripción	El administrador es quien configura el Sistema Ingresa toda la información, modifica, mira y elimina, crea usuarios y roles.
Actor	Administrador.
Condiciones previas	Tener un rol de administrador.
Flujo básico eventos	Registrar, actualizar o inactivar usuarios. Crea, modifica, Visualiza o eliminar información de todos los módulos del sistema.

Flujos alternativos	Se mostrará un mensaje de error si: ✓ Si hay una excepción

Fuente: Propia.

2.6.4.2 CASO DE USO DOCENTE

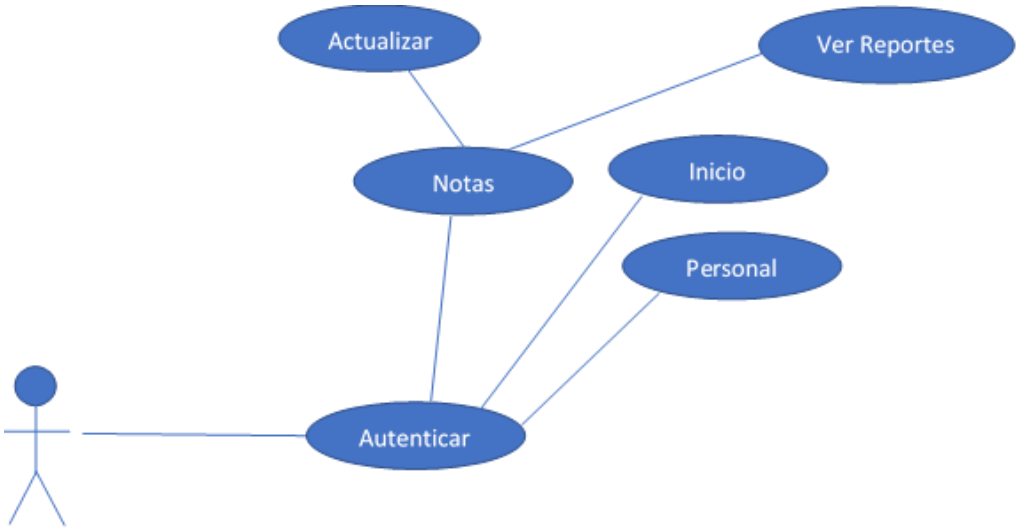


Figura 37: Caso de uso docente.
Fuente: Propia.

Tabla 34: Descripción caso de uso usuario administrador

Caso de uso	Usuario Docente.
Descripción	El docente edita las notas de sus cursos, mira reportes, mira su perfil y horarios.
Actor	Docente.
Condiciones previas	Tener un rol de docente.

Flujo básico eventos	Modifica notas, Visualiza información de 2 módulos del sistema.
Flujos alternativos	Se mostrará un mensaje de error si: ✓ Si hay una excepción

Fuente: Propia.

2.6.4.3 CASO DE USO ESTUDIANTE

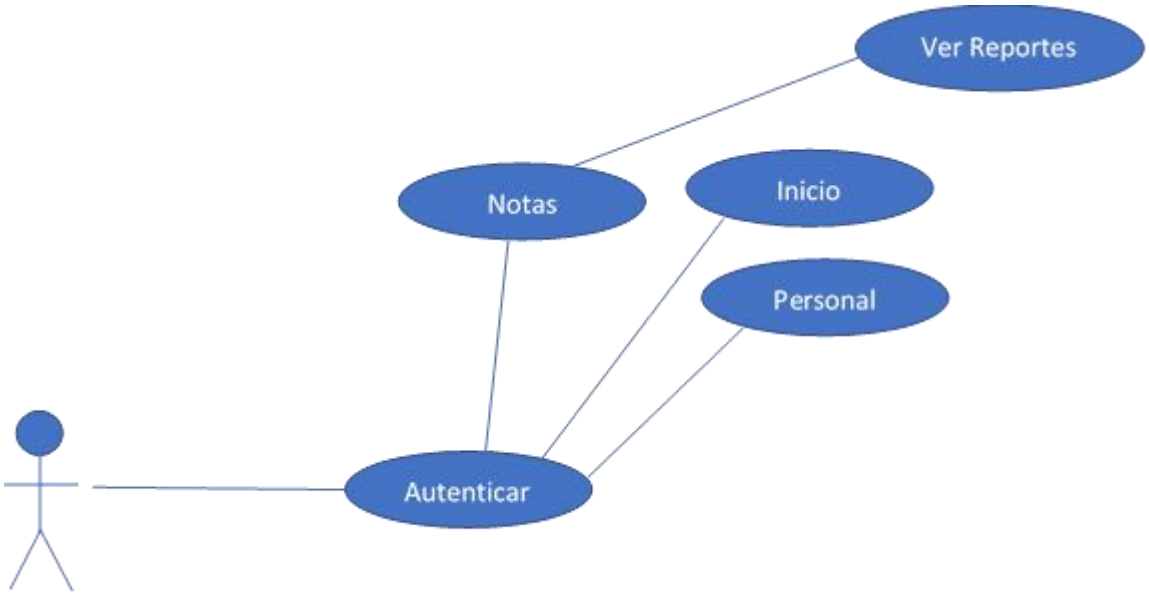


Figura 38: Caso de uso estudiante.
Fuente: Propia.

Tabla 35: Descripción caso de uso usuario estudiante

Caso de uso	Usuario estudiante.
Descripción	El docente mira las notas de su curso, mira reportes, mira su perfil y horarios.
Actor	Estudiante.

Condiciones previas	Tener un rol de Estudiante.
Flujo básico eventos	Visualiza información de 3 módulos del sistema.
Flujos alternativos	Se mostrará un mensaje de error si: ✓ Si hay una excepción

Fuente: Propia.

2.7. CONSTRUCCIÓN DE LA APLICACIÓN

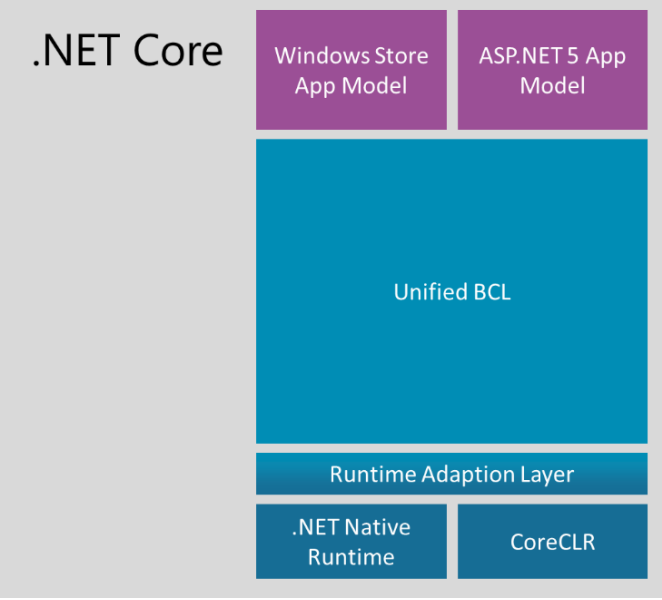
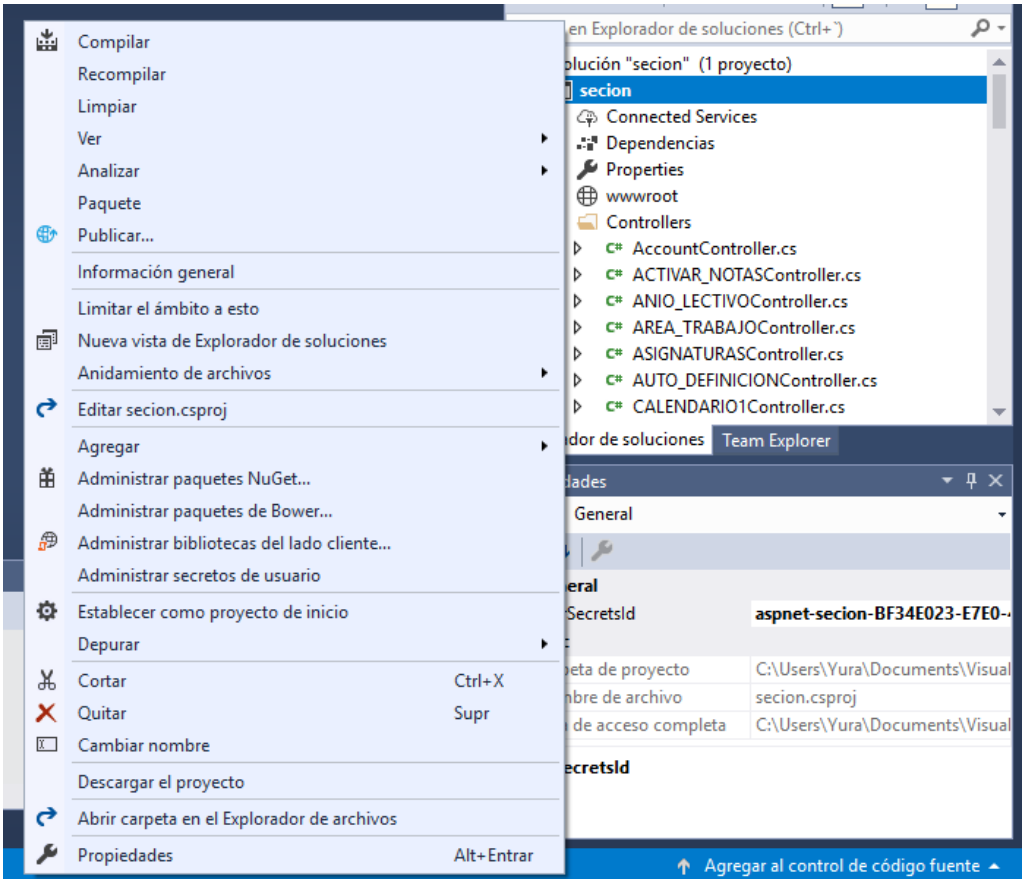


Figura 39: Modelo de construcción de una Aplicación ASP.NET CORE.
Fuente: http://www.diveria.com/blog/2016/08/net-core-whats-new-in-net-world/

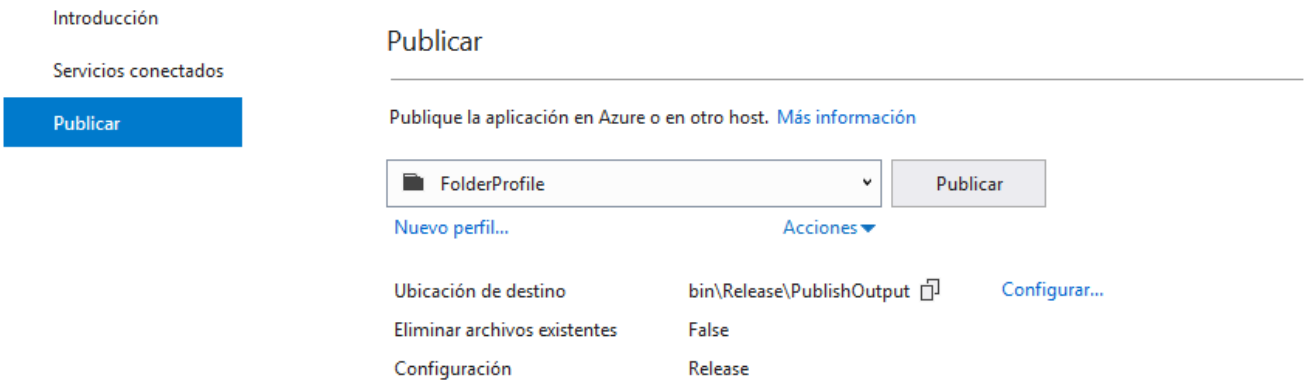
2.8. IMPLEMENTACIÓN

Para realizar la Implementación se detallan los siguientes pasos.

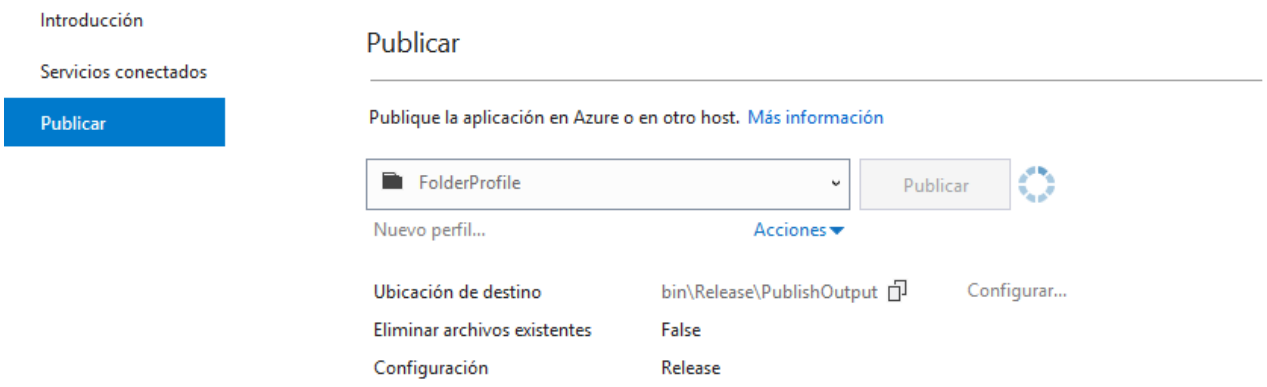
- Click derecho sobre el proyecto



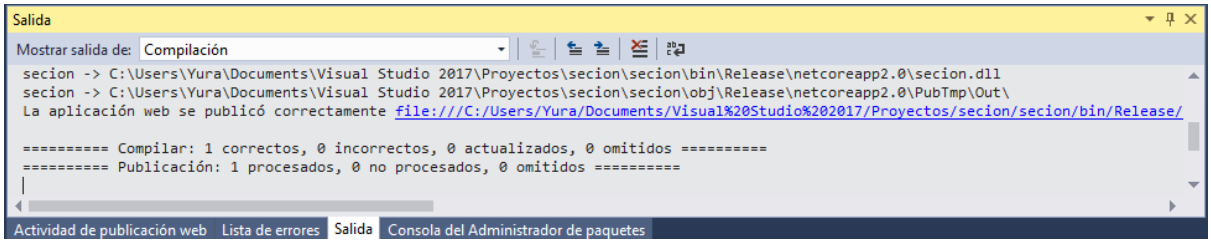
- Publicar
- Creamos un Perfil si no lo tenemos



- Dejamos todo por defecto y publicar



- Si no hay errores el proceso es correcto



- Podemos Publicar localmente o en la Web

Link Disponible: http://sistemaacademico.apphb.com/

CAPÍTULO III

3. VALIDACIÓN Y RESULTADOS

3.1. DIAGNÓSTICO

Con el uso de la herramienta ASP.NET CORE 2 de Visual Studio 2017 se va pudo generar una aplicación web aprovechando en gran medida el potencial de dicha herramienta, además mejoramos los servicios a todos los involucrados en la institución educativa y dimos solución a sus problemas ya que el sistema tiene un conjunto de opciones prácticas que ayudan en la gestión académica, brindando así un mejor servicio a la comunidad institucional ya que se gestiona sus procesos e información de forma automática mediante esta nueva tecnología, con la que está desarrollado el Sistema Académico para las personas dentro y fuera de la institución.

3.2. EVALUCIÓN Y ANÁLISIS

El análisis productivo se determina como excelente ya que se mejoró la gestión en los procesos educativos de la institución.

Tiempo de desarrollo: Lo que tomó el desarrollo en cuestión del tiempo es bajo ya que se pudo realizar la implementación con rapidez, y permitiendo a la institución hacer uso del sistema rápidamente.

Facilidad de manejo: la solución web es de fácil manejo por lo que los involucrados disponen de más tiempo para invertirlo en otras actividades y merando así el proceso de gestión.

Mejora de procesos: La funcionalidad del Sistema Académico permite realizar todos los procesos de gestión con calidad y rapidez teniendo buenas reacciones de toso los usuarios del sistema tanto internos como externos.

3.3. RESULTADOS

El sistema puede gestionar y administrar la información de los estudiantes y docentes, se permite realizar matriculas, pases de nivel, ingresar notas, visualizar el rendimiento académico, asistencia por estudiante, genera informes automáticos no solo por alumno sino también por curso, todos estos datos se los puede observar desde la web en cualquier parte

del mundo con tan solo hacer un clic ya que está alojado en AppHarbor que es un hosting gratuito, beneficiándose de esta manera directamente a la Escuela Eufrasia Pelletier.

3.4. IMPACTOS

Impacto Social

Como la Escuela Eufrasia Pelletier, no contaba con ningún Sistema Académico aprovechamos al máximo los beneficios de la herramienta antes descrita, para mitigar los problemas que existían en la institución y dejamos un pequeño estudio práctico de esta novedosa herramienta para uso de la comunidad.

Impacto Ambiental

Con la implantación del Sistema Académico ayudamos al medio ambiente ya que no se está utilizando hojas de papel normal de impresión, si no que se utiliza hojas hecha a base de caña de azúcar para preservar la naturaleza, además como el Sistema permite la consulta de notas online se ha reducido el consumo de papel.

Impacto Económico

Con el Sistema Académico la escuela Eufrasia Pelletier tiene un ahorro de dinero, ya que se entiende, que una solución de este tipo se está gestionando los procesos, recursos humanos e información de padres de familia y estudiantes, por lo que la gestión resulta más eficiente conllevando a una reducción tanto en tiempo como dinero, por la adecuada gestión del personal que labora en la institución.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- La herramienta utilizada de fácil manejo y optimiza el tiempo de desarrollo
- Una solución Web hecha ASP.NET CORE garantiza el uso de sus módulos y muestra una interfaz amigable con el usuario.
- En uso de frameworks de diseño y eventos se facilita mucho ya que ASP.NET CORE los tiene integrados.
- Optimiza la escritura de código ya que genera módulos funcionales únicamente con generar un scaffold.
- La aplicación fue todo un éxito.

RECOMENDACIONES

- Se recomienda a la institución hacer un presupuesto económico, para cuando el limite gratuito de base de datos se supere.
- Se recomienda hacer uso de esta herramienta por todos los beneficios anteriormente detallados.
- Se recomienda a la carrera integrar esta esta herramienta en mayor cantidad en los niveles de estudio.
- Se recomienda a los usuarios de la herramienta apoyar en el desarrollo de software libre ya que ASP.NET CORE está bajo esa licencia.
- Se recomienda estudiar más a profundidad esta novedosa herramienta.

REFERENCIAS BIBLIOGRÁFICAS

Anderson, R. (12 de 04 de 2017). *Microsoft Docs*. Obtenido de Microsoft Docs: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/validation?view=aspnetcore-2.1

Anderson, R. (13 de 03 de 2018). *Microsoft*. Obtenido de Microsoft: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/?view=aspnetcore-2.1&tabs=aspnetcore2x

Appharbor. (01 de 01 de 2018). *Appharbor*. Obtenido de Appharbor: https://appharbor.com/page/about-us

Appharbor. (01 de 01 de 2018). *Appharbor*. Obtenido de Appharbor: https://appharbor.com/

BERSANO, D. (08 de 25 de 2014). *DIEGO BERSANO*. Obtenido de DIEGO BERSANO: https://diegobersano.com.ar/2014/08/25/appharbor-y-azure-alternativas-para-publicar-aplicaciones-en-la-nube-de-forma-gratuita/

Chiaretta, S., & Lattanzi, U. (01 de 01 de 2017). *Syncfusion.* Obtenido de https://www.syncfusion.com/ebooks/asp_net_core_succinctly/what-are-net-core-and-asp-net-core

Constituyente Asamblea Nacional, E. (2011). *Educar*. Obtenido de http://www.educar.ec/servicios/regla_loei-5.html

Danysoft. (02 de 14 de 2017). *Danysoft*. Obtenido de Danysoft: https://www.danysoft.com/asp-net-core-nuevo-marco-de-codigo-abierto-y-multiplataforma/

Ecured. (01 de 01 de 2016). *Ecured*. Obtenido de Ecured: https://www.ecured.cu/Lenguaje_de_Programaci%C3%B3n_C_Sharp

Enríquez, S. (25 de Abril de 2017). *Repositorio UTN*. Obtenido de Repositorio UTN: http://repositorio.utn.edu.ec/bitstream/123456789/6904/1/04%20ISC%20439%20TRABAJO%20DE%20GRADO.pdf

Enríquez, S. (01 de 01 de 2017). *Repositorio UTN.* Obtenido de http://repositorio.utn.edu.ec/bitstream/123456789/6904/1/04%20ISC%20439%20TRABAJO%20DE%20GRADO.pdf

Freddie. (14 de Junio de 2014). *Cristalab*. Obtenido de Cristalab: http://www.cristalab.com/blog/que-significa-backend-y-frontend-en-el-diseno-web-c106224l/

Holguera, J. (22 de 11 de 2010). *Desarrollo Web*. Obtenido de Desarrollo Web: https://desarrolloweb.com/articulos/la-sintaxis-razor.html

Informatica, L. R. (01 de 01 de 2015). *La Revista Informatica*. Obtenido de La Revista Informatica: http://www.larevistainformatica.com/C1.htm

Makesoft. (01 de 01 de 2017). *Makesoft*. Obtenido de Makesoft: https://www.makesoft.es/es/productos/microsoft-sql-server/

Microsoft. (08 de 08 de 2017). *Introduction to ASP.NET Core*. Obtenido de Microsoft: https://docs.microsoft.com/en-us/aspnet/core/

Microsoft. (01 de 01 de 2018). *Msdn*. Obtenido de Msdn: https://msdn.microsoft.com/es-es/library/hh833994(v=vs.108).aspx

Miller, R. (22 de 02 de 2018). *Docs*. Obtenido de Docs: https://docs.microsoft.com/es-es/ef/core/providers/

Ollero Sánchez, C. (2014). *Manual. Creación de páginas web con el lenguaje de marcas (UF1302/MF0950_2). Certificados de profesionalidad. Confección y publicación de páginas Web (IFCD0110).* Mexico: Editorial CEP, S.L.

Posadas, M. (15 de 11 de 2017). *Bit*. Obtenido de Bit: https://www.bit.es/knowledge-center/novedades-para-el-desarrollo-web-en-net-core-2-0/
Quelal, D. (20 de 01 de 2019). *Sistema Académico*. Obtenido de http://sistemaacademico.apphb.com/
Rouse, M. (01 de Enero de 2017). *SearchDataCenter*. Obtenido de SearchDataCenter: https://searchdatacenter.techtarget.com/es/definicion/Internet-de-las-cosas-IoT
Sanderson, S. (20 de 02 de 2018). *Microsoft Docs*. Obtenido de Microsoft Docs: https://docs.microsoft.com/en-us/aspnet/core/client-side/spa/angular?view=aspnetcore-2.1&tabs=visual-studio
Smith, S. (07 de 01 de 2018). *Microsoft Docs*. Obtenido de Microsoft Docs: https://docs.microsoft.com/es-es/aspnet/core/mvc/overview?view=aspnetcore-2.1
Smith, S. (27 de 06 de 2018). *Microsoft Docs*. Obtenido de Microsoft Docs: https://docs.microsoft.com/es-es/dotnet/standard/modern-web-apps-azure-architecture/develop-asp-net-core-mvc-apps
Téllez, M. d. (2014). *Bases de datos distribuidas.* LA Tunas: Editorial Universitaria.
Wenzel, M. (30 de 03 de 2017). *Microsoft*. Obtenido de Microsoft: https://docs.microsoft.com/es-es/dotnet/framework/get-started/net-core-and-open-source
Wikipedia. (25 de Junio de 2017). *Wikipedia*. Obtenido de Wikipedia: https://en.wikipedia.org/wiki/.NET_Foundation