Implementación de un portal web que automatice los procesos para las empresas del sector del calzado en la ciudad de Cúcuta

Henry Nuncira Rincón

Trabajo de grado para optar el título de Ingeniero de Sistemas

Director

Carlos Adolfo Beltrán

Ingeniero de Sistemas

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Ingeniería de Sistemas

Bucaramanga

2022

Dedicatoria

A Dios por permanecer por enseñarme a escuchar y ser paciente. Todo lo que he logrado se lo debo a Él.

A mi madre Adalgisa, por ser incondicional, por estar al 100% conmigo por confiar en mí y motivarme cada día a seguir luchando por mis sueños.

A mi padre Henry, por dar su máximo esfuerzo para que nunca faltara en mi mesa el pan, por ser ejemplo de responsabilidad y lucha, por todo el sacrificio y esfuerzo que hizo para verme alcanzar esta meta, por ser incondicional y demostrar que se puede amar a los hijos. Siempre será mi guía y mi inspiración

A mis tíos Jesús y Victoria, siempre estuvieron, desde el día 0 en este proceso, siempre encontré en ellos una familia y ese calor de hogar, a ellos sumamente agradecido por siempre tener puertas abiertas para mi

Agradecimientos

Al profesor Ing. Carlos Adolfo Beltrán, por confiar en mí y apoyarme en todo el proceso.

A la Universidad Industrial de Santander por ser una institución digna de orgullo, a la escuela de Ingeniería de Sistemas por la formación que me brindó y que ahora culmina, dejando invaluables enseñanzas.

Finalmente agradezco a cada una de las personas, familiares y amigos que estuvieron de alguna manera en este proceso.

Y por ultimo me agradezco a mí, por ser resiliente, tener claro hacia donde voy y luchar por alcanzarlo.

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Resumen

Título: Implementación de un portal web que automatice los procesos para las empresas del sector del calzado en la ciudad de Cúcuta*

Autor: Henry Nuncira Rincón**

Palabras Clave: Api, rest, software, microservicios, comercializadora de calzado.

Descripción: Basado en el rápido crecimiento de las ventas por internet y la necesidad de competir en un mercado ágil como es el eCommerce , surge una necesidad en todas las comercializadoras de calzado la cual es tener la información en tiempo real y que sus productos puedan ser exhibidos en páginas de ventas por internet como Facebook, Instagram etc; es por ello que en este proyecto decidimos abordar el problema creando una página web que lleve el registro de ventas, productos y sea administrable para de esta forma dar solución a dicho problema.

Se puede observar que se implementó bajo arquitecturas API REST por su facilidad en el sostenimiento y mantenimiento del software para de esta forma dar también sostenibilidad en el tiempo y podamos crecer a la velocidad que la tecnología y el internet nos lo permiten.

Tecnologías de alto impacto y crecimiento como SQL Server, ASPX.NET y ANGULAR se diseñó un aplicativo sencillo novedoso no solo dando solución al problema sino se creó un primer prototipo que simula una comercializadora en el cual podemos ver registros de ventas, estadísticas, control de productos, inventarios toda esta información es obtenida en tiempo real y puede ser consumida desde clientes web y móviles debido a su diseño responsive.

Manejamos un control de versiones del software desde su diseño y desarrollo llevando un control de cambios y versiones del mismo software. En esta tesis de grado podemos ver el prototipo CCCVentas.1.0.0 versión.

* Trabajo de grado.

**Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas. Director: Carlos Adolfo Beltrán. Ingeniero de Sistemas.

Abstract

Title: Implementation of a web portal that automates processes for companies in the footwear sector in the city of Cúcuta*

Author: Henry Nuncira Rincón**

Key Words: Api, rest, software, microservices, footwear marketer.

Description: Based on the rapid growth of internet sales and the need to compete in an agile market such as eCommerce, a need arises in all footwear marketers which is to have information in real time and that their products can be displayed in Internet sales pages such as Facebook, Instagram, etc; That is why in this project I addressed the problem by creating a web page that keeps track of sales, products and is manageable in order to solve this problem.

It can be seen that it was implemented under API REST architectures for its ease in supporting and maintaining the software in order to also provide sustainability over time and we can grow at the speed that technology and the internet allow us.

High-impact technologies such as SQL Server, ASPX.NET and ANGULAR developed a new simple application not only solving the problem but also creating a first prototype that simulates a marketer in which we can see sales records, statistics, control of products, inventories, all this information is obtained in real time and can be consumed from web and mobile clients due to its responsive design.

We manage a version control of the software from its design and development, carrying out a control of changes and versions of the same software. In this degree thesis we can see the prototype version CCCVentas.1.0.0.

* Degree Word.

** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas. Director: Carlos Adolfo Beltrán. Ingeniero de Sistemas.

1. Planteamiento y Justificación del Problema

En Norte de Santander el sector del calzado es muy conocido a nivel regional y nacional por su buena calidad, bajos precios y variedad, es por esto por lo que, en el área metropolitana de la capital norte santandereana, existen muchas microempresas quese dedican a la producción y otras a la comercialización de calzado. Este sector tiene una tradición de muchos años que ha ido creciendo con el tiempo, mejorando la economía de la región y aumentando la demanda de este producto. Esto ha llevado a que muchas de estas pequeñas y medianas empresas, tengan dificultades a la hora de llevar sus cuentas, debido a que han adoptado modelos tradicionales económicos no digitales. Lo anterior ha causado un mal manejo de inventarios, también dificultadesa la hora de calcular el sueldo a sus empleados, llevar un control del flujo de sus ventas y de la existencia del calzado disponible en las bodegas, al igual que la información de contacto y los pendientes con sus proveedores. Todo esto acumulado afecta a la empresa, a tal punto que se han generado pérdidas económicas y en otras ocasiones una demora en la atención a sus clientes, evitando la expansión de dichas empresas aumentando la complejidad a la hora de querer abordar diversos mercados. En el mercado actual, existen páginas web y programas de escritorio que podrían ayudar a automatizar muchas de estas funciones, pero la mayoría de los locales no cuentan con dichos softwares debido a su inadaptabilidad con sus necesidades y pocos espacios para instalar computadores, sin contar que este tipo de programas tiene costos elevados en la adquisiciónde sus licencias para ser utilizados. Por esto surge la necesidad de un software que pueda ser utilizado en cualquier lugar y momento como lo son las plataformas web, que cuentan con formatos "responsive"que son más prácticos y fáciles de manejar y se pueden ejecutar en cualquier aparato que tenga conexión a internet. Una plataforma que pueda acoplarse a sus

necesidades de manera fácil, sencilla y que cumpla con todas las funcionalidades requeridas para dar una solución satisfactoria.

En el siguiente trabajo de grado se propone la creación de un prototipo WEB multiplataforma a la medida, versátil e intuitivo, de fácil manejo, enfocado en la automatización y digitalización de los registros diarios de ventas, gastos e inventarios de una empresa distribuidora de calzado en Cúcuta. Además, la aplicación brinda la posibilidad de llevar un registro de los proveedores y los clientes que compran en esta empresa. De esta manera la empresa lleva registros más ordenados que pueden ser visualizados de manera fácil y rápida, evitando pérdidas económicas y brindando un mejor servicio a los clientes.

La aplicación permitirá actualizar en tiempo real la base de datos, en donde se lleva un registro del calzado que se compra y se vende (al por mayor o detal), esto permitirá que al final del día se puedacalcular las ganancias y los gastos que tuvo la empresa, minimizando las pérdidas tanto del inventariocomo monetarias. Además, la base de datos podrá dar acceso desde cualquier dispositivo y permitiráa otros empleados realizar búsquedas y filtros en tiempo real haciendo más corta y eficaz la atencióna los clientes.

Finalmente, para que el registro de compra y venta de calzado sea más fácil de llenar, la aplicación contará con un modo de llenado manual y automatizado, según los datos ingresados. Esto permitirá llevar un control estadístico sobre todos sus movimientos.

Cabe resaltar que es fundamental que la empresa debe hacer registros durante el día, todos los días, pues de esta manera la aplicación va a ser más eficiente y de fácil manejo tanto para los empleados como para la administradora, con esto se quiere que la aplicación tenga un constante

flujo de entrada y salida de datos lo que favorecerá a la empresa al momento de sacar sus cuentas diarias o como tradicionalmente se dice a cuadrar caja.

2. Objetivos

2.1. Objetivo General

Diseñar e implementar un software que permita administrar y gestionar con el fin de mejorar el acceso a la información, la gestión de las operaciones y la reducción en los tiempos de respuesta en las actividades de las pequeñas y medianas empresas del sector del calzado en la ciudad de Cúcuta.

2.2. Objetivos Específicos

• Diseño de software API RESTFULL con arquitectura Microservicios.

• Desarrollo de software que permita el registro detallado de clientes e inventarios, estadísticas;

usando tecnologías ASPX.NETCORE Y Angular 11.2.14

• Facilitar, la búsqueda, visualización y actualización del inventario del calzado, clientes y proveedores que se encuentran disponibles.

• Brindar seguridad y acceso a la información desde cualquier dispositivo con el fin de mantener el inventario de la empresa actualizado.

• Crear portal web como una vitrina digital que exponga servicios y productos, información corporativa que le den tono y cuerpo a las empresas.

3. Marco de Referencia

3.1. Industria del calzado

La industria del calzado se basa en una serie de sectores diferentes que ayudan a crear diseños de calzado y venderlos a los clientes. Las facetas de la industria incluyen el diseño, el marketing y la fabricación de calzado, así como las ventas minoristas[1].

Dentro de la industria del calzado se destaca la producción de diferentes tipos de calzado, entre los que se encuentran, calzado deportivo, botas, calzado casual para hombre y dama, zapatillas y calzadopara niño.

En los últimos años, ha habido una tendencia en el sector del calzado especialmente en países en víade desarrollo, con un aumento en la exportación hacia países desarrollados. Gracias a esto la demandaen la producción de calzado dentro de estos países en desarrollo también ha crecido. Las empresas locales con menor acceso a la información técnica han adoptado métodos de fabricación artesanales a expensas de tecnologías más adecuadas a la demanda local, especialmente a escalas de producciónbajas.

3.2. Importancia histórica, económica y cultural de la industria del calzado en Colombia y la región norte santandereana

A lo largo de la historia, los países del mundo han tenido que adaptarse a las nuevas demandas que elproceso de globalización e industrialización les ha traído para ser más competitivos y Colombia no es la excepción. La zapatería entra al País mediante la importación de zapatos que los artesanos locales aprenden a reparar sobre pies de hierro. Posteriormente, a principios de 1900, se empieza conhormas importadas la producción local que, por su calidad ha llegado a ser apreciada

en los mercadosinternacionales[2]. Gracias a esto, han surgido pequeñas y medianas empresas, que a través de los añosse han fortalecido y han generado muchos empleos.

Actualmente la industria del calzado está conformada en su mayoría por empresas familiares. Este sector es muy importante para la economía, ya que aporta no solamente crecimiento, sino bienestar social debido a que es uno de los principales generadores de empleo en el territorio nacional.

Por su parte, la industria del calzado en Santander y norte de Santander es de las más representativas en Colombia[3]. Además, muchas familias obtienen el sustento de cadadía con esta actividad económica, que no sólo da empleo directo a los fabricantes, sino también a quienes compran este producto a las fábricas y lo venden en los distintos puntos comerciales de la ciudad, permitiendo que esto genere un flujo constante en el comercio y permita la generación de nuevos empleos y a su vez un crecimiento económico en la región.

3.3. Distribución y Comercio

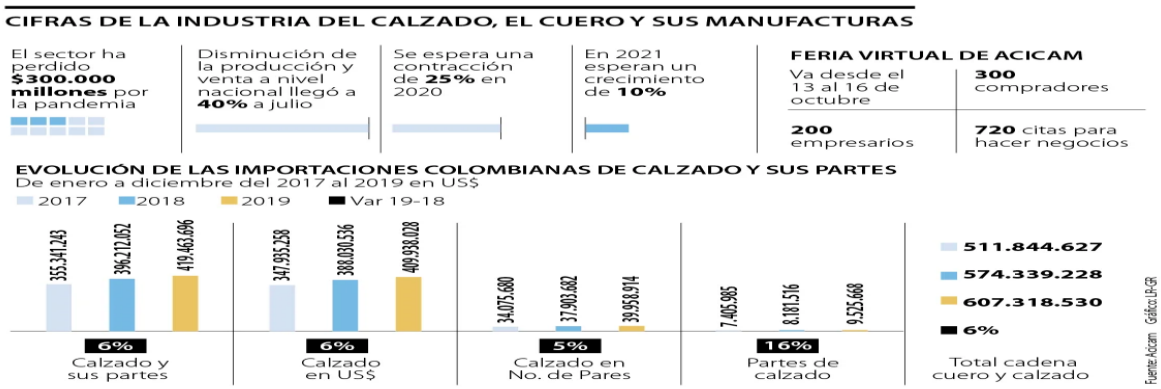
Hoy, en el departamento de Santander hay más de 6.000 fábricas de zapatos, entre familiares, micros, pequeñas, medianas y grandes empresas, que producen calzado sobre pedido para su comercialización, ya que para los fabricantes no es rentable hacer calzado en cantidades mínimas y es aquí donde entran los pequeños distribuidores y vendedores quienes son los encargados de ser el puente entre los fabricantes y el consumidor final.

Este flujo de compra y venta es más fuerte en ciertas épocas del año, en las cuales se genera una grandemanda de este producto, bien sea por necesidad o por consumismo impulsado por algunas fechas especiales y culturales. Gracias a esto, hay más inversión en el sector e incentiva la creación

de nuevas fábricas y la adquisición de mejores tecnologías, generando así la necesidad de más personal para suplir las demandas en producción y distribución del calzado.

Figura 1.

Cifras de la industria del calzado, el cuero y sus manufacturas



Nota: Tomado de ACICAM reportó que el sector del calzado y el cuero ha perdido cerca de $300.000 millones *por La República.*

El presidente de la junta también resaltó que han tenido una disminución en la producción y la venta a nivel nacional que oscila entre 38% y 40%, acumulado a julio, y explicó que esperan “una contracción de 25% este año en el sector, y que en 2021 podamos tener un crecimiento de por lo menos 10%”. La industria tuvo que transformar sus hábitos hacia las ventas y canales digitales, pero también se han visto cambios en el consumidor por la nueva realidad. por este motivo cada día el sector del calzado se ve en la obligación de adoptar formas digitales de abordar sus mercados para poder seguir siendo competitivos en un mundo tecnológico.

3.4. Técnicas y Tecnologías de Desarrollo

Al momento de buscar la mejor tecnología con la cual se empezará a desarrollar el aplicativo o software, primero se tuvieron en cuenta los recursos tecnológicos, económicos y necesidades que existen, se adoptaron tecnologías con un crecimiento notable y sostenido en el tiempo que se

caracterizan por el soporte, fácil escalabilidad y su baja adherencia, aprovechando las novedades que tenemos en pleno siglo XXI y se concluyó que la mejor tecnología para desarrollar el proyecto es una aplicación que soporte multiplataforma con portal web, ya que el dispositivo tecnológico más práctico y de uso frecuente para todas las personas son los teléfonos inteligentes; y dicho portal web no solo nos permite administrar dicha empresa, también permite tener una vitrina virtual en el mundo del internet lo cual coloca los negocios sin importar su tamaño en el radar de los consumidores de manera local, nacional e internacional. Partiendode lo anterior, en la actualidad existen diversas tecnologías que al hacer uso de ella nos permite crear aplicaciones robustas con orientación web y que dependiendo de la complejidad del proyecto y los costos computacionales y económicos se escogerá alguna de estas.

3.5. Apps Web

Existen las aplicaciones web, su principal ventaja es que, al no requerir ser compiladasen un sistema operativo específico, son multiplataforma y son fáciles de actualizar y lo único que se necesitan es de un navegador de internet para poder acceder a ellas. Una de las desventajas más significativas es que al no estar instalada en el dispositivo si no tenemos conexión de internet no pueden ser utilizadas y su velocidad y fluidez también dependen de la estabilidad de la red. Otra desventaja es que no puede acceder a todos los recursos del hardware del dispositivo

Por otro lado, actualmente está tomando fuerza el proyecto Progressive Web Apps o PWA, el cual permite usar casi todas las funcionalidades móviles incluyendo algunas sin la necesidad de tener unaconexión permanente a internet. El lenguaje comúnmente utilizado para programar estas aplicacionesweb es HTML5, CSS3 y JavaScript.

3.6. Apps Híbridas

Las aplicaciones híbridas son una combinación de las dos anteriores, por lo tanto, se traducen en aplicaciones web que están inmersas en una aplicación nativa y de esta manera se puede acceder a la gran mayoría del hardware del dispositivo. La ventaja principal es que es una aplicación construida para ser utilizada o implementada en cualquier sistema operativo (IOS, Android, Windows Phone, entre otros), evitando la tarea de crear nuevos códigos para cada uno.

Existen varias tecnologías que permiten utilizar los lenguajes HTML5, CSS3 y JavaScript para lograrcompilar nativamente una aplicación web en un teléfono móvil, una de éstas se llama Apache cordova, el cual es un Framework de desarrollo móvil de código abierto creado por Google. Estas aplicaciones se ejecutan dentro de envolturas para cada plataforma y dependen de enlaces estándaresAPI para acceder a sensores, datos y estado de la red.

3.7. Tecnologías

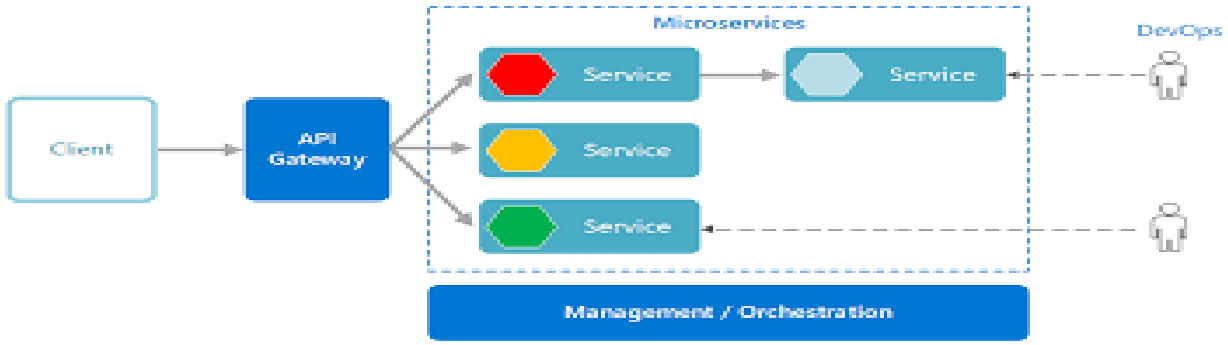
Para la creación de software nativo existen varias tecnologías que nos facilitan el desarrollo de aplicativos móviles, algunas de éstas trabajan bajo licencias de pago y otras trabajan con versiones de código abierto, entre las más populares se encuentran Flutter, Ionic, Unity, React Native, Angular entre otras. A continuación, se describirán las que se van a usar en la realización deeste proyecto de grado.

3.8. Estilo de arquitectura de microservicios

Una arquitectura de microservicios consta de una colección de servicios autónomos y pequeños. Los servicios son independientes entre sí y cada uno debe implementar una funcionalidad de negocio individual.

Figura 2.

Diagrama lógico del estilo de arquitectura de microservicios



Nota: Tomado de Estilo de arquitectura de microservicios.

3.8.1. ¿Qué son los microservicios?

Los microservicios son pequeños e independientes, y están acoplados de forma imprecisa. Un único equipo reducido de programadores puede escribir y mantener un servicio.

- Cada servicio es un código base independiente, que puede administrarse por un equipo de desarrollo pequeño.
- Los servicios pueden implementarse de manera independiente. Un equipo puede actualizar un servicio existente sin tener que volver a generar e implementar toda la aplicación.
- Los servicios son los responsables de conservar sus propios datos o estado externo. Esto difiere del modelo tradicional, donde una capa de datos independiente controla la persistencia de los datos.
- Los servicios se comunican entre sí mediante API bien definidas. Los detalles de la implementación interna de cada servicio se ocultan frente a otros servicios.

- No es necesario que los servicios compartan la misma pila de tecnología, las bibliotecas o los marcos de trabajo.

Además de los propios servicios, hay otros componentes que aparecen en una arquitectura típica de microservicios:

3.8.1.1. Administración e implementación. Este componente es el responsable de la colocación de servicios en los nodos, la identificación de errores, el reequilibrio de servicios entre nodos, etc. Normalmente, este componente es una tecnología estándar, como Kubernetes, en lugar de algo creado de forma personalizada.

3.8.1.2. Puerta de enlace de API. La puerta de enlace de API es el punto de entrada para los clientes. En lugar de llamar a los servicios directamente, los clientes llaman a la puerta de enlace de API, que reenvía la llamada a los servicios apropiados en el back-end.

Entre las ventajas de usar una puerta de enlace de API se encuentran las siguientes:

- Desacoplan los clientes de los servicios. Los servicios pueden cambiar de versión o refactorizarse sin necesidad de actualizar todos los clientes.
- Los servicios pueden utilizar los protocolos de mensajería que no son fáciles de usar para un servicio web, como AMQP.
- La puerta de enlace de API puede realizar otras funciones transversales como la autenticación, el registro, la terminación SSL y el equilibrio de carga.

Ventajas

Agilidad. Dado que microservicios se implementan de forma independiente, resulta más fácil de administrar las correcciones de errores y las versiones de características. Puede actualizar un servicio sin volver a implementar toda la aplicación y revertir una actualización si algo va mal. En muchas aplicaciones tradicionales, un error en una parte de la aplicación puede bloquear todo el proceso de lanzamiento. Es posible que se requieran nuevas características a la espera de que se integre, pruebe y publique una corrección de errores.

Equipos pequeños y centrados. Un microservicio debe ser lo suficientemente pequeño como para que un solo equipo de características lo pueda compilar, probar e implementar. Los equipos pequeños favorecen la agilidad. Los equipos grandes suelen ser menos productivos, porque la comunicación es más lenta, aumenta la sobrecarga de administración y la agilidad disminuye.

Base de código pequeña. En las aplicaciones monolíticas, con el paso del tiempo se da la tendencia de que las dependencias del código se acaben enredarse, por lo que para agregar una nueva característica, es preciso tocar el código en muchos lugares. Al no compartir el código ni los almacenes de datos, la arquitectura de microservicios minimiza las dependencias y resulta más fácil agregar nuevas características.

Mezcla de tecnologías. Los equipos pueden elegir la tecnología que mejor se adapte al servicio de una combinación de pilas de tecnología, según corresponda.

Aislamiento de errores. Si un microservicio individual no está disponible, no interrumpe toda la aplicación, siempre que los microservicios de nivel superior estén diseñados para controlar los errores correctamente (por ejemplo, mediante la implementación de disyuntores).

Escalabilidad. Los servicios se pueden escalar de forma independiente, lo que permite escalar horizontalmente los subsistemas que requieren más recursos, sin tener que escalar horizontalmente toda la aplicación. Mediante un orquestador como Kubernetes o Service Fabric se puede empaquetar una mayor densidad de servicios en un solo host, lo que aumenta la eficacia en el uso de los recursos.

Aislamiento de los datos. Al verse afectado solo un microservicio, es mucho más fácil realizar actualizaciones del esquema. En una aplicación monolítica, las actualizaciones del esquema pueden ser muy complicadas, ya que las distintas partes de la aplicación pueden tocar los mismos datos, por lo que realizar modificaciones en el esquema resulta peligroso.

Desafíos

Las ventajas de los microservicios tienen un "precio". Estos son algunos de los aspectos que deben tenerse en cuenta antes de embarcarse en una arquitectura de microservicios:

Complejidad. Una aplicación de microservicios tiene más partes en movimiento que la aplicación monolítica equivalente. Cada servicio es más sencillo, pero el sistema como un todo es más complejo.

Desarrollo y pruebas. La escritura de un servicio pequeño que utilice otros servicios dependientes requiere un enfoque que no sea escribir una aplicación tradicional monolítica o en capas. Las herramientas existentes no siempre están diseñadas para trabajar con dependencias de servicios. La refactorización en los límites del servicio puede resultar difícil. También supone un desafío probar las dependencias de los servicios, especialmente cuando la aplicación está evolucionando rápidamente.

Falta de gobernanza. El enfoque descentralizado para la generación de microservicios tiene ventajas, pero también puede causar problemas. Puede acabar con tantos lenguajes y marcos de trabajo diferentes que la aplicación puede ser difícil de mantener. Puede resultar útil establecer algunos estándares para todo el proyecto sin restringir excesivamente la flexibilidad de los equipos. Esto se aplica especialmente a las funcionalidades transversales, como el registro.

Congestión y latencia de red. El uso de muchos servicios pequeños y detallados puede dar lugar a más comunicación interservicios. Además, si la cadena de dependencias del servicio se hace demasiado larga (el servicio A llama a B, que llama a C..), la latencia adicional puede constituir un problema. Tendrá que diseñar las API con atención. Evite que las API se comuniquen demasiado, piense en formatos de serialización y busque lugares para utilizar patrones de comunicación asincrónica.

Integridad de datos. Cada microservicio es responsable de la conservación de sus propios datos. Como consecuencia, la coherencia de los datos puede suponer un problema. Adopte una coherencia final cuando sea posible.

Administración. Para tener éxito con los microservicios se necesita una cultura de DevOps consolidada. El registro correlacionado entre servicios puede resultar un desafío. Normalmente, el registro debe correlacionar varias llamadas de servicio para una sola operación de usuario.

Control de versiones. Las actualizaciones de un servicio no deben interrumpir servicios que dependen de él. Es posible que varios servicios se actualicen en cualquier momento; por lo tanto, sin un cuidadoso diseño, podrían surgir problemas con la compatibilidad con versiones anteriores o posteriores.

Conjunto de habilidades. Los microservicios son sistemas muy distribuidos. Evalúe cuidadosamente si el equipo tiene los conocimientos y la experiencia para desenvolverse correctamente.

3.9. Procedimientos recomendados

- Adapte los servicios al dominio empresarial.
- Descentralice todo. Los equipos individuales son responsables de diseñar y compilar servicios. Evite el uso compartido de código o esquemas de datos.
- El almacenamiento de datos debería ser privado para el servicio que posee los datos. Use el almacenamiento recomendado para cada tipo de servicio y de datos.
- Los servicios se comunican a través de API bien diseñadas. Evite la pérdida de detalles de la implementación. Las API deben modelar el dominio, no la implementación interna del servicio.
- Evite el acoplamiento entre servicios. Entre las causas de acoplamiento se encuentran los protocolos de comunicación rígidos y los esquemas de bases de datos compartidos.
- Descargue a la puerta de enlace de cuestiones transversales, como la autenticación y la terminación SSL.
- Conserve el conocimiento del dominio fuera de la puerta de enlace. La puerta de enlace debe controlar y enrutar las solicitudes de cliente sin ningún conocimiento de las reglas de negocios o la lógica de dominio. En caso contrario, la puerta de enlace se convierte en una dependencia y puede provocar el acoplamiento entre servicios.

- Los servicios deben tener un acoplamiento flexible y una alta cohesión funcional. Las funciones que es probable que cambien juntas se deben empaquetar e implementar en conjunto. Si residen en distintos servicios, estos terminan estrechamente acoplados, porque un cambio en un servicio requerirá la actualización del otro servicio. La comunicación demasiado intensa entre dos servicios puede ser un síntoma de un acoplamiento estrecho y una cohesión baja.
- Aísle los errores. Utilice estrategias de resistencia para impedir que los errores dentro de un servicio se reproduzcan en cascada. Consulte Patrones de resistencia y Diseño de aplicaciones confiables.

3.10. Diseñar un microservicio orientado a DDD

El diseño impulsado por dominios (DDD) aboga por el modelado basado en la realidad empresarial como relevante para sus casos de uso. En el contexto de la creación de aplicaciones, DDD habla de problemas como dominios. Describe las áreas de problemas independientes como contextos limitados (cada contexto limitado se correlaciona con un microservicio) y enfatiza un lenguaje común para hablar sobre estos problemas. También sugiere muchos conceptos y patrones técnicos, como entidades de dominio con modelos ricos (sin modelo de dominio anémico), objetos de valor, agregados y reglas de raíz agregada (o entidad raíz) para respaldar la implementación interna. Esta sección presenta el diseño y la implementación de esos patrones internos.

A veces, estas reglas y patrones técnicos de DDD se perciben como obstáculos que tienen una curva de aprendizaje pronunciada para implementar enfoques de DDD. Pero lo importante no son los patrones en sí mismos, sino la organización del código para que esté alineado con los problemas comerciales y utilizando los mismos términos comerciales (lenguaje ubicuo).

Además, los enfoques de DDD deben aplicarse solo si está implementando microservicios complejos con reglas comerciales importantes. Las responsabilidades más simples, como un servicio CRUD, se pueden administrar con enfoques más simples.

Dónde trazar los límites es la tarea clave al diseñar y definir un microservicio. Los patrones DDD le ayudan a comprender la complejidad del dominio. Para el modelo de dominio para cada contexto limitado, usted identifica y define las entidades, los objetos de valor y los agregados que modelan su dominio. Usted construye y refina un modelo de dominio que está contenido dentro de un límite que define su contexto. Y eso es explícito en forma de microservicio. Los componentes dentro de esos límites terminan siendo sus microservicios, aunque en algunos casos un BC o microservicios comerciales pueden estar compuestos por varios servicios físicos. DDD se trata de límites y también de microservicios.

Mantenga los límites del contexto del microservicio relativamente pequeños.

Determinar dónde colocar los límites entre contextos delimitados equilibra dos objetivos en competencia. Primero, desea crear inicialmente los microservicios más pequeños posibles, aunque ese no debería ser el controlador principal; debe crear un límite alrededor de las cosas que necesitan cohesión. En segundo lugar, desea evitar las comunicaciones conversadoras entre microservicios. Estos objetivos pueden contradecirse entre sí. Debe equilibrarlos descomponiendo el sistema en tantos microservicios pequeños como pueda hasta que vea que los límites de comunicación crecen rápidamente con cada intento adicional de separar un nuevo contexto limitado. La cohesión es clave dentro de un único contexto acotado.

Es similar al olor del código de intimidad inapropiada al implementar clases. Si dos microservicios necesitan colaborar mucho entre sí, probablemente deberían ser el mismo microservicio.

Otra forma de ver este aspecto es la autonomía. Si un microservicio debe depender de otro servicio para atender directamente una solicitud, no es verdaderamente autónomo.

3.11. Capas en microservicios DDD

La mayoría de las aplicaciones empresariales con una complejidad técnica y empresarial significativa están definidas por múltiples capas. Las capas son un artefacto lógico y no están relacionadas con la implementación del servicio. Existen para ayudar a los desarrolladores a gestionar la complejidad del código. Las diferentes capas (como la capa del modelo de dominio frente a la capa de presentación, etc.) pueden tener diferentes tipos, que exigen traducciones entre esos tipos.

Por ejemplo, una entidad podría cargarse desde la base de datos. Luego, parte de esa información, o una agregación de información que incluye datos adicionales de otras entidades, se puede enviar a la interfaz de usuario del cliente a través de una API web REST. El punto aquí es que la entidad de dominio está contenida dentro de la capa del modelo de dominio y no debe propagarse a otras áreas a las que no pertenece, como la capa de presentación.

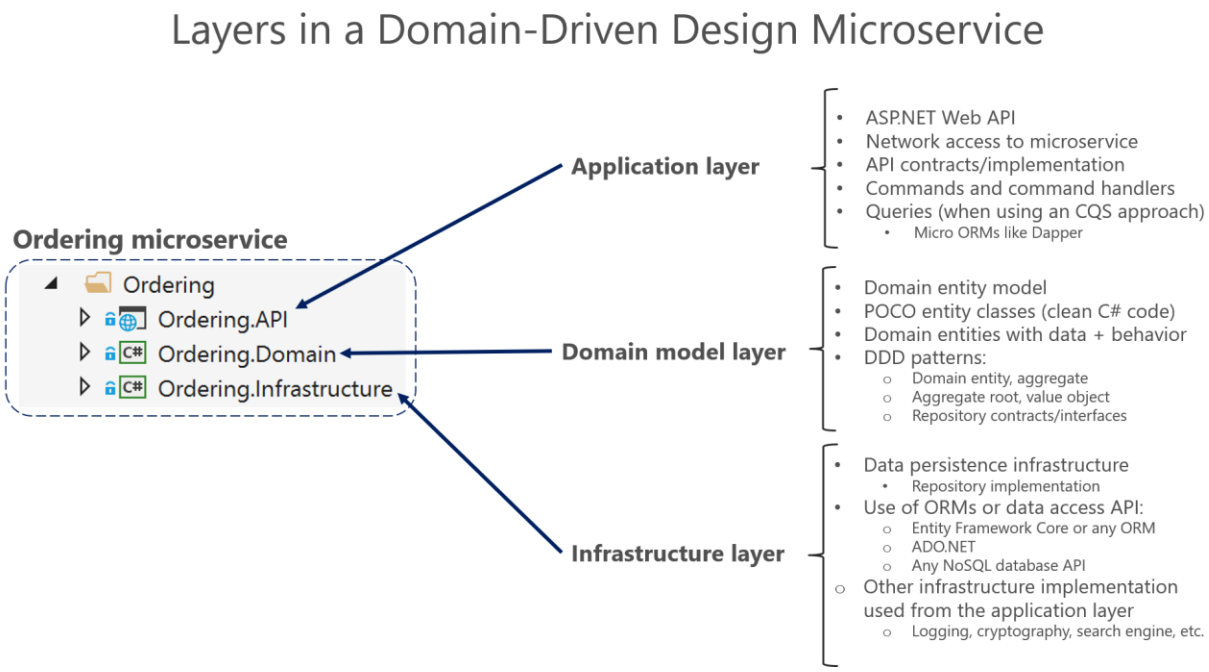
Además, debe tener entidades siempre válidas (consulte Diseño de validaciones en la sección de capa de modelo de dominio) controladas por raíces agregadas (entidades raíz). Por lo tanto, las entidades no deben estar vinculadas a las vistas del cliente, porque en el nivel de la interfaz de usuario es posible que algunos datos aún no estén validados. Esta es la razón para la que sirve ViewModel. ViewModel es un modelo de datos exclusivamente para las necesidades de la capa

de presentación. Las entidades de dominio no pertenecen directamente a ViewModel. En su lugar, necesita traducir entre ViewModels y entidades de dominio y viceversa.

Al abordar la complejidad, es importante tener un modelo de dominio controlado por raíces agregadas que asegure que todas las invariantes y reglas relacionadas con ese grupo de entidades (agregadas) se realicen a través de un único punto de entrada o puerta, la raíz agregada.

Figura 3.

Capas DDD en el microservicio de pedidos en eShopOnContainers



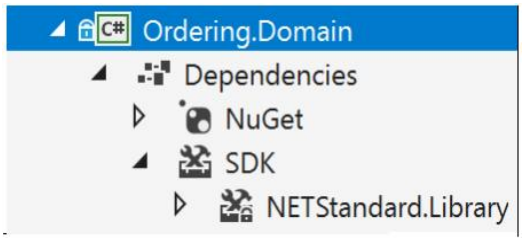
Nota: Tomado de Implement a microservice domain model with .NET.

Las tres capas de un microservicio DDD como Ordering. Cada capa es un proyecto VS: la capa de aplicación es Ordering.API, la capa de dominio es Ordering.Domain y la capa de infraestructura es Ordering.Infrastructure. Desea diseñar el sistema para que cada capa se comunique solo con algunas otras capas. Ese enfoque puede ser más fácil de aplicar si las capas se implementan como bibliotecas de clases diferentes, porque puede identificar claramente qué dependencias se

establecen entre las bibliotecas. Por ejemplo, la capa del modelo de dominio no debe depender de ninguna otra capa (las clases del modelo de dominio deben ser clases de Objetos CLR antiguos simples o POCO). Como se muestra en la Figura 7-6, el dominio Ordering. La biblioteca de capas tiene dependencias solo en las bibliotecas .NET o los paquetes NuGet, pero no en ninguna otra biblioteca personalizada, como la biblioteca de datos o la biblioteca de persistencia.

Figura 4.

Capas implementadas como bibliotecas



Nota: Tomado de Implement a microservice domain model with .NET.

3.12. La capa del modelo de dominio

El excelente libro de Eric Evans Domain Driven Design dice lo siguiente sobre la capa de modelo de dominio y la capa de aplicación.

Capa de modelo de dominio: responsable de representar conceptos del negocio, información sobre la situación del negocio y reglas de negocio. Aquí se controla y utiliza el estado que refleja la situación del negocio, aunque los detalles técnicos de su almacenamiento se delegan a la infraestructura. Esta capa es el corazón del software empresarial.

La capa del modelo de dominio es donde se expresa el negocio. Cuando implementa una capa de modelo de dominio de microservicio en .NET, esa capa se codifica como una biblioteca de

clases con las entidades de dominio que capturan los datos más el comportamiento (métodos con lógica).

Siguiendo los principios de Ignorancia de persistencia e Ignorancia de infraestructura, esta capa debe ignorar por completo los detalles de persistencia de datos. Estas tareas de persistencia deben ser realizadas por la capa de infraestructura. Por lo tanto, esta capa no debe tener dependencias directas de la infraestructura, lo que significa que una regla importante es que las clases de entidad de su modelo de dominio deben ser POCO.

Las entidades de dominio no deben tener ninguna dependencia directa (como derivar de una clase base) en ningún marco de infraestructura de acceso a datos como Entity Framework o NHibernate. Idealmente, las entidades de su dominio no deben derivar ni implementar ningún tipo definido en ningún marco de infraestructura.

La mayoría de los marcos ORM modernos como Entity Framework Core permiten este enfoque, de modo que las clases de modelo de dominio no estén acopladas a la infraestructura. Sin embargo, tener entidades POCO no siempre es posible cuando se usan ciertas bases de datos y marcos NoSQL, como Actors y Reliable Collections en Azure Service Fabric.

Incluso cuando es importante seguir el principio de Ignorancia de persistencia para su modelo de dominio, no debe ignorar las preocupaciones de persistencia. Sigue siendo importante comprender el modelo de datos físicos y cómo se asigna a su modelo de objeto de entidad. De lo contrario, puede crear diseños imposibles.

Además, este aspecto no significa que pueda tomar un modelo diseñado para una base de datos relacional y moverlo directamente a una base de datos NoSQL u orientada a documentos. En algunos modelos de entidad, el modelo puede ajustarse, pero normalmente no es así. Aún existen

restricciones que su modelo de entidad debe cumplir, basadas tanto en la tecnología de almacenamiento como en la tecnología ORM.

3.13. La capa de aplicación

Pasando a la capa de aplicación, podemos citar nuevamente el libro de Eric Evans Domain Driven Design:

3.13.1. Capa de aplicación

Define los trabajos que se supone que debe realizar el software y dirige los objetos de dominio expresivo para resolver problemas. Las tareas de las que es responsable esta capa son significativas para la empresa o necesarias para la interacción con las capas de aplicación de otros sistemas. Esta capa se mantiene delgada. No contiene reglas de negocio o conocimientos, solo coordina tareas y delega el trabajo a colaboraciones de objetos de dominio en la siguiente capa. No tiene un estado que refleje la situación empresarial, pero puede tener un estado que refleje el progreso de una tarea para el usuario o el programa.

La capa de aplicación de un microservicio en .NET se codifica comúnmente como un proyecto de API web ASP.NET Core. El proyecto implementa la interacción del microservicio, el acceso remoto a la red y las API web externas que se utilizan desde la interfaz de usuario o las aplicaciones cliente. Incluye consultas si se usa un enfoque CQRS, comandos aceptados por el microservicio e incluso la comunicación impulsada por eventos entre microservicios (eventos de integración). La API web ASP.NET Core que representa la capa de la aplicación no debe contener reglas comerciales o conocimientos de dominio (especialmente reglas de dominio para transacciones o actualizaciones); estos deben ser propiedad de la biblioteca de clases del modelo de dominio. La

capa de aplicación solo debe coordinar tareas y no debe contener ni definir ningún estado de dominio (modelo de dominio).

Básicamente, la lógica de la aplicación es donde implementa todos los casos de uso que dependen de una interfaz determinada. Por ejemplo, la implementación relacionada con un servicio de API web.

El objetivo es que la lógica del dominio en la capa del modelo de dominio, sus invariantes, el modelo de datos y las reglas comerciales relacionadas deben ser completamente independientes de las capas de presentación y aplicación. Sobre todo, la capa del modelo de dominio no debe depender directamente de ningún marco de infraestructura.

3.13.2. La capa de infraestructura

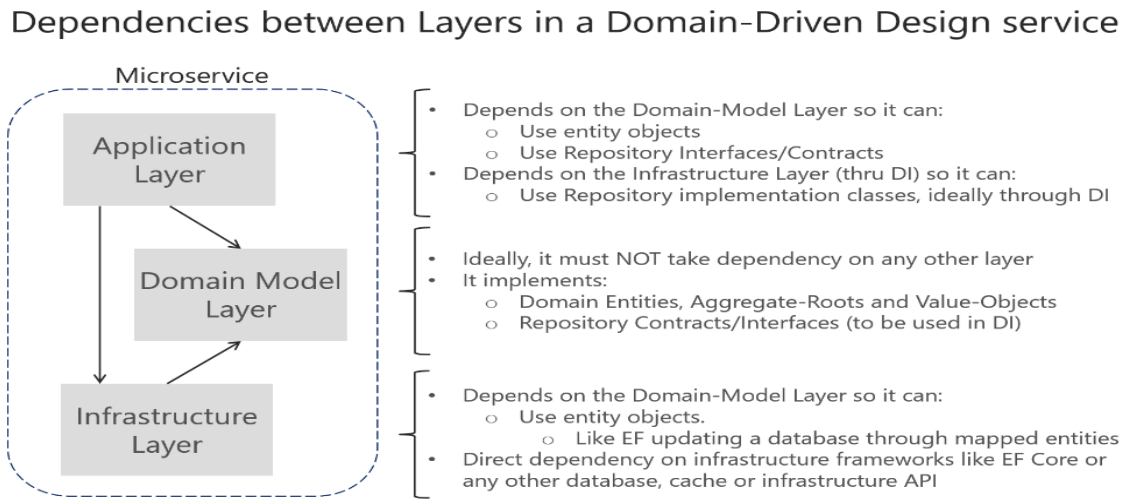
La capa de infraestructura es cómo los datos que se mantienen inicialmente en las entidades de dominio (en la memoria) se conservan en las bases de datos u otro almacén persistente. Un ejemplo es usar el código de Entity Framework Core para implementar las clases de patrón de repositorio que usan un DBContext para conservar los datos en una base de datos relacional.

De acuerdo con los principios de Ignorancia de persistencia e Ignorancia de infraestructura mencionados anteriormente, la capa de infraestructura no debe "contaminar" la capa de modelo de dominio. Debe mantener las clases de entidad del modelo de dominio independientes de la infraestructura que usa para conservar los datos (EF o cualquier otro marco) al no tomar dependencias estrictas en los marcos. Su biblioteca de clases de capa de modelo de dominio debe tener solo su código de dominio, solo clases de entidad POCO que implementen el corazón de su software y estén completamente desacopladas de las tecnologías de infraestructura.

Por lo tanto, sus capas o bibliotecas de clases y proyectos deberían depender en última instancia de su capa de modelo de dominio (biblioteca), no al revés, como se muestra en la Figura 5.

Figura 5.

Dependencias existentes entre niveles en DDD



Nota: Tomado de Implement a microservice domain model with .NET.

Dependencias en un servicio DDD, la capa de aplicación depende del dominio y la infraestructura, y la infraestructura depende del dominio, pero el dominio no depende de ninguna capa. Este diseño de capa debe ser independiente para cada microservicio. Como se mencionó anteriormente, puede implementar los microservicios más complejos siguiendo patrones DDD, mientras implementa microservicios basados en datos más simples (CRUD simple en una sola capa) de una manera más simple.

Figura 6.

Diseño de un microservicio orientado a DDD

Nota: Tomado de Implement a microservice domain model with .NET.

3.14. Métodos de petición

HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Aunque estos también pueden ser sustantivos, estos métodos de solicitud a veces son llamados HTTP verbs. Cada uno de ellos implementan una semántica diferente, pero algunas características similares son compartidas por un grupo de ellos: ej. un request method puede ser safe, idempotent, o cacheable.

GET. El método GET solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

HEAD. El método HEAD pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.

POST. El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

PUT. El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

DELETE. El método DELETE borra un recurso en específico.

CONNECT. El método CONNECT establece un túnel hacia el servidor identificado por el recurso.

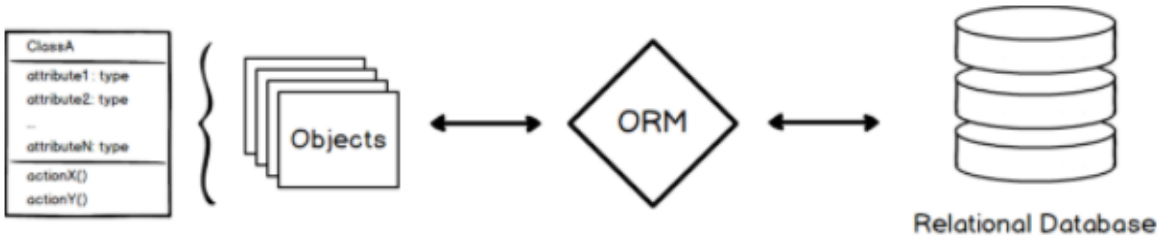
OPTIONS. El método OPTIONS es utilizado para describir las opciones de comunicación para el recurso de destino.

TRACE. El método TRACE realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.

PATCH. El método PATCH es utilizado para aplicar modificaciones parciales a un recurso.

Figura 7.

ORM.



Nota: Tomado de ¿Qué es un ORM? *Por Deloitte.*

Un ORM es un modelo de programación que permite mapear las estructuras de una base de datos relacional (SQL Server, Oracle, MySQL, etc.), en adelante RDBMS (Relational Database Management System), sobre una estructura lógica de entidades con el objeto de simplificar y acelerar el desarrollo de nuestras aplicaciones.

Las estructuras de la base de datos relacional quedan vinculadas con las entidades lógicas o base de datos virtual definida en el ORM, de tal modo que las acciones CRUD (Create, Read, Update, Delete) a ejecutar sobre la base de datos física se realizan de forma indirecta por medio del ORM.

La consecuencia más directa que se infiere del párrafo anterior es que, además de "mapear", los ORMs tienden a "liberarnos" de la escritura o generación manual de código SQL (Structured Query Language) necesario para realizar las queries o consultas y gestionar la persistencia de datos en el RDBMS.

Así, los objetos o entidades de la base de datos virtual creada en nuestro ORM podrán ser manipulados por medio de algún lenguaje de nuestro interés según el tipo de ORM utilizado, por

ejemplo, LINQ sobre Entity Framework de Microsoft. La interacción con el RDBMS quedará delegada en los métodos de actualización correspondientes proporcionados por el ORM. Los ORMs más completos ofrecen servicios para persistir todos los cambios en los estados de las entidades, previo seguimiento o tracking automático, sin escribir una sola línea de SQL.

Llegados a este punto, parece claro que el hecho de "atacar" directamente a las entidades de la base de datos virtual sin necesidad de generar código SQL nos reportará importantes ventajas a la hora de acelerar el desarrollo o implementación de nuestras aplicaciones, pero me gustaría anticipar dos cuestiones básicas a desarrollar en sendos artículos posteriores:

- ¿En qué condiciones encaja mejor el uso de un ORM?
- Si decidimos usar un ORM sobre un RDBMS, ¿nos "olvidamos" completamente entonces del SQL?

3.15. Motivos para utilizar un ORM

Llegados a este punto, podemos hacernos esta pregunta: ¿por qué usar un ORM?

Te exponemos una serie de ventajas por las que utilizar un ORM, ya que esta herramienta permite adaptarse a los nuevos tiempos y darnos una mayor versatilidad a la hora de manejar los datos.

- Lo primero que el ORM funciona como una capa intermedia totalmente separada de la base de datos. Esto permite que puedas centrarte únicamente en el desarrollo de la aplicación.
- Hace que una migración sea más sencilla. Si en algún momento se quiere cambiar la aplicación, no habría que reconstruir todas las consultas: podríamos migrar esa aplicación a otra base de datos sin problema.

- El programador no necesita saber un lenguaje para cada base de datos: el ORM unifica el lenguaje.
- Nos permite tener una mayor velocidad a la hora de hacer tareas básicas en cuanto al acceso a los datos.
- Es un código más legible y con menos líneas.
- Otra ventaja importante es la seguridad. Al ser una capa independiente a los datos, nos permite protegerlos de ciberataques al no estar en el mismo nivel.
- Nos evita escribir a mano las consultas de SQL necesarias.
- Facilita el trabajo con acciones simples y básicas de acceso a los datos. Como funciones básicas de bases de datos consideramos lo que se conoce como CRUD: Create, Read, Update y Delete.
- Por último y principalmente, una de las ventajas más atractiva para los desarrolladores a la hora de usarlo es que un ORM nos guarda y carga toda la información de una base de datos relacional automáticamente.

3.16. ORMs en el mercado actual

- Microsoft Entity Framework 6.0: Microsoft Entity Framework se publicó por primera vez en 2008, como parte de .NET Framework 3.5 SP1 y Visual Studio 2008 SP1. A partir de la versión 4.1, se ha distribuido como paquete NuGet independiente convirtiéndose en uno de los más populares en NuGet.org. Sólo para plataforma Windows. Muy estable. Open Source. Proporciona servicios avanzados de modelado y seguimiento de estados de entidades, persistencia automática de cambios, caching, gestión de transacciones, etc.

- Microsoft Entity Framework Core 2.0: Nueva implementación de Entity Framework en versión Core. Más ligera, extensible y multiplataforma (Windows, Linux, Mac). Ofrece un rendimiento más potente en comparación con la versión desarrollada específicamente para .Net Framework. En continua evolución. Open Source. Puede ejecutarse sobre .Net Framework 4.6.1, .Net Core 2.0 o posteriores.
- Microsoft Entity Framework Core 2.1: Versión más reciente de Entity Framework Core lanzada a finales de mayo de 2018. Incluye mejoras funcionales y de rendimiento, así como correcciones sobre la versión anterior. No obstante, no toda la funcionalidad de Entity Framework 6.0 ha sido migrada todavía a esta última versión de Entity Framework Core.
- NHibernate: Conversión de Hibernate en Java para lenguaje C# compatible con plataforma .Net. Estable. Open Source.
- Dapper: Micro-ORM que proporciona métodos de extensión a las clases de .Net Framework para mapear resultados y persistir datos, previa inyección de código SQL personalizado. No nos libera pues de la implementación de nuestro código SQL, actuando como simple mapeador, pero a cambio ofrece un buen rendimiento. Es un ORM creado y usado en producción por el conocido portal web Stack Overflow.

3.17. SPA Single Page Application

Es un tipo de aplicación web donde todas las pantallas las muestra en la misma página, sin recargar el navegador.

Técnicamente, una SPA es un sitio donde existe un único punto de entrada, generalmente el archivo index.html. En la aplicación no hay ningún otro archivo HTML al que se pueda acceder

de manera separada y que nos muestre un contenido o parte de la aplicación, toda la acción se produce dentro del mismo index.html.

Varias vistas, no varias páginas

Aunque solo tengamos una página, lo que sí tenemos en la aplicación son varias vistas, entendiendo por vista algo como lo que sería una pantalla en una aplicación de escritorio. En la misma página, por tanto, se irán intercambiando vistas distintas, produciendo el efecto de que tienes varias páginas, cuando realmente todo es la misma, intercambiando vistas.

SPA ofrece una experiencia de usuario tan agradable

Al pesar muy poco los datos, mucho menos que si estuvieran mezclados dentro de un complejo código HTML y CSS para definir su presentación, las transmisiones son muy rápidas y las comunicaciones entre cliente y servidor se realizan muy fluidas. Nuevamente ayuda a que las páginas respondan muy velozmente al visitante, creando una experiencia de usuario muy agradable.

Importante:

El lenguaje con el que habitualmente comunicas los datos crudos desde el servidor hacia el cliente es JSON. De todos modos, nada impide usar otro lenguaje como XML, también muy popular en los Web Services. Aunque JSON se ha establecido como un estándar, JSON es una notación de objeto Javascript, por lo que es algo muy cercano a la web. Es ligero y tiene soporte en la totalidad de los lenguajes usados en la web.

Las páginas de gestión, o administración de cualquier tipo de servicio, paneles de control y cosas así son muy adecuadas para las SPA. El resultado es una aplicación web se comporta muy parecido a una aplicación de escritorio.

3.18. Lenguajes y tecnologías para producir una SPA

Una SPA se realiza en Javascript. No existe ningún otro tipo de lenguaje en el que puedas realizar una SPA, ya que básicamente es una aplicación web que se ejecuta del lado del cliente.

también se realiza usando HTML + CSS, para la presentación, ya que son los lenguajes que entiende el navegador.

Ya luego, dentro de Javascript, existen diversas librerías y frameworks que facilitan mucho el desarrollo de una SPA, entre los que podemos mencionar:

- AngularJS
- Angular 2
- React
- Polymer
- EmberJS

3.19. Comportamientos tradicionales y de SPA admitidos

Las aplicaciones web tradicionales apenas contaban con comportamiento del lado cliente, y en su lugar se basaban en el servidor para todas las operaciones de navegación, consultas y actualizaciones que la aplicación tuviera que realizar. Cada operación nueva realizada por el usuario se convertiría en una nueva solicitud web, con el resultado de una recarga de página completa en el explorador del usuario final. Los marcos de controlador de vista de modelos (MVC)

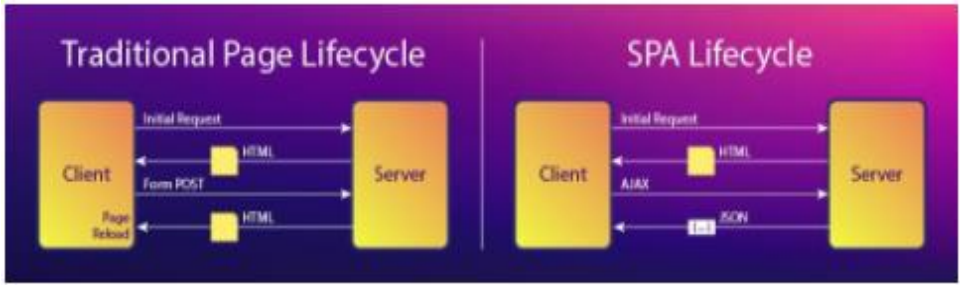
clásicos normalmente siguen este enfoque, en el que cada solicitud nueva se corresponde a otra acción de controlador, lo que a su vez podría funcionar con un modelo y devolver una vista. Es posible que algunas operaciones individuales en una página determinada se mejoraran con funcionalidad de AJAX (JavaScript asincrónico y XML), pero la arquitectura global de la aplicación usaba muchas vistas MVC distintas y extremos de URL. Además, ASP.NET Core MVC también admite Razor Pages, una forma más sencilla de organizar las páginas de tipo MVC.

Las aplicaciones de página única (SPA), por el contrario, implican muy pocas cargas de página generadas de forma dinámica en el lado de servidor (si existen). Muchas SPA se inicializan en un archivo HTML estático que carga las bibliotecas de JavaScript necesarias para iniciar y ejecutar la aplicación. Estas aplicaciones hacen un uso intensivo de las API web para sus necesidades de datos y pueden proporcionar experiencias de usuario mucho más enriquecidas.

Muchas aplicaciones web implican una combinación del comportamiento de aplicación web tradicional (normalmente para el contenido) y SPA (para la interactividad). ASP.NET Core admite MVC (basado en vistas o páginas) y las API web en la misma aplicación y usa el mismo conjunto de herramientas y bibliotecas de marco subyacentes.

Figura 8.

Diferencia entre SPA y MPA.



Nota: Tomado de Single Page Application vs Multi Page Application.

3.20. Angular

Es un framework opensource desarrollado por Google para facilitar la creación y programación de aplicaciones web de una sola página, las webs SPA (Single Page Application).

Angular separa completamente el frontend y el backend en la aplicación, evita escribir código repetitivo y mantiene todo más ordenado gracias a su patrón MVC (Modelo-Vista-Controlador) asegurando los desarrollos con rapidez, a la vez que posibilita modificaciones y actualizaciones.

En una web SPA aunque la velocidad de carga puede resultar un poco lenta la primera vez que se abre, navegar después es totalmente instantáneo, ya que se ha cargado toda la página de golpe.

Solamente es una ruta la que se tiene que enviar el servidor, y Angular lo que hace 'por debajo' es cambiar la vista al navegar para que dé la apariencia de una web normal, pero de forma más dinámica.

Entre otras ventajas, este framework es modular y escalable adaptándose a nuestras necesidades y al estar basado en el estándar de componentes web, y con un conjunto de interfaz de programación de aplicaciones (API) permite crear nuevas etiquetas HTML personalizadas que pueden reutilizarse.

El lenguaje principal de programación de Angular es Typescript, y así toda la sintaxis y el modo de hacer las cosas en el código es el mismo, lo que añade coherencia y consistencia a la información, permitiendo, por ejemplo, la incorporación de nuevos programadores, en caso de ser necesarios, ya que pueden continuar su trabajo sin excesiva dificultad.

Como ya se ha indicado, las plantillas de Angular almacenan por separado el código de la interfaz del usuario (front-end) y el de la lógica de negocio (back-end), que entre otros beneficios permite utilizar mejor otras herramientas anteriormente existentes.

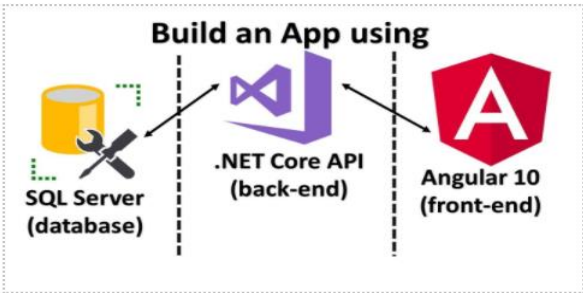
Y, por si fuera poco, los principales editores y entornos de desarrollo integrado (IDEs) ofrecen ya extensiones para poder trabajar con este framework con mayor comodidad.

Por su programación reactiva, la vista se actualiza automáticamente tras realizar los cambios. Además, Angular dispone de asistente por línea de comandos para poder crear proyectos base y también se integra bien con herramientas de testing y con Ionic, lo que facilita la creación de web-responsive, es decir, adaptadas a móviles.

Este aspecto cada día adquiere mayor importancia tanto por el creciente uso de estos dispositivos para acceder a internet como por la penalización que Google realiza de aquellas páginas que no facilitan su visita en cualquier dispositivo.

Figura 9.

Compra y uso de la app.



Nota: Tomado de *Angular.*

4. Metodología

Actualmente existen varias metodologías para el desarrollo de software que nos ayudan a estructurar, planear y controlar este proceso y de esta manera desarrollar efectiva y eficientemente el software, así como para documentar y poder rendir cuentas de los resultados obtenidos. Teniendo en cuenta los objetivos y las especificaciones del proyecto se adoptaron las siguientes metodologías para un desarrollo más productivo y eficaz.

4.1. Metodología de Prototipos

Un prototipo es una versión preliminar o implementación parcial en este caso de la aplicación con fines de demostración o evaluación. Es construido de una manera rápida tal como sea posible y comparativamente económico. Su gran ventaja es que permite que los usuarios que van a utilizar la aplicación puedan dar una vista preliminar y experimentar parte del software y de esta manera hacer retroalimentación sobre lo que a ellos les gustó y no les gustó acerca del prototipo proporcionado y hacer los cambios convenientes hasta que el usuario quede satisfecho. A continuación, se presentan las etapas para el desarrollo del software:

4.1.1. Planeación

En esta etapa se recolecta requisitos, el cliente define los objetivos globales del software y más específicos que se desean destacar en el prototipo, de esta manera planificar las actividades a realizar en cada iteración. De esta etapa surge una memoria llamada SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos.

4.1.2. Modelado

Nuevamente, en esta fase se realizará un diseño centrado en los aspectos del software visible al usuario, como la arquitectura del sistema y la definición de la estructura de las interfaces, que posteriormente deberá desarrollarse de forma detallada, al producto quese pretende obtener.

4.1.3. Elaboración del Prototipo

Ya que contamos con la planeación de lo que vamos a realizar yel modelado rápido, entonces es momento de elaborar el prototipo. Esta etapa es importanteya que se construirá el prototipo para probar la eficiencia de los algoritmos que se van a implementar y para comprobar el rendimiento de un determinado componente del sistema.

4.1.4. Desarrollo

Posterior a contar con el prototipo elaborado y mostrado al cliente, es momento de comenzar el desarrollo. Es importante realizarle pruebas convenientes y que los usuarios lo evalúen, para de esta manera garantizar la viabilidad de una aplicación que cumpla con las normas y permitir concretar, refinar los requisitos y revelar las áreas críticas de mejora del software.

4.1.5. Entrega y Retroalimentación

Una de las cosas con las que cuenta el modelo de prototipos, es que, una vez entregado el proyecto, debemos darle al cliente cierta retroalimentación sobrecómo utilizarlo y ciertamente es una fase que se encuentra dentro de las etapas de desarrollo de software esta metodología.

4.1.6. Comunicación con el Cliente

Es importante que, una vez entregado el proyecto, tengamos cierta comunicación con el cliente, básicamente para que nos indique si el proyecto es correcto o si desea agregarle ciertas funciones.

Se produce un proceso iterativo en el que el prototipo es refinado para que satisfaga las necesidades del cliente, al tiempo que facilita unmejor conocimiento del sistema.

4.1.7. Entrega del Producto Final

Por último, solamente quedará entregar el sistema elaborado mediante esta metodología. Se comprueba que la aplicación cumpla con los requerimientos expuestos inicialmente por el cliente y los usuarios. Además, se espera que el cliente haya quedado satisfecho con el resultado del software final.

4.2. Metodología Scrum

La metodología de trabajo de Scrum tiene sus principios fundamentales en la década de 1980, la cual fue desarrollada por su necesidad en procesos de reingeniería por Goldratt, Takeuchi y Nonaka.

El concepto de Scrum tiene su origen sobre los nuevos procesos de desarrollo utilizados en productos exitosos en Japón y los Estados. Los equipos que desarrollaron estos productos partían de requisitos muy generales, así como novedosos, y debían salir al mercado en mucho menos del tiempo del que se tardó en lanzar productos anteriores. Estos equipos seguían patrones de ejecución de proyecto muy similares. En este estudio se comparaba la forma de trabajo de estos equipos altamente productivos y multidisciplinares con la colaboración entre los jugadores de Rugby y su formación de Scrum, de la cual se tomó su nombre. Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

Podríamos decir que SCRUM se basa en cierto caos controlado, pero establece ciertos mecanismos para controlar esta indeterminación, manipular lo impredecible y controlar la flexibilidad.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad es fundamental.

Figura 10.

Proceso de Scrum

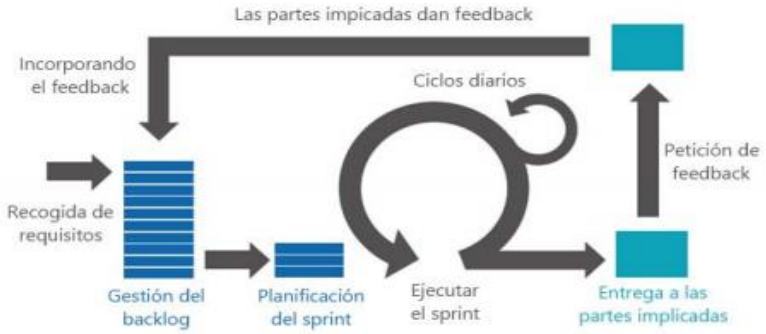


Figura Nº 14 – Proceso de Scrum.

Nota: Tomado de Metodologías de desarrollo de software *por Maida y Pacienzia.*

Las actividades que se llevan a cabo en Scrum son las siguientes:

- Planificación de la iteración (Sprint Planning)
- Ejecución de la iteración (Sprint)
- Reunión diaria de sincronización del equipo (Daily meeting)

- Demostración de requisitos completados (Sprint Review)
- Retrospectiva (Sprint Retrospective)
- Replanificación del proyecto (Product Backlog Refinement)

La planificación de las tareas a realizar en la iteración se divide en dos partes:

Primera parte de la reunión. Se realiza en un timebox promedio de 1 a 2 horas.

El cliente presenta al equipo la lista de requisitos priorizada (Product Backlog) del producto o proyecto, pone nombre a la meta de la iteración (de manera que ayude a tomar decisiones durante su ejecución) y propone los requisitos más prioritarios a desarrollar en ella

El equipo examina la lista, pregunta al cliente las dudas que le surgen, añaden más condiciones de satisfacción y selecciona los objetivos/requisitos más prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

Segunda parte de la reunión. Se realiza en un timebox promedio de 1 a 2 horas.

El equipo planifica la iteración, elabora la táctica que le permitirá conseguir el mejor resultado posible con el mínimo esfuerzo. Esta actividad la realiza el equipo dado que ha adquirido un compromiso, es el responsable de organizar su trabajo y es quien mejor conoce cómo realizarlo.

Define las tareas necesarias para poder completar cada objetivo/requisito, creando la lista de tareas de la iteración (Sprint Backlog).

Se realiza una estimación conjunta del esfuerzo necesario para realizar cada tarea (Pocker Planning).

Cada miembro del equipo se autoasigna a las tareas que puede realizar.

Este equipo está constituido por los siguientes perfiles:

- Desarrollador
- Analista funcional
- Analista Técnico
- Tester
- Arquitecto
- Proyect Manager
- Team Leader

Herramientas que se utilizan en Scrum:

- Lista de objetivos / requisitos priorizada (Product Backlog)
- Lista de tareas de la iteración (Sprint Backlog)
- Gráficos de trabajo pendiente (Burndown charts)

Ventajas de utilizar Scrum:

- Entregas parciales a corto plazo de resultados
- Gestión regular de las expectativas del cliente y basada en resultados tangibles.
- Resultados anticipados.
- Flexibilidad y adaptación respecto a las necesidades del cliente, cambios en el mercado, etc.
- Gestión sistemática del Retorno de Inversión (ROI).

- Mitigación sistemática de los riesgos del proyecto.
- Productividad y calidad.
- Alineamiento entre el cliente y el equipo de desarrollo.
- Equipo motivado.

5. Resultados y Discusión

5.1 Identificación de Procesos y Necesidades

El desarrollo de la actividad comercial propia del sector de calzado supone una serie de requerimientos para la sistematización de procedimientos. Uno de estos ámbitos es el ligado a la gestión de Ingresos/egresos y la generación de facturas. En este sentido, se identifica la necesidad de desarrollar un sistema que permita: gestionar de forma automatizada el inventario, que permita por ende identificar la disponibilidad de productos, el costo al que se comercializarán, y que en el proceso dinamice procesos como la comunicación entre las personas que despachan, fabrican y comercializan el calzado.

Partiendo de lo anterior, a continuación, se especifican los diagramas de uso en función de los cuales se amplían las necesidades identificadas. Cabe resaltar que el software desarrollado se compone de un modelo de 3 capaz: Modelo de datos, reglas de negocio y Vistas, donde se han llevado al desarrollo un módulo de administrador, modulo vendedor y uno de cliente en este sentido, los diagramas de usos se diferencian en función de dichos módulos.

5.1.1 Módulo Administrador

Tabla 1.

Diagrama de uso Inicio de Sesión

Descripción	**El administrador debe iniciar sesión con una cuenta de tipo administrador predeterminada. Y el sistema deberá tener una cuenta creada por un administrador**	
Nombre	Inicio de sesión	
Actores	Administrador.	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dirigirse al botón "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso contrario le mostrara un mensaje con el error.
Alternativa	N/A	

Precondición	Acceder al sistema
Postcondición	Se accedió al sistema

Nota: Elaboración propia.

Tabla 2.

Diagrama de Uso Registro de Usuarios

Descripción	**El administrador deberá registrar los usuarios para que puedan tener acceso a las funciones de la plataforma**	
Nombre	Registro de usuarios (administrador, vendedor,cliente)	
Actores	Administrador	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1.Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dirigirse al botón "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso

		contrario le mostrara un mensaje con el error.
	4.Dirigirse al menú lateral	Despliega las opciones del menú
	5.Dirigirse al botón "Usuarios del Sistema"	Muestra el módulo de usuarios
	6.Dirigirse al botón "Nuevo"	Despliega un formulario de registro en una ventana modular
	7.Diligenciar el formulario de registro	
	8.Dirigirse al botón de guardar	Valida los campos del formulario y guarda un registro en la base de datos con los datos del nuevo usuario y mostrará un mensaje donde dirá que el usuario ha sido agregado con éxito. En caso contrario mostrara los errores del formulario
Alternativa	N/A	

Precondición	Se ha contratado un nuevo empleado que necesita una cuenta para acceder al sistema
Postcondición	Se ha agregado un nuevo usuario al sistema

Nota: Elaboración propia.

Tabla 3.

Diagrama de Uso Modificar Usuario

Descripción	**Modifica los datos del usuario, cambiarle el rol, desactivar o activar**	
Nombre	Modificar usuario	
Actores	Administrador	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dirigirse al botón "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso contrario le mostrara un mensaje con el error.

	4.Dirigirse al menú lateral	Despliega las opciones del menú
	5.Dirigirse al botón "Usuarios del Sistema"	Muestra el módulo de usuarios. Se mostrará una tabla con los usuarios registrados
	6. Dirigirse a la columna de acciones y dar click en "editar" del registro a modificar	Despliega un formulario con los datos del usuario en una ventana modular
	7.Modifique los campos que quiera actualizar	
	8.Dar click en actualizar	Validara los datos ingresados, actualizara los datos del usuario en la base de datos y mostrara un mensaje donde dirá que el usuario ha sido actualizado con éxito. Caso contrario mostrara un mensaje con el error.
Alternativa	N/A	
Precondición	Se requiere la actualización de datos de un usuario	

Postcondición	Se han actualizado los datos del usuario

Nota: Elaboración propia.

Tabla 4.

Diagrama de Uso Agregar una Nueva Categoría

Descripción	**Crea nuevas categorías para usarlas al momento de crear o actualizar un producto**	
Nombre	Agregar una nueva categoría	
Actores	Administrador	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dar click en "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso

		contrario le mostrara un mensaje con el error.
	4.Dirigirse al menú lateral	Despliega las opciones del menú
	5.Dirigirse al botón "Categorias de Productos " y dar click	Muestra el módulo de categorías.
	6. Dar click en "Nuevo"	Muestra un campo para ingresar el nombre de la categoría
	7.Digite el nombre de la categoría	
	8.Dar click en "Guardar"	Valida el campo, agregará un nuevo registro en la base de datos y mostrará mensaje donde dirá que la categoría ha sido agregada con éxito. Caso contrario mostrara un mensaje con el error.
Alternativa		N/A

Precondición	Se necesita agregar una categoría para el registro de un producto
Postcondición	Se han agregado una nueva categoría a la base de datos

Nota: Elaboración propia.

Tabla 5.

Diagrama de Uso Actualizar una Categoría

Descripción	**Actualizar las categorías para identificar mejor los productos**	
Nombre	actualizar una categoría	
Actores	Administrador	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	

2. Digitar las credenciales de la cuenta	
3. Dar click en "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso contrario le mostrara un mensaje con el error.
4.Dirigirse al menú lateral	Despliega las opciones del menú
5.Dar click en "Categorías de Productos "	Muestra el módulo de categorías. Se mostraran todas las categorías registradas.
6. Dirigirse a la categoría a editar y darle click en el icono de "editar"	Muestra un campo para con el nombre de la categoría seleccionada
7.Modifique el nombre de la categoría	
8.Dar click en el botón "Guardar" para actualizar la categoría	Valida el campo, actualiza la categoría en la base de datos y mostrará mensaje donde dirá que la categoría ha sido actualizada con éxito. Caso contrario mostrara un mensaje con el error.

Alternativa	N/A
Precondición	Se necesita agregar una categoría para el registro de un producto
Postcondición	Se han agregado una nueva categoría a la base de datos

Nota: Elaboración propia.

Tabla 6.

Diagrama de Uso Agregar Nuevo Producto

Descripción	**Registra productos, selecciona una categoría, stock, precio y una descripción relacionada al producto y lleva el control del inventario**
Nombre	Agregar nuevo producto

Actores	Administrador	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dar click en "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso contrario le mostrara un mensaje con el error.
	4.Dirigirse al menú lateral	Despliega las opciones del menú
	5.Dar click en "Productos"	Muestra el módulo de productos.
	6. Dar click en "Nuevo"	Muestra un formulario con los campos para registrar un producto en una ventana modal
	7.Digite los campos con los datos del nuevo producto	

	8.Dar click en "Guardar"	Valida los datos ingresados, crea un nuevo registro en la base de datos y mostrará mensaje donde dirá que el producto ha sido agregado con éxito. Caso contrario mostrara un mensaje con el error.
Alternativa	N/A	
Precondición	Se necesita agregar un nuevo producto	
Postcondición	Se han agregado un nuevo registro con los datos del producto en la base de datos	

Nota: Elaboración propia.

Tabla 7.

Diagrama de Uso Actualizar Producto

Descripción	**Actualiza los datos del producto como el stock, la categoría, el precio la cantidad del inventario**	
Nombre	Actualizar producto	
Actores	Administrador	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dar click en "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso contrario le mostrara un mensaje con el error.
	4.Dirigirse al menú lateral	Despliega las opciones del menú
	5.Dar click en "Productos"	Muestra el módulo de productos. Se cargara una tabla con los productos registrados
	6. Dirigirse a la columna de acciones y darle click en el icono	Muestra un formulario con los datos del producto seleccionado en una ventana modular

	de "editar" en el registro a actualizar	
	7.Modifique los datos	
	8.Dar click en "Actualizar"	Valida los datos ingresados, actualiza registro con los nuevos datos en la base de datos y mostrará un mensaje donde dirá que el producto ha sido actualizado con éxito. Caso contrario mostrara un mensaje con el error.
Alternativa	N/A	
Precondición	Se necesita actualizar un producto	
Postcondición	Se ha actualizado los datos del producto en la base de datos	

Nota. Elaboración propia.

Tabla 8.

Diagrama de Uso Agregar Proveedores

Descripción	**Agrega nuevas proveedores**	
Nombre	Agregar proveedores	
Actores	Administrador	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dar click en "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso contrario le mostrara un mensaje con el error.
	4.Dirigirse al menú lateral	Despliega las opciones del menú
	5.Dirigirse al botón "Proveedores" y dar click	Muestra el módulo de proveedores.

	6. Dar click en "Nuevo"	Muestra los campos para ingresar el nombre y la descripción del proveedor
	7.Digite los campos del formulario	
	8.Dar click en "Guardar"	Valida los datos ingresados en el formulario, agregará un nuevo registro en la base de datos y mostrará mensaje donde dirá que el proveedor ha sido agregada con éxito. Caso contrario mostrara un mensaje con el error.
Alternativa	N/A	
Precondición	Se necesita agregar más proveedores	
Postcondición	Se han agregado un nuevo registro de proveedor a la base de datos	

Nota: Elaboración propia.

Tabla 9.

Diagrama de Uso Actualizar Proveedor

Descripción	**Actualiza nuevos proveedores**	
Nombre	Actualizar proveedores	
Actores	Administrador	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dar click en "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso contrario le mostrara un mensaje con el error.
	4.Dirigirse al menú lateral	Despliega las opciones del menú

	5.Dirigirse al botón "Proveedores" y dar click	Muestra el módulo de Proveedores.
	6. Dirigirse al registro a modificar y darle click al botón "editar"	Muestra el formulario con los datos del proveedor seleccionado
	7.Modifique los datos.	
	8.Dar click en el botón del "Guardar"	Valida los datos ingresados en el formulario, agregará actualiza el registro en la base de datos y mostrará un mensaje donde dirá que la mesa ha sido actualizada con éxito. Caso contrario mostrara un mensaje con el error.
Alternativa	N/A	
Precondición	Se necesita actualizar un proveedor	
Postcondición	Se ha actualizado un registro de proveedor en la base de datos	

Nota: Elaboración propia.

Tabla 10.

Diagrama de Uso Agregar Vendedores

Descripción	**Agrega nuevos vendedores**	
Nombre	Agregar vendedores	
Actores	Administrador	
Flujo de eventos	**Eventos actores**	**Eventos sistema**
	1. Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dar click en "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso contrario le mostrara un mensaje con el error.

	4.Dirigirse al menú lateral	Despliega las opciones del menú
	5.Dirigirse al botón " “Vendedores " y dar click	Muestra el módulo de proveedores.
	6. Dar click en "Nuevo"	Muestra los campos para ingresar el nombre y la descripción del vendedor
	7.Digite los campos del formulario	
	8.Dar click en "Guardar"	Valida los datos ingresados en el formulario, agregará un nuevo registro en la base de datos y mostrará mensaje donde dirá que el vendedor ha sido agregada con éxito. Caso contrario mostrara un mensaje con el error.
Alternativa	N/A	
Precondición	Se necesita agregar más vendedores	
Postcondición	Se han agregado un nuevo registro de proveedor a la base de datos	

Nota: Elaboración propia.

Tabla 11.

Diagrama de Uso Actualizar Proveedor

Descripción	**Actualiza nuevos proveedores**	
Nombre	Actualizar proveedores	
Actores	Administrador	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dar click en "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso

		contrario le mostrara un mensaje con el error.
	4.Dirigirse al menú lateral	Despliega las opciones del menú
	5.Dirigirse al botón " “Vendedores " y dar click	Muestra el módulo de vendedores.
	6. Dirigirse al registro a modificar y darle click al botón "editar"	Muestra el formulario con los datos del proveedor seleccionado
	7.Modifique los datos.	
	8.Dar click en el botón del "Guardar"	Valida los datos ingresados en el formulario, agregará actualiza el registro en la base de datos y mostrará un mensaje donde dirá que la mesa ha sido actualizada con éxito. Caso contrario mostrara un mensaje con el error.
Alternativa	N/A	

Precondición	Se necesita actualizar un vendedor
Postcondición	Se ha actualizado un registro de vendedor en la base de datos

Nota: Elaboración propia.

Tabla 12.

Diagrama de Uso Crear Venta y sus detalles

Descripción	**Crea factura digital para los clientes y lleva un registro de tus ventas**	
Nombre	Crear factura	
Actores	vendedor	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	

2. Digitar las credenciales de la cuenta	
3. Dar click en "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso contrario le mostrara un mensaje con el error.
4.Dirigirse al menú lateral	Despliega las opciones del menú
5.Dirigirse al botón "Nueva Venta" y dar click	Muestra el módulo de facturación.
6. Dirigirse al apartado donde dice clientes.	
7.Dar click en el campo	Despliega una lista con los clientes previamente registrados en el sistema
8.Selecciona la opción de búsqueda	
9.Dar click en el campo "Vendedor"	Despliega una lista con las vendedores

10.Selecciona el botón “Detalles”	Cargar una vista con los detalles de la venta
11.Dale click en el campo de producto	Despliega una lista con los productos
12.Dirigirse a la columna de acciones y dar click al botón "agregar"	
14.En el apartado izquierdo de factura en la sección de información del producto dar click en el campo "cantidad"	pasará el registro seleccionado al apartado izquierdo de factura donde se listara en la sección de "Información del producto".
15.Digite la cantidad del producto	
16.Digite el descuento en caso de que quiera cambiar el precio	Calcula el valor total del ítem
18.Dar click en "Remover"	Formatea dicha ventana

19.Dar click en "Guardar factura"	Calcula el valor total del ítem
21.Dar click en "Volver a la lista"	Regresa a la lista de todas las ventas
N/A	Crea un registro en la base de datos con los datos, actualiza el inventario de los productos agregados a la factura, muestra un mensaje con que dirá que la factura ha sido creada con éxito y desplegara una ventana modal con dos botones. Caso contrario mostrara un mensaje de error.
N/A Se requiere ingresar una factura al software e imprimir la factura	Abrirá una nueva ventana donde se mostrará el tipo de factura y tipo de venta junto con un consecutivo único por venta y es anexado a la facura.
N/A Se requiere ingresar una factura al software e imprimir la factura Se ha creado un registro en la base de datos con los datos de la factura y se ha impreso la factura	Guarda la factura

	N/A	Cierra la ventana modal
	Se requiere ingresar una factura al software e imprimir la factura	
	Se ha creado un registro en la base de datos con los datos de la factura y se ha impreso la factura	
Alternativa	N/A	
	Se requiere ingresar una factura al software e imprimir la factura	
	Se ha creado un registro en la base de datos con los datos de la factura y se ha impreso la factura	
Precondición		
Postcondición		

Nota: Elaboración propia.

5.1.2 Módulo Login

Tabla 13.

Diagrama de Uso Inicio de Sesión de un Usuario

Descripción	**El usuario deberá tener una cuenta creada por un administrador**	
Nombre	Inicio de sesión	
Actores	usuario	
Flujo de eventos	**Eventos actor**	**Eventos sistema**
	1. Abrir la plataforma	
	2. Digitar las credenciales de la cuenta	
	3. Dirigirse al botón "Iniciar sesión"	Valida las credenciales. Si son correctas lo dirigirá a la ventana principal. Caso contrario le mostrara un mensaje con el error.
	4. Identificación de usuario	El sistema tiene la capacidad de identificar que tipo de usuario es, es decir el rol al cual fue asignado, también valida si esta activo o no, y dependiendo dichas validaciones tiene acceso a ciertas áreas del sistema permitiéndole ciertos permisos por rol
Alternativa	N/A	

Precondición	Acceder al sistema
Postcondición	Se accedió al sistema

Nota: Elaboración propia.

5.2 Definición de metodología: Herramientas

5.2.1 Soporte de Backend

El backend seleccionado para el desarrollo del software CCCVentas fue .NetCore con IDE Visual estudio 2019. Esta es una herramienta que permite el desarrollo de plataformas web api ofreciendo la posibilidad software con arquitectura de microservicios. Este comprende servidor de bases de datos como lo es SQL Server, y opera bajo la utilización de lenguajes de programación de código abierto haciendo de infinitas posibilidades, ya que de la misma forma cada microservicio puede ser creado en numerosas tecnologías yo he decidido trabajar en .NetCore.

Así mismo, crear aplicaciones con Angular y reutilice su código y permite crear aplicaciones para cualquier objetivo de implementación. Para web, web móvil, móvil nativo y escritorio nativo. ASP.**NET Core** es una nuevo framework web, open-source (GitHub) y multiplataforma pensado para crear aplicaciones web modernas, con foco en aprovechar la nube así como en solucionar algunos de los nuevos desafíos como IoT y backends para mobile apps. a la seguridad y la

modularidad del desarrollo a partir de librerías, esto dado que se usaron mecanismos de hash y salt para encriptar a partir de librerías.

5.2.2 Arquitectura del Sistema

REST es una de las más habituales en nuestros días. Para algunas personas REST es una arquitectura, para otras es un patrón de diseño, para otras un API. REST exactamente? REST es un ESTILO de Arquitectura a la hora de realizar una comunicación entre cliente y servidor. Se envía una información y se espera un resultado.

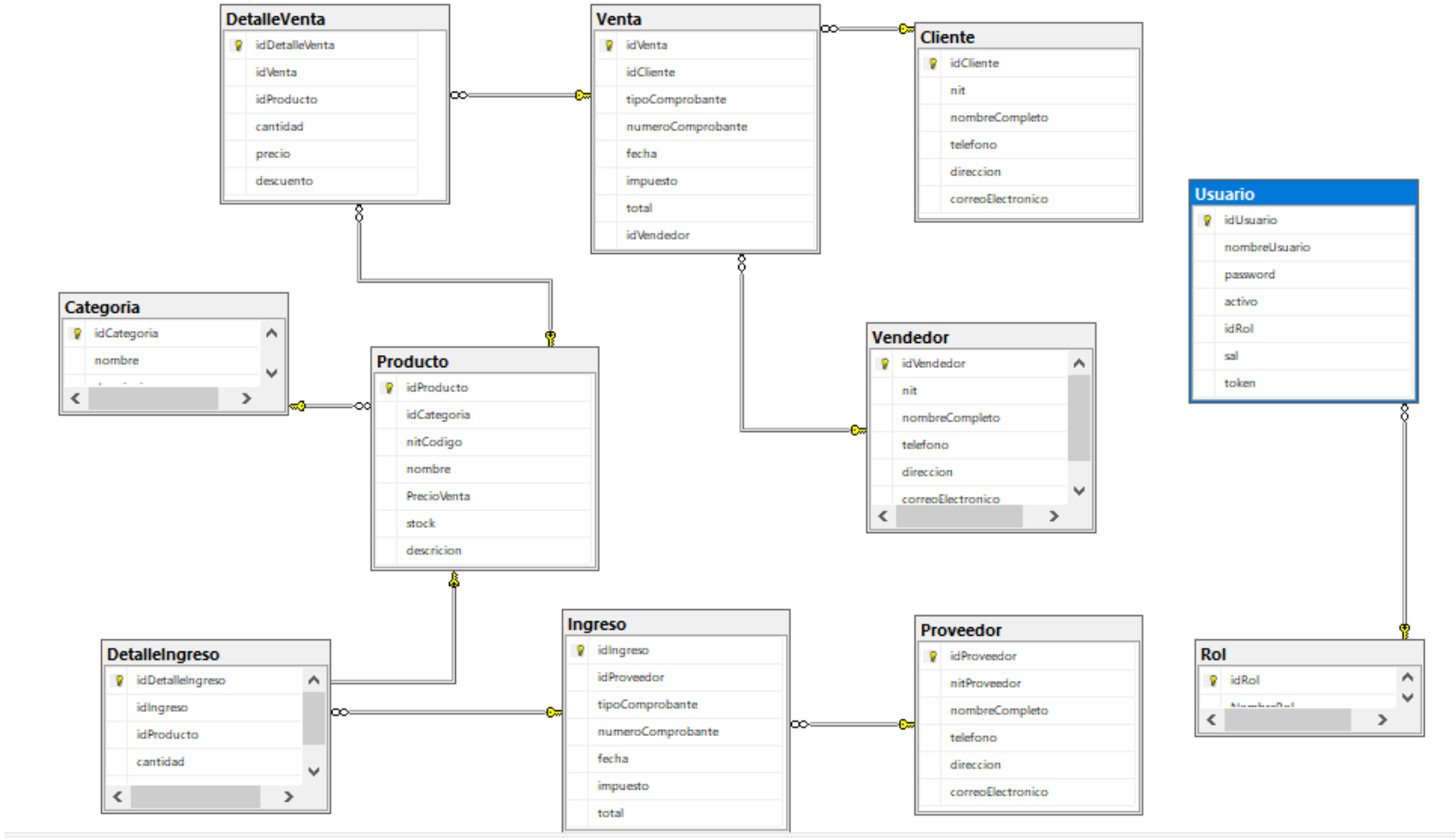
5.2.3 Arquitectura de la Plataforma Web

El funcionamiento de la arquitectura de la plataforma web se ejemplifica a través de una serie de diagramas, siendo el principal el modelo de base de datos. El tipo de base de datos desarrollada parte de la utilización de SQLServer y se caracteriza como relacional, es decir, evita la redundancia de registros en la base de datos a partir de llaves foráneas. Lo anterior contribuye a que la base de datos sea menos pesada y por ende permite un manejo más eficiente de los datos. Así mismo, supone ventajas como el manejo de grandes volúmenes de datos con puntos de relación entre sí que se gestionan de modo uniforme y el manejo de datos uniformes en todas las aplicaciones y copias de la misma base, denominadas instancias.

A continuación, se presenta el maquetado de la base de datos desarrollada:

Figura 11.

Modelo Base de Datos CCCVentas



Nota: Elaboración propia.

5.3 Diseño Web Responsive

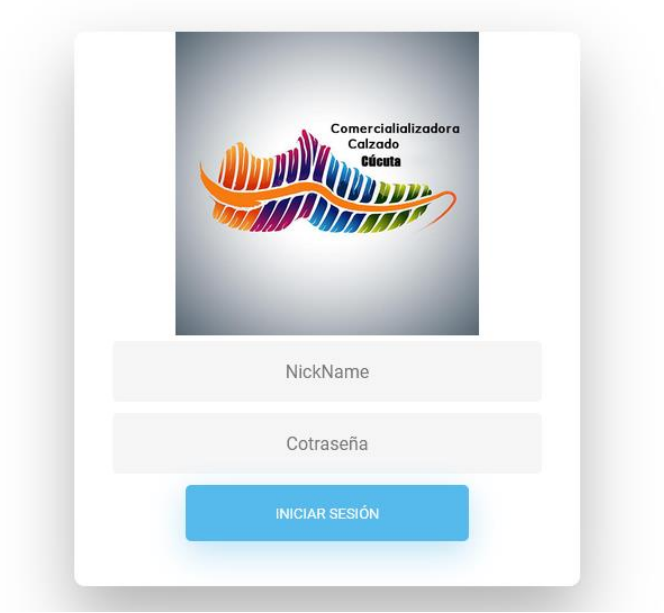
El diseño web responsive o adaptativo es una técnica de diseño web que busca la correcta visualización de una misma página en distintos dispositivos. Desde ordenadores de escritorio a tablets y móviles.

El diseño responsive permite reducir el tiempo de desarrollo, evita los contenidos duplicados, y aumenta la viralidad de los contenidos ya que permite compartirlos de una forma mucho más rápida y natural.

5.3.1 Interfaz Gráfica Módulo Administrador

Figura 12.

Interfaz Registro de Usuario



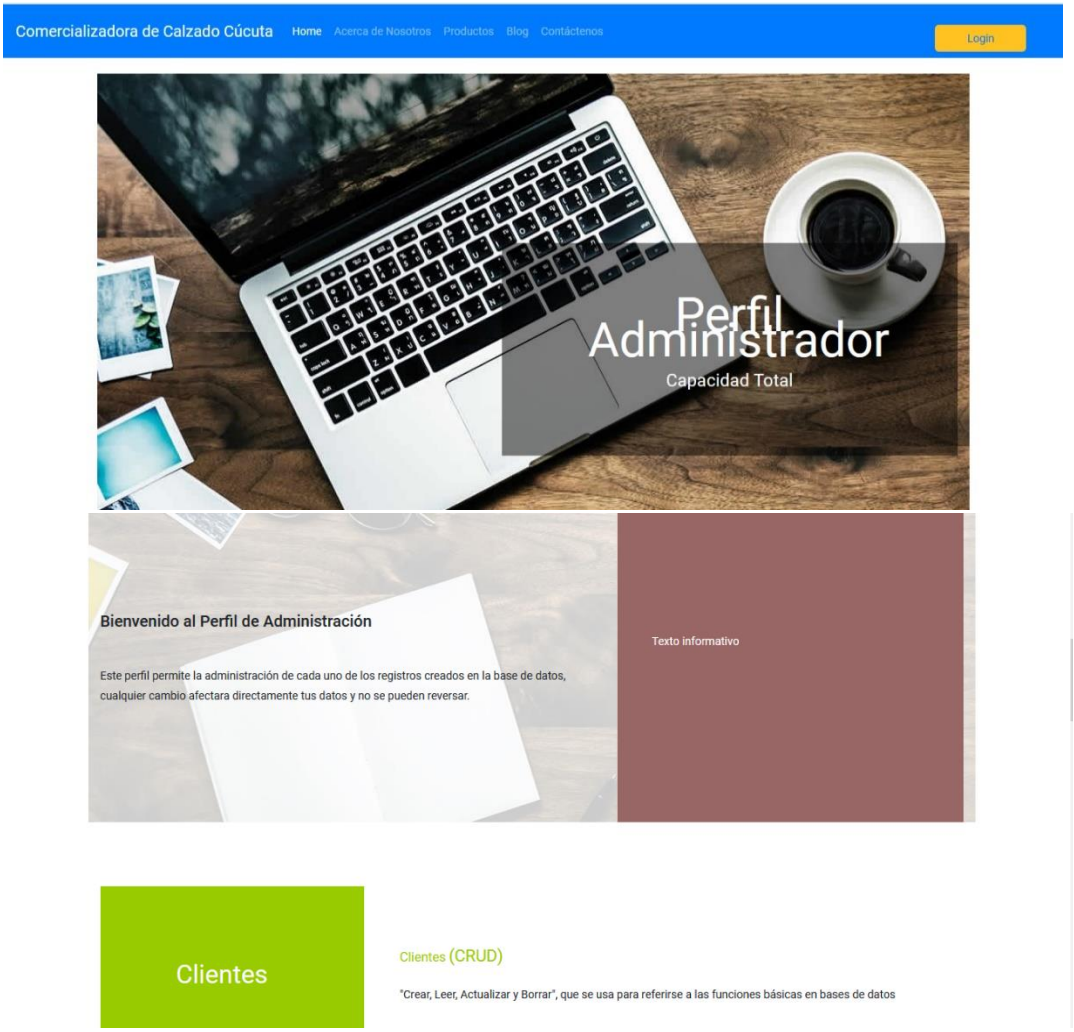
Nota: Elaboración propia.

En la Figura 4 se presenta el login que da acceso a la plataforma a los usuarios registrados como administrador. Esta se compone de: campo de NickName y contraseña, botón para la selección de

recordatorio, funcionalidad en caso de que el usuario olvide su contraseña y botón de inicio de sesión.

Figura 13.

Interfaz Vista Principal rol Administrador



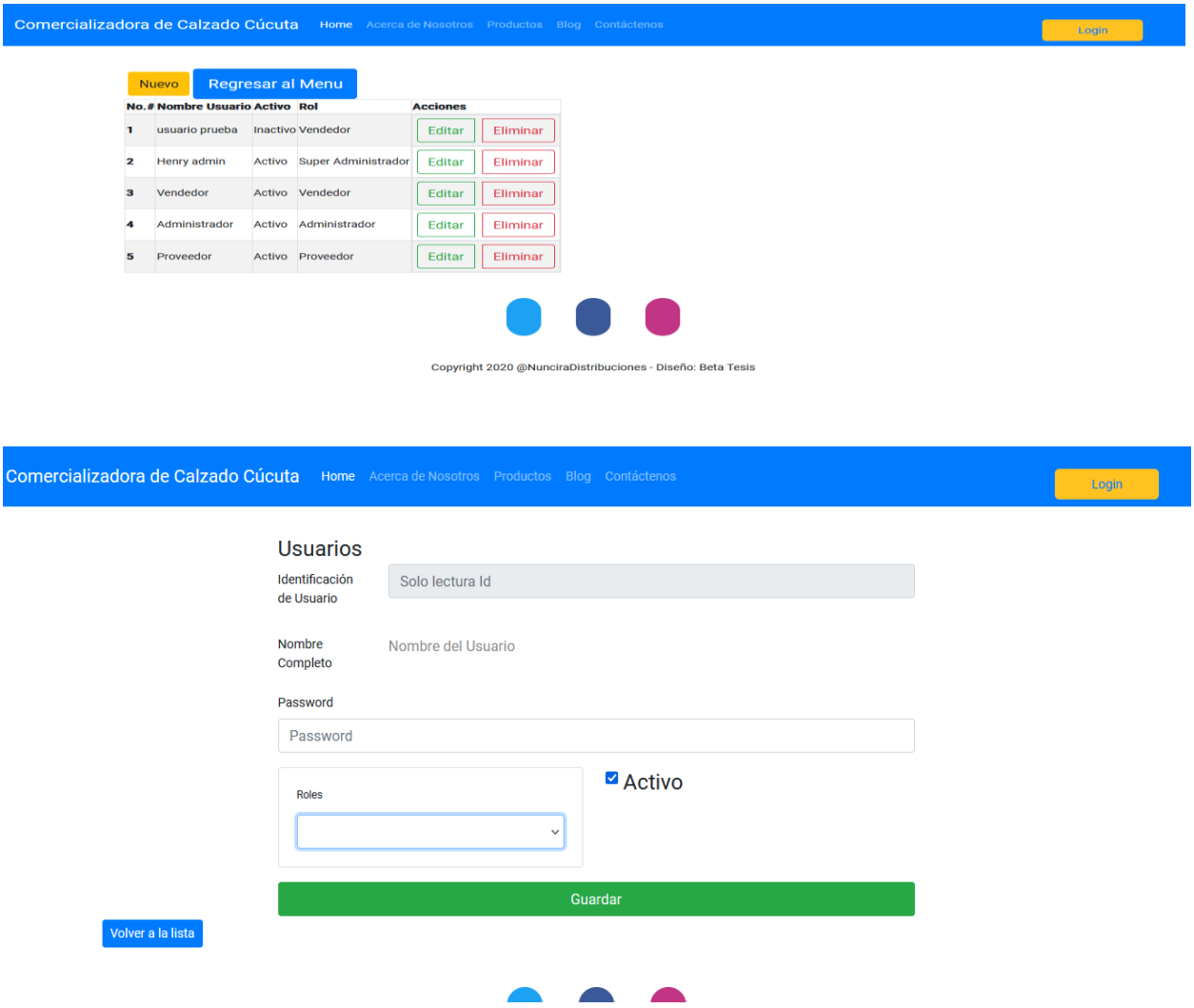
Nota: Elaboración propia.

En la figura 5 se presenta la interfaz principal a la que tiene acceso el usuario registrado como administrador. Esta se compone de: Usuarios, Categorías, Productos, Proveedores y Registro de ventas. Así mismo contiene un apartado denominado AD, en el cual el administrador podrá

cambiar su contraseña y demás datos de usuario personales, y una sección identificada con una lupa a través de la cual el usuario podrá facilitar sus búsquedas en la plataforma.

Figura 14.

Interfaz Usuarios



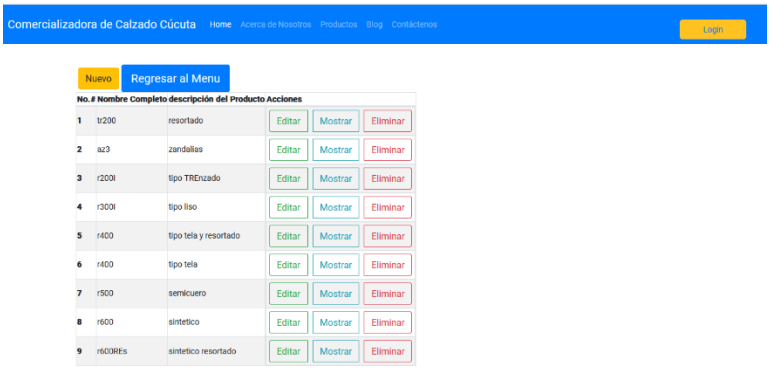
Nota: Elaboración propia.

En la figura 6 se refleja la interfaz de usuario, en la cual el administrador puede visualizar elementos como el nombre, correo, perfil y estado de los diferentes usuarios registrados en la

plataforma. Así mismo, contiene la opción de modificar los registros ya existentes a partir del botón de acciones y presenta un campo para facilitar la búsqueda de usuarios registrados.

Figura 15.

Interfaz Categorías



Nota: Elaboración propia.

En la figura 7 se presenta la interfaz de categorías, en esta el administrador y el vendedor podrá visualizar las categorías existentes y crear nuevas. Cabe resaltar que estas son necesarias para la emisión de facturas dado que, partiendo de estas, se organiza el inventario de la comercializadora con el objeto de agilizar la búsqueda de los productos.

Figura 16.

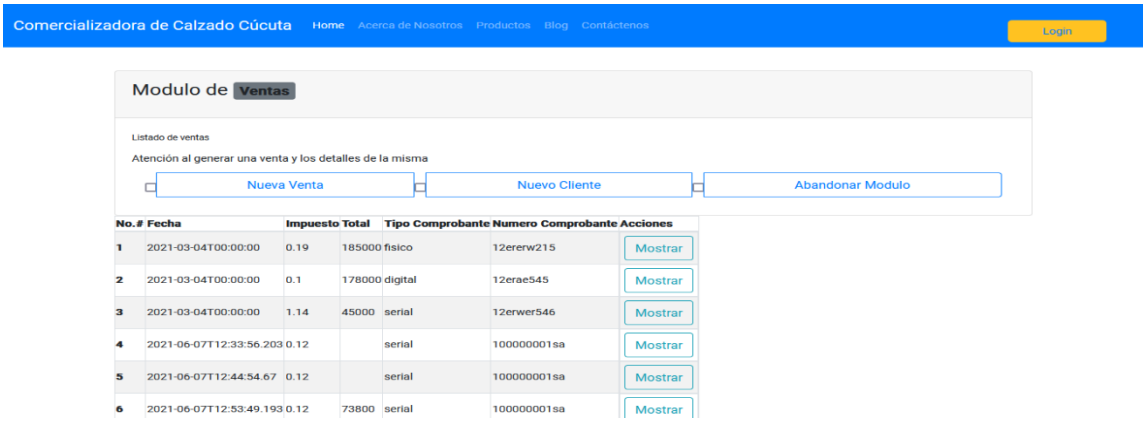
Interfaz de Productos

Nota: Elaboración propia.

En la figura 8 se presenta la interfaz denominada productos, a partir de esta se registran los diferentes productos, en función de las categorías anteriormente creadas, y se especifican elementos como su nombre, código, descripción, stock disponible, precio de compra, precio de venta. Así mismo, se permite la modificación de cada uno de los productos registrados y el filtrado de estos a partir de un campo de búsqueda.

Figura 17.

Interfaz Registro de Ventas



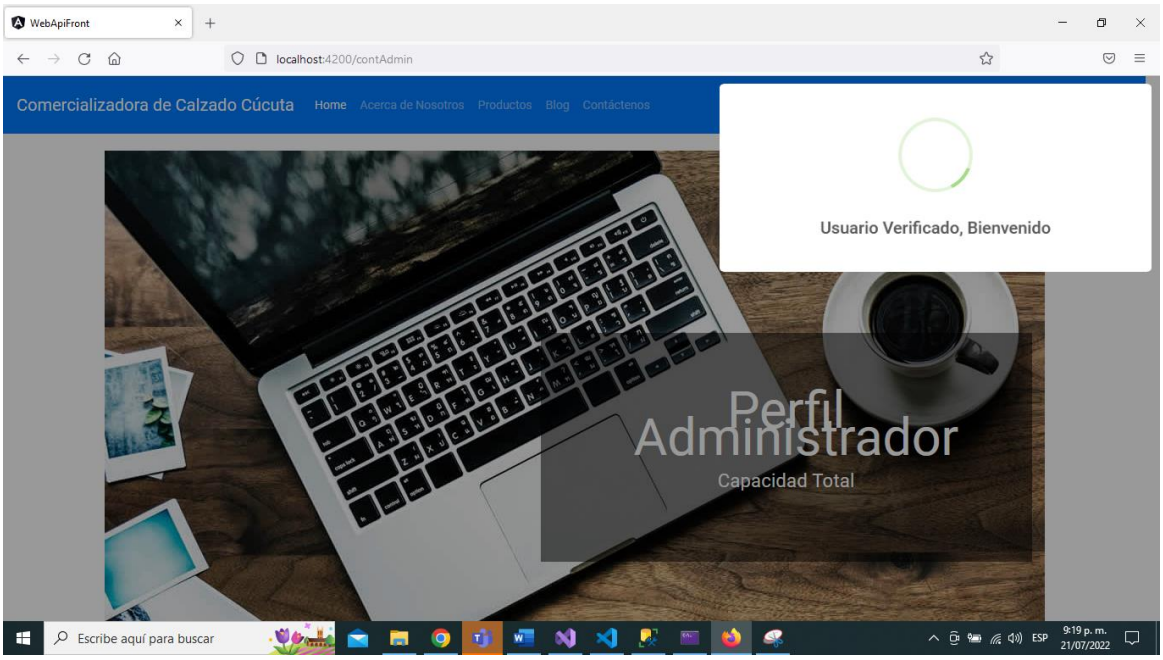
Nota: Elaboración propia.

En la figura 10 se presenta la interfaz de la sección de registro de ventas. Esta consta de una tabla en la que se presentarán la totalidad de las facturas emitidas por los diferentes usuarios y se especificará elementos como el número de ticket al que están asociadas junto con su tipo de comprobante, la fecha de realización, el vendedor que la emitió, el valor total. Así mismo, el apartado permite que el administrador haga una búsqueda rápida a partir del número del ticket, el vendedor a cargo o la fecha en la que se realizó la factura.

5.3.2 Interfaz Gráfica Módulo Login

Figura 18.

Interfaz Login Mesero

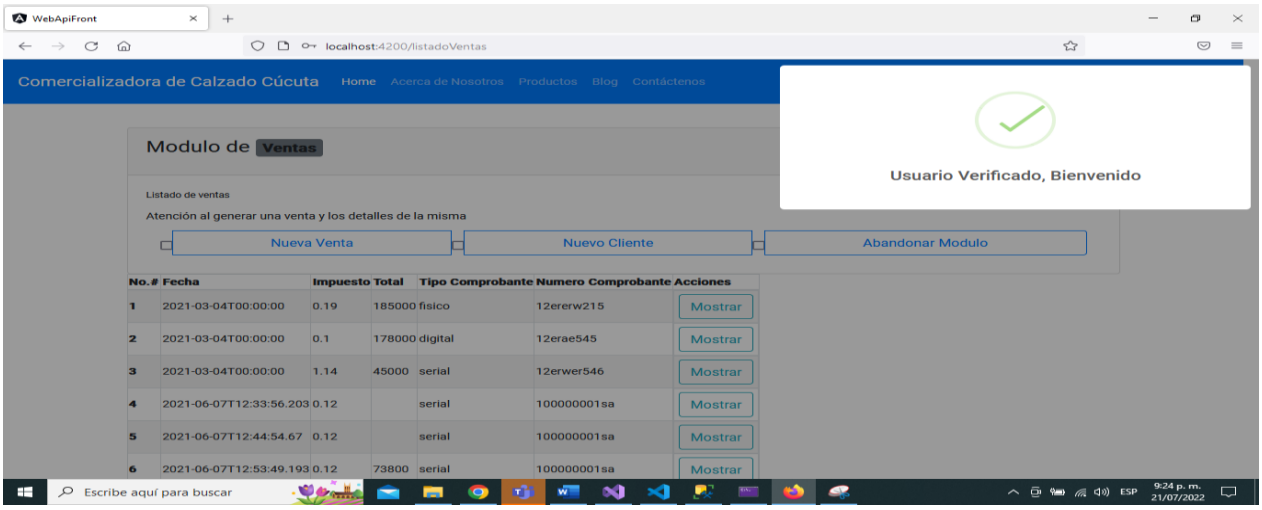


Nota: Elaboración propia.

En la figura 11 se presenta la primera interfaz con la que tiene contacto el Usuario, este se compone de dos campos principales, email y contraseña, y permite recordar el usuario, obtener ayuda en caso de olvidar la contraseña y a través del botón de iniciar sesión acceder a la plataforma. Una vez el usuario se logea se le dará acceso a diferentes pantallas dependiendo su rol, en esta imagen vemos el rol Administrador:

Figura 19.

Interfaz Gráfica Usuario Vendedor

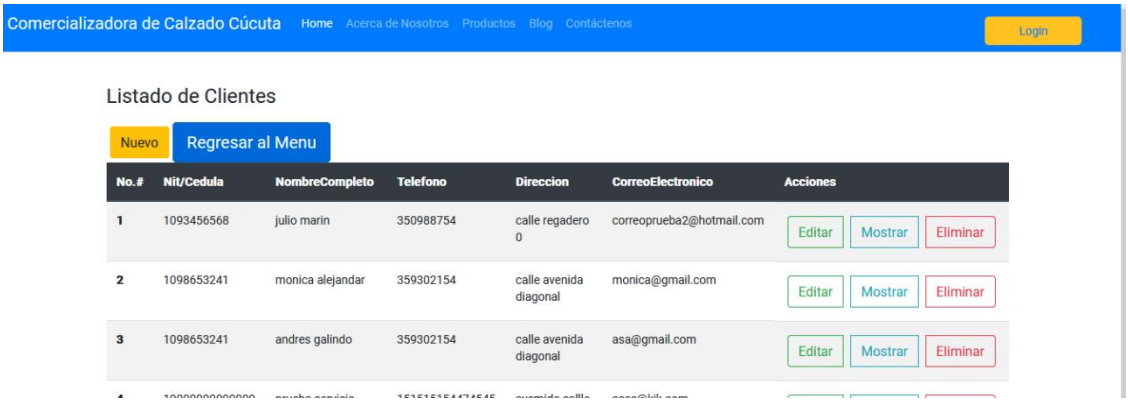


Nota: Elaboración propia.

En la figura 12 se presenta la interfaz principal a la que tiene acceso el personal Vendedor. Esta especifica la cantidad de ventas realizadas por la comercializadora y a partir de allí puede agregar nuevos clientes.

Figura 20.

Interfaz Gráfica nuevo cliente

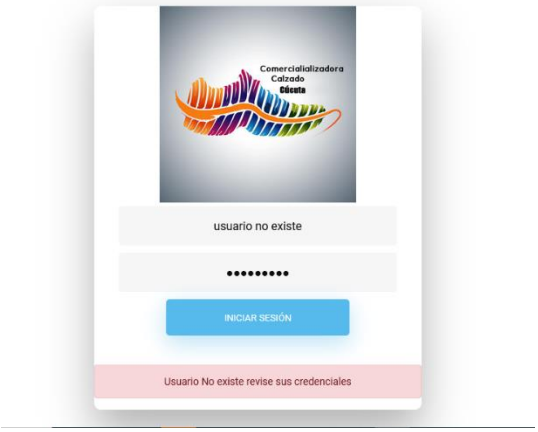


Nota: Elaboración propia.

En la figura 13 se presenta el apartado en el cual los usuarios administradores y vendedores pueden acceder para agregar nuevos clientes agregar o eliminar dicho cliente.

Figura 21.

Interfaz gráfica evento login usuario incorrecto

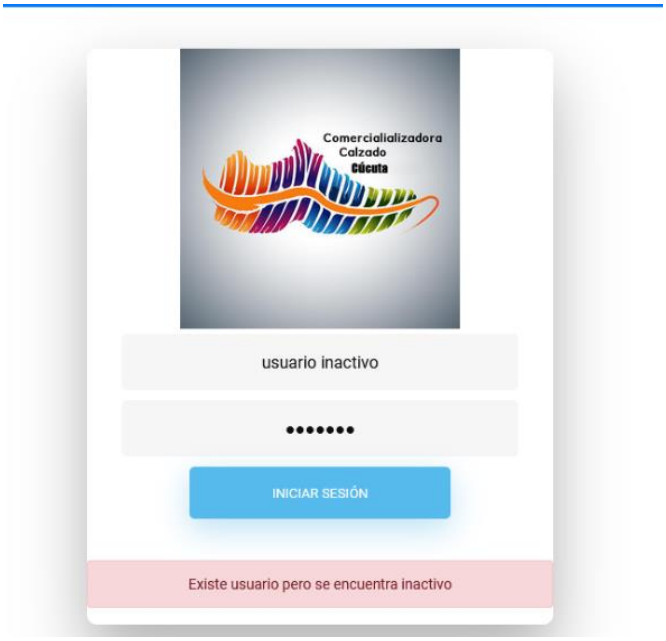


Nota: Elaboración propia.

En la figura 14 se presenta la interfaz del evento usuario inexistente dado que es un usuario que no está registrado o el dato nickname o contraseña son incorrectos.

Figura 22.

Interfaz gráfica evento login usuario inactivo

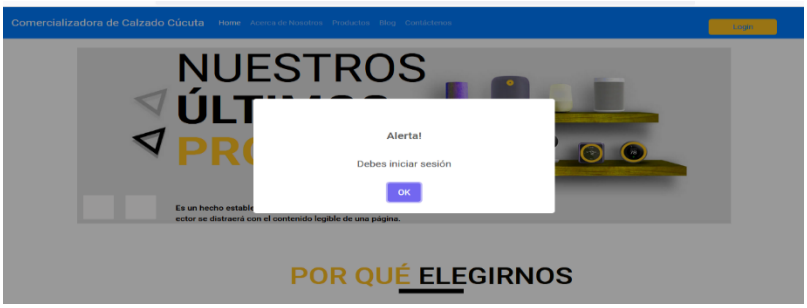


Nota: Elaboración propia.

En la figura 15 se presenta la interfaz del evento usuario inexistente cuando el usuario tiene un nickname valido y contraseña, pero ha sido inactivado por un administrador esto le quita permisos para acceder y le dirá que se encuentra inactivo.

Figura 23.

Interfaz Principal de guardianes para las páginas de acceso privado



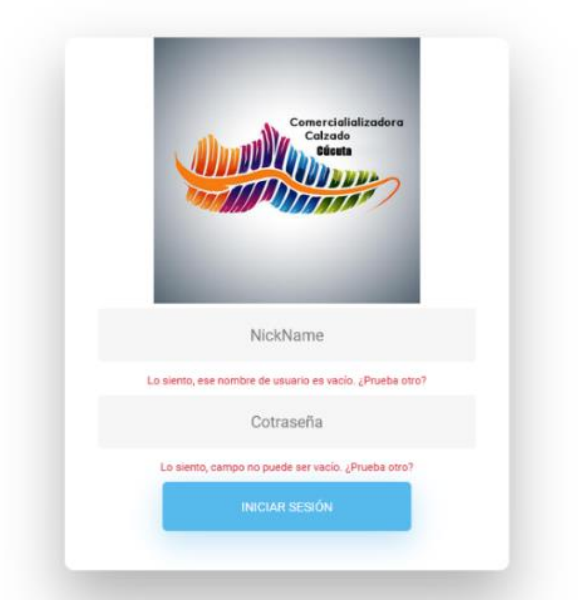
Nota: Elaboración propia.

En la figura 16 se presenta la interfaz principal donde se es redirigido cuando se intenta acceder a una URL en la cual no tiene permisos o no se encuentra con una sesión iniciada eso se puede interpretar como un token nulo.

5.3.4 Interfaz Gráfica validaciones

Figura 24.

Interfaz Validaciones de casillas

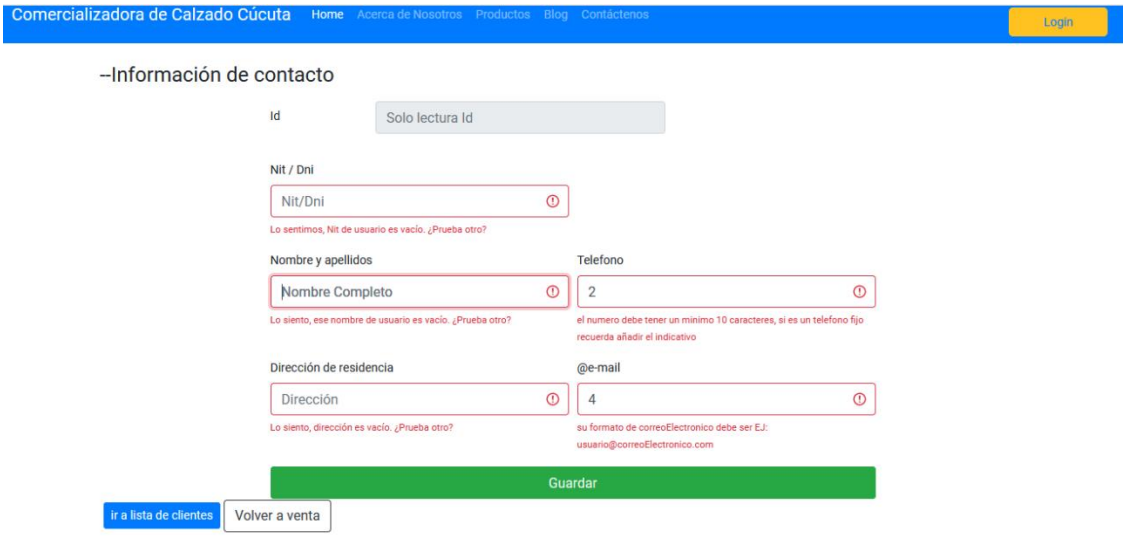


Nota: Elaboración propia.

En la figura 17 se muestra la interfaz donde se le indica al usuario que tipo de datos debe ingresar en los campos, en este caso login debe ser mayor a 6 dígitos y su contraseña y Nick no pueden ser vacíos.

Figura 25.

Interfaz Validaciones cliente



Nota: Elaboración propia.

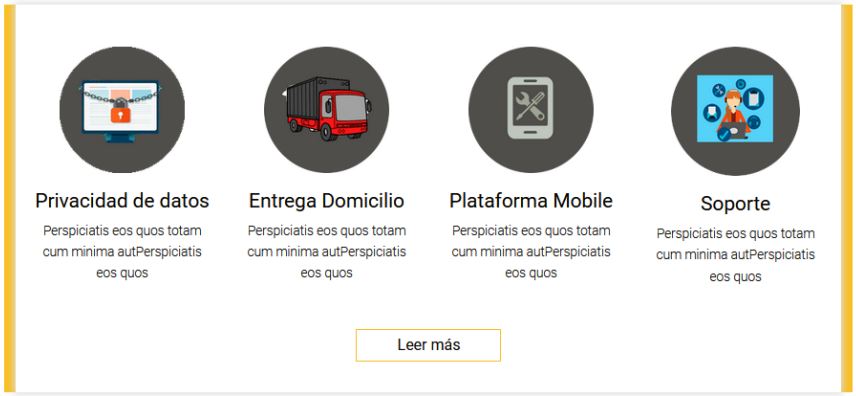
En la figura 18 se presenta la primera interfaz donde se valida que los nombres son solo alfabéticos los correos deben ser correos verdaderos el número de teléfono no puede ser diferente a 10 dígitos y empezar por 3.

Figura 26.

Interfaz Principal del sistema (home)

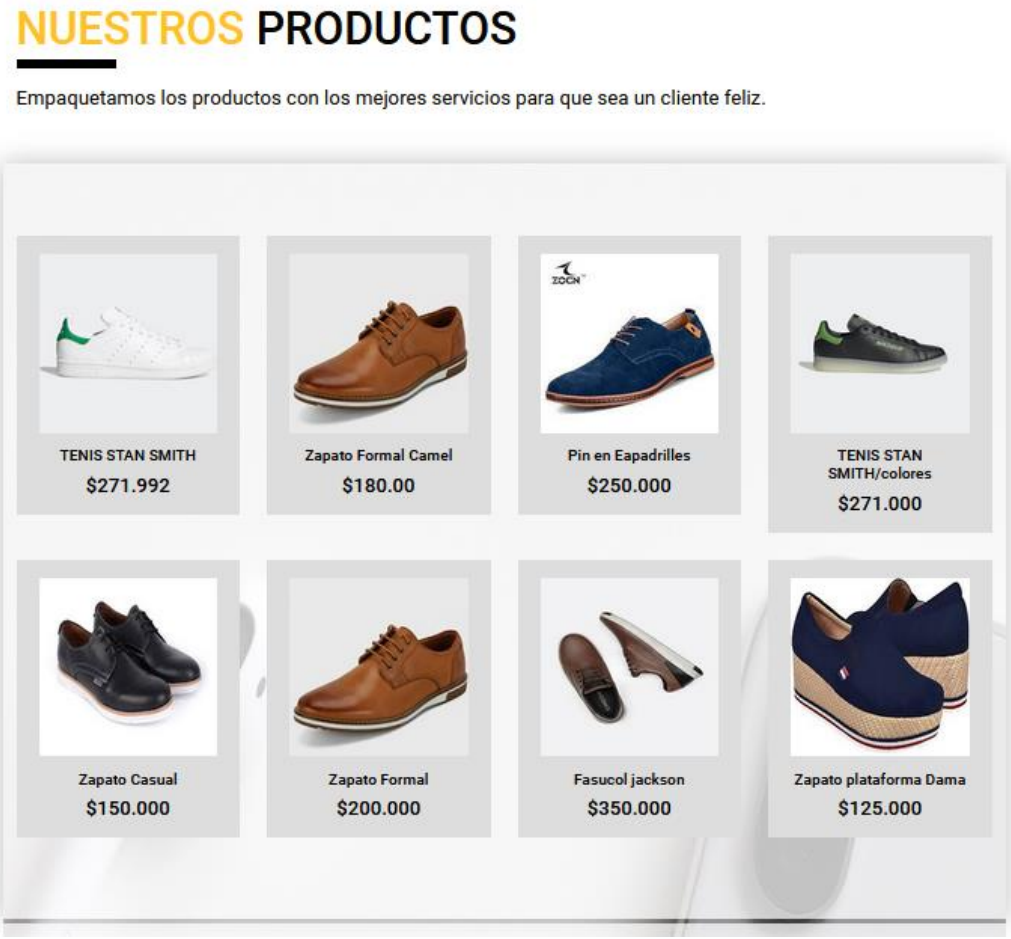
POR QUÉ ELEGIRNOS

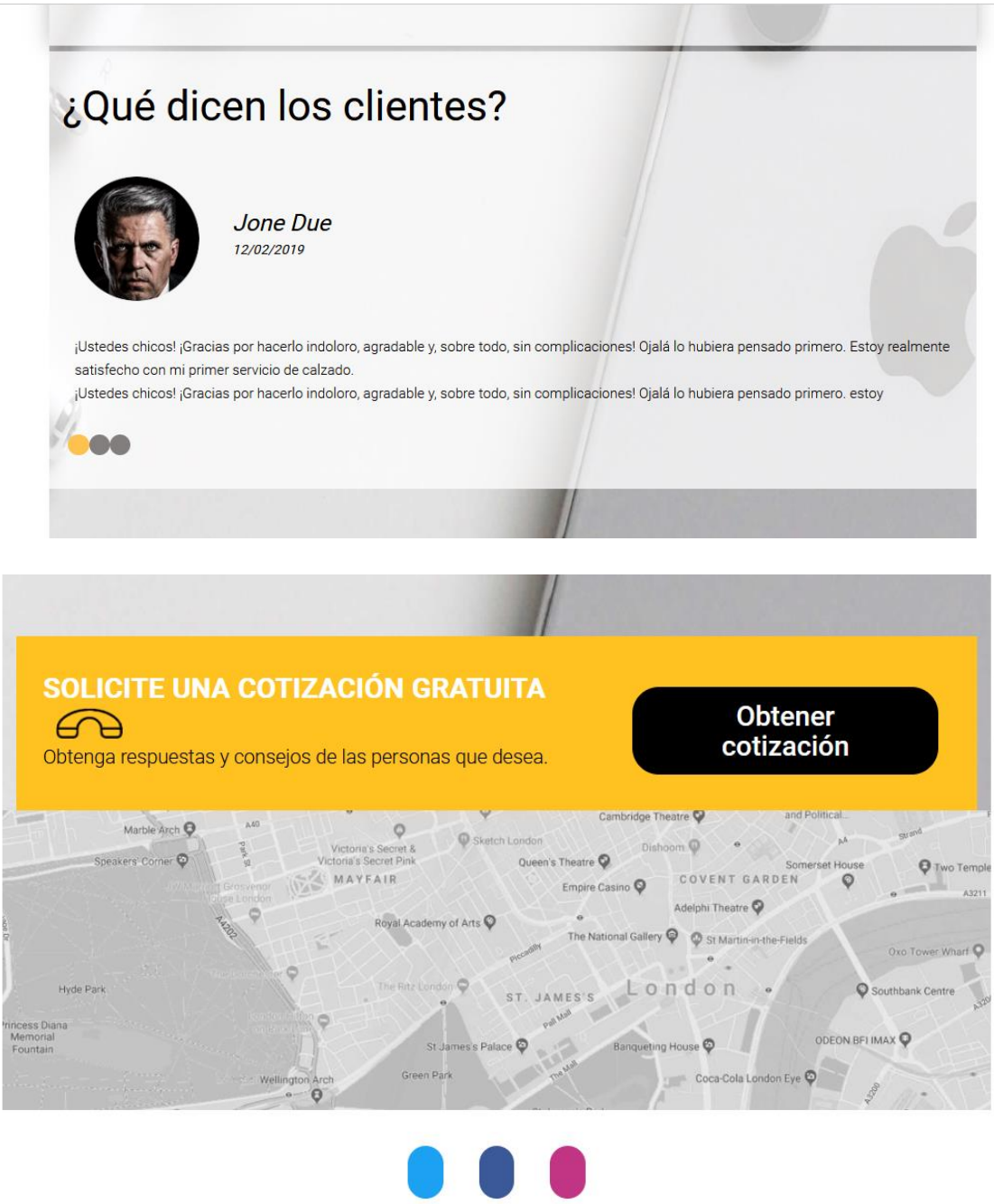
¡El servicio de calzado más rápido al mejor precio!



SERVICIOS/ PROCESOS

Manera fácil y efectiva de adquirir su calzado


NUESTROS PRODUCTOS
Empaquetamos los productos con los mejores servicios para que sea un cliente feliz.
TENIS STAN SMITH
$271.992
Zapato Formal Camel
$180.00
Pin en Eapadrilles
$250.000
TENIS STAN SMITH/colores
$271.000
Zapato Casual
$150.000
Zapato Formal
$200.000
Fasucol jackson
$350.000
Zapato plataforma Dama
$125.000

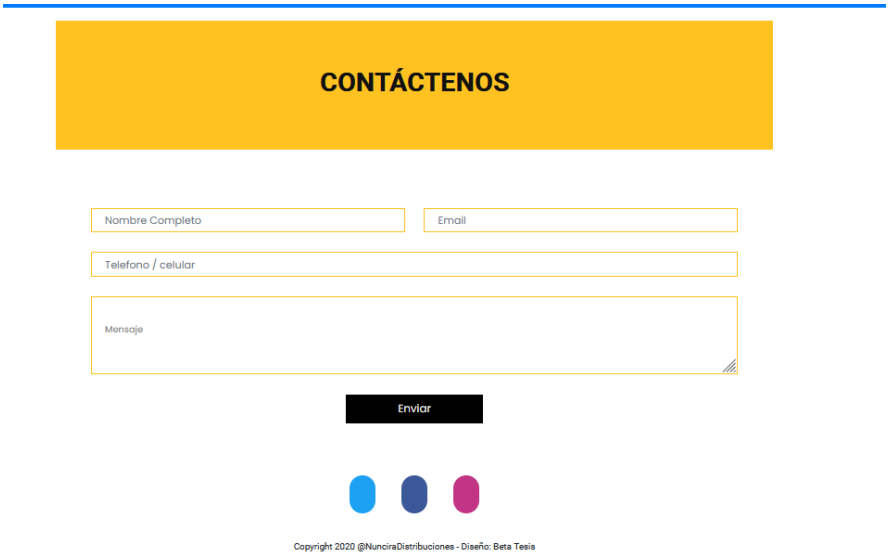


Nota: Elaboración propia.

En la figura 19 se presenta la interfaz principal del sistema donde las personas podrán ver un breve contenido de la comercializadora su información como empresa, productos y experiencias de otros usuarios o clientes.

Figura 27.

Interfaz Sesión contáctenos



Nota: Elaboración propia.

En la figura 20 se presenta la interfaz de contáctenos donde las personas interesadas van a poder colocarse en contacto inmediatamente con la comercializadora.

Tabla 14.

Rejilla Evaluación Software

Variables a Evaluar	1	2	3	4	5
Las consultas y reportes que le generó el software son confiables.				X	
La información que se gestiona en el software es de suma importancia para su actividad laboral, contar con esta facilita considerablemente su labor.					X

La funcionalidad ofrecida por el software apoya de manera adecuada la operación de su comercializadora.				X
La navegación del software es sencilla.			X	
La apariencia del software es estética y agradable.			X	
La operación del software requirió una capacitación extensa.		X		
El software presenta errores mientras se opera	X			
Los tiempos en los cuales opera el software son rápidos.			X	

Nota: Elaboración propia.

6. Conclusiones y Trabajo Futuro

- La arquitectura de microservicios se aprovechó para dar uso a las api's o para el consumo de cada servicio dentro del sistema de esta forma se logró desarrollar una primera versión de fácil acceso mantenimiento y escalabilidad debido a las tecnologías usadas para el desarrollo de dicho problema.
- Respecto al segundo objetivo, se lograron identificar determinadas necesidades que permitieron crear, con base en ellas, cuatro módulos que facilitaron la operación de comercializadoras de calzado frente a la gestión de pedidos y facturación, los cuales son Administrador, vendedor, proveedores y clientes.
- Para dar cumplimiento al segundo objetivo propuesto, se trabajó bajo la luz de los parámetros de la metodología Scrum, a través de la cual se definieron actividades, requisitos e interacciones que permitieron el logro del mismo.
- Los demás objetivos se lograron gracias a la creación de una plataforma web que permite gestionar la toma de pedidos asociados a la operación de una comercializadora de calzado, ya sea en el mismo lugar o a domicilio, esto a través de los módulos mencionados anteriormente.
- Se logra la creación de un software funcional e integral que cumple con el último objetivo propuesto, facilitando no solo la gestión de pedidos y facturas en el sector comercial del calzado, sino que, además, logra operar con eficiencia y rapidez en la ejecución de las actividades y desarrollo amigable con el usuario.

Referencias Bibliográficas

Adapptative. (2018). Tipos de aplicaciones móviles o apps. https://adapptative.com/tipos-de aplicaciones-moviles-o-apps/

Bautista, M. (2009). Caracterización competitiva de las empresas exportadoras del sector calzado en Bucaramanga y su área metropolitana. https://repository.upb.edu.co/bitstream/handle/20.500.11912/861/digital_19196.pdf

Cordova. Overview. https://cordova.apache.org/docs/en/11.x/guide/overview/

Deloitte. ¿Qué es un OMR? https://www2.deloitte.com/es/es/pages/technology/articles/que-es orm.html

De la Torre, C., Wagner, B. & Rousos, M. .NET Microservices: Architecture for Containerized .NET Applications. https://docs.microsoft.com/es-es/dotnet/architecture/microservices/

Fielding, R., & Reschke, J. (2014). Internet Engineering Task Force (IETF). https://datatracker.ietf.org/doc/html/rfc7231#section-4

IMEBU. (2010). Industria del calzado y su visualización internacional. http://santandercompetitivo.org/media/fe3fd42cd1377168f726a3e07091c3293caab0d6.pdf

La República. (15 de octubre de 2020). Acicam reportó que el sector del calzado y el cuero ha perdido cerca de $300.000 millones. https://www.larepublica.co/empresas/acicam-reporto-que-el-sector-del-calzado-y-el-cuero-ha-perdido-cerca-de-300000-millones-3073939

Maida, E. & Pacienzia, J. (2015. Metodologías de desarrollo de software.

https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

Métodos de petición HTTP. https://developer.mozilla.org/es/docs/Web/HTTP/Methods

Moore, K. (2022). What Is the Footwear Industry? https://www.infobloom.com/what-is-the footwear-industry.htm