Universidad de Monterrey



Bases de Datos Avanzadas

Proyecto Final

**Sistema de autogestionamiento para microempresas de la comercialización de calzado**

**Maestro:** Raúl Morales Salcedo

**Equipo:** 3

**Integrantes:**

Margarita Concepción Cuervo Citalán #581771

Pedro Soto Barboza #658298

**Carrera:** ITC, 5to Semestre

**Día y hora del grupo de la clase:** lunes y jueves 08:30h

24 de noviembre del 2023.

*Doy mi palabra de haber realizado esta actividad con integridad académica.*

**Índice**

Introducción ………………………………………………………………………………………… 02

Sector ……………………………………………………………………………………………….. 02

Análisis del problema ………………………………………………………………………………. 03

Objetivo general y objetivos particulares …………………………………………………………… 04

Detalles y requerimientos …………………………………………………………………………… 05

Funcionalidades ……………………………………………………………………………………... 07

Flujograma de la aplicación desarrollada ……………………………………………………………. 09

Diseño E-R inicial …………………...………………………………………………………………. 11

Pasos de normalización hasta 3FN …………………………………….……………………………. 11

Diseño E-R final ……………………………..……...…………………….………………………… 21

Diccionario de datos ………………………………………………………………………………… 23

Creación de Base de datos …………………….…………………………………………………….. 23

Datos Iniciales ………………………….………….………………………………………………… 24

Sentencias SQL y su explicación …………….……………………………………………………… 26

Procedimientos almacenados y su explicación ……………………………………………………… 34

Comandos para replicar la instancia ………………………………………………………………… 63

Descripción de la aplicación desarrollada ………… .……………………………………………….. 64

Screenshots …………………………………………………………………..……………………… 65

Conclusiones ………………………………………………………………………………………… 76

Video de demostración (10 min máx) ………………………………………………………………. 77

Referencias ………………………………………………………………………………………….. 78

**Introducción**

El presente documento tiene como objetivo explicar de manera detallada los pormenores del proyecto a desarrollar titulado ‘*Sistema de autogestionamiento para microempresas de la comercialización de calzado’* hecho por nosotros, Pedro y Margarita, con finalidad de que no se tenga duda acerca de los pasos llevados a cabo durante el desarrollo del mismo y que se cuente con la información requerida para entender el propósito de la realización de este proyecto.

Por tanto, estos pormenores incluyen el sector al que está enfocado el micro negocio, el análisis del problema a desarrollar, el objetivo general y sus objetivos particulares, los detalles y requerimientos tomados en consideración. Después, se pasa a explicar las funcionalidades de la aplicación, su flujograma una vez desarrollada. En cuanto a la base de datos, se mostrará el diseño E-R inicial, los datos iniciales insertados en la base datos, los pasos de la normalización hasta la 3FN de las tablas lo cual conlleva al diagrama E-R final, su diccionario de datos, la creación de la base de datos, las sentencias SQL y su explicación, sus procedimientos almacenados y su explicación. Una vez terminado con todo lo relacionado a la base de datos, se pasa de nuevo a la descripción de la aplicación Flask desarrollada en la nube de Google, screenshots de como se ve esta una vez finalizada y un video de demostración así como las conclusiones a las que llegamos como equipo tras dar por concluido el proyecto.

**Sector**

Identificar el giro al que se dedica nuestra microempresa y entender cómo funciona el sector en el que este opera es crucial a la hora de tomar decisiones en cualquier proyecto que se les desarrolle. Por ende, es pertinente decir que el sector al que se dedica el micronegocio del cual estaremos desarrollando nuestro proyecto de gestión de micronegocios es el sector de comercio, más específicamente de la industria del calzado. De acuerdo a la Organización Internacional del Trabajo (OIT) (2023), el sector del comercio, o servicios de distribución, es un sector de la economía que tiene un fuerte círculo entre los productores de los bienes y sus consumidores, por lo que tanto empresas grandes como pequeñas pueden dedicarse a este sector. Por tanto, se puede decir que las empresas o negocios que se encuentran dentro del sector del comercio se dedican exclusivamente a la compra y venta de productos.

En este caso, la industria del calzado es el conjunto de actividades de diseño, fabricación, distribución, comercialización, y venta de todo tipo de calzado para el pie. De acuerdo con la Secretaría de Economía de México (2015), la industria del calzado es una importante actividad comercial en nuestro país, la cual genera una cadena de proveeduría altamente competitiva, al ver como esta industria puede agruparse por segmentos de productos muy diversificados al ver que abarca también una gran variedad de materiales y consumidores como hombres, mujeres, niños, deportistas de alto rendimiento, personas con diabetes o que ocupan calzado especial, etc. Además esta diversidad de productos finales no es sino fiel reflejo de la multitud de procesos industriales, empresas y estructuras de mercado existentes.

**Análisis del problema**

Debido a que al negocio al que le estaremos haciendo una aplicación para que éste pueda autogestionarse se dedica a la comercialización dentro de la industria del calzado, es importante saber más acerca de los retos y tendencias que este sector ha tenido que atravesar durante los últimos años y que siguen perdurando, sobre todo tras la pandemia del COVID-19, para que con ello podemos tener una perspectiva más informada y detallada del panorama actual.

Por tanto, de acuerdo a Dussel E. Pérez L. (2020), explican que tras la recaída de la economía en el 2020 debido al COVID-19, las condiciones actuales internas a las que se enfrenta la la cadena global del valor del calzado (CVC) México, se concentran en concretamente dos apartados: la creación del Plan de Transformación de la Industria, en base a los resultados de la consultoría de Roland Berger para la La Cámara de la Industria del Calzado del Estado de Guanajuato (CICEG) en 2019, y el Decreto de 2019. Ambos establecen condiciones importantes para la comprensión de la CVC en México e incluso antes de la pandemia internacional.

Entre los principales resultados plasmados en el Plan de Transformación de la Industria (2019) sobresalen:

1. México se encuentra entre los 10 principales productores a nivel mundial y con una orientación hacia el mercado nacional (90% de su producción).
2. El sector calzado mexicano ha ido creciendo en los últimos años más en valor (5%) que en volumen (4%) y refleja una evolución positiva en sus precios.
3. El segmento de la CVC con mayor crecimiento es el calzado para dama con aproximadamente 9% en volumen y 8% en valor, mientras el calzado deportivo y casual siguen ganando cuota en comparación con los estilos más tradicionales. Asimismo, se ve una tendencia de la venta del calzado de niños debido a que la población de México es mayoritariamente de adultos de la tercera edad.
4. Los canales también se distinguen por los mark ups (diferencia entre el precio final y el precio de fábrica) aplicados, que reflejan las distintas estrategias comerciales e influyen en la capacidad de venta y negociación.
5. En la CVC en México destaca el canal minoristas tradicionales, que se mantiene como el canal más relevante para el calzado (53%). A pesar de representar el mayor canal de consumo, el retail presenta una baja concentración de actores a diferencia de los canales E-commerce, departamentales y catálogos.

Asimismo se encontraron diversos retos en diversas áreas que enfrentan las empresas dentro de la industria del calzado, y como estas en cierta forma afectan a nuestro cliente, las cuales se enlistan a continuación:

1. **Proveeduría:** 
   1. El total de los productores consultados consideran la proveeduría como un desafío del sector. Aproximadamente 30% de los consultados de los fabricantes están integrados hacia la proveeduría (hacia abajo). La proveeduría es pieza fundamental en la producción, dada la complejidad de diferentes componentes y modelos, por ejemplo, se suelen requerir más de 10 proveedores por par.
   2. Las principales consecuencias de la difícil relación con la proveeduría es el bajo nivel de servicio, la acumulación de stock, el aumento de ineficiencia y de costos; lo que plantea la necesidad de buscar alternativas.
2. **Diseño y desarrollo (D&D):** 
   1. No existe una clara estrategia de identificación de tendencias en el mercado y consumidores, por ende, existe una falta de agilidad en aterrizar las necesidades del mercado.
3. **Conocimiento del consumidor:** 
   1. Poca información sobre las preferencias y el comportamiento del consumidor final. Se estima una baja inversión en mercadeo y publicidad (aproximadamente 1% de las ventas).
   2. Falta de competencias de marketing, en un ambiente cada vez más competitivo y globalizado algunos fabricantes están perdiendo fuerza por falta de mercadotecnia.
4. **Innovación y tecnología:** 
   1. La mayoría de las empresas apuestan por invertir en sistemas de diseño, sin embargo, existe poca inversión en sistemas de análisis de mercado y conocimiento del consumidor que impactan la innovación.
5. **Fragmentación del sector:** 
   1. La fragmentación y predominancia de pequeñas empresas influye en la baja capacidad para influir en la movilización del sector. Más del 50% de las empresas consultadas identificó la cultura empresarial como poco unida y colaborativa marcada por un cierto nivel de desconfianza y en ocasiones con actitudes sin ética empresarial.
   2. Bajo poder de negociación con los grandes compradores debido a su pequeña escala.
6. **E-Commerce y gestión de canales de distribución:** 
   1. Gestión de canales poco desarrollada. Existe una amplia gama de tipos de canales (minoristas, retail, departamentales, catálogos y comercio electrónico principalmente) con características específicas y con los que es posible definir diferentes mezclas. Existe una gran concentración de uso de terceros para la distribución del producto (aproximadamente 80%).
   2. Lento desarrollo en el canal de comercio electrónico. Cuando se compara con sus pares y referencias, se identifica una oportunidad de penetración del calzado mexicano en E-commerce.

**Objetivo general y objetivos particulares**

Por tanto, tras observar los retos a los que se enfrenta nuestro micronegocio, queremos establecer los siguientes objetivos que alcanzaremos con la realización del proyecto

**Objetivo general:** Brindar a cualquier microempresa dentro del sector de la comercialización de calzado, la capacidad de poder autogestionar sus operaciones de compra y venta al proporcionar una base de datos que permita realizar una manipulación de datos conforme a las operaciones CRUD y que la aplicación que haga uso de esta base de datos, sea funcional e intuitiva para los distintos usuarios que la usen.

**Objetivos específicos:**

* Que los usuarios tengan distintos niveles de acceso a la información para garantizar la integridad de la información contenida y que no pueda ser vulnerada tan fácilmente.
* Hacer el diseño de la base de datos conforme a la normalización de sus tablas para corregir datos duplicados y anomalías, evitar la creación y actualización de relaciones y dependencias no deseadas entre datos. prevenir borrados indeseados de datos y optimizar el espacio de almacenamiento.
* Haciendo uso de los procedimientos almacenados, los usuarios por medio de la interfaz de la aplicación, serán capaces de acceder cuantas veces lo requieran a información relevante del negocio de forma que las consultas puedan ser realizadas de forma más rápida y eficaz.
* Que la interfaz de la aplicación a desarrollar sea fácil, intuitiva de usar sin dejar de lado que esta tendrá una estética adecuada pero llamativa de acuerdo a los tipos de productos que ofrece el negocio.
* Que esta aplicación les permita a los administradores y empleados del negocio poder autogestionar su inventario de productos, compras, ventas, locales, proveedores, otros empleados y clientes.

**Detalles y requerimientos**

Antes de detallar los requerimientos del proyecto, se deben de tomar en cuenta ciertos detalles o características que conforman a las microempresas. Por ende, de acuerdo a BBVA (2023) y González J. C. (2023), en México, las microempresas son negocios que operan con menos de 10 trabajadores. Generalmente funcionan de forma local, ofreciendo sus productos y servicios a personas cercanas a su centro de operación.

De acuerdo con el Instituto Nacional de Estadística y Geografía, el 37% de las personas que trabajan en México están empleadas en una microempresa. Además, este tipo de emprendimiento aporta el 15% del Producto Interno Bruto, según la Secretaría de Economía.

Algunas de sus principales características son:

* Son negocios de entre 1 y 10 trabajadores.
* Dos Niveles Jerárquicos: La estructura organizativa de las microempresas se divide generalmente entre el dueño o jefe y los demás colaboradores. Aunque hay excepciones, esta jerarquía simplificada permite una toma de decisiones ágil y directa.
* Nicho de Mercado Definido: Las microempresas tienden a enfocarse en un nicho de mercado específico, lo que las lleva a operar principalmente a nivel local. Esto les permite conocer a sus clientes y atender sus necesidades de manera más personalizada.
* Cada año generan ganancias por máximo 4 millones de pesos.
* No tienen maquinaria para producir en masa, por lo cual hacen mayor uso de la mano de obra.
* Usualmente llevan su logística con ayuda de otras empresas, como plataformas de envío a domicilio o de asesoría legal.
* Necesitan una menor cantidad de inversión que otros tipos de empresas para seguir creciendo si así lo desean.
* Son adaptables pues tienen facilidad para cambiar el rumbo del negocio en caso de necesitarlo.
* Entre sus áreas de oportunidad están los mecanismos de control de calidad, los sueldos para sus empleados y la dificultad para llegar a otras partes del mundo.
* Asimismo las microempresas tienes 3 subcategorías principales
  + Supervivencia: Estas microempresas surgen por necesidad económica. No cuentan con un presupuesto, inversión, financiamiento o capital base. Se caracterizan por una producción inmediata y limitada. Un ejemplo sería alguien que hace y vende pasteles desde su hogar.
  + Expansión: A diferencia de las microempresas de supervivencia, este tipo cuenta con recursos para crecer gradualmente. Pueden contratar más personal y expandir sus operaciones a medida que generan ingresos. Por ejemplo, alguien que puede delegar la tarea de hacer pasteles a un empleado.
  + Transformación: Este tipo de microempresa experimenta un crecimiento significativo en facturación y comienza a acumular capital. Pueden expandir sus operaciones y diversificar su oferta. Un ejemplo sería alguien que alquila un local para abrir una pastelería y contrata un equipo para ofrecer una variedad de pasteles.

En el caso del punto anterior, nuestra microempresa pertenece al sector de transformación ya que al desarrollar está aplicación, está puede expandir sus operaciones y diversificar su oferta.

Por tanto, ahora tras saber más acerca de qué compone una microempresa en México y los antecedentes del problema, podemos pasar a detallar los requerimientos del proyecto por medio del uso de historias de usuario:

* Base de datos:
  + Como administrador y empleado necesito realizar de manera sencilla, rápida consultas a la información almacenada de la base de datos como productos, stock, y precios de forma que estas se puedan hacer de manera recurrente. Para ello se hará uso de stored procedures para hacer todas estas distintas consultas.
  + Como administrador necesito que la aplicación tenga la capacidad de visualizar los distintos locales, empleados, productos, proveedores, ventas, clientes y categorías de productos para poder expandir y gestionar de manera más sencilla las operaciones de puntos de venta físicos o capacidad de inventario entre los distintos productos que tengo.
  + Como administrador y empleado necesito que se puedan ver cuantas unidades quedan en stock de los productos ofrecidos para evitar la acumulación de stock, reducir el bajo nivel del servicio brindado, la ineficiencia y los costos de mantener este stock.
  + Como administrador necesito que el diseño de las tablas de la base de datos tenga la capacidad de expandirse para que si en un futuro el tamaño de esta microempresa crece, est aplicación siga siendo funcional y no tenga que invertir más dinero en mejorar esta base de datos que usará la aplicación o en el peor de los casos, pagar por que me desarrollen otra.
* Aplicación:
  + Como dueño o administrador necesito que la base de datos y la aplicación sea capaz de crear diferentes niveles de acceso para el usuario administrador y el usuario empleado para que la información sensible que está pueda contener no sea visible ni editable por todos, sino solamente al administrador y así poder garantizar la integridad de los mismos.
  + Como administrador y empleado necesito que el diseño de esta aplicación desarrollada sea atractiva visualmente, sea intuitiva de usar y accesible en cuanto a costos, para fomentar el comercio en línea de los productos que ofrecemos al cliente y tener un mejor autogestionamiento de nuestro inventario, ventas, pedidos, locales, empleados, proveedores y clientes.
  + Como administrador necesito que con la información recopilada a través de esta aplicación, me sea más fácil saber cuánto fueron las ventas totales por mes y año de cada local que tenga.
  + Como administrador y empleado necesito que esta aplicación se pueda acceder desde el navegador para que me sea fácil acceder a esta desde cualquier dispositivo.
  + Como administrador necesito tener la capacidad de quitar derechos de acceso a la aplicación creada para evitar que empleados que ya no trabajen para la empresa puedan acceder a la información.

**Funcionalidades**

Una vez detallados los requerimientos, pasamos a explicar cuáles serán las funcionalidades que tendrían

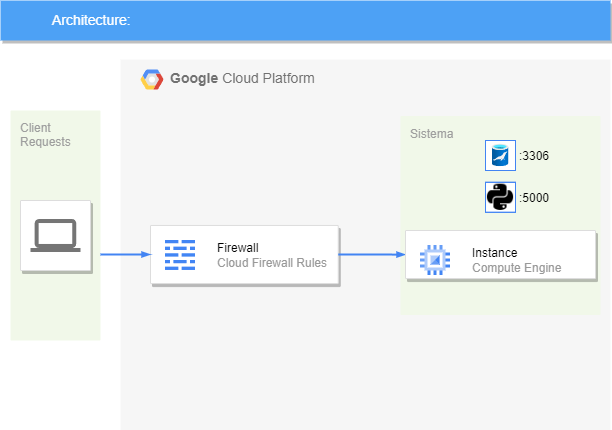
**Base de datos:**

* Su arquitectura de software es casi igual a la arquitectura de google gcp
* Hay distintos SP para diferentes tablas, pero en general todas nos permiten encontrar datos por id usando parámetros de entrada, insertar, editar o eliminar registros de esas tablas, activar o desactivar ciertos artículos o usuarios. Del mismo modo, nos permiten hacer gráficos que analicen las ventas totales por mes y año de han hecho cada uno de los locales, por lo que todos los SP son de grana ayuda para hacer consultas de la información de manera más sencilla, rápida, con menor ocupación de almacenamiento de los datos para posteriormente, estos datos y gráficas puedan ser analizados a la conveniencia del negocio.
* Para validar el funcionamiento de la Base de datos y los SP, se mandaron a llamar estos haciendo diversas pruebas con varios datos válidos e inválidos. En caso de detectar errores, estos se corrigen.

**Aplicación:**

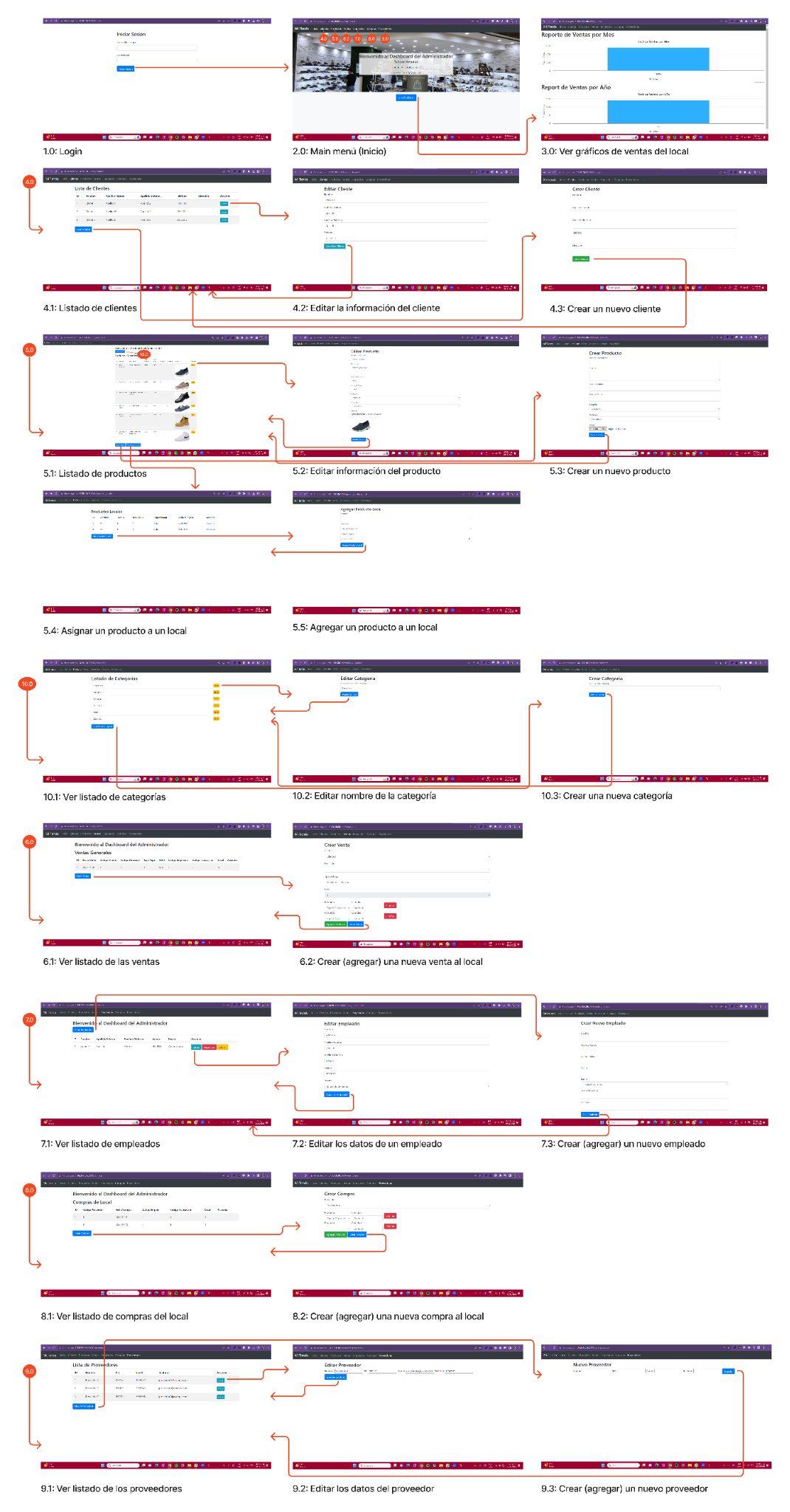
* Su arquitectura de software es casi igual a la arquitectura de google gcp
* Para acceder a esta app, el usuario se registra con su mail y contraseña en el login y para salir, se añade al url de la aplicación ‘/logout’.
* Los comportamientos de cada acción que se puede realizar con los productos, categorías, proveedores, empleados, clientes, ventas y compras van de acuerdo a los funcionamientos de los SP que están en el código fuente de la app, que hace que funcione la app.
* Para validar el correcto funcionamiento, si la acción que el usuario quiere realizar no se puede ejecutar, no se despliega el mensaje de cambio, eliminación o agregación exitosa.

**Arquitectura:**



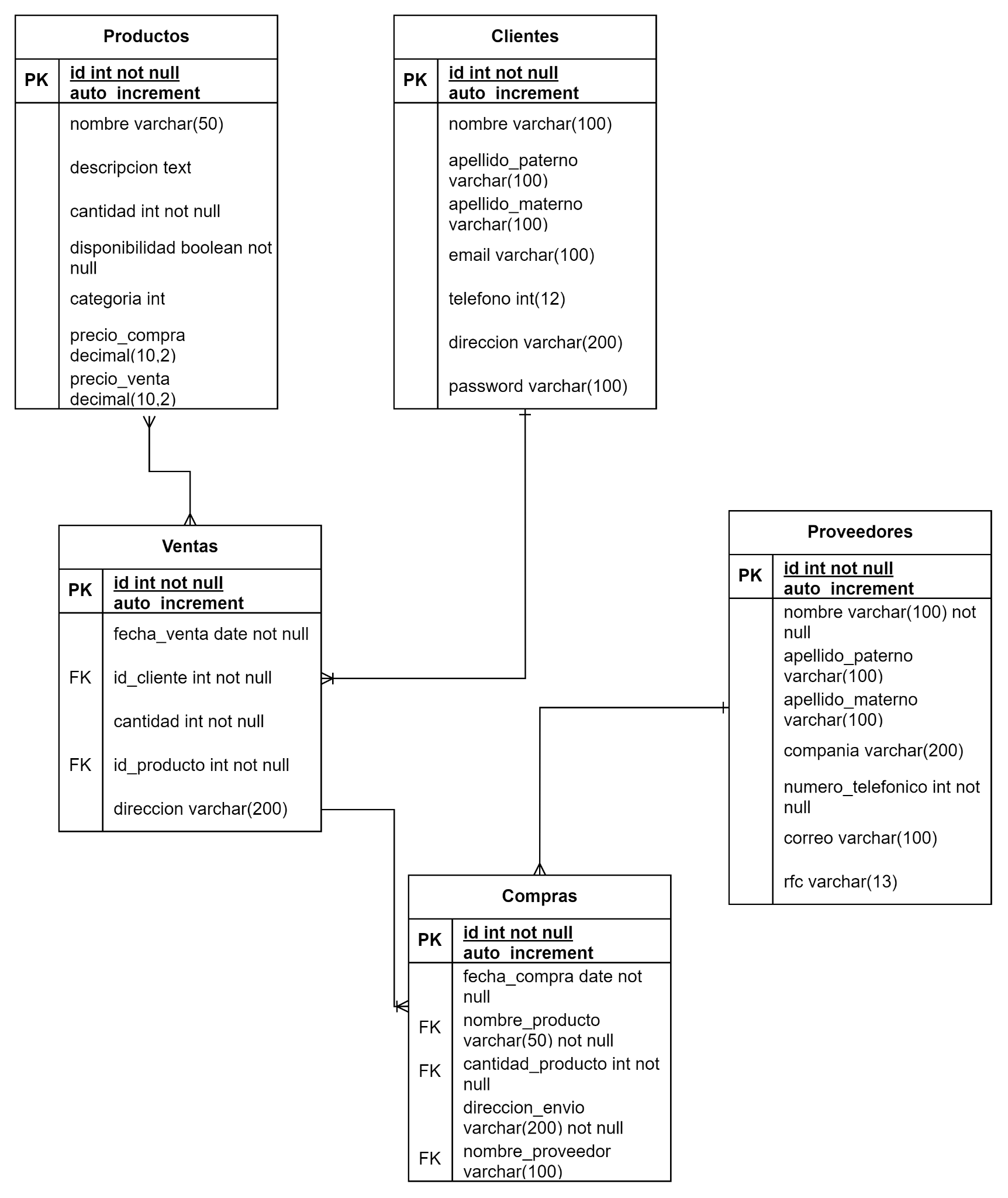
**Flujograma de la aplicación desarrollada**

A continuación, se muestra la foto del flujograma de la aplicación desarrollada, en donde muestra la interacción entre los distintos botones y pestañas de la aplicación. También, se anexa el pdf de este flujograma en el archivo comprimido del proyecto.



**Diseño E-R inicial**

Tras leer las instrucciones dadas en blackboard, se pasa al modelo E-R de la base de datos, que establece la creación de las tablas originales que identificamos pertinentes incluir en la base de datos de nuestro micronegocio con sus respectivos atributos, tipos de datos, y relaciones entre las tablas. Por tanto, el modelo E-R inicial es el siguiente:



**Pasos de normalización hasta 3FN**

Una vez que se tiene el diagrama E-R inicial, pasamos a la normalización de las tablas para corregir la relación varios a varios entre productos y ventas, así como para poder identificar maps tablas que se requieran hacer tras identificar las necesidades del cliente en torno al sector al que pertenece y los retos como lo rodean, así como para saber si la información contenida cuenta con las tres reglas de normalización: atomicidad de los datos, crear tablas independientes para conjuntos de valores que se apliquen a varios registros al relacionar estas tablas con una clave externa y eliminar dependencias que no dependan de la clave. Por tanto para hacer la normalización hasta la 3FN, se usó una hoja de Excel la cuál se explica a continuación:

**Tablas originales:** En este apartado, las tablas que se muestran a continuación en la hoja de Excel, son las tablas con las que se hizo el diagrama E-R inicial.

| ***Productos*** |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **nombre** | **descripción** | **cantidad** | **disponibilidad** | **categoria** | **precio\_compra** | **precio\_venta** |
| 1 | Zapato Deportivo | Perfectos para correr | 20 | VERDADERO | Deportivos | 50 | 100 |
| 2 | Zapato Casual | Ideal para el uso diario | 15 | VERDADERO | Casuales | 40 | 80 |
| 3 | Zapato Formal | Elegantes para ocasiones especiales | 18 | VERDADERO | Formales | 60 | 120 |
| 4 | Sandalias Verano | Cómodas y frescas | 25 | VERDADERO | Sandalías | 30 | 60 |
| 5 | Botas de Invierno | Calientes y resistentes al agua | 22 | VERDADERO | Botas | 70 | 140 |
| 6 | Zapatillas para Correr | Diseñadas para atletas profesionales | 30 | VERDADERO | Zapatillas | 80 | 160 |

| ***Clientes*** |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **nombre** | **apellido\_paterno** | **appelido\_materno** | **email** | **telefono** | **direccion** | **password** |
| 1 | Cliente 1 | Apellido 1 | Apellido 2 | adminA@example.com | 11111111 | Dirección Cliente 1 | 123 |
| 2 | Cliente 2 | Apellido 3 | Apellido 4 | adminB@example.com | 22222222 | Dirección Cliente 2 | 123 |
| 3 | Cliente 3 | Apellido 5 | Apellido 6 | adminC@example.com | 33333333 | Dirección Cliente 3 | 123 |

| ***Ventas*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **fecha\_venta** | **id\_cliente** | **cantidad** | **id\_producto** | **direccion** |
| 1 | 2023-11-01 | 1 | 2 | 1 | Direccion Cliente 1 |
| 2 | 2023-11-02 | 2 | 2 | 2 | Direccion Cliente 2 |
| 3 | 2023-11-03 | 3 | 2 | 3 | Direccion Cliente 3 |

| ***Compras*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **fecha\_compra** | **nombre\_producto** | **cantidad\_producto** | **direccion\_envio** | **nombre\_proveedor** |
| 1 | 2023-11-01 | Zapato Deportivo | 20 | Dirección Cliente 1 | Proovedor A |
| 2 | 2023-11-02 | Zapato Casual | 15 | Dirección Cliente 2 | Proovedor B |
| 3 | 2023-11-03 | Zapato Formal | 18 | Dirección Cliente 3 | Proovedor C |
| 4 | 2023-10-10 | Sandalias Verano | 25 | Dirección Cliente 1 | Proovedor A |
| 5 | 2023-10-15 | Botas de Invierno | 22 | Dirección Cliente 2 | Proovedor B |
| 6 | 2023-10-20 | Zapatillas para Correr | 30 | Dirección Cliente 3 | Proovedor C |

| ***Proveedores*** |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **nombre** | **apellido\_paterno** | **appelido\_materno** | **compania** | **numero\_telefónico** | **correo** | **rfc** |
| 1 | Proovedor A | Apellido 1 | Apellido 2 | Elmo's Shoes | 11111111 | proveedorA@example.com | 1234567 |
| 2 | Proovedor B | Apellido 3 | Apellido 4 | Zapateria Javier | 22222222 | proveedorB@example.com | 9876540 |
| 3 | Proovedor C | Apellido 5 | Apellido 6 | Charro Zapateria | 33333333 | proveedorC@example.com | 5555555 |

**1era Forma Normal: (1FN):** Una vez que se tienen las tablas originales se identificó que los datos ya son atómicos (separados por valores singulares), en todas las tablas, sin embargo también se notó que:

* El email y el password no depende del id de la tabla Clientes
* La categoría no depende del id de la tabla Productos
* Se hizo la creación de dos tablas: Categorías y Usuarios.

| ***Productos*** |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **nombre\_producto** | **descripción** | **cantidad** | **disponibilidad** | **categoria\_id** | **precio\_compra** | **precio\_venta** |
| 1 | Zapato Deportivo | Perfectos para correr | 20 | VERDADERO | 1 | 50 | 100 |
| 2 | Zapato Casual | Ideal para el uso diario | 15 | VERDADERO | 2 | 40 | 80 |
| 3 | Zapato Formal | Elegantes para ocasiones especiales | 18 | VERDADERO | 3 | 60 | 120 |
| 4 | Sandalias Verano | Cómodas y frescas | 25 | VERDADERO | 4 | 30 | 60 |
| 5 | Botas de Invierno | Calientes y resistentes al agua | 22 | VERDADERO | 5 | 70 | 140 |
| 6 | Zapatillas para Correr | Diseñadas para atletas profesionales | 30 | VERDADERO | 6 | 80 | 160 |

| ***Categorias*** |  |
| --- | --- |
| **id** | **nombre\_categoria** |
| 1 | Deportivos |
| 2 | Casuales |
| 3 | Formales |
| 4 | Sandalías |
| 5 | Botas |
| 6 | Zapatillas |

| ***Clientes*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **nombre** | **apellido\_paterno** | **appelido\_materno** | **telefono** | **direccion** |
| 1 | Cliente 1 | Apellido 1 | Apellido 2 | 11111111 | Dirección Cliente 1 |
| 2 | Cliente 2 | Apellido 3 | Apellido 4 | 22222222 | Dirección Cliente 2 |
| 3 | Cliente 3 | Apellido 5 | Apellido 6 | 33333333 | Dirección Cliente 3 |

| ***Usuarios*** |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **email** | **nombre** | **password** | **activo** |
| 1 | adminA@example.com | Cliente 1 | 123 | VERDADERO |
| 2 | adminB@example.com | Cliente 2 | 123 | VERDADERO |
| 3 | adminC@example.com | Cliente 3 | 123 | VERDADERO |

**2nda Forma Normal (2FN):** En este apartado, se identificaron varias cosas como:

* La dirección del cliente no depende de la tabla Clientes, ni de las tablas Compras o Ventas
* Debido al punto anterior, se creó la tabla Direcciones
* Se asignan id a todas las tablas para garantizar su unicidad
* Eliminamos y/o cambiamos nombres de los datos en las tablas Proveedores, Clientes, Ventas y Compras

| ***Direcciones\_clientes*** |  |  |
| --- | --- | --- |
| **id** | **direccion** | **cliente\_id** |
| 1 | Direccion Cliente 1 | 1 |
| 2 | Direccion Cliente 2 | 2 |
| 3 | Direccion Cliente 3 | 3 |

| ***Clientes*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **nombre** | **apellido\_paterno** | **appelido\_materno** | **telefono** | **direccion\_id** |
| 1 | Cliente 1 | Apellido 1 | Apellido 2 | 11111111 | 1 |
| 2 | Cliente 2 | Apellido 3 | Apellido 4 | 22222222 | 2 |
| 3 | Cliente 3 | Apellido 5 | Apellido 6 | 33333333 | 3 |

| ***Ventas*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **fecha\_venta** | **cliente\_id** | **cantidad** | **producto\_id** | **direccion\_id** |
| 1 | 2023-11-01 | 1 | 2 | 1 | 1 |
| 2 | 2023-11-02 | 2 | 2 | 2 | 2 |
| 3 | 2023-11-03 | 3 | 2 | 3 | 3 |

| ***Compras*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **fecha\_compra** | **producto\_id** | **cantidad** | **direccion\_id** | **proveedor\_id** |
| 1 | 2023-11-01 | 1 | 20 | 1 | 1 |
| 2 | 2023-11-02 | 2 | 15 | 2 | 2 |
| 3 | 2023-11-03 | 3 | 18 | 3 | 3 |
| 4 | 2023-10-10 | 4 | 25 | 1 | 1 |
| 5 | 2023-10-15 | 5 | 22 | 2 | 2 |
| 6 | 2023-10-20 | 6 | 30 | 3 | 3 |

| ***Proveedores*** |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **nombre** | **rfc** | **numero\_telefónico** | **correo** |
| 1 | Proovedor A | 1234567 | 11111111 | proveedorA@example.com |
| 2 | Proovedor B | 9876540 | 22222222 | proveedorB@example.com |
| 3 | Proovedor C | 5555555 | 33333333 | proveedorC@example.com |

**3era Forma Normal (3FN):** Por último, en este apartado se identifican las siguientes cosas:

* Se determina cuáles columnas que no son llave no dependen de la llave primaria de la tabla o no tienen llave primaria
* Se eliminan esas columnas de la tabla base
* Se crea una nueva tabla con esas columnas. Asimismo, se consideran otros factores que implican la gestión de un micronegocio como empleados, tipos de pagos, locales y transacciones, etc. y se realizan sus tablas.
* Eliminamos y/o cambiamos nombres de los datos en las tablas
* Se asignan id a todas las tablas para garantizar su unicidad

| **15 tablas en la 3FN** |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Productos** | id | nombre\_producto | descripcion | categoria\_id | precio\_compra | precio\_venta |  |  |  |
| **Categorias** | id | nombre\_categoria |  |  |  |  |  |  |  |
| **Clientes** | id | nombre | apellido\_paterno | apellido\_materno | telefono |  |  |  |  |
| **Usuarios** | id | email | password | rol | activo |  |  |  |  |
| **Ventas** | id | fecha\_venta | cliente\_id | direccion\_id | tipo\_pago\_id | total\_venta | empleado\_id | transaccion\_id | local\_id |
| **Detalles\_Ventas** | id | venta\_id | cantidad\_producto | producto\_id |  |  |  |  |  |
| **Direcciones\_Clientes** | id | direccion | cliente\_id |  |  |  |  |  |  |
| **Transacciones** | id | tipo\_transaccion | monto | fecha\_transaccion | referencia | usuario\_id |  |  |  |
| **Compras** | id | proveedor\_id | fecha\_compra | empleado\_id | transaccion\_id | local\_id |  |  |  |
| **Productos\_Locales** | id | cantidad | local\_id | producto\_id | disponibilidad | fecha\_ingreso |  |  |  |
| **Locales** | id | nombre | direccion | imagen |  |  |  |  |  |
| **Empleados** | id | nombre | apellido\_paterno | apellido\_materno | salario | puesto | local\_id | usuario\_id |  |
| **Proveedores** | id | nombre | rfc | telefono | email |  |  |  |  |
| **Detalles\_Compras** | id | compra\_id | producto\_id | cantidad |  |  |  |  |  |
| **Tipos\_Pagos** | id | nombre |  |  |  |  |  |  |  |

| ***Productos*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **nombre\_producto** | **descripcion** | **categoria\_id** | **precio\_compra** | **precio\_venta** |
| 1 | Zapato Deportivo | Perfectos para correr | 1 | 50 | 100 |
| 2 | Zapato Casual | Ideal para el uso diario | 2 | 40 | 80 |
| 3 | Zapato Formal | Elegantes para ocasiones especiales | 3 | 60 | 120 |
| 4 | Sandalias Verano | Cómodas y frescas | 4 | 30 | 60 |
| 5 | Botas de Invierno | Calientes y resistentes al agua | 5 | 70 | 140 |
| 6 | Zapatillas para Correr | Diseñadas para atletas profesionales | 6 | 80 | 160 |

| ***Categorias*** |  |
| --- | --- |
| **id** | **nombre\_categoria** |
| 1 | Deportivos |
| 2 | Casuales |
| 3 | Formales |
| 4 | Sandalías |
| 5 | Botas |
| 6 | Zapatillas |

| ***Clientes*** |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **nombre** | **apellido\_paterno** | **appelido\_materno** | **telefono** |
| 1 | Cliente 1 | Apellido 1 | Apellido 2 | 11111111 |
| 2 | Cliente 2 | Apellido 3 | Apellido 4 | 22222222 |
| 3 | Cliente 3 | Apellido 5 | Apellido 6 | 33333333 |

| ***Usuarios*** |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **email** | **password** | **rol** | **activo** |
| 1 | adminA@example.com | 123 | admin | VERDADERO |
| 2 | adminB@example.com | 123 | admin | VERDADERO |
| 3 | adminC@example.com | 123 | admin | VERDADERO |

| ***Ventas*** |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **fecha\_venta** | **cliente\_id** | **direccion\_id** | **tipo\_pago\_id** | **total\_venta** | **empleado\_id** | **transaccion\_id** | **local\_id** |
| 1 | 2023-11-01 | 1 | 1 | 1 | 120 | 1 | 3 | 1 |
| 2 | 2023-11-02 | 2 | 2 | 2 | 160 | 2 | 4 | 2 |
| 3 | 2023-11-03 | 3 | 3 | 3 | 200 | 3 | 5 | 3 |

| ***Detalles\_Ventas*** |  |  |  |
| --- | --- | --- | --- |
| **id** | **venta\_id** | **cantidad\_producto** | **producto\_id** |
| 1 | 2 | 1 | 60 |
| 2 | 2 | 2 | 80 |
| 3 | 2 | 3 | 100 |

| ***Direcciones\_Clientes*** |  |  |
| --- | --- | --- |
| **id** | **direccion** | **cliente\_id** |
| 1 | Direccion Cliente 1 | 1 |
| 2 | Direccion Cliente 2 | 2 |
| 3 | Direccion Cliente 3 | 3 |

| **Transacciones** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **tipo\_transaccion** | **monto** | **fecha\_transaccion** | **referencia** | **usuario\_id** |
| 1 | Venta | 120 | 2023-11-01 | ReferenciaVenta1 | 1 |
| 2 | Venta | 160 | 2023-11-02 | ReferenciaVenta2 | 2 |
| 3 | Venta | 200 | 2023-11-03 | ReferenciaVenta3 | 3 |

| ***Compras*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **proveedor\_id** | **fecha\_compra** | **empleado\_id** | **transaccion\_id** | **local\_id** |
| 1 | 1 | 2023-11-01 | 1 | 1 | 1 |
| 2 | 2 | 2023-11-02 | 2 | 2 | 2 |
| 3 | 3 | 2023-11-03 | 3 | 3 | 3 |
| 4 | 1 | 2023-10-10 | 1 | 1 | 1 |
| 5 | 2 | 2023-10-15 | 2 | 2 | 2 |
| 6 | 3 | 2023-10-20 | 3 | 3 | 3 |

| ***Productos\_Locales*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **cantidad** | **local\_id** | **producto\_id** | **disponibilidad** | **fecha\_ingreso** |
| 1 | 20 | 1 | 1 | VERDADERO | 2023-10-01 |
| 2 | 15 | 1 | 2 | VERDADERO | 2023-10-01 |
| 3 | 18 | 2 | 3 | VERDADERO | 2023-10-02 |
| 4 | 25 | 2 | 4 | VERDADERO | 2023-10-02 |
| 5 | 22 | 3 | 5 | VERDADERO | 2023-10-03 |
| 6 | 30 | 3 | 6 | VERDADERO | 2023-10-03 |

| ***Locales*** |  |  |  |
| --- | --- | --- | --- |
| **id** | **nombre** | **direccion** | **imagen** |
| 1 | Tienda A | Calle Principal 123 | TiendaA.jpg |
| 2 | Tienda B | Avenida Central 456 | TiendaB.jpg |
| 3 | Tienda C | Plaza Comercial 789 | TiendaC.jpg |

| ***Empleados*** |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **nombre** | **apellido\_paterno** | **apellido\_materno** | **salario** | **puesto** | **local\_id** | **usuario\_id** |
| 1 | Admin A | Apellido | Paterno | 50000 | Administrador | 1 | 1 |
| 2 | Admin B | Apellido | Paterno | 50000 | Administrador | 2 | 2 |
| 3 | Admin C | Apellido | Paterno | 50000 | Administrador | 3 | 3 |

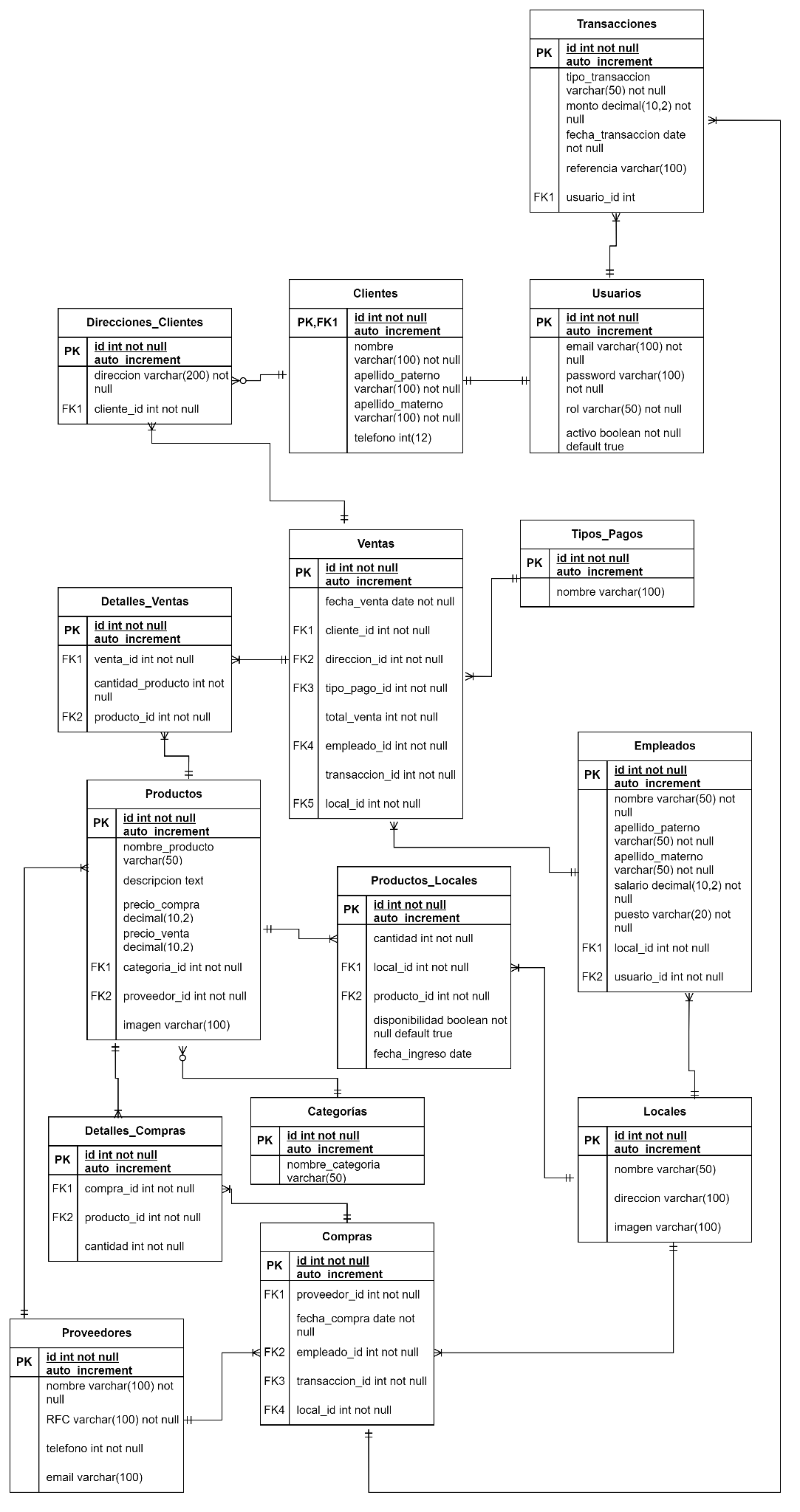
| ***Proveedores*** |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **nombre** | **rfc** | **telefono** | **email** |
| ***1*** | Proovedor A | RFC123 | 1234567 | proveedorA@example.com |
| 2 | Proovedor B | RFC456 | 9876540 | proveedorB@example.com |
| 3 | Proovedor C | RFC789 | 5555555 | proveedorC@example.com |

| ***Detalles\_Compras*** |  |  |  |
| --- | --- | --- | --- |
| **id** | **compra\_id** | **producto\_id** | **cantidad** |
| 1 | 1 | ***1*** | 20 |
| 2 | 2 | 3 | 18 |
| 3 | 3 | 5 | 22 |
| 4 | 4 | 2 | 15 |
| 5 | 5 | 4 | 25 |
| 6 | 6 | 6 | 30 |

| ***Tipos\_Pagos*** |  |
| --- | --- |
| **id** | **nombre** |
| 1 | Efectivo |
| 2 | Tarjetas de Crédito |
| 3 | Transferencia Bancaria |
| 4 | Pago Móvil |

**Diseño E-R final**

Una vez realizado, la normalización hasta la 3FN de las tablas de la base de datos del proyecto, se crea el modelo E-R final que establece la creación de las tablas normalizadas que resultaron en la base de datos de nuestro micronegocio con sus respectivos atributos, tipos de datos, y relaciones entre las tablas. Por tanto, el modelo E-R final es el siguiente:



**Diccionario de datos**

Ahora bien, cuando se terminó de realizar la 3FN de las tablas de la base de datos, al final del Excel se incluye un diccionario de datos que tiene la finalidad de dar precisión semántica sobre los datos que se manejan en la base de datos del sistema de de autogestionamiento para microempresas de la comercialización de calzado, evitando que puedan darse diferentes interpretaciones o ambigüedades de los datos. Dicho diccionario de datos, es el siguiente:

| **Diccionario de Datos** | |  | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Productos** | **precio\_compra =** precio al que se compró el producto del proveedor, **precio\_venta =** precio al que se ofrece el producto al cliente | | | | | | | | | | | | | | |
| **Usuarios** | **rol =** para saber si el usuario es administrador o empleado (y saber qué accesos se le otorgan), **activo =** para saber si el usuario aún pertenece a la empresa, en caso de que este ya esté jubilado, lo corrieron, o otros factores se desactiva (false) para que este ya no tenga acceso a la info (sirve más que nada dentro de la lógica de la app flask, ya que no tiene datos en el archivo sql) | | | | | | | | | | | | | | |
| **Transacciones** | **tipo\_transaccion =** saber si la transacción es de venta o compra, **referencia =** a que se asocia esta transacción. Puede ser 'Venta', 'Compra', 'Cancelación', 'En Proceso' o 'Reembolso'. | | | | | | | | | | | | | | |
| **Productos\_Locales** | **cantidad =** cantidad del producto, **disponibilidad =** verdadero si aún hay productos y falso si ya no hay en existencia, **fecha\_ingreso =** cuando se ingreso el producto al local en el sistema | | | | | | | | | | | | | | |
| **Locales** | **imagen =** saber el nombre del archivo de la imagen que tendrá cada local en la app | | | | | | | | | | | | | | |

**Creación de Base de datos**

Una vez terminado con la normalización de la base de datos, pasamos a la creación de la base de datos, para ello creamos un archivo sql llamado ‘schemaTienda.sql’ para guardar aquí toda la estructura de la base de datos en donde básicamente se encuentran las sentencias sql para la creación de la base de datos ‘Tienda’, las las 15 tablas que la conforman con sus respectivos atributos y tipos de datos y la creación del usuario administrador.

Luego en el archivo sql llamado ‘dataTienda.sql’ se guardaron todas las sentencias sql para insertar los datos respectivos de las tablas. Una vez teniendo estos dos archivos, se hizo el restore de estos desde la base de datos de MariaDB instalado en el SSH de la instancias virtual de Google Cloud.

Una vez, realizado el restore, se pasó a la creación de los procedimientos almacenados, los cuales antes de ser insertados en la base de datos de Google Cloud, también se creó un archivo sql llamado ‘storedProcedures.sql’. Dentro de este documento se encuentran los procedimientos para calcular, obtener o realizar lo siguiente:

* Insertar categorías, editar categoría, eliminar categoría, obtener categoría por id, obtener todas las categorías
* Insertar (agregar) un producto, editar un producto, eliminar un producto, encontrar producto por id o buscar todos los productos, obtener producto por id del proveedor, activar un producto
* Insertar usuario, editar usuario, eliminar usuario, obtener usuario por id o todos los usuarios
* Insertar cliente, editar cliente, eliminar cliente, obtener cliente por id o todos los clientes,
* Obtener todas las direcciones del cliente por id, insertar dirección del cliente o editarlo
* Insertar proveedor, editarlo o eliminarlo, obtener proveedor por id o todos los proveedores
* Insertar producto del local, obtener precio del producto, obtener productos de local por id del local, obtener productos disponibles por id del local, desactivar o activar producto del local, obtener producto local por id
* Obtener ventas por id del local, registrar venta, registrar detalle venta, editar venta, eliminar venta, obtener venta por id, obtener todas las ventas y también obtener todas las ventas por id del local
* Registrar compra, registrar detalle compra, obtener compras por id del local, editar compra, eliminar compra, obtener compra por id o todas las compras
* Insertar detalle de compra, editar detalle de compra, eliminar detalle de compra, obtener detalle de compra por id, obtener todos los detalles de compra
* Obtener empleado por id, obtener a los empleados por id del local, insertar empleado a un local, desactivar o activar a un empleado, editar a un empleado
* Obtener usuario por email
* Obtener local por id, obtener todos productos por id del local
* Obtener los tipos de pagos
* obtener ventas totales por cliente, obtener ventas totales por mes u obtener ventas totales por año

Una vez hechos todos estos procedimientos en el archivo sql, también se hizo el restore de estos a la base de datos. La explicación de todas estas sentencias sql y procedimientos almacenados se explicarán en apartados posteriores.

**Datos Iniciales**

Por otro lado, como se observó en el apartado de la normalización de las tablas de la base de datos, los datos iniciales corresponden a los datos insertados en las tablas originales, que se modelaron en el modelo E-R inicial. Estos datos iniciales fueron inventados por nosotros, Pedro y Margarita, con el propósito de asegurarnos que de esta forma se estuvieran cumpliendo las propiedades de la normalización en cada paso a seguir, siguiendo la lógica del tipo de datos que se insertan en cada columna así como tomando en consideración el giro del micronegocio, la comercialización de calzado. Por tanto, los datos iniciales son los siguientes:

| ***Productos*** |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **nombre** | **descripción** | **cantidad** | **disponibilidad** | **categoria** | **precio\_compra** | **precio\_venta** |
| 1 | Zapato Deportivo | Perfectos para correr | 20 | VERDADERO | Deportivos | 50 | 100 |
| 2 | Zapato Casual | Ideal para el uso diario | 15 | VERDADERO | Casuales | 40 | 80 |
| 3 | Zapato Formal | Elegantes para ocasiones especiales | 18 | VERDADERO | Formales | 60 | 120 |
| 4 | Sandalias Verano | Cómodas y frescas | 25 | VERDADERO | Sandalías | 30 | 60 |
| 5 | Botas de Invierno | Calientes y resistentes al agua | 22 | VERDADERO | Botas | 70 | 140 |
| 6 | Zapatillas para Correr | Diseñadas para atletas profesionales | 30 | VERDADERO | Zapatillas | 80 | 160 |

| ***Clientes*** |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **nombre** | **apellido\_paterno** | **appelido\_materno** | **email** | **telefono** | **direccion** | **password** |
| 1 | Cliente 1 | Apellido 1 | Apellido 2 | adminA@example.com | 11111111 | Dirección Cliente 1 | 123 |
| 2 | Cliente 2 | Apellido 3 | Apellido 4 | adminB@example.com | 22222222 | Dirección Cliente 2 | 123 |
| 3 | Cliente 3 | Apellido 5 | Apellido 6 | adminC@example.com | 33333333 | Dirección Cliente 3 | 123 |

| ***Ventas*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **fecha\_venta** | **id\_cliente** | **cantidad** | **id\_producto** | **direccion** |
| 1 | 2023-11-01 | 1 | 2 | 1 | Direccion Cliente 1 |
| 2 | 2023-11-02 | 2 | 2 | 2 | Direccion Cliente 2 |
| 3 | 2023-11-03 | 3 | 2 | 3 | Direccion Cliente 3 |

| ***Compras*** |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **fecha\_compra** | **nombre\_producto** | **cantidad\_producto** | **direccion\_envio** | **nombre\_proveedor** |
| 1 | 2023-11-01 | Zapato Deportivo | 20 | Dirección Cliente 1 | Proovedor A |
| 2 | 2023-11-02 | Zapato Casual | 15 | Dirección Cliente 2 | Proovedor B |
| 3 | 2023-11-03 | Zapato Formal | 18 | Dirección Cliente 3 | Proovedor C |
| 4 | 2023-10-10 | Sandalias Verano | 25 | Dirección Cliente 1 | Proovedor A |
| 5 | 2023-10-15 | Botas de Invierno | 22 | Dirección Cliente 2 | Proovedor B |
| 6 | 2023-10-20 | Zapatillas para Correr | 30 | Dirección Cliente 3 | Proovedor C |

| ***Proveedores*** |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **nombre** | **apellido\_paterno** | **appelido\_materno** | **compania** | **numero\_telefónico** | **correo** | **rfc** |
| 1 | Proovedor A | Apellido 1 | Apellido 2 | Elmo's Shoes | 11111111 | proveedorA@example.com | 1234567 |
| 2 | Proovedor B | Apellido 3 | Apellido 4 | Zapateria Javier | 22222222 | proveedorB@example.com | 9876540 |
| 3 | Proovedor C | Apellido 5 | Apellido 6 | Charro Zapateria | 33333333 | proveedorC@example.com | 5555555 |

**Sentencias SQL y su explicación**

Como bien se mencionó en el apartado de ‘Creación de la base de datos’, en esta sección se insertarán los las sentencias sql que pertenecen a los archivos del esquema y los datos de la base de datos con comentarios que expliquen qué significan y qué hace cada sentencia.

**schemaTienda.sql**

-- Se crea la base de datos Tienda si no existe

CREATE DATABASE IF NOT EXISTS Tienda;

-- Usamos la BD Tienda

USE Tienda;

--- Creación del usuario Administrador en el servidor local con su password

CREATE USER IF NOT EXISTS 'admin'@'localhost' IDENTIFIED BY '123';

-- Le damos todos todos los privilegios a este usuario administrador

GRANT ALL PRIVILEGES ON Tienda.\* TO 'admin'@'localhost' WITH GRANT OPTION;

-- Para desactivar las restricciones de las llaves primarias momentáneamente

SET FOREIGN\_KEY\_CHECKS = 0;

--- Creación de tablas

-- Tabla principal que almacena las categorías de los productos

CREATE TABLE Categorias (

id INT NOT NULL AUTO\_INCREMENT,

nombre\_categoria VARCHAR(50),

PRIMARY KEY (id)

);

-- Tabla que almacena los usuarios que se le asigna a cada cliente para registrarse al sistema

CREATE TABLE Usuarios (

id INT NOT NULL AUTO\_INCREMENT,

email VARCHAR(100) NOT NULL,

password VARCHAR(100) NOT NULL,

rol VARCHAR(50) NOT NULL,

activo BOOLEAN NOT NULL DEFAULT TRUE,

PRIMARY KEY (id)

);

-- Tabla que almacena a los clientes del negocio

CREATE TABLE Clientes (

id INT NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(100) NOT NULL,

apellido\_paterno VARCHAR(100) NOT NULL,

apellido\_materno VARCHAR(100) NOT NULL,

telefono INT(12) NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (id) REFERENCES Usuarios (id)

);

-- Tabla que almacena las diferentes direcciones que pueden guardar los clientes en el sistema

CREATE TABLE Direcciones\_Clientes (

id INT NOT NULL AUTO\_INCREMENT,

direccion VARCHAR(200) NOT NULL,

cliente\_id INT NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (cliente\_id) REFERENCES Clientes (id)

);

-- Tabla que almacena lso distintos tipos de pagos que se pueden aplicar en las ventas

CREATE TABLE Tipos\_Pagos(

id INT NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(100),

PRIMARY KEY (id)

);

-- Tabla que almacena las ventas del negocio (los pedidos de los clientes)

CREATE TABLE Ventas (

id INT NOT NULL AUTO\_INCREMENT,

fecha\_venta DATE NOT NULL,

cliente\_id INT NOT NULL,

direccion\_id INT NOT NULL,

tipo\_pago\_id INT NOT NULL,

total\_venta INT NOT NULL,

empleado\_id INT NOT NULL,

transaccion\_id INT NOT NULL,

local\_id INT NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (cliente\_id) REFERENCES Clientes (id),

FOREIGN KEY (direccion\_id) REFERENCES Direcciones\_Clientes (id),

FOREIGN KEY (tipo\_pago\_id) REFERENCES Tipos\_Pagos (id),

FOREIGN KEY (empleado\_id) REFERENCES Empleados(id),

FOREIGN KEY (local\_id) REFERENCES Locales (id)

);

-- Tabla que almacena los distintos proveedores de calzado que tiene el negocio

CREATE TABLE Proveedores (

id INT NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(100) NOT NULL,

RFC VARCHAR(100) NOT NULL,

telefono INT NOT NULL,

email VARCHAR(100),

PRIMARY KEY (id)

);

-- Tabla que almacena la información de los distintos locales que tiene el micronegocio

CREATE TABLE Locales (

id INT NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(50),

direccion VARCHAR(100),

imagen VARCHAR(100),

PRIMARY KEY (id)

);

-- Tabla que almacena los distintos productos (tipos de calzado) que ofrece el micronegocio

CREATE TABLE Productos (

id INT NOT NULL AUTO\_INCREMENT,

nombre\_producto VARCHAR(50),

descripcion TEXT,

precio\_compra DECIMAL (10,2),

precio\_venta DECIMAL (10,2),

categoria\_id INT NOT NULL,

proveedor\_id INT NOT NULL,

imagen VARCHAR(100),

PRIMARY KEY (id),

FOREIGN KEY (categoria\_id) REFERENCES Categorias (id),

FOREIGN KEY (proveedor\_id) REFERENCES Proveedores (id)

);

-- Tabla que almacena los detalles de las ventas realizadas

CREATE TABLE Detalles\_Ventas(

id INT NOT NULL AUTO\_INCREMENT,

venta\_id INT NOT NULL,

cantidad\_producto INT NOT NULL,

producto\_id INT NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (producto\_id) REFERENCES Productos (id),

FOREIGN KEY (venta\_id) REFERENCES Ventas(id)

);

-- Tabla que almacena el inventario que tiene cada uno de los locales del negocio

CREATE TABLE Productos\_Locales (

id INT NOT NULL AUTO\_INCREMENT,

cantidad INT NOT NULL,

local\_id INT NOT NULL,

producto\_id INT NOT NULL,

disponibilidad BOOLEAN NOT NULL DEFAULT TRUE,

fecha\_ingreso DATE,

PRIMARY KEY (id),

FOREIGN KEY (local\_id) REFERENCES Locales (id),

FOREIGN KEY (producto\_id) REFERENCES Productos (id)

);

-- Tabla que almacena los empleados que tiene el negocio repartidos en lso distintos locale

CREATE TABLE Empleados (

id INT NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(50) NOT NULL,

apellido\_paterno VARCHAR(50) NOT NULL,

apellido\_materno VARCHAR(50) NOT NULL,

salario DECIMAL (10,2) NOT NULL,

puesto VARCHAR(20) NOT NULL,

local\_id INT NOT NULL,

usuario\_id INT NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (local\_id) REFERENCES Locales (id),

FOREIGN KEY (usuario\_id) REFERENCES Usuarios (id)

);

-- Tabla principal para registrar transacciones financieras que se generan de las ventas y compras del negocio

CREATE TABLE Transacciones (

id INT NOT NULL AUTO\_INCREMENT,

tipo\_transaccion VARCHAR(50) NOT NULL, -- Puede ser 'Venta', 'Compra', 'Cancelación', 'En Proceso' o 'Reembolso'

monto DECIMAL(10,2) NOT NULL,

fecha\_transaccion DATE NOT NULL,

referencia VARCHAR(100),

usuario\_id INT, -- Nueva columna para asociar la transacción con un usuario

PRIMARY KEY (id),

FOREIGN KEY (usuario\_id) REFERENCES Usuarios (id)

);

-- Tabla que almacena las distintas compras (pedidos a los proveedores) que realiza el negocio

CREATE TABLE Compras (

id INT NOT NULL AUTO\_INCREMENT,

proveedor\_id INT NOT NULL,

fecha\_compra DATE NOT NULL,

empleado\_id INT NOT NULL,

transaccion\_id INT NOT NULL,

local\_id INT NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (proveedor\_id) REFERENCES Proveedores (id),

FOREIGN KEY (empleado\_id) REFERENCES Empleados (id),

FOREIGN KEY (transaccion\_id) REFERENCES Transacciones (id),

FOREIGN KEY (local\_id) REFERENCES Locales (id)

);

-- Tabla que almacena los detalles de las compras realizadas por el negocio

CREATE TABLE Detalles\_Compras (

id INT NOT NULL AUTO\_INCREMENT,

compra\_id INT NOT NULL,

producto\_id INT NOT NULL,

cantidad INT NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (compra\_id) REFERENCES Compras (id),

FOREIGN KEY (producto\_id) REFERENCES Productos (id)

);

-- Para activar de vuelta las restricciones de las llaves primarias

SET FOREIGN\_KEY\_CHECKS = 1;

**dataTienda.sql**

-- Usamos la BD Tienda

USE Tienda;

-- Para desactivar las restricciones de las llaves primarias momentáneamente

SET FOREIGN\_KEY\_CHECKS = 0;

-- Insertar datos para crear varios locales con administradores y usuarios ficticios

INSERT INTO Locales (nombre, direccion, imagen) VALUES

('Tienda A', 'Calle Principal 123', 'TiendaA.jpg'),

('Tienda B', 'Avenida Central 456', 'TiendaB.jpg'),

('Tienda C', 'Plaza Comercial 789', 'TiendaC.jpg');

-- Insertar datos para empleados ficticios, con un salario, puesto, local asignado y usuario

INSERT INTO Empleados (nombre, apellido\_paterno, apellido\_materno, salario, puesto, local\_id, usuario\_id)

VALUES

('Admin A', 'Apellido', 'Paterno', 50000.00, 'Administrador', 1,1),

('Admin B', 'Apellido', 'Paterno', 50000.00, 'Administrador', 2,2),

('Admin C', 'Apellido', 'Paterno', 50000.00, 'Administrador', 3,3);

-- Insertar datos para crear usuarios administradores ficticios

INSERT INTO Usuarios (email, password, rol) VALUES

('adminA@example.com', '123', 'admin'),

('adminB@example.com', '123', 'admin'),

('adminC@example.com', '123', 'admin');

-- Insertar datos para crear diversos tipos de pagos

INSERT INTO Tipos\_Pagos (nombre) VALUES

('Efectivo'),

('Tarjeta de Crédito'),

('Transferencia Bancaria'),

('Pago Móvil');

-- Insertar datos para crear diversas categorías de calzado

INSERT INTO Categorias (nombre\_categoria) VALUES

('Deportivos'),

('Casuales'),

('Formales'),

('Sandalias'),

('Botas'),

('Zapatillas');

-- Insertar datos para crear varios proveedores ficticios

INSERT INTO Proveedores (nombre, RFC, telefono, email) VALUES

('Proveedor A', 'RFC123', 1234567, 'proveedorA@example.com'),

('Proveedor B', 'RFC456', 9876540, 'proveedorB@example.com'),

('Proveedor C', 'RFC789', 5555555, 'proveedorC@example.com');

-- Insertar datos para crear varios lclientes ficticios

INSERT INTO Clientes (nombre, apellido\_paterno, apellido\_materno, telefono) VALUES

('Cliente 1', 'Apellido1', 'Apellido2', 11111111),

('Cliente 2', 'Apellido3', 'Apellido4', 22222222),

('Cliente 3', 'Apellido5', 'Apellido6', 33333333);

-- Insertar datos para crear varias direcciones ficticias de los clientes

INSERT INTO Direcciones\_Clientes (direccion, cliente\_id) VALUES

('Direccion Cliente 1', 1),

('Direccion Cliente 2', 2),

('Direccion Cliente 3', 3);

-- Insertar datos para crear varios productos ficticios

INSERT INTO Productos (nombre\_producto, descripcion, precio\_compra, precio\_venta, categoria\_id, proveedor\_id, imagen) VALUES

('Zapato Deportivo', 'Perfectos para correr', 50.00, 100.00, 1, 1, 'zapato1.jpg'),

('Zapato Casual', 'Ideal para uso diario', 40.00, 80.00, 2, 2, 'zapato2.jpg'),

('Zapato Formal', 'Elegantes para ocasiones especiales', 60.00, 120.00, 3, 1, 'zapato3.jpg'),

('Sandalias Verano', 'Cómodas y frescas', 30.00, 60.00, 4, 2, 'sandalias.jpg'),

('Botas de Invierno', 'Calientes y resistentes al agua', 70.00, 140.00, 5, 3, 'botas1.jpg'),

('Zapatillas para Correr', 'Diseñadas para atletas profesionales', 80.00, 160.00, 1, 1, 'zapatillas1.jpg');

-- Insertar datos para crear varias ventas a clientes ficticias

INSERT INTO Ventas (fecha\_venta, cliente\_id, direccion\_id, tipo\_pago\_id, total\_venta, empleado\_id, transaccion\_id, local\_id) VALUES

('2023-11-01', 1, 1, 1, 120.00, 2, 3, 1),

('2023-11-02', 2, 2, 2, 160.00, 3, 4, 2),

('2023-11-03', 3, 3, 3, 200.00, 4, 5, 3);

-- Insertar datos para crear varios detalles de ventas ficticios

INSERT INTO Detalles\_Ventas (venta\_id, cantidad\_producto, producto\_id) VALUES

(1, 2, 1),

(2, 2, 2),

(3, 2, 3);

-- Insertar datos para crear varias transacciones ficticias

INSERT INTO Transacciones (tipo\_transaccion, monto, fecha\_transaccion, referencia, usuario\_id) VALUES

('Venta', 120.00, '2023-11-01', 'ReferenciaVenta1', 2),

('Venta', 160.00, '2023-11-02', 'ReferenciaVenta2', 3),

('Venta', 200.00, '2023-11-03', 'ReferenciaVenta3', 4);

-- Insertar datos para crear varias compras a proveedores ficticias

INSERT INTO Compras (proveedor\_id, fecha\_compra, empleado\_id, transaccion\_id, local\_id) VALUES

(1, '2023-11-01', 2, 5, 1),

(2, '2023-11-02', 3, 6, 2),

(3, '2023-11-03', 4, 7, 3);

-- Insertar más datos ficticios en Compras (añadiendo más registros)

INSERT INTO Compras (proveedor\_id, fecha\_compra, empleado\_id, transaccion\_id, local\_id) VALUES

(1, '2023-10-10', 2, 8, 1),

(2, '2023-10-15', 3, 9, 2),

(3, '2023-10-20', 4, 10, 3);

-- Insertar datos para crear varios productos en diversos locales para tener un inventario ficticio

INSERT INTO Productos\_Locales (cantidad, local\_id, producto\_id, disponibilidad, fecha\_ingreso) VALUES

(20, 1, 1, TRUE, '2023-10-01'),

(15, 1, 2, TRUE, '2023-10-01'),

(18, 2, 3, TRUE, '2023-10-02'),

(25, 2, 4, TRUE, '2023-10-02'),

(22, 3, 5, TRUE, '2023-10-03'),

(30, 3, 6, TRUE, '2023-10-03');

-- Para activar de vuelta las restricciones de las llaves primarias

SET FOREIGN\_KEY\_CHECKS = 1;

**Procedimientos almacenados y su explicación**

Asimismo, se mencionó en el apartado de ‘Creación de la base de datos’, que en esta sección se insertarán los las sentencias sql que pertenecen a el archivo de los procedimientos almacenados de la base de datos con comentarios que expliquen qué significan y qué hace cada procedimiento.

**storedProcedures.sql**

-- Usamos la BD Tienda

USE Tienda;

-- Variable para en lugar de usar ; como delimitador o termionador de una secuencia, ahora sean los símbolos $$

DELIMITER $$

-- Se crea el procedimiento insertar categoria

CREATE PROCEDURE insertCategoria(

-- Parametro de entrada para el nombre de la categoria

IN nombreCategoria VARCHAR(50)

)

-- se empieza el SP

BEGIN

-- declaramos la variable categoria existente

DECLARE categoriaExistente INT;

-- Verificar si la categoría ya existe

SELECT COUNT(\*) INTO categoriaExistente

FROM Categorias

WHERE nombre\_categoria = nombreCategoria;

-- Si la categoría no existe, insertarla (estructura IF)

IF categoriaExistente = 0 THEN

INSERT INTO Categorias (nombre\_categoria)

VALUES (nombreCategoria);

ELSE

-- Puedes manejar la lógica de error o mensaje aquí, por ejemplo:

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'La categoría ya existe';

END IF;

END $$

-- Terminamos el SP al delimitar la terminación de este con $$

DELIMITER ;

-- Devolvemos el ; como el delimitador de las sentencias sql

DELIMITER $$

-- Se crea el procedimiento editar categoría

CREATE PROCEDURE updateCategoria(

-- parámetros de entrada

IN id INT,

IN nombreCategoria VARCHAR(50)

)

BEGIN

-- Actualizar el nombre de la categoría si el insertado y registrado son el mismo en donde el id es el mismo

UPDATE Categorias

SET nombre\_categoria = nombreCategoria

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- Se crea el procedimiento eliminar categoria

CREATE PROCEDURE deleteCategoria(

-- Parámetro id de entrada

IN id INT

)

BEGIN

-- Eliminar categoria cuando el id sea el mismo

DELETE FROM Categorias

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- Se crea el procedimiento obtiene categoria por id

CREATE PROCEDURE getCategoriaById(

-- parámetro de entrada id

IN id INT

)

BEGIN

-- seleccionar todo de categorias si tiene el mismo id registrado e insertado por el usuario

SELECT \*

FROM Categorias

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtiene todas las categorias

CREATE PROCEDURE getAllCategorias(

)

BEGIN

-- seleccionar todo de categorias

SELECT \*

FROM Categorias;

END $$

DELIMITER ;

--- PRODUCTOS

DELIMITER $$

-- se crea el procedimiento insertar producto

CREATE PROCEDURE insertProducto(

-- parámetros de entrada

IN nombreProducto VARCHAR(50),

IN descripcion TEXT,

IN precioCompra DECIMAL(10,2),

IN precioVenta DECIMAL(10,2),

IN categoriaId INT,

IN proveedorId INT, -- Nuevo parámetro para el proveedor

IN imagen\_path VARCHAR(100) -- Nuevo parámetro para la imagen

)

BEGIN

-- declaración de variable

DECLARE productoExistente INT;

-- Verificar si el producto ya existe

SELECT COUNT(\*) INTO productoExistente

FROM Productos

WHERE nombre\_producto = nombreProducto;

-- Si el producto no existe, insertarlo

IF productoExistente = 0 THEN

INSERT INTO Productos (nombre\_producto, descripcion, precio\_compra, precio\_venta, categoria\_id, proveedor\_id, imagen)

VALUES (nombreProducto, descripcion, precioCompra, precioVenta, categoriaId, proveedorId, imagen\_path);

ELSE

-- Manejar la lógica de error o mensaje aquí

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'El producto ya existe';

END IF;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimeinto para editar un producto

CREATE PROCEDURE updateProducto(

-- parámetros de entrada

IN productoId INT,

IN nombreProducto VARCHAR(50),

IN descripcion TEXT,

IN precioCompra DECIMAL(10,2),

IN precioVenta DECIMAL(10,2),

IN categoriaId INT,

IN proveedorId INT, -- Nuevo parámetro para el proveedor

IN imagen\_path VARCHAR(100) -- Nuevo parámetro para la imagen

)

BEGIN

-- actualizar productos cuando los parametros insertados sean los mismos que los registrados cuando id sea igual al id del producto

UPDATE Productos

SET nombre\_producto = nombreProducto,

descripcion = descripcion,

precio\_compra = precioCompra,

precio\_venta = precioVenta,

categoria\_id = categoriaId,

proveedor\_id = proveedorId, -- Agrega esta línea

imagen = IFNULL(imagen\_path, imagen) -- Modifica esta línea

WHERE id = productoId;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento eliminar producto

CREATE PROCEDURE deleteProducto(

-- parámetro de entrada id

IN id INT

)

BEGIN

-- eliminar productos donde el parámetro id sea igual al id registrado

DELETE FROM Productos

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtiene producto por id con parámetro de entrada id

CREATE PROCEDURE getProductoById(IN id INT)

BEGIN

-- seleccionar todo de productos donde el parámetro id sea igual al id registrado

SELECT \*

FROM Productos

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtiene todos los productos

CREATE PROCEDURE getAllProductos()

BEGIN

-- seleccionar todos los productos

SELECT \*

FROM Productos;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtiene productos por id del proveedor

CREATE PROCEDURE getProductosByProveedorId(

IN proveedorId INT -- parámetro de entrada id proveedor

)

BEGIN

-- seleccionar todo de productos si el parametro del id proveedor es el mismo que el registrado en la tabla

SELECT \*

FROM Productos

WHERE proveedor\_id = proveedorId;

END $$

DELIMITER ;

DELIMITER $$

--- Usuarios

-- se crea el procedimiento insertar (agregar) un usuario

CREATE PROCEDURE insertUsuario (

-- parámetros de entrada

IN email VARCHAR(100),

IN password VARCHAR(100),

IN rol VARCHAR(50)

)

BEGIN

-- insertar en la tabla usuarios los valores que contienen los parámetros de entrada en las columnas correspondientes

INSERT INTO Usuarios (email, password, rol)

VALUES (email, password, rol);

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento editar un usuario

CREATE PROCEDURE updateUsuario (

-- parámetros de entrada

IN id INT,

IN email VARCHAR(100),

IN password VARCHAR(100),

IN rol VARCHAR(50)

)

BEGIN

-- actualizar usuarios al establecer el email, password y rol siempre y cuando se tenga el mismo id

UPDATE Usuarios

SET email = email,

password = password,

rol = rol

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento eliminar usuario

CREATE PROCEDURE deleteUsuario(

IN id INT -- parámetro de entrada id

)

BEGIN

-- borrar de usuarios el que tenga el id registrado que se inserto en el parámetro

DELETE FROM Usuarios

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener usuario por id

CREATE PROCEDURE getUsuarioById(

IN id INT -- parámetro de entrada id

)

BEGIN

-- seleccionar todo de usuarios si se tiene el mismo id

SELECT \*

FROM Usuarios

WHERE id = id;

END$$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtiene todos los empleados

CREATE PROCEDURE getAllUsuarios()

BEGIN

-- seleccionar todo de usuarios

SELECT \*

FROM Usuarios;

END $$

DELIMITER ;

-- Cliente

DELIMITER //

-- se crea el procedimiento insertar (agregar) cliente

CREATE PROCEDURE insertCliente(

-- parámetros de entrada

IN nombre\_cliente VARCHAR(100),

IN apellido\_paterno VARCHAR(100),

IN apellido\_materno VARCHAR(100),

IN telefono INT,

IN dir VARCHAR(100))

BEGIN

-- declaramos variable

DECLARE cliente\_id INT;

-- Insertar cliente

INSERT INTO Clientes (nombre, apellido\_paterno, apellido\_materno, telefono)

VALUES (nombre\_cliente, apellido\_paterno, apellido\_materno, telefono);

-- Obtener el ID del cliente recién insertado

SET cliente\_id = LAST\_INSERT\_ID();

-- Insertar direcciones

INSERT INTO Direcciones\_Clientes (direccion, cliente\_id)

VALUES (dir, cliente\_id);

END //

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento editar cliente

CREATE PROCEDURE updateCliente(

-- parámetros de entrada

IN cliId INT,

IN nombre VARCHAR(100),

IN apellidoPaterno VARCHAR(100),

IN apellidoMaterno VARCHAR(100),

IN telefono INT(12)

)

BEGIN

-- actualizar clientes si los parametros de entrada coinciden con alguno de los registrados cuando el id del cliente sea el mismo

UPDATE Clientes

SET nombre = nombre,

apellido\_paterno = apellidoPaterno,

apellido\_materno = apellidoMaterno,

telefono = telefono

WHERE id = cliId;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento eliminar cliente

CREATE PROCEDURE deleteCliente(

IN id INT -- parámetro de entrada id

)

BEGIN

-- borrar de clientes cuando los id coincidan

DELETE FROM Clientes

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtiene cliente por id

CREATE PROCEDURE getClienteById(

-- parametro de entrada id

IN id INT

)

BEGIN

-- seleccionar todo cuando los id coincidan

SELECT \*

FROM Clientes

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener todos los clientes

CREATE PROCEDURE getAllClientes()

BEGIN

-- seleccionar todo de la tabla clientes

SELECT \*

FROM Clientes;

END $$

DELIMITER ;

--- Direcciones\_Clientes

DELIMITER $$

-- se crea el procedimiento obtener todas las direcciones del cliente por id

CREATE PROCEDURE getAllDireccionesById(

IN cliente\_id INT

)

BEGIN

-- selecciona todo lo que esta en la tabla clientes siempre que los id coincidan

SELECT \*

FROM Clientes

WHERE id = cliente\_id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el porocedimient insertar direccion del cliente

CREATE PROCEDURE insertDireccionCliente(

-- parámetros de entrada

IN cliente\_id INT,

IN nueva\_direccion VARCHAR(100)

)

BEGIN

-- insertar en direcciones clientes la direccion y id cliente los valores que estan en los parámetros

INSERT INTO Direcciones\_Clientes (direccion, cliente\_id)

VALUES (nueva\_direccion, cliente\_id);

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento actualizar la direccion del cliente

CREATE PROCEDURE updateDireccionCliente(

-- parámetros de entrada

IN dirId INT,

IN nueva\_direccion VARCHAR(100)

)

BEGIN

-- actualizar la tabla con la nueva direccion siempre que los id coincidan

UPDATE Direcciones\_Clientes

SET direccion = nueva\_direccion

WHERE id = dirId;

END $$

DELIMITER ;

-- Proveedores

DELIMITER $$

-- se crea el procedimiento insertar proveedor

CREATE PROCEDURE insertProveedor(

-- parámetros de entrada

IN nombreProveedor VARCHAR(100),

IN rfcProv VARCHAR(15),

IN correoProv VARCHAR(50),

IN telefonoProv INT

)

BEGIN

-- insertar en proveedores nombre, rfc, email, telefono con los valores de los parámetros

INSERT INTO Proveedores (nombre, RFC, email, telefono)

VALUES (nombreProveedor, rfcProv, correoProv, telefonoProv);

END$$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento actualizar proveedor

CREATE PROCEDURE updateProveedor(

-- parámetros de entrada

IN proveedor\_id INT,

IN nombreProveedor VARCHAR(100),

IN rfcProv VARCHAR(15),

IN correoProv VARCHAR(50),

IN telefonoProv INT(12)

)

BEGIN

-- actualizar proveedores con los nuevos valores que tienen los parámetros siempre y cuando el id coincida

UPDATE Proveedores

SET

nombre = nombreProveedor,

RFC = rfcProv,

email = correoProv,

telefono = telefonoProv

WHERE id = proveedor\_id;

END$$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento eliminar proveedor

CREATE PROCEDURE deleteProveedor(

-- parámetro de entrada id

IN id INT

)

BEGIN

-- eliminar de proveedores siempre y cuando los id coincidan

DELETE FROM Proveedores

WHERE id = id;

END$$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener el proveedor por id

CREATE PROCEDURE getProveedorById(

-- parámetro de entrada id

IN id INT

)

BEGIN

-- seleccionar todo de proveedores si los id coinciden

SELECT \*

FROM Proveedores

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener todos los proveedores

CREATE PROCEDURE getAllProveedores()

BEGIN

-- seleccionar todo de proveedores

SELECT \*

FROM Proveedores;

END $$

DELIMITER ;

--- Inventarios

DELIMITER //

-- se crea el procedimiento insertar productos al local

CREATE PROCEDURE insertProductoLocal(

- parámetros de entrada

IN p\_cantidad INT,

IN p\_local\_id INT,

IN p\_producto\_id INT,

IN p\_disponibilidad TINYINT,

IN p\_fecha\_ingreso DATE

)

BEGIN

-- insertar en productos locales cantidad, local id, producto id, disponibilidad, fecha ingreso los valores de los parámetros

INSERT INTO Productos\_Locales (cantidad, local\_id, producto\_id, disponibilidad, fecha\_ingreso)

VALUES (p\_cantidad, p\_local\_id, p\_producto\_id, p\_disponibilidad, p\_fecha\_ingreso);

END //

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener precio por producto

CREATE PROCEDURE getPrecioProducto(

-- parámetro de entrada producto id

IN productoId INT

)

BEGIN

-- seleccionar el precio de venta de productos si los id coinciden

SELECT precio\_venta FROM Productos WHERE id = productoId;

END $$

DELIMITER ;

DELIMITER //

-- se crea el procedimiento almacenado obtener productos de los locales por id del local

CREATE PROCEDURE getProductosLocalesByLocalId(

-- parámetro de entrada

IN p\_local\_id INT

)

BEGIN

-- seleccionar todo de productos locales si los id de los locales coinciden

SELECT \* FROM Productos\_Locales WHERE local\_id = p\_local\_id;

END //

DELIMITER ;

DELIMITER //

-- se crea el procedimiento obtener productos disponibles por local

CREATE PROCEDURE getProductosDisponiblesByLocalId(

-- parámetro de entrada

IN p\_local\_id INT

)

BEGIN

-- seleccionar todo de productos locales si los id de los locales coinciden, la cantidad de productos es mayor a 0 y hay disponibilidad

SELECT \* FROM Productos\_Locales WHERE local\_id = p\_local\_id AND cantidad > 0

AND disponibilidad = 1;

END //

DELIMITER ;

DELIMITER //

-- se crea el procedimiento obtener productos en local con parámetro de entrada

CREATE PROCEDURE obtener\_productos\_en\_local(IN p\_local\_id INT)

BEGIN

-- seleccionar el id y nombre del producto de la unión de productos locales y productos si el id del local coincide

SELECT p.id, p.nombre\_producto

FROM Productos\_Locales pl

JOIN Productos p ON pl.producto\_id = p.id

WHERE pl.local\_id = p\_local\_id;

END //

DELIMITER ;

DELIMITER //

-- se crea el procedimiento desactivar producto del local

CREATE PROCEDURE deactivateProductoLocal(

-- parámetro de entrada

IN p\_producto\_local\_id INT

)

BEGIN

-- actualizar disponibilidad a 0 (que ya no hay) en productos locales si los id coinciden

UPDATE Productos\_Locales SET disponibilidad = 0 WHERE id = p\_producto\_local\_id;

END //

DELIMITER ;

DELIMITER //

-- se crea el procedimiento activar producto del local

CREATE PROCEDURE activateProductoLocal(

-- parámetro de entrada

IN p\_producto\_local\_id INT

)

BEGIN

-- actualizar disponibilidad a 1 (que si hay) en productos locales si los id coinciden

UPDATE Productos\_Locales SET disponibilidad = 1 WHERE id = p\_producto\_local\_id;

END //

DELIMITER ;

DELIMITER //

-- se crea el procedimiento obtener productos local por id

CREATE PROCEDURE getProductoLocalById(

-- parámetro de entrada

IN p\_producto\_local\_id INT

)

BEGIN

-- seleccionar todo de productos locales si los id coinciden

SELECT \* FROM Productos\_Locales WHERE id = p\_producto\_local\_id;

END //

DELIMITER ;

--- Ventas

DELIMITER $$

-- se crea el procedimiento obtener ventas por id del local

CREATE PROCEDURE getVentasByLocalId(

-- parámetro de entrada

IN localId INT

)

BEGIN

-- seleccionar todo de ventas si los id coinciden

SELECT \* FROM Ventas

WHERE local\_id = localId;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento registrar venta

CREATE PROCEDURE registrar\_venta(

-- parámteros de entrada

IN p\_cliente\_id INT,

IN p\_direccion\_id INT,

IN p\_tipo\_pago\_id INT,

IN p\_total\_venta DECIMAL(10, 2),

IN p\_empleado\_id INT,

IN p\_local\_id INT

)

BEGIN

-- Iniciar transacción

START TRANSACTION;

-- Registrar la transacción financiera asociada a la venta

INSERT INTO Transacciones (tipo\_transaccion, monto, fecha\_transaccion, usuario\_id)

VALUES ('Venta', p\_total\_venta, NOW(), p\_empleado\_id);

-- Obtener el ID de la transacción registrada

SET @transaccion\_id = LAST\_INSERT\_ID();

-- Registrar la venta principal

INSERT INTO Ventas (cliente\_id, fecha\_venta, direccion\_id, tipo\_pago\_id, total\_venta, empleado\_id, transaccion\_id, local\_id)

VALUES (p\_cliente\_id, NOW(), p\_direccion\_id, p\_tipo\_pago\_id, p\_total\_venta, p\_empleado\_id, @transaccion\_id, p\_local\_id);

-- Obtener el ID de la venta registrada

SET @venta\_id = LAST\_INSERT\_ID();

-- Commit de la transacción

COMMIT;

-- Devolver el ID de la venta registrado

SELECT @venta\_id AS venta\_id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento registrar detalle de la venta

CREATE PROCEDURE registrar\_detalle\_venta(

-- parámetros de entrada

IN p\_venta\_id INT,

IN p\_producto\_id INT,

IN p\_cantidad\_producto INT

)

BEGIN

-- Registrar el detalle de la venta

INSERT INTO Detalles\_Ventas (venta\_id, producto\_id, cantidad\_producto)

VALUES (p\_venta\_id, p\_producto\_id, p\_cantidad\_producto);

-- Restar la cantidad vendida de Productos\_Locales

UPDATE Productos\_Locales

SET cantidad = cantidad - p\_cantidad\_producto

WHERE producto\_id = p\_producto\_id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento actualizar venta

CREATE PROCEDURE updateVenta(

-- parámetros de entrada

IN id INT,

IN productoId INT,

IN clienteId INT,

IN cantidad INT,

IN fechaVenta DATE,

IN totalVenta DECIMAL(10,2)

)

BEGIN

-- actualizar ventas al poner los valores de los parámteros en los registros correspondientes si los id coinciden

UPDATE Ventas

SET producto\_id = productoId,

cliente\_id = clienteId,

cantidad = cantidad,

fecha\_venta = fechaVenta,

total\_venta = totalVenta

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- eliminar venta

CREATE PROCEDURE deleteVenta(

-- parámetro de entrada

IN id INT

)

BEGIN

-- borrar todo de ventas si el id coincide

DELETE FROM Ventas

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener venta por id

CREATE PROCEDURE getVentaById(

-- parámetro de entrada

IN id INT

)

BEGIN

-- seleccionar todo de ventas si coinciden los id

SELECT \*

FROM Ventas

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener todas las ventas

CREATE PROCEDURE getAllVentas()

BEGIN

-- seleccionar todo de ventas

SELECT \*

FROM Ventas;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener todas las ventas por id del local

CREATE PROCEDURE getAllVentasByLocalId(

-- parámetro de entrada

IN localId INT

)

BEGIN

-- seleccionar todo de venta si los id del local coinciden

SELECT \* FROM Ventas WHERE local\_id = localId;

END $$

DELIMITER ;

-- Compras

DELIMITER $$

-- se crea el procedimiento registrar compra

CREATE PROCEDURE registrar\_compra(

IN proveedorId INT,

IN totalCompra DECIMAL(10,2),

IN empleadoId INT,

IN localId INT

)

BEGIN

-- se empieza la transacción

START TRANSACTION;

-- insertar en transacciones tipo de transacción, monto, fecha de transacción y usuario id los valores de los parámetros

INSERT INTO Transacciones (tipo\_transaccion, monto, fecha\_transaccion, usuario\_id)

VALUES ('Compra', totalCompra, NOW(), empleadoId);

-- Obtener el ID

SET @transaccion\_id = LAST\_INSERT\_ID();

-- insertar en compras el proveedor id, fecha de compra, empleado id, transacción id y local id los valores de los parámetros

INSERT INTO Compras (proveedor\_id, fecha\_compra, empleado\_id, transaccion\_id, local\_id)

VALUES (proveedorId, NOW(), empleadoId, @transaccion\_id, localId);

-- Obtener el ID de la compra

SET @compra\_id = LAST\_INSERT\_ID();

-- guardar cambios en la transaccion

COMMIT;

-- establecer variable de entorno

SELECT @compra\_id AS compra\_id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento registrar detalle de compra

CREATE PROCEDURE registrar\_detalle\_compra(

-- parámetros de entrada

IN compraId INT,

IN productoId INT,

IN cantidad INT

)

BEGIN

-- insertar en detalles compras compra id, producto id y cantidad los valores de los parámetros

INSERT INTO Detalles\_Compras (compra\_id, producto\_id, cantidad)

VALUES (compraId, productoId, cantidad);

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener compras por id del local

CREATE PROCEDURE getComprasByLocalId(

-- parámetro de entrada

IN localId INT

)

BEGIN

-- seleccionar todo de compras si los id coinciden

SELECT \* FROM Compras WHERE local\_id = localId;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento actualizar compra

CREATE PROCEDURE updateCompra(

-- parámetros de entrada

IN id INT,

IN proveedorId INT,

IN fechaCompra DATE,

IN totalCompra DECIMAL(10,2)

)

BEGIN

-- actualizar compras donde los registros tengan los valores de los parámetros de entrada si el id coincide

UPDATE Compras

SET proveedor\_id = proveedorId,

fecha\_compra = fechaCompra,

total\_compra = totalCompra

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento eliminar compra

CREATE PROCEDURE deleteCompra(

-- parámetro de entrada

IN id INT

)

BEGIN

-- eliminar de compras si los id coinciden

DELETE FROM Compras

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener compra por id

CREATE PROCEDURE getCompraById(

-- parámetro de entrada

IN id INT

)

BEGIN

-- seleccionar todo de compras si los id coinciden

SELECT \*

FROM Compras

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener todas las compras

CREATE PROCEDURE getAllCompras()

BEGIN

-- se selecciona todo de compras

SELECT \*

FROM Compras;

END $$

DELIMITER ;

-- Detalles de Compras

DELIMITER $$

-- se crea el procedimiento insertar detalles de compras

CREATE PROCEDURE insertDetalleCompra(

-- parámetros de entrada

IN compraId INT,

IN productoId INT,

IN cantidad INT

)

BEGIN

-- insertar en detalles compras compra id, producto id y cantidad con los valores de los parámetros

INSERT INTO Detalles\_Compras (compra\_id, producto\_id, cantidad)

VALUES (compraId, productoId, cantidad);

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento actualizar detalles de compra

CREATE PROCEDURE updateDetalleCompra(

-- parámetros de entrada

IN id INT,

IN compraId INT,

IN productoId INT,

IN cantidad INT

)

BEGIN

-- actualizar detalles compras en donde los registros tengan los valores de los parámetros si los id coinciden

UPDATE Detalles\_Compras

SET compra\_id = compraId,

producto\_id = productoId,

cantidad = cantidad

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento eliminar detalle de la compra

CREATE PROCEDURE deleteDetalleCompra(

-- parámetro de entrada

IN id INT

)

BEGIN

-- borrar de detalles compras si los id coinciden

DELETE FROM Detalles\_Compras

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener detalles de compra por id

CREATE PROCEDURE getDetalleCompraById(

-- parámetro de entrada

IN id INT

)

BEGIN

-- seleccionar todo de detalles compras si los id coinciden

SELECT \*

FROM Detalles\_Compras

WHERE id = id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener todos los detalles de compras

CREATE PROCEDURE getAllDetallesCompras()

BEGIN

-- seleccionar todo de detalles de compras

SELECT \*

FROM Detalles\_Compras;

END $$

DELIMITER ;

--- Empleados

DELIMITER $$

-- se crea el procedimiento obtener empleado por id

CREATE PROCEDURE getEmpleadoById(

-- parámetro de entrada

IN IdEmpleado INT)

BEGIN

-- seleccionar todo de empleados si el id coincide

SELECT \* FROM Empleados

WHERE id = IdEmpleado;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento obtener empleados por id del local

CREATE PROCEDURE getEmpleadosByLocalId(

-- parámetro de entrada

IN Id INT)

BEGIN

-- seleccionar todo de empleados si los id coinciden

SELECT \* FROM Empleados WHERE local\_id = Id;

END $$

DELIMITER ;

DELIMITER $$

-- Procedimiento para insertar un nuevo empleado asociado a un usuario

CREATE PROCEDURE insertEmpleadoInLocal(

-- parámetros de entrada

IN name VARCHAR(100),

IN appPaterno VARCHAR(100),

IN appMaterno VARCHAR(100),

IN sal DECIMAL(10,2),

IN puesto VARCHAR(100),

IN localId INT,

IN userId INT

)

BEGIN

-- Insertar un nuevo empleado asociado al usuario

INSERT INTO Empleados (nombre, apellido\_paterno, apellido\_materno, salario, puesto, local\_id, usuario\_id)

VALUES (name, appPaterno, appMaterno, sal, puesto, localId, userId);

END $$

DELIMITER ;

-- Conseguir Usuario

DELIMITER $$

-- se crea el procedimiento obtener el usuario con el email

CREATE PROCEDURE getUsuarioByEmail(

-- parámetro de entrada

IN correo VARCHAR(100)

)

BEGIN

-- seleccionar todo de usuarios si los correos coinciden

SELECT \* FROM Usuarios

WHERE email = correo;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento desactivar empleado

CREATE PROCEDURE deactivateEmpleado(

-- parámetro de entrada

IN usuario\_id INT

)

BEGIN

-- actualizar usuarios y poner la variable activo como 0 (no est activo) si los id coinciden

UPDATE Usuarios

SET activo = 0

WHERE id = usuario\_id;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento activar empleado

CREATE PROCEDURE activateEmpleado(

-- parámetro de entrada

IN usuario\_id INT

)

BEGIN

-- actualizar usuarios y a variable activo como 1 (esta activo) si los id coinciden

UPDATE Usuarios

SET activo = 1

WHERE id = usuario\_id;

END $$

DELIMITER ;

-- Actualizar el Empleado

DELIMITER $$

CREATE PROCEDURE updateEmpleado(

-- parámetros de entrada

IN IdEmpleado INT,

IN nombre VARCHAR(100),

IN appPaterno VARCHAR(100),

IN appMaterno VARCHAR(100),

IN salario DECIMAL(10,2),

IN puesto VARCHAR(100)

)

BEGIN

-- actualizar los registros de la tabla usuarios si con los valores de los parámteros si los id coinciden

UPDATE Empleados

SET nombre = nombre,

apellido\_paterno = appPaterno,

apellido\_materno = appMaterno,

salario = salario,

puesto = puesto

WHERE id = IdEmpleado;

END $$

DELIMITER ;

--- Locales

DELIMITER $$

-- se crea el procedimient obtener local por id

CREATE PROCEDURE getLocalById(

-- parámetro de entrada

IN IdLocal INT

)

BEGIN

-- seleccionar todo de locales si los id coinciden

SELECT \* FROM Locales

WHERE id = IdLocal;

END $$

DELIMITER ;

--- Productos

DELIMITER $$

-- se crea el procedimiento obtener productos

CREATE PROCEDURE getProductos(

-- parámetro de entrada

IN IdLocal INT

)

BEGIN

-- seleccionar todo de la combinación de tablas de productos y productos locales si el id del local coincide

SELECT \* FROM Productos p JOIN Productos\_Locales pl

ON p.id = pl.producto\_id WHERE pl.local\_id = IdLocal;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento para desactivar producto

CREATE PROCEDURE deactivateProducto(

-- parámetro de entrada

IN IdProductoLocal INT

)

BEGIN

-- actualizar productos locales al establecer la disponibilidad en o (no hay) si los id coinciden

UPDATE Productos\_Locales

SET disponibilidad = 0

WHERE id = IdProductoLocal;

END $$

DELIMITER ;

DELIMITER $$

-- se crea el procedimiento para activar un producto

CREATE PROCEDURE activateProducto(

-- parámetro de entrada

IN IdProductoLocal INT

)

BEGIN

-- actualizar productos locales al establecer la dispinibilidad en 1 (si hay) si los id coinciden

UPDATE Productos\_Locales

SET disponibilidad = 1

WHERE id = IdProductoLocal;

END $$

DELIMITER ;

--- Tipos de pago

DELIMITER $$

-- se crea el procedimiento obtener tipos de pago

CREATE PROCEDURE getTiposPagos()

BEGIN

-- seleccionar todo de la tabla tipos de pago

SELECT \* FROM Tipos\_Pagos;

END $$

DELIMITER ;

-- Direcciones

DELIMITER $

-- se crea el procedimiento obtener direcciones del cliente por su id

CREATE PROCEDURE getDireccionesByClienteId(

-- parámetro de entrada

IN clienteId INT

)

BEGIN

-- se selecciona todo de direcciones clientes si los id del cliente coinciden

SELECT \* FROM Direcciones\_Clientes

WHERE cliente\_id = clienteId;

END $$

DELIMITER ;

-- Reporte financiero

DELIMITER //

-- se crea el procedimiento obtener ventas totales por cliente con parámetro de entrada

CREATE PROCEDURE getTotalSalesByClient(IN clientId INT)

BEGIN

-- se seleccina nombre, ambos apellidos y en total de las ventas de la combinación de las tablas clientes y ventas si el id del cliente coincide

SELECT c.nombre, c.apellido\_paterno, c.apellido\_materno, SUM(v.total\_venta) AS total\_ventas

FROM Clientes c

JOIN Ventas v ON c.id = v.cliente\_id

WHERE c.id = clientId;

END //

DELIMITER ;

DELIMITER //

-- se crea el procedimiento obtener ventas totales por mes

CREATE PROCEDURE GetTotalSalesByMonth()

BEGIN

-- se selecciona la fecha venta del mes, año y la suma de las ventas totales de ventas agrupándolas por mes y año

SELECT

MONTH(fecha\_venta) AS mes,

YEAR(fecha\_venta) AS anio,

SUM(total\_venta) AS total\_ventas

FROM Ventas

GROUP BY mes, anio;

END //

DELIMITER ;

DELIMITER //

-- se crea el procedimiento obtener ventas totales por año

CREATE PROCEDURE GetTotalSalesByYear()

BEGIN

-- se selecciona la fecha venta del año y la suma de las ventas totales de ventas agrupándolas por año

SELECT

YEAR(fecha\_venta) AS anio,

SUM(total\_venta) AS total\_ventas

FROM Ventas

GROUP BY anio;

END //

DELIMITER ;

**Comandos para replicar la instancia**

Se brindarán los comandos necesarios para la replicación del programa en una instancia nueva. De este modo, se tiene la instalación de paquetes y librerías

sudo dnf update

sudo dnf install mariadb-server

sudo dnf install

sudo dnf install pip

sudo dnf install mariadb\*

sudo dnf groupinstall “Development Tools”

pip install flask

pip install request

pip install flask\_mysqldb

pip install -r requirements.txt

Una vez instalados los paquetes y componentes necesarios para la aplicación, se deberá de realizar la restauración de la base de datos correspondiente.

mysql -u root -p < schemaTienda.sql

Con ello se crearán las tablas correspondientes a la base de datos, una vez se tenga ello, se deberá de realizar la inserción de los datos

mysql -u root -p < dataTienda.sql

Luego de ello, se deberá de restablecer las sentencias correspondientes a los procedimientos almacenados

mysql -u root -p < storedProcedure.sql

Con ello, ya se tendría toda la información necesaria para la ejecución del programa, ahora se deberá de ejecutar el archivo correspondiente.

./inicia.sh

En caso no se tenga, se deberá de iniciar con:

flask run –host=0.0.0.0

Al ingresar se considerará un usuario login para ello, se deberá de ingresar con:

[adminA@example.com](mailto:adminA@example.com)

123

**Descripción de la aplicación desarrollada**

Una vez hecha la base de datos, se pasa a desarrollar la app. Para ello usamos las instancias de las máquinas virtuales que proporciona Google Cloud, en donde hicimos una instancia que se conecta a la red local para acceder a internet.

Dentro de la pestaña de la máquina virtual, creamos el directorio de proyecto el cual tiene los subdirectorios de static/images, dónde está la carpeta de productos y dentro de está se almacenan todas las fotos que se usaron para la creación de la aplicación del proyecto como las fotos de los productos y de los locales. El segundo subdirectorio es la carpeta de templates en donde se encuentran todos los archivos html con el código fuente para la creación de las diferentes pestañas que conforman el menú de la barra superior de inicio.

Por último, dentro del directorio principal del proyecto, se encuentra la app, codificada con python. Dentro de este archivo .py se encuentra la configuración del acceso de la aplicación a la base de datos y todas las rutas CRUD utilizadas como login o inicio de sesión del usuario, el registro del mismo, autentificación del usuario que trata de iniciar sesión, acceder al panel de control y para cerrar sesión.

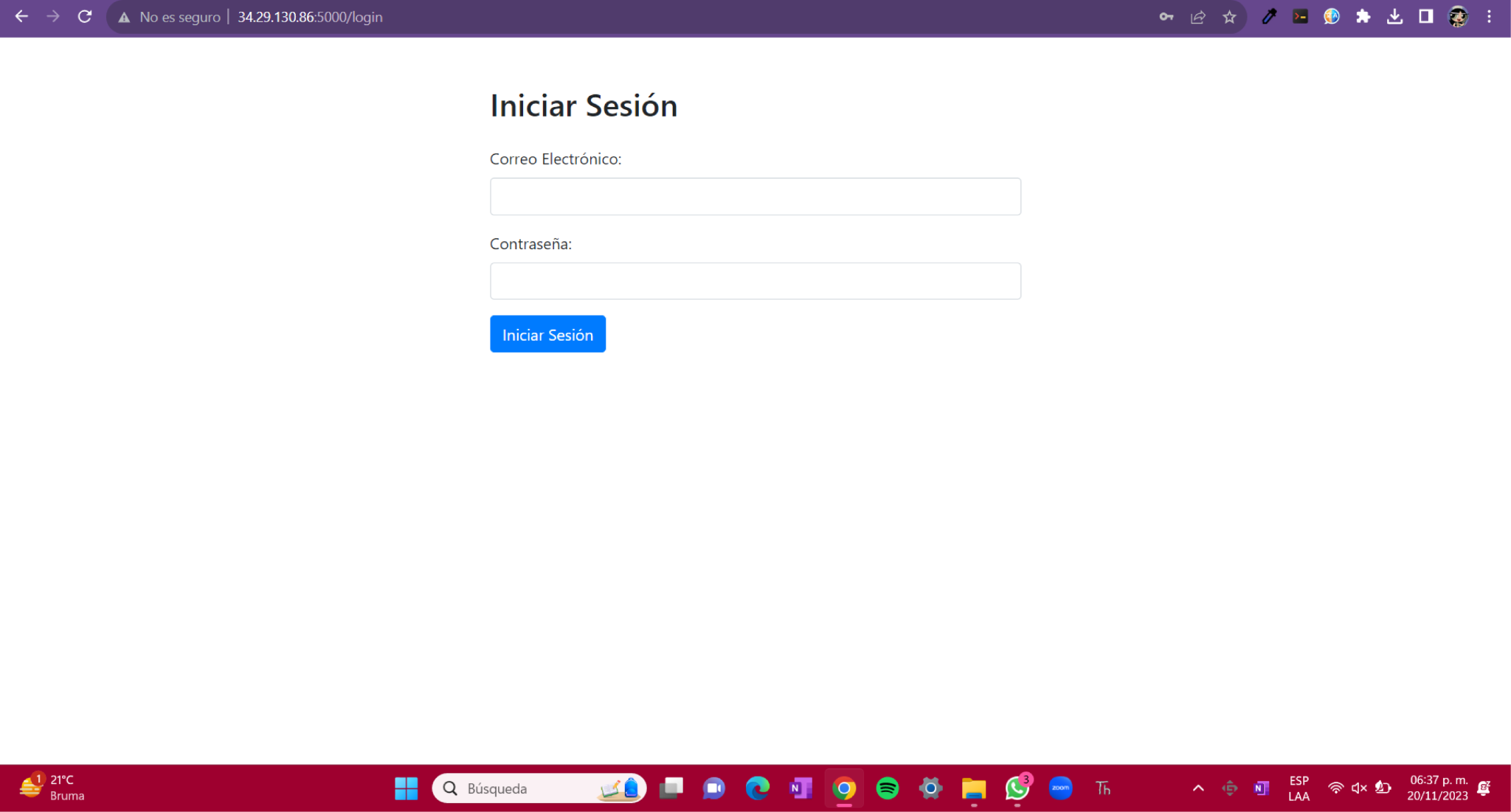
En cuanto a las acciones e información que puede acceder el empleado, también se especifican dentro de este archivo y todas las acciones que se pueden desempeñar dentro de cada pestaña del menú. Estas acciones que se pueden realizar se explican de mejor manera en el flujograma de la app o en el siguiente apartado de screenshots.

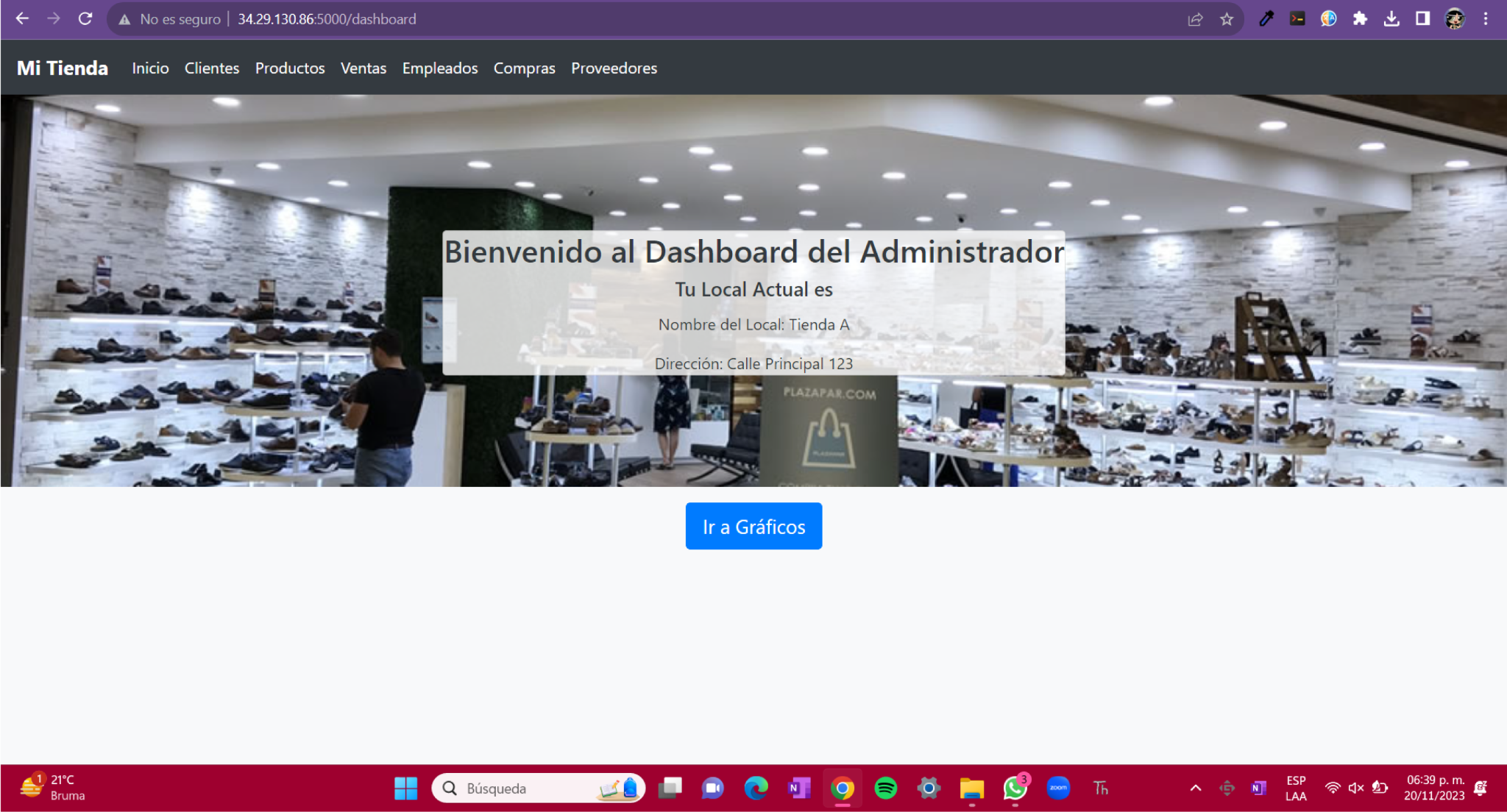
**Screenshots**

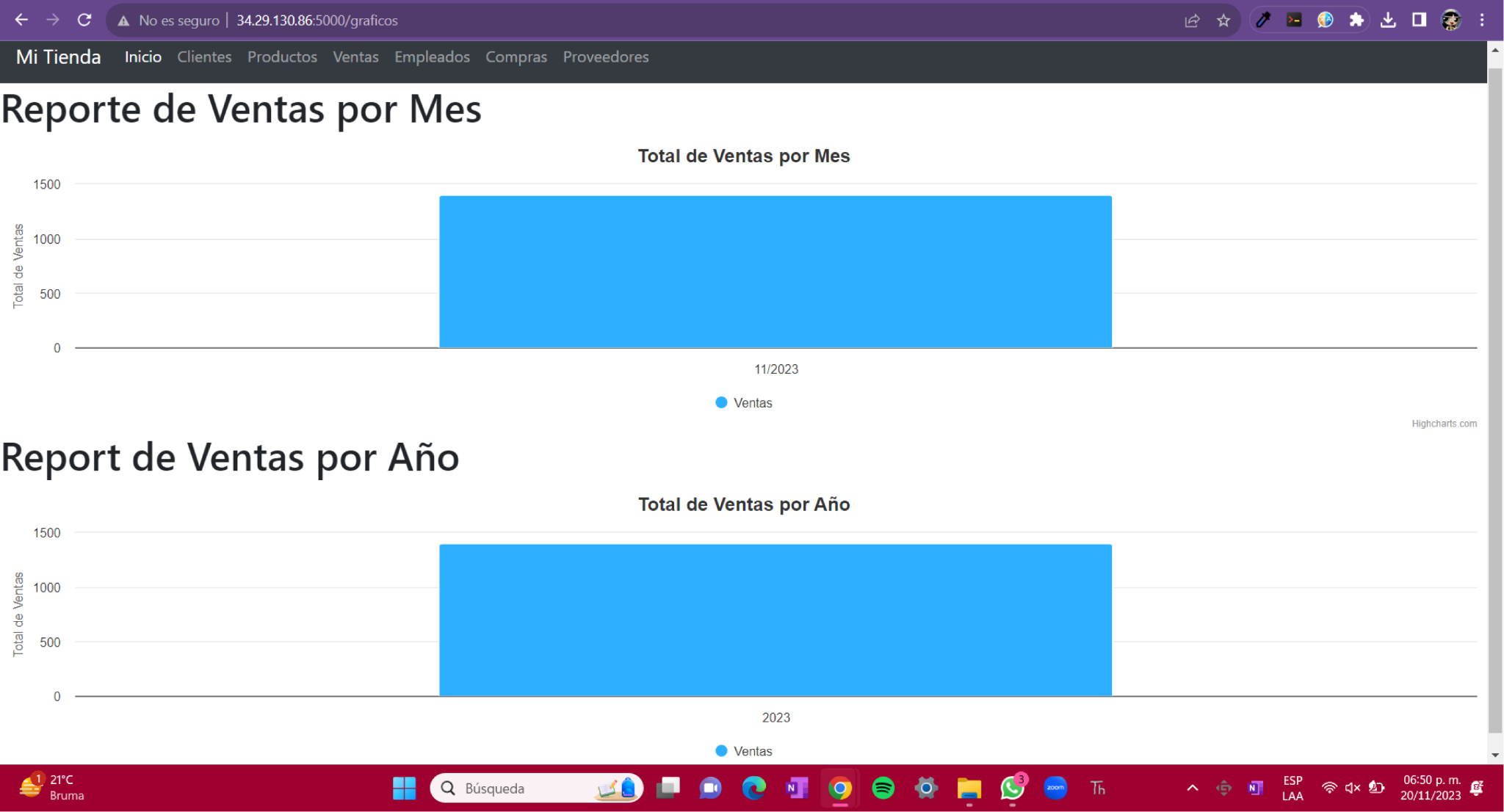
En esta sección se anexará screenshots de como luce la aplicación ya desarrollada, corriendo, se utiliza el usuario:

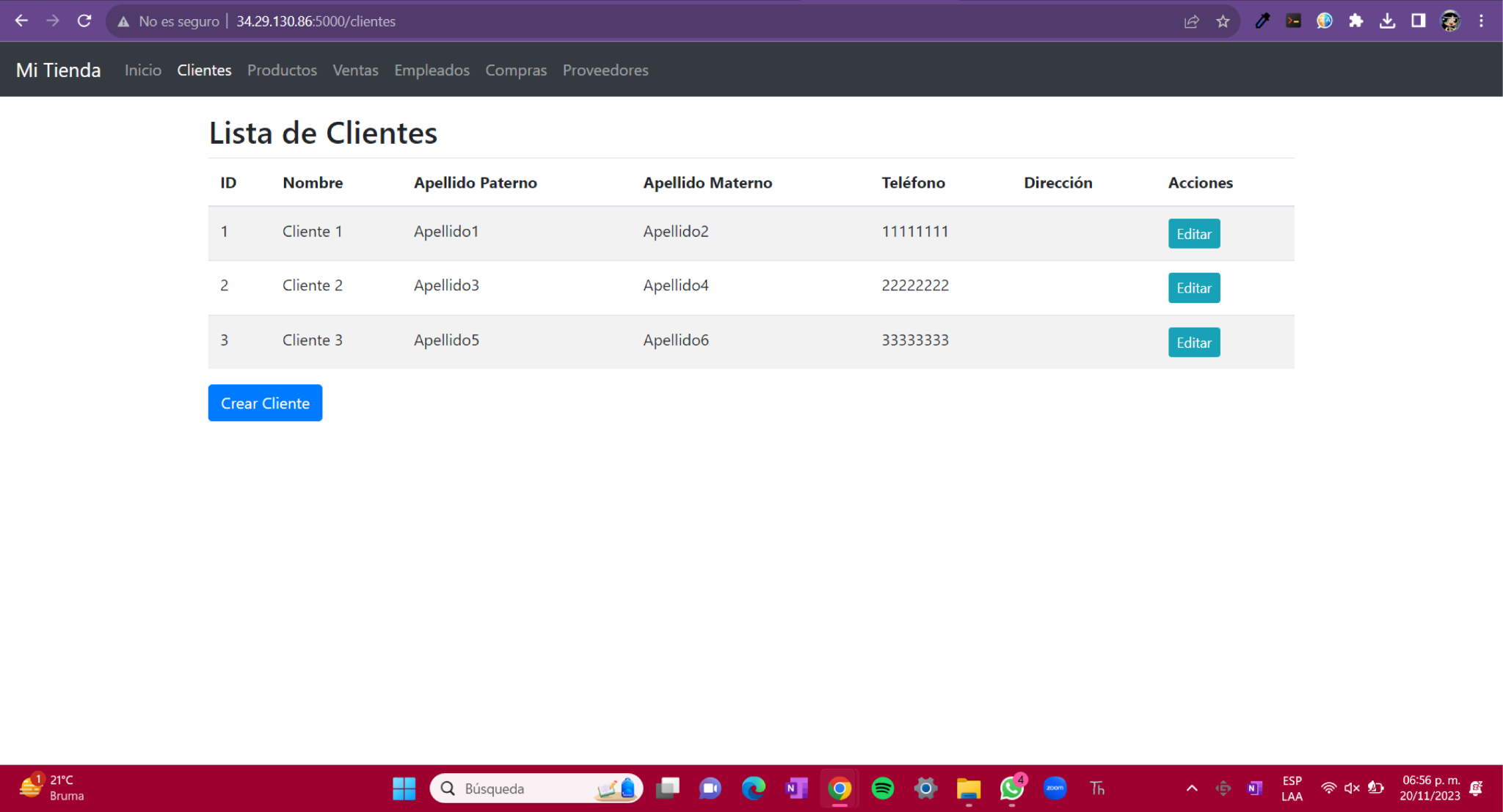
correo: [adminA@example.com](mailto:adminA@example.com)

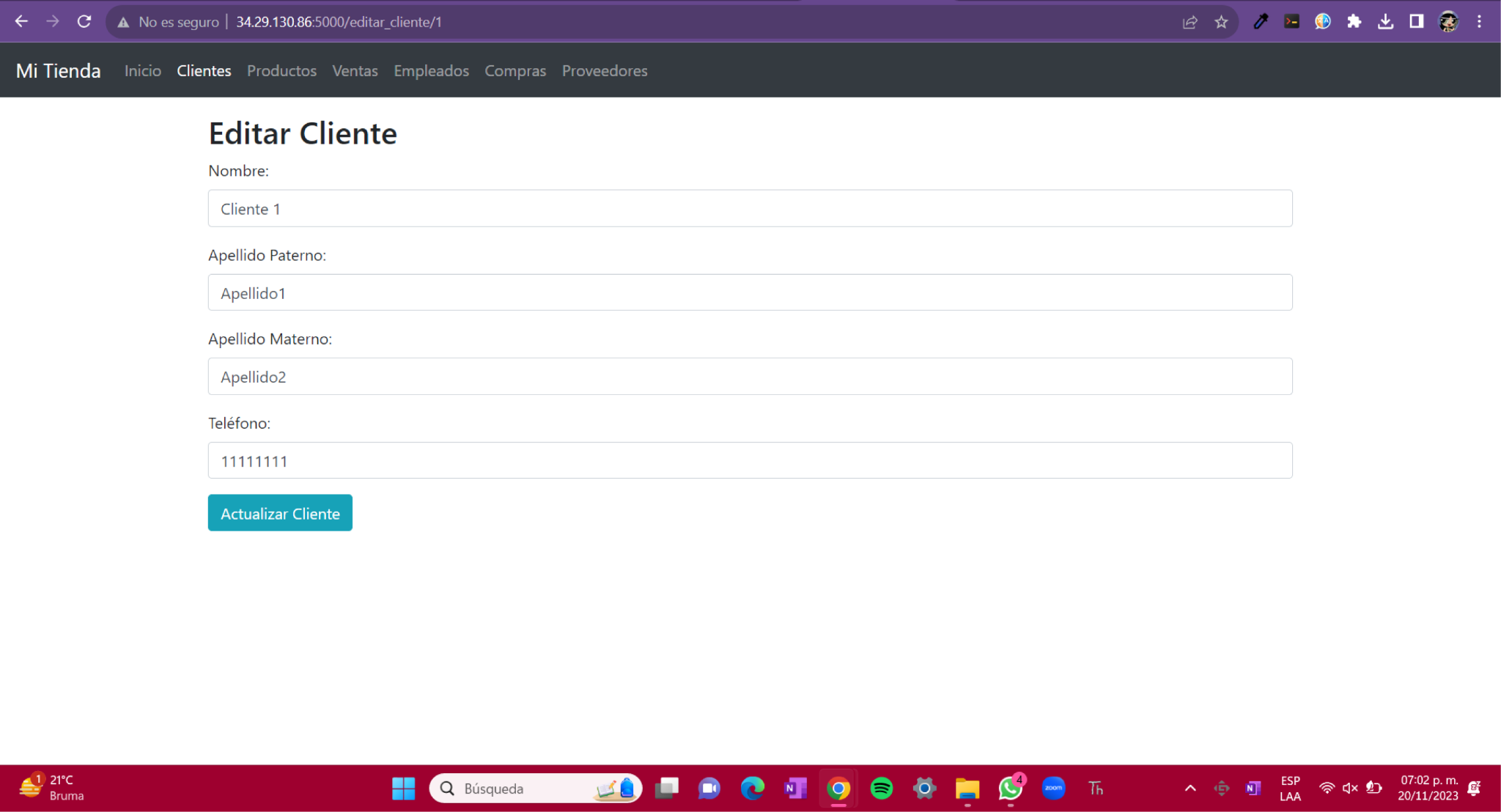
contraseña: 123

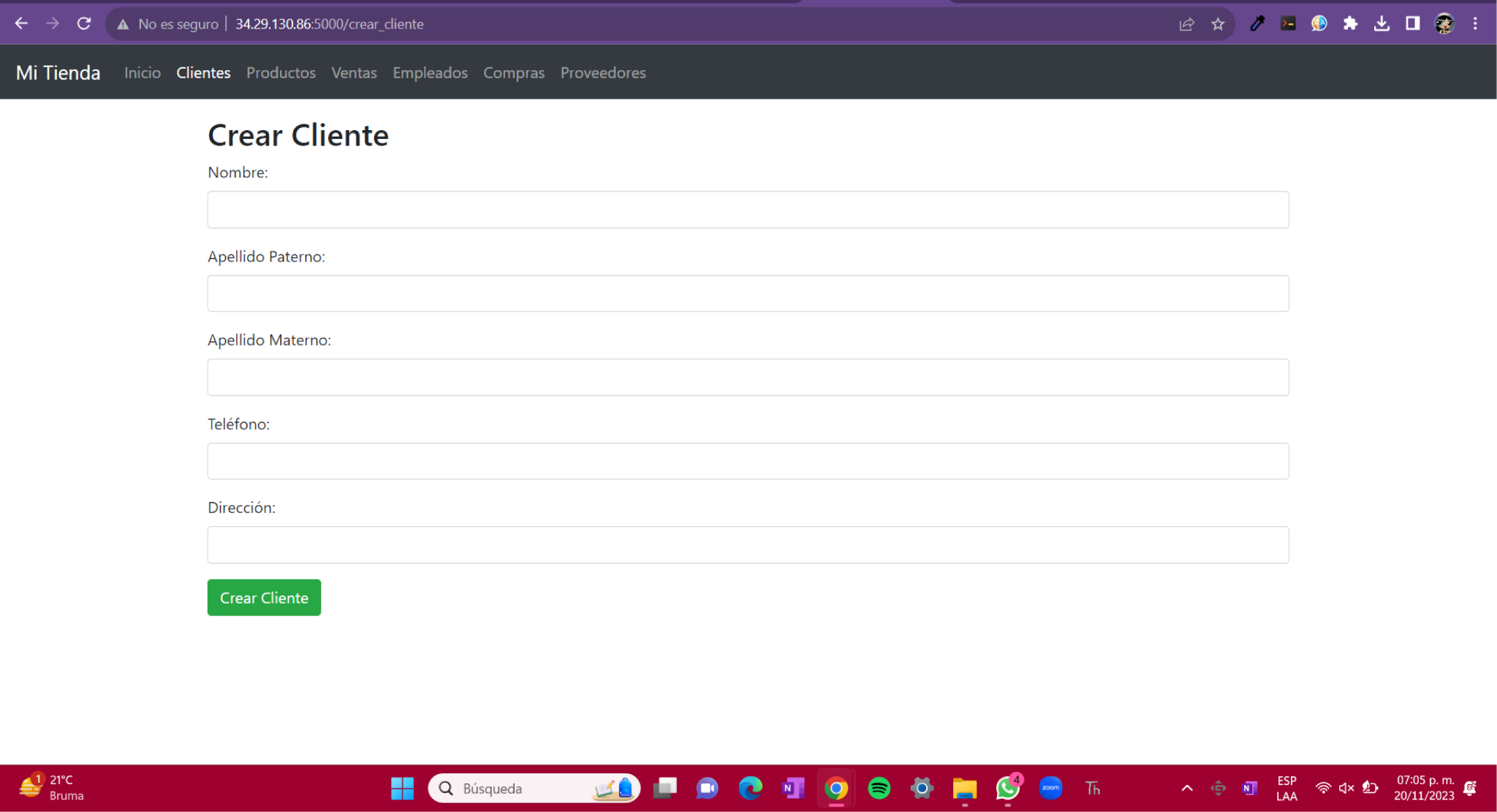


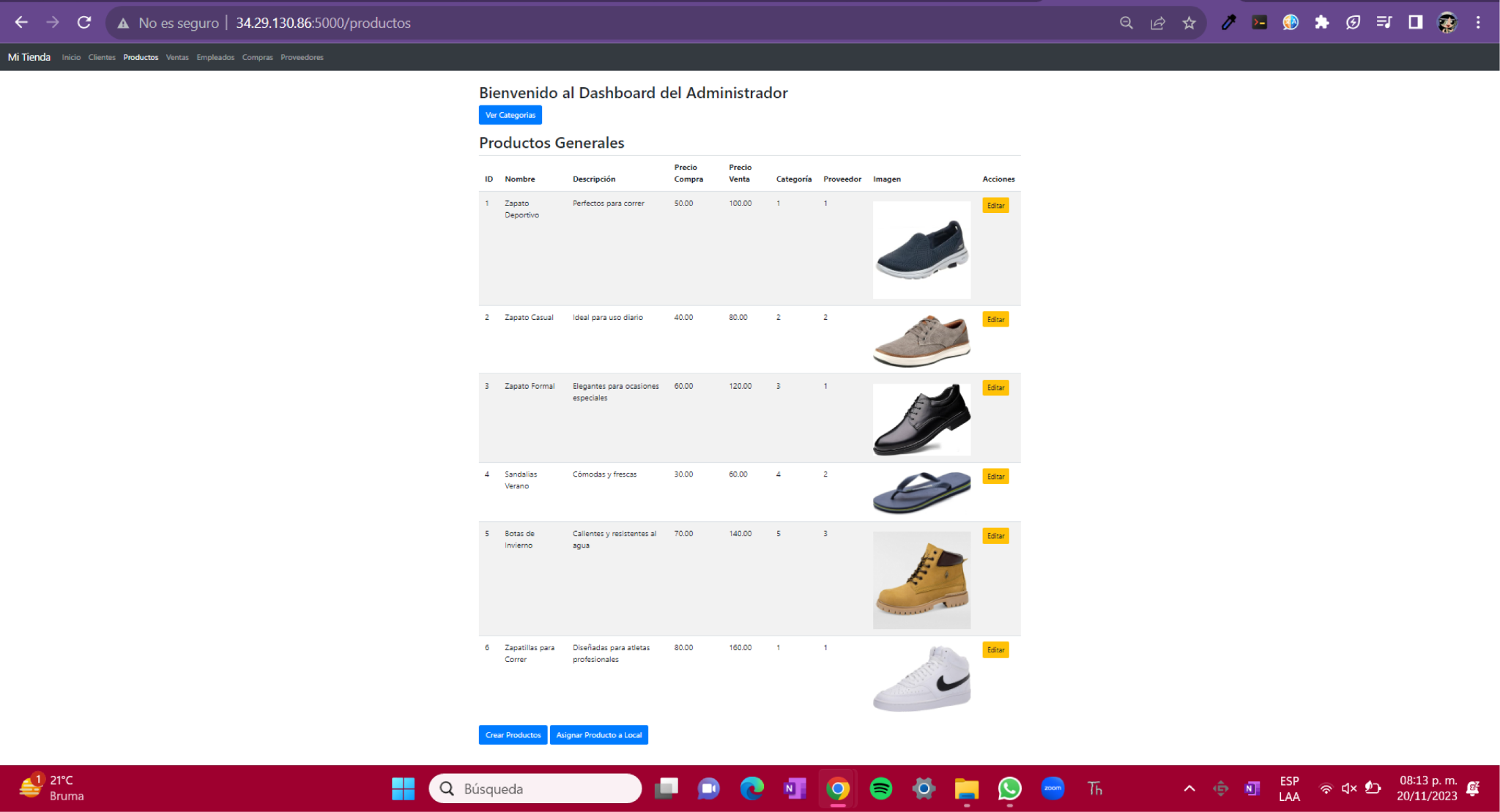


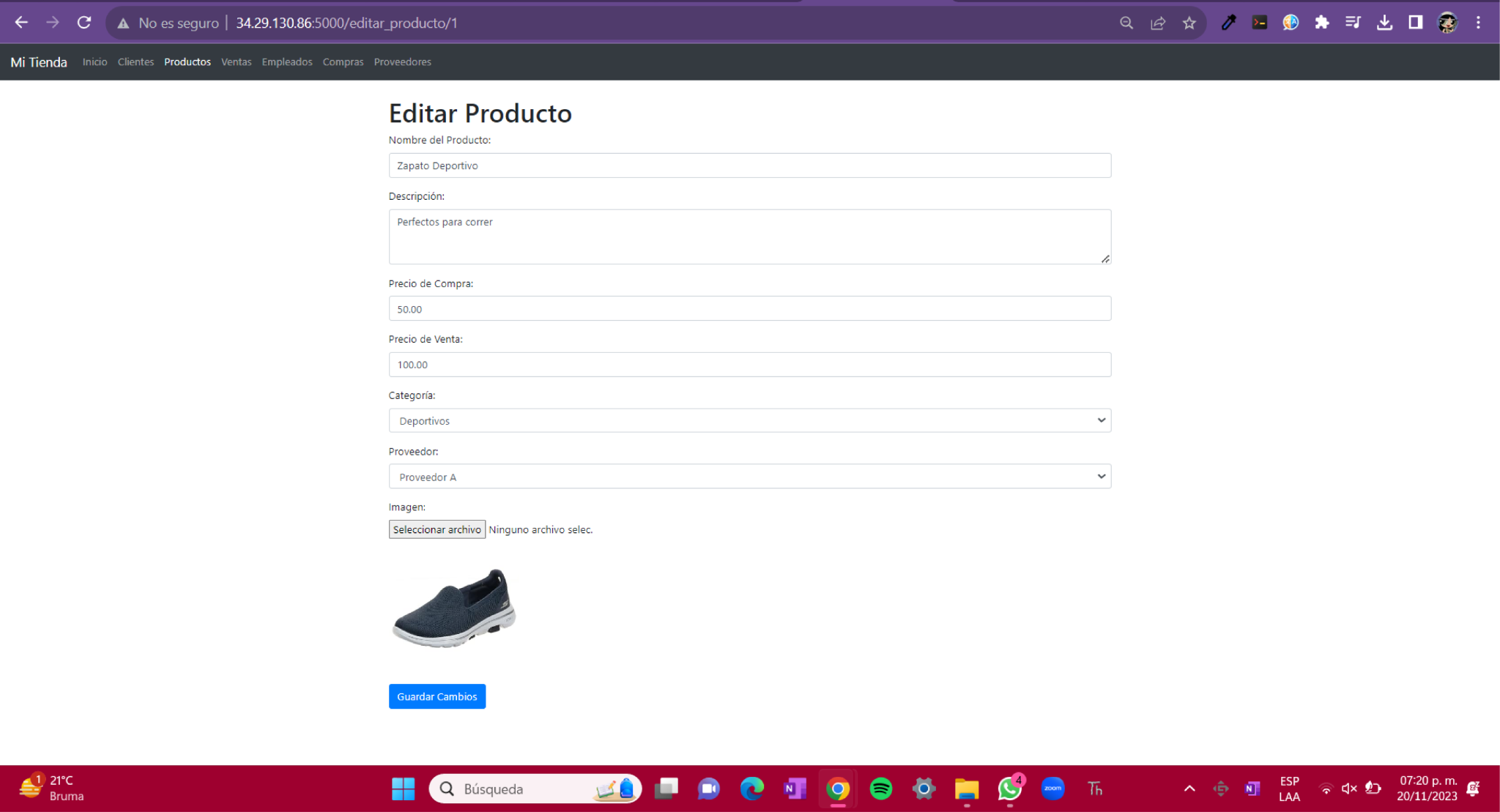


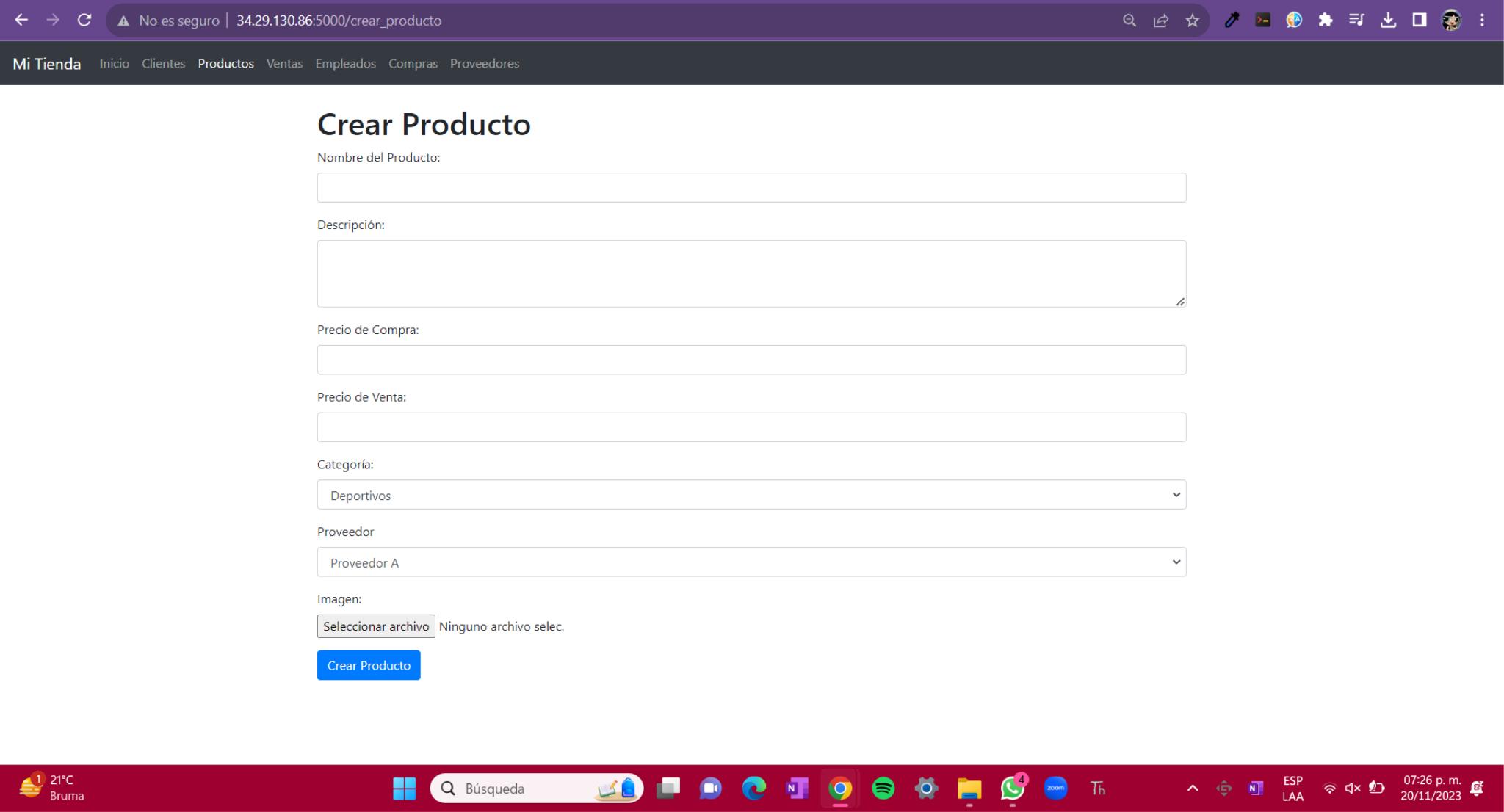


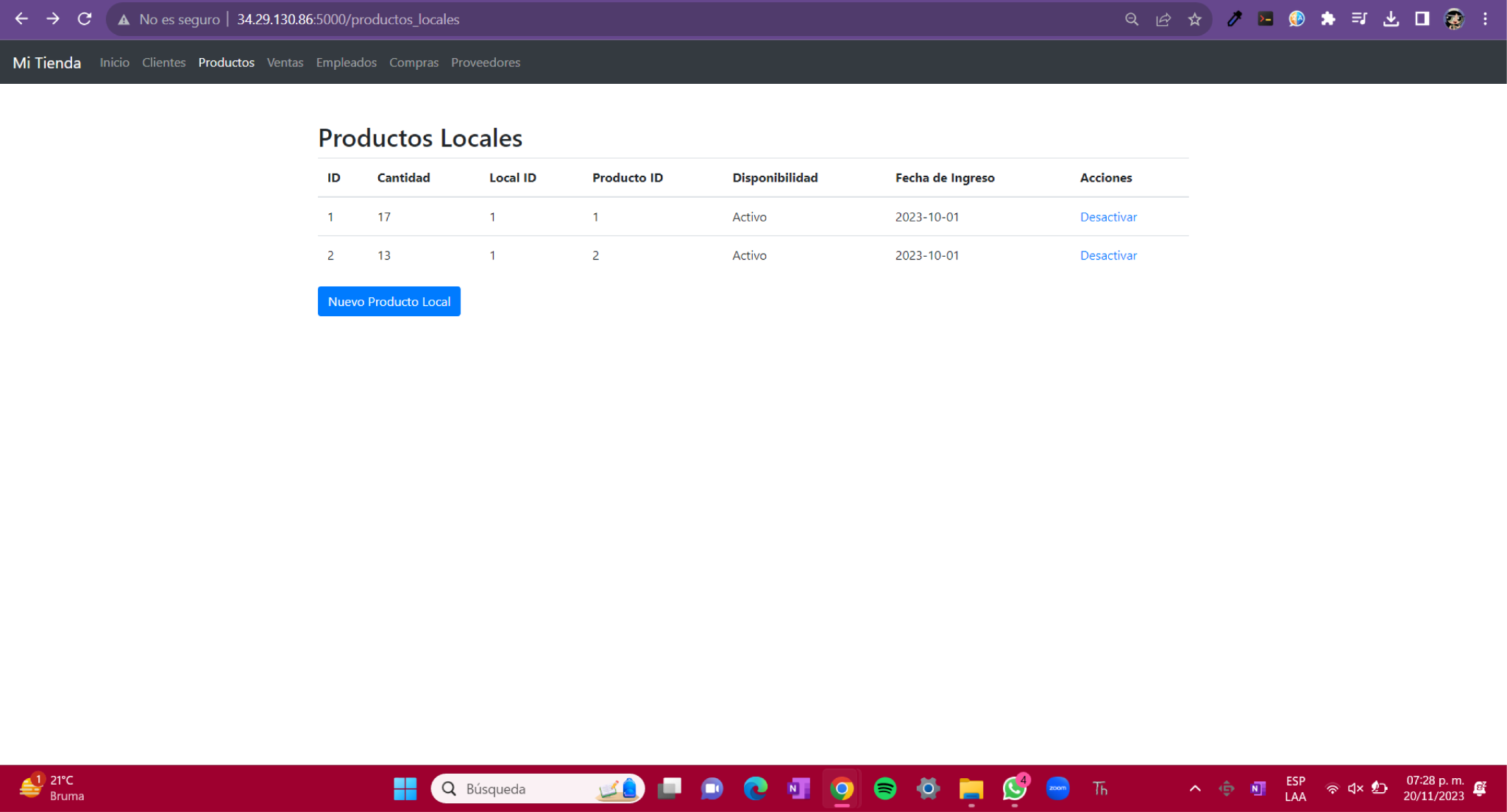


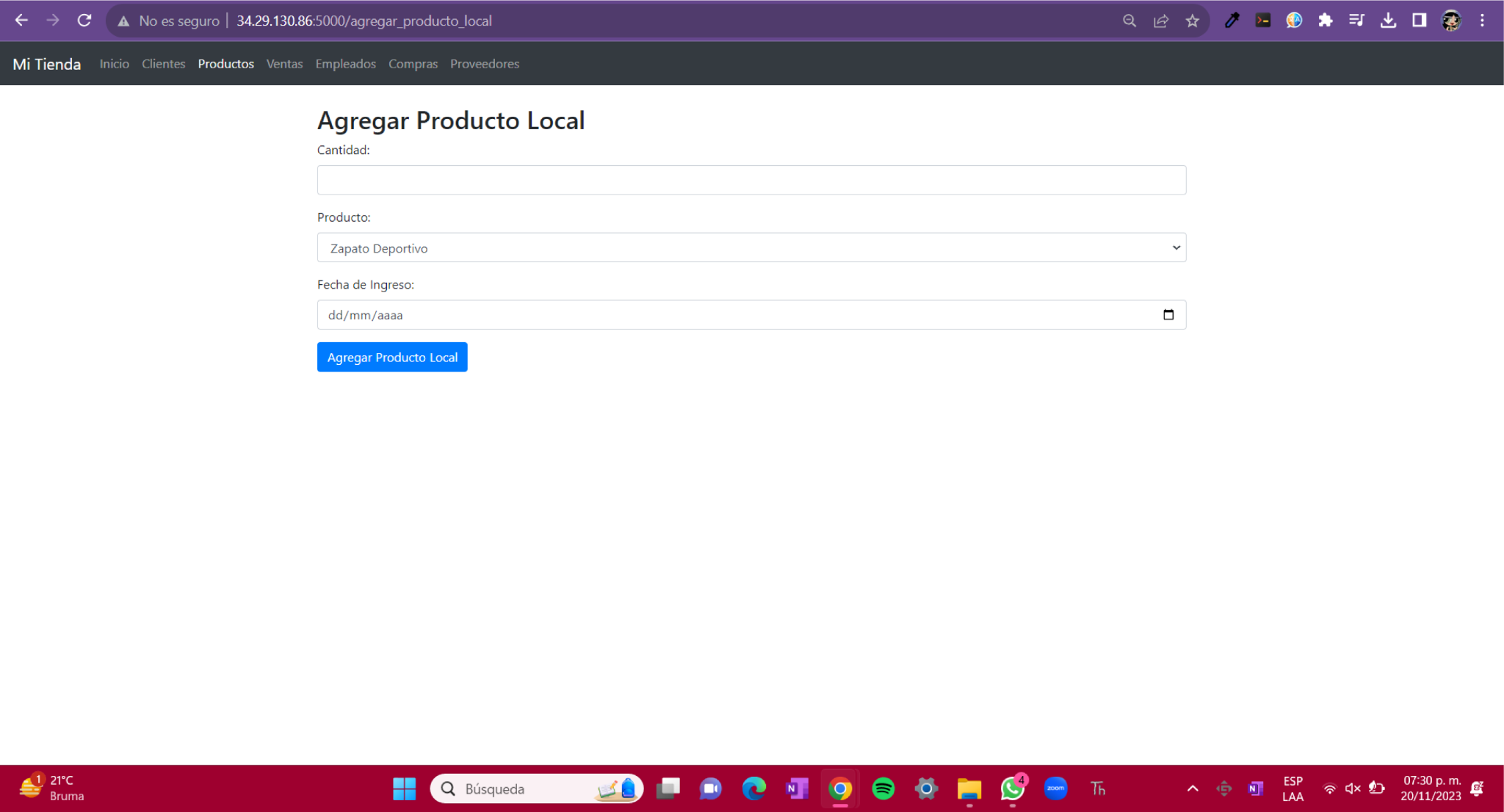


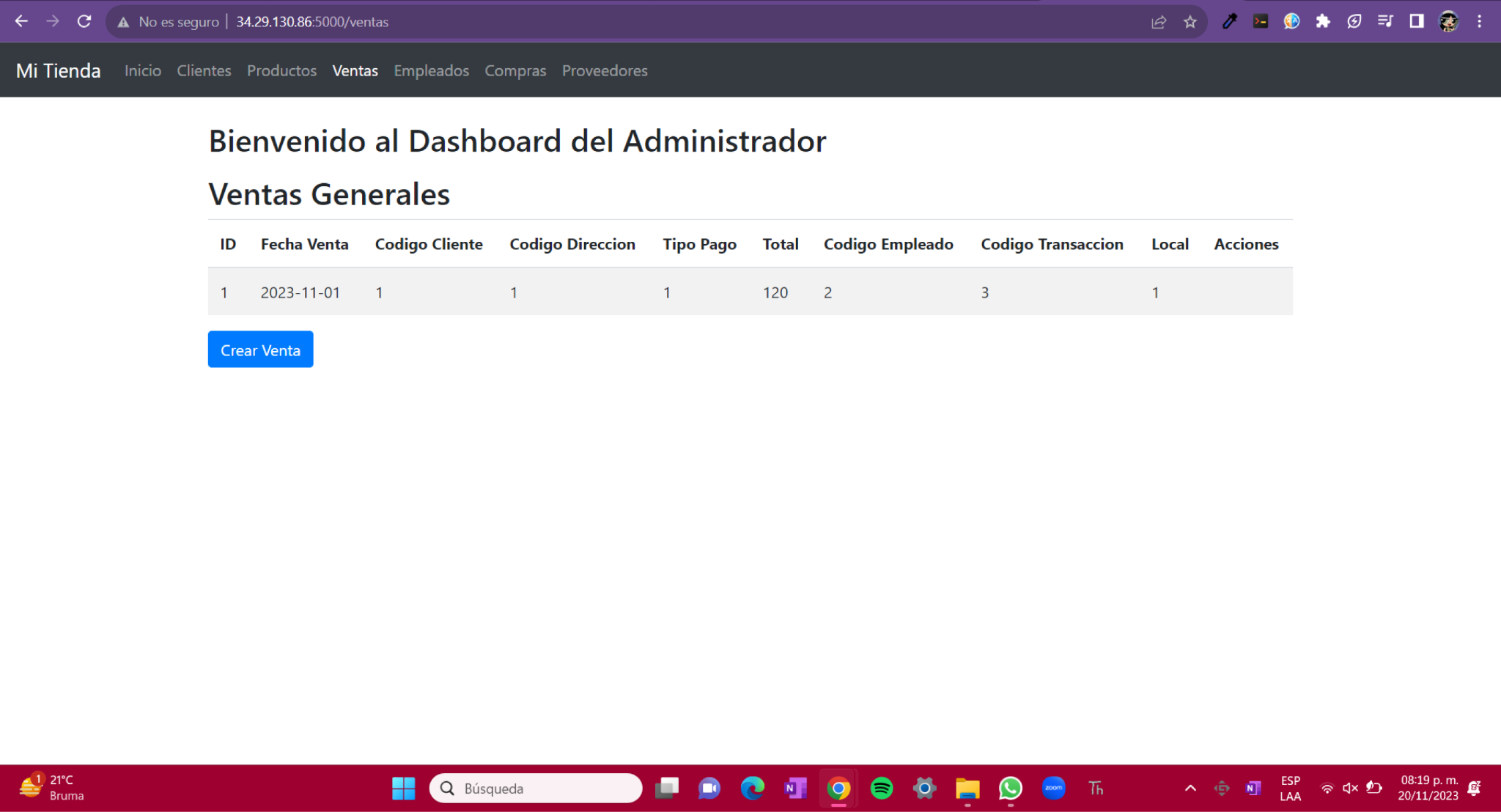


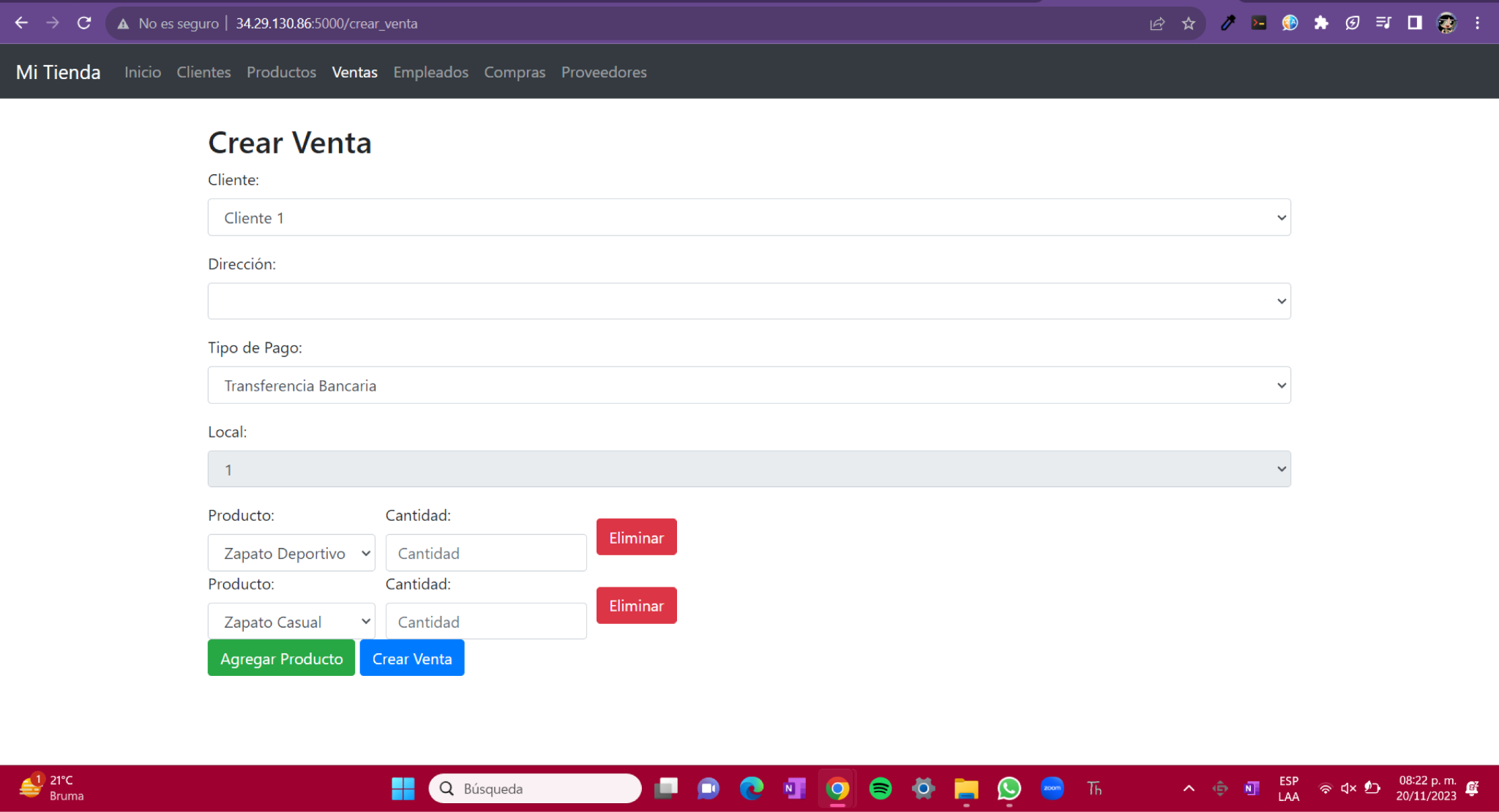


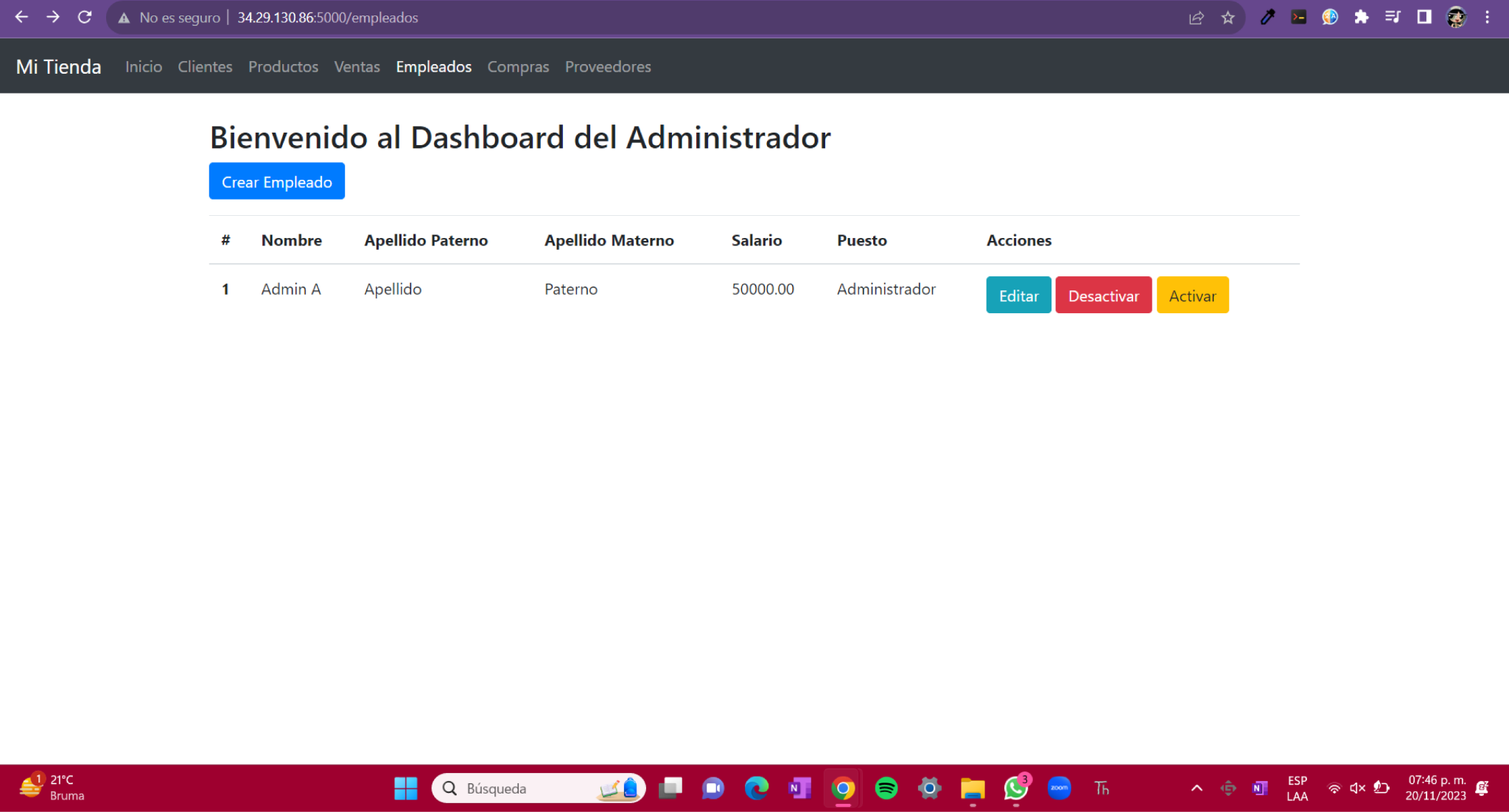


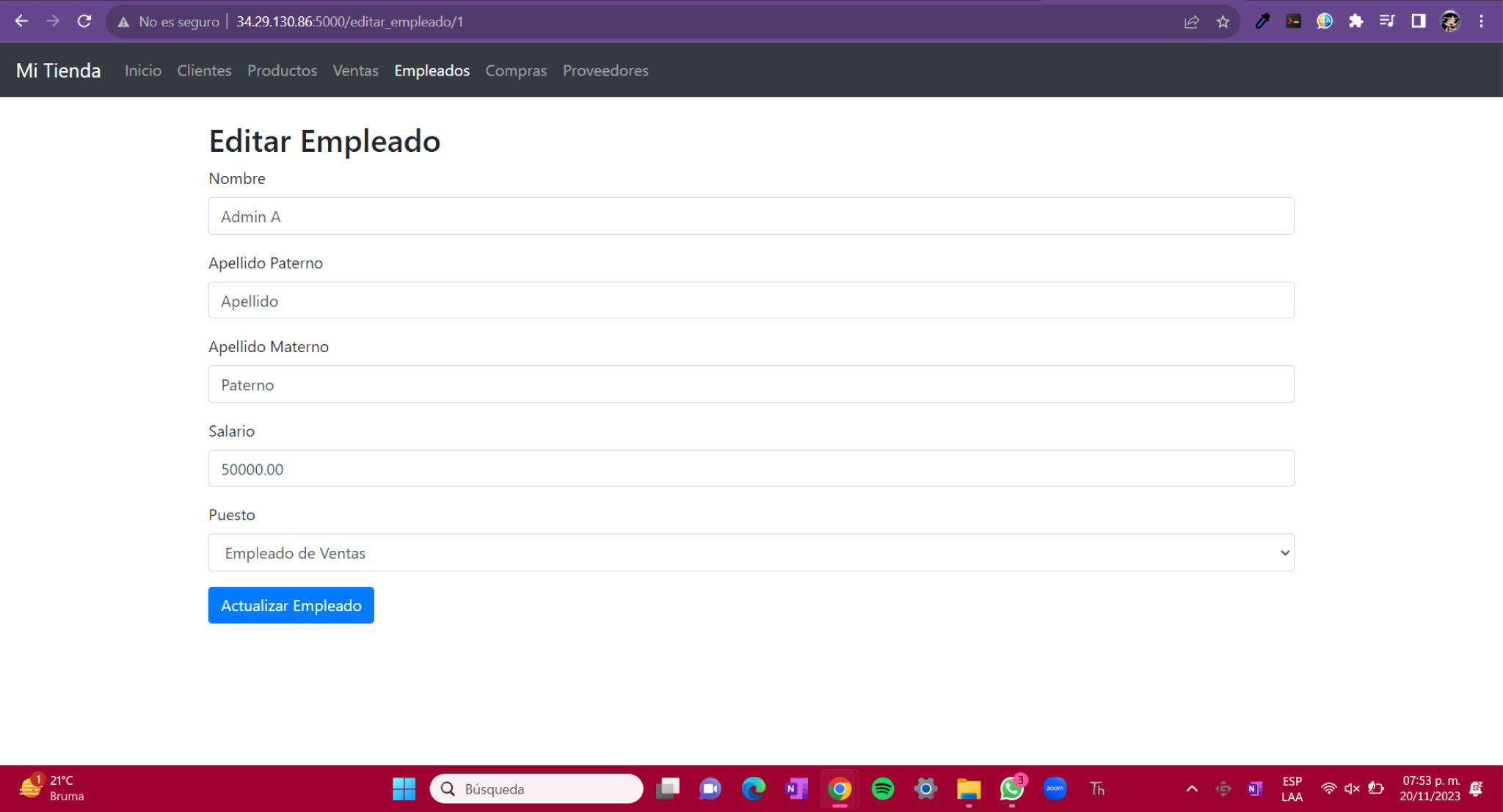


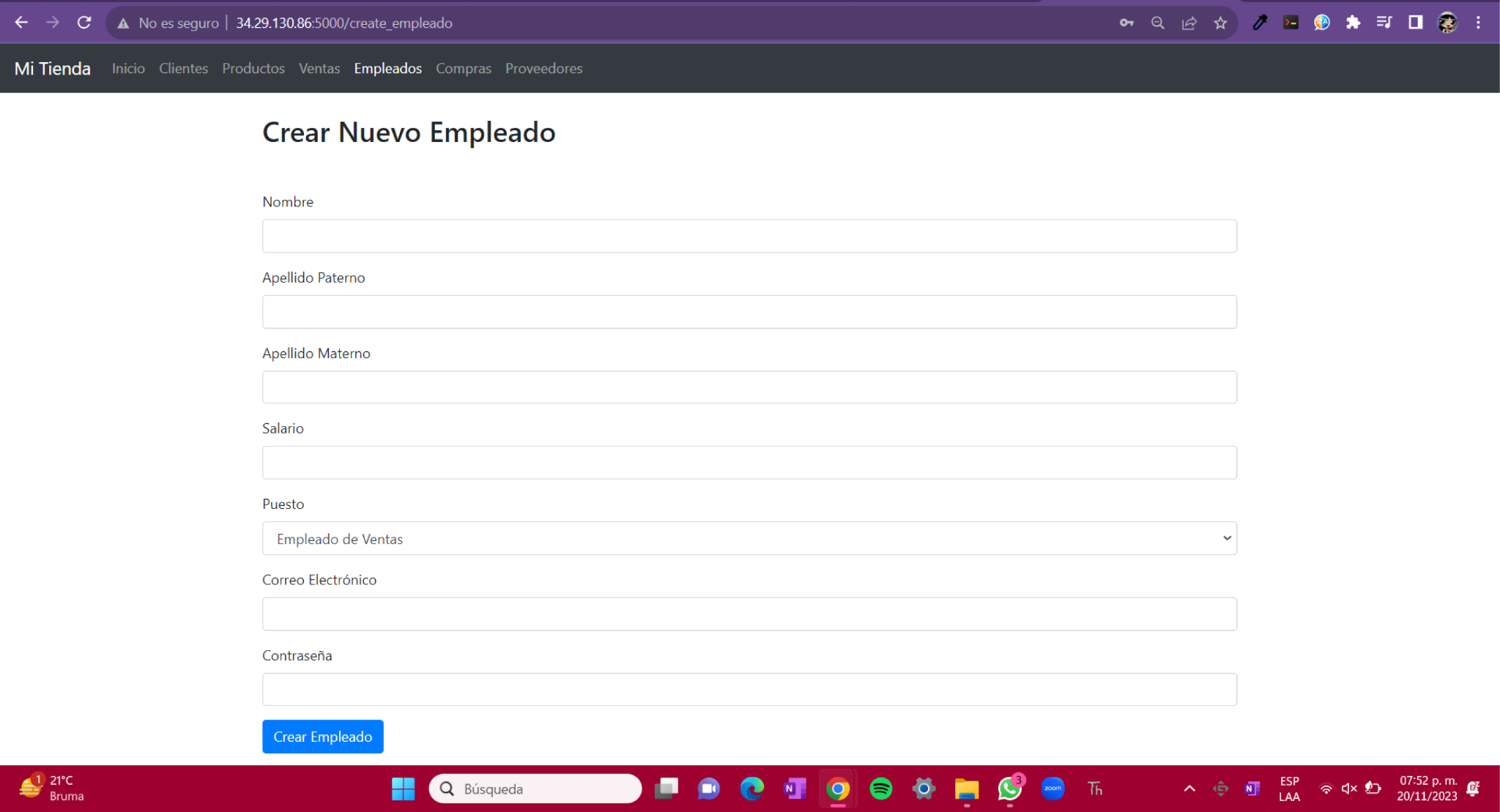


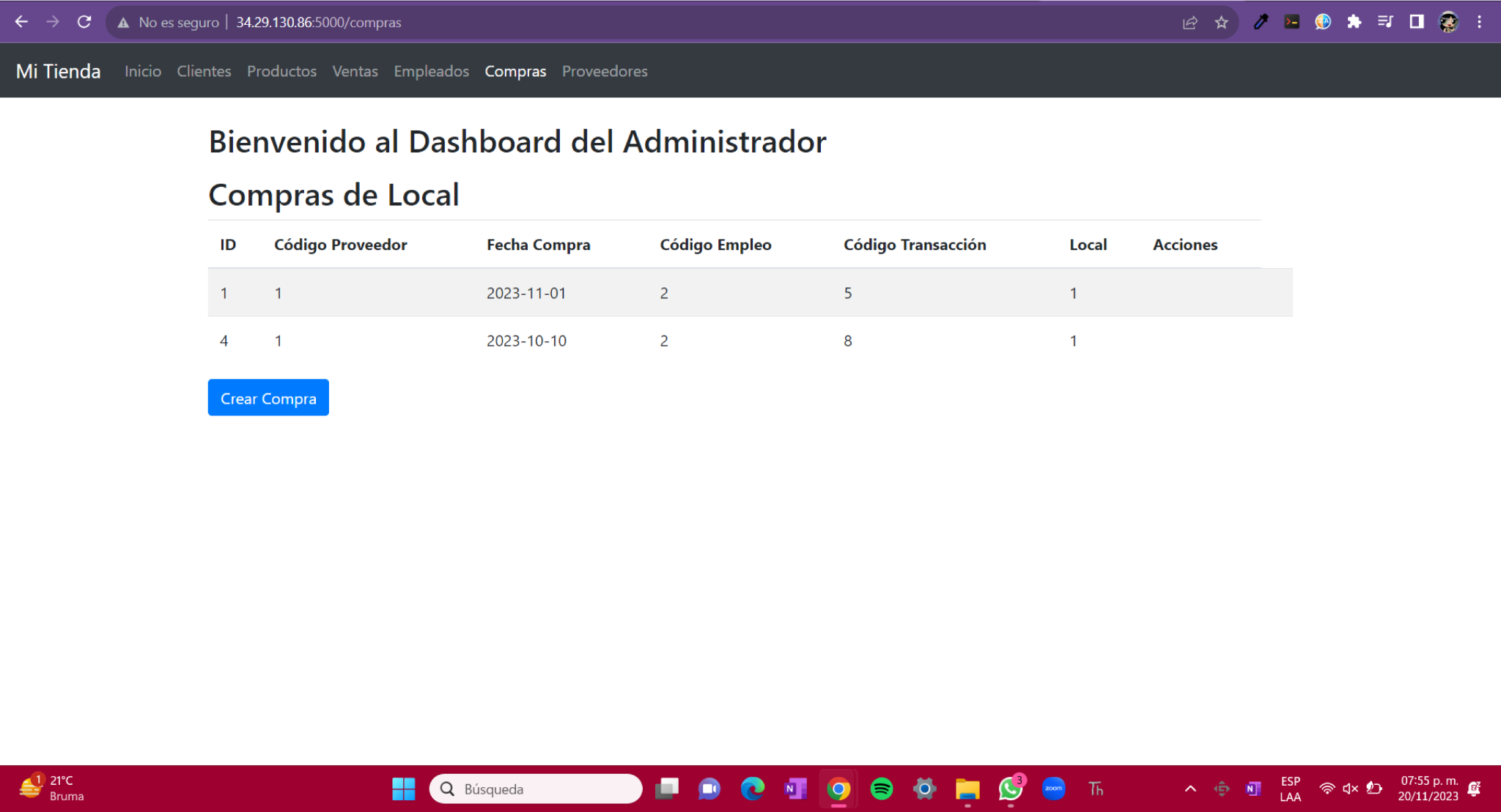


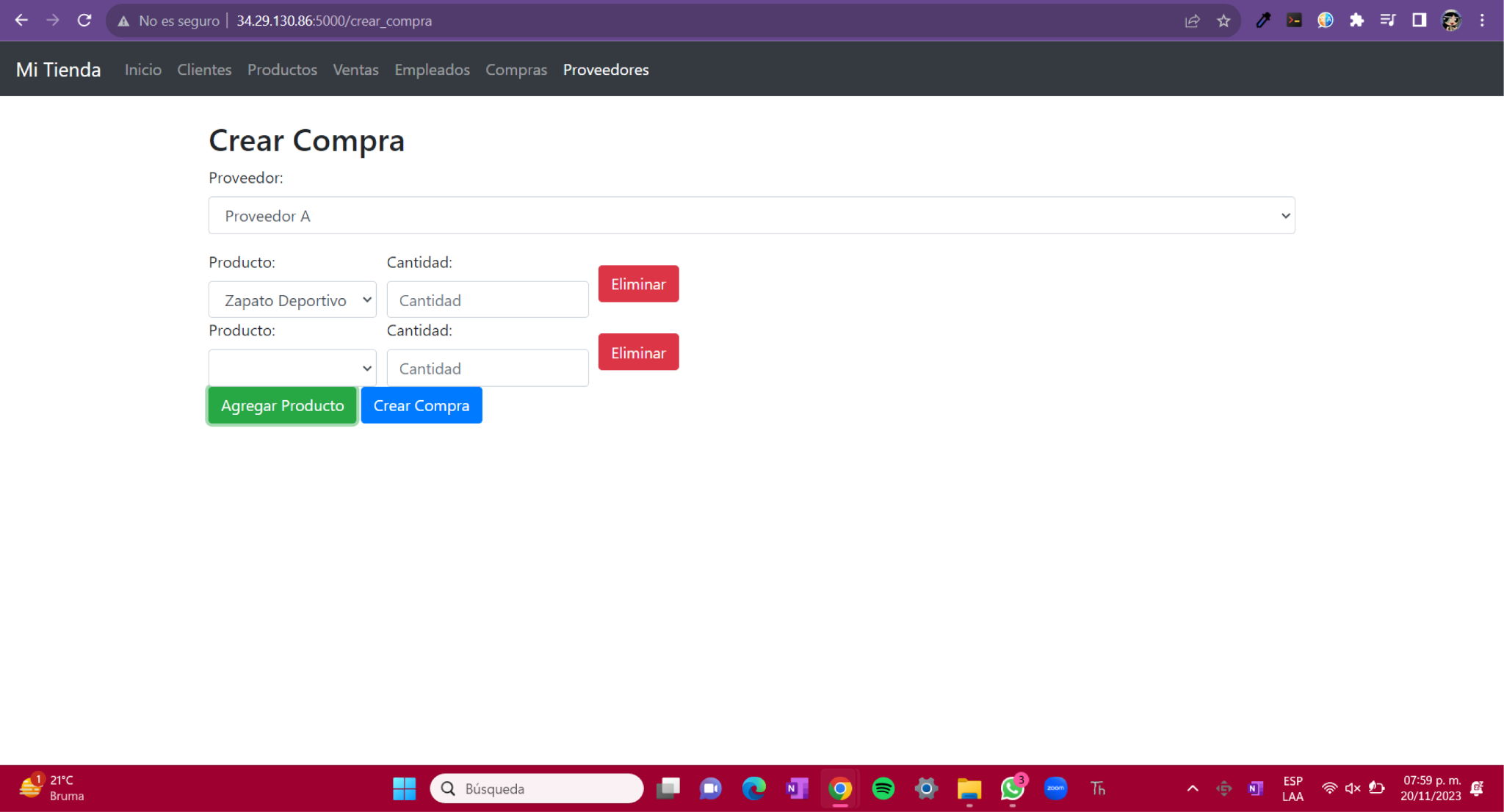


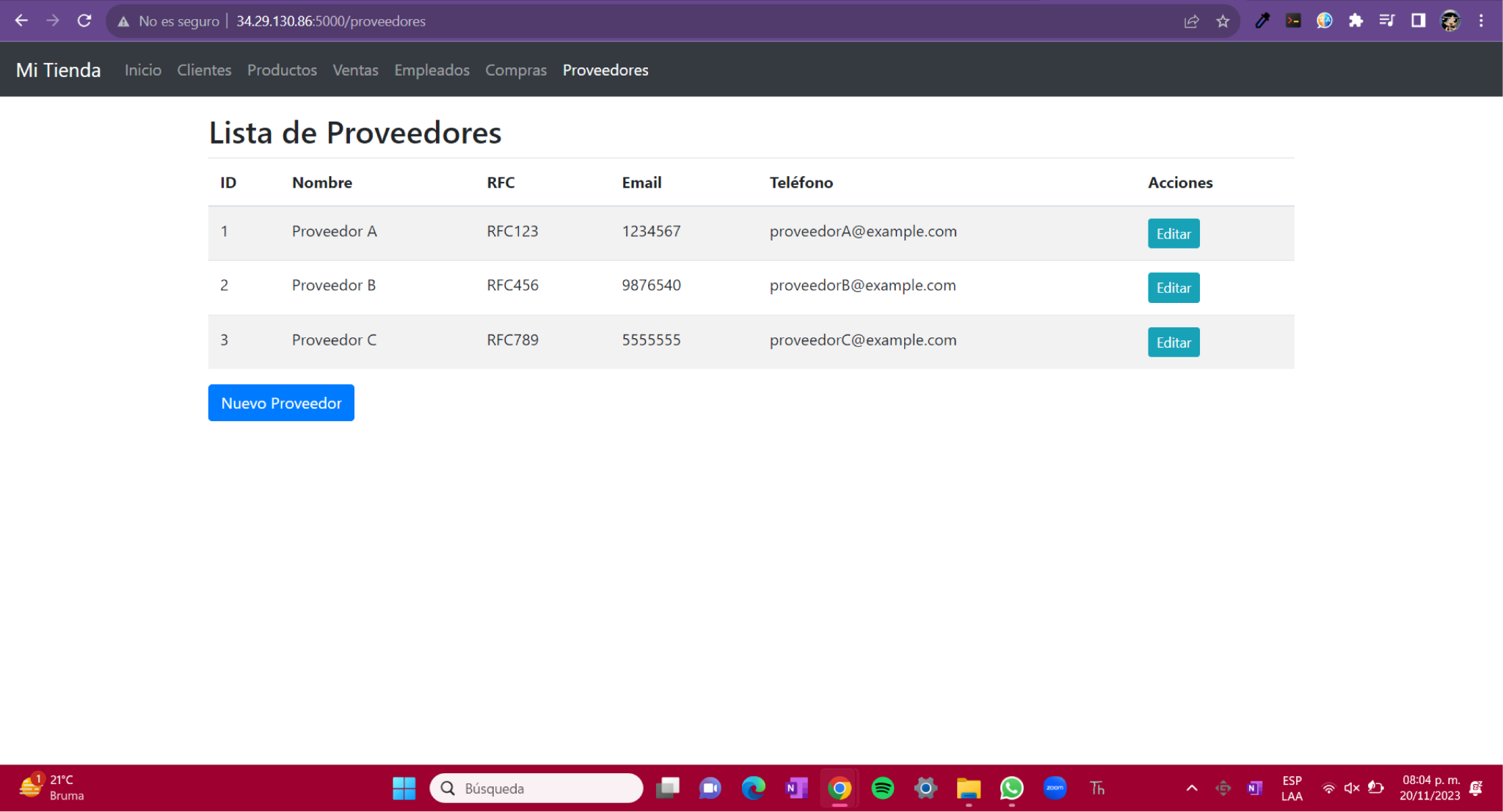


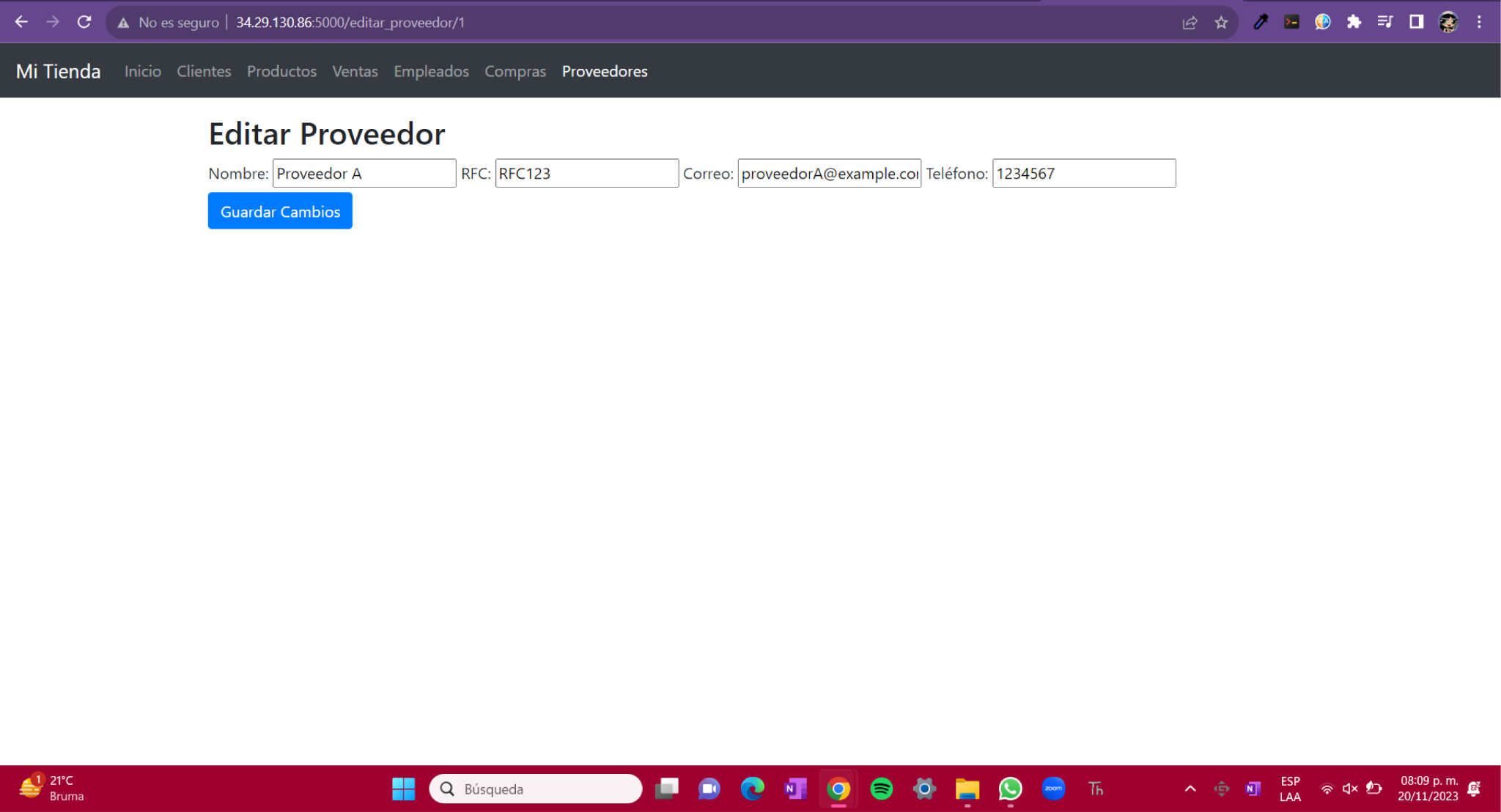


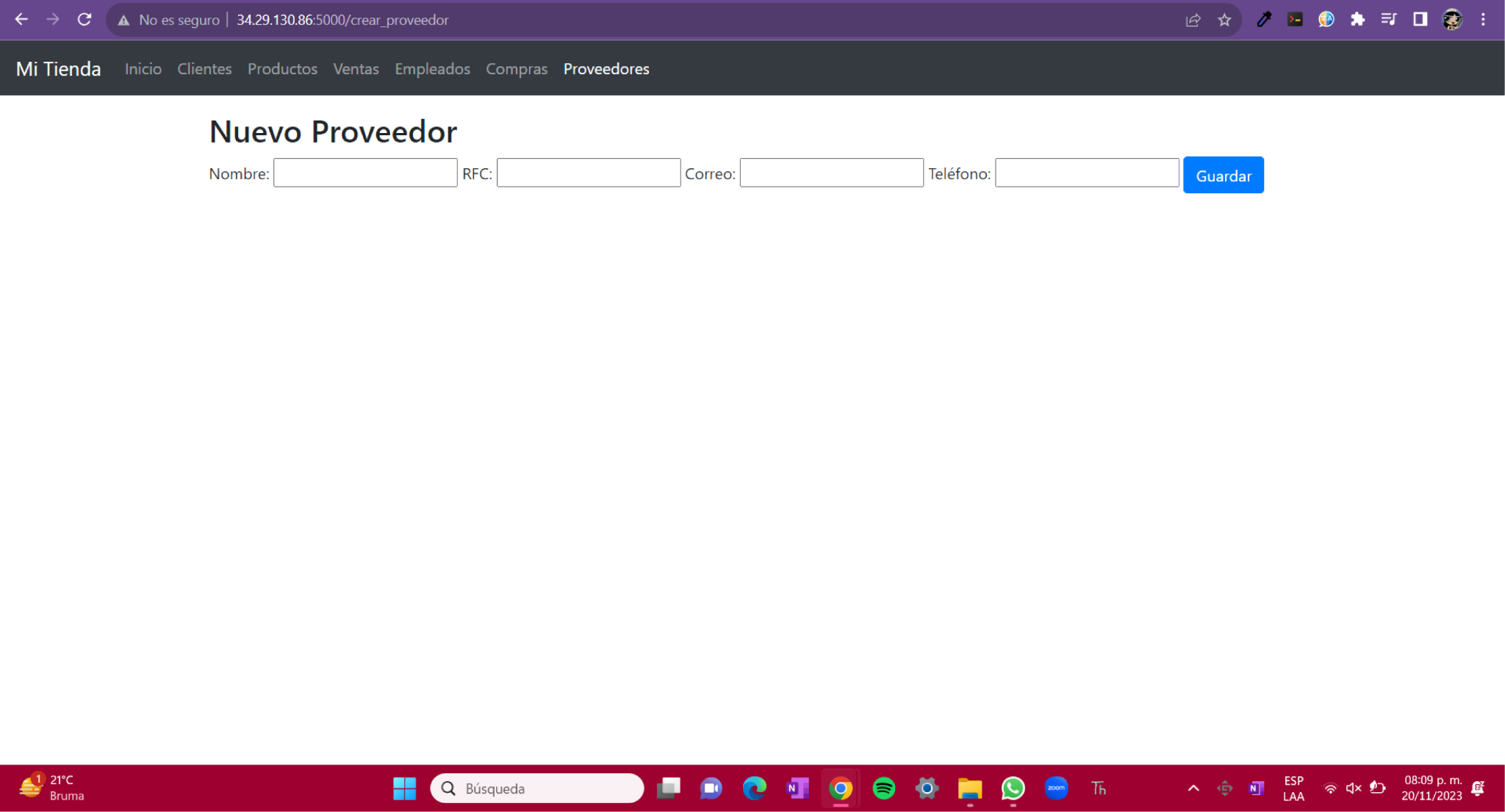


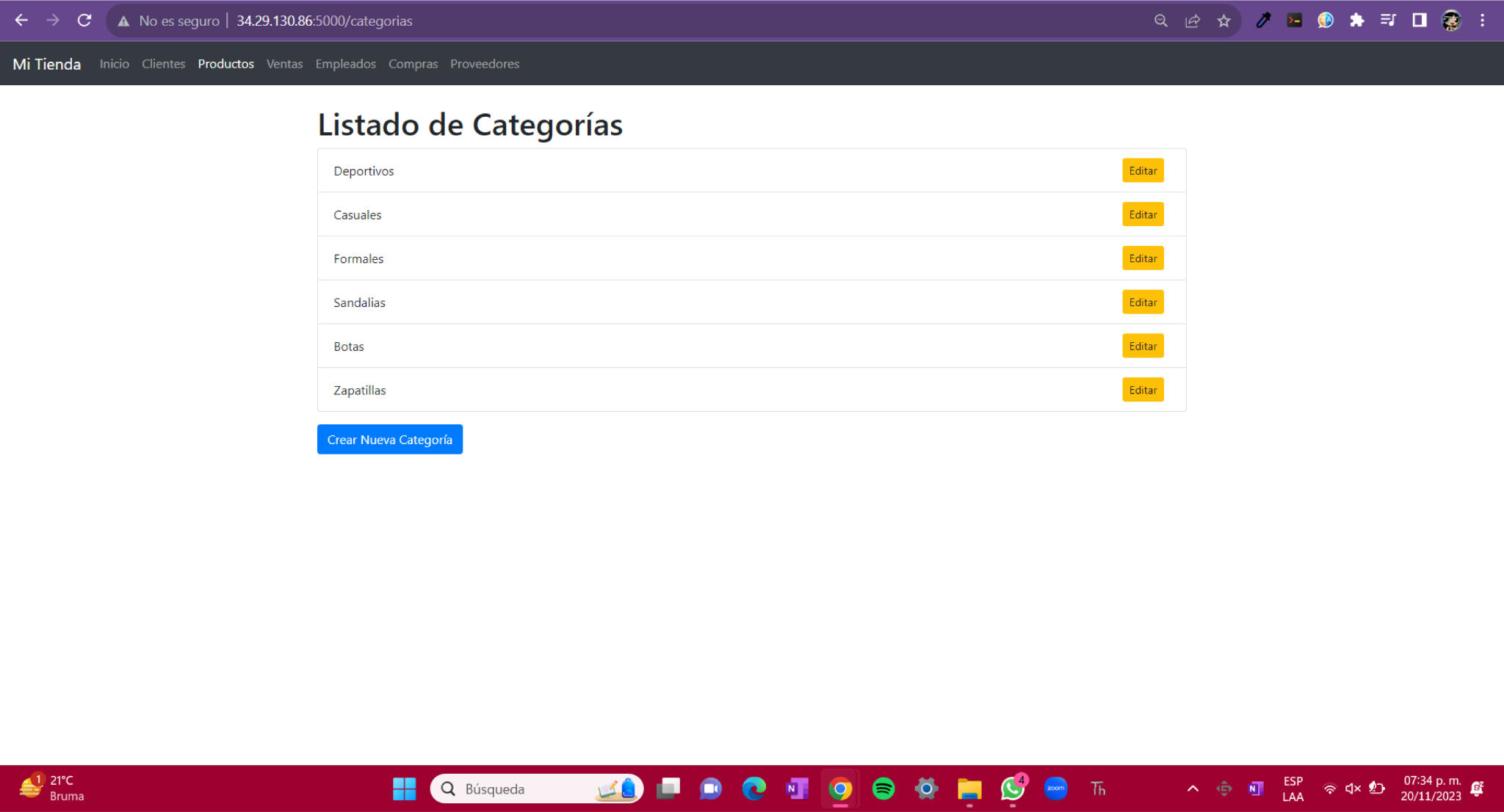


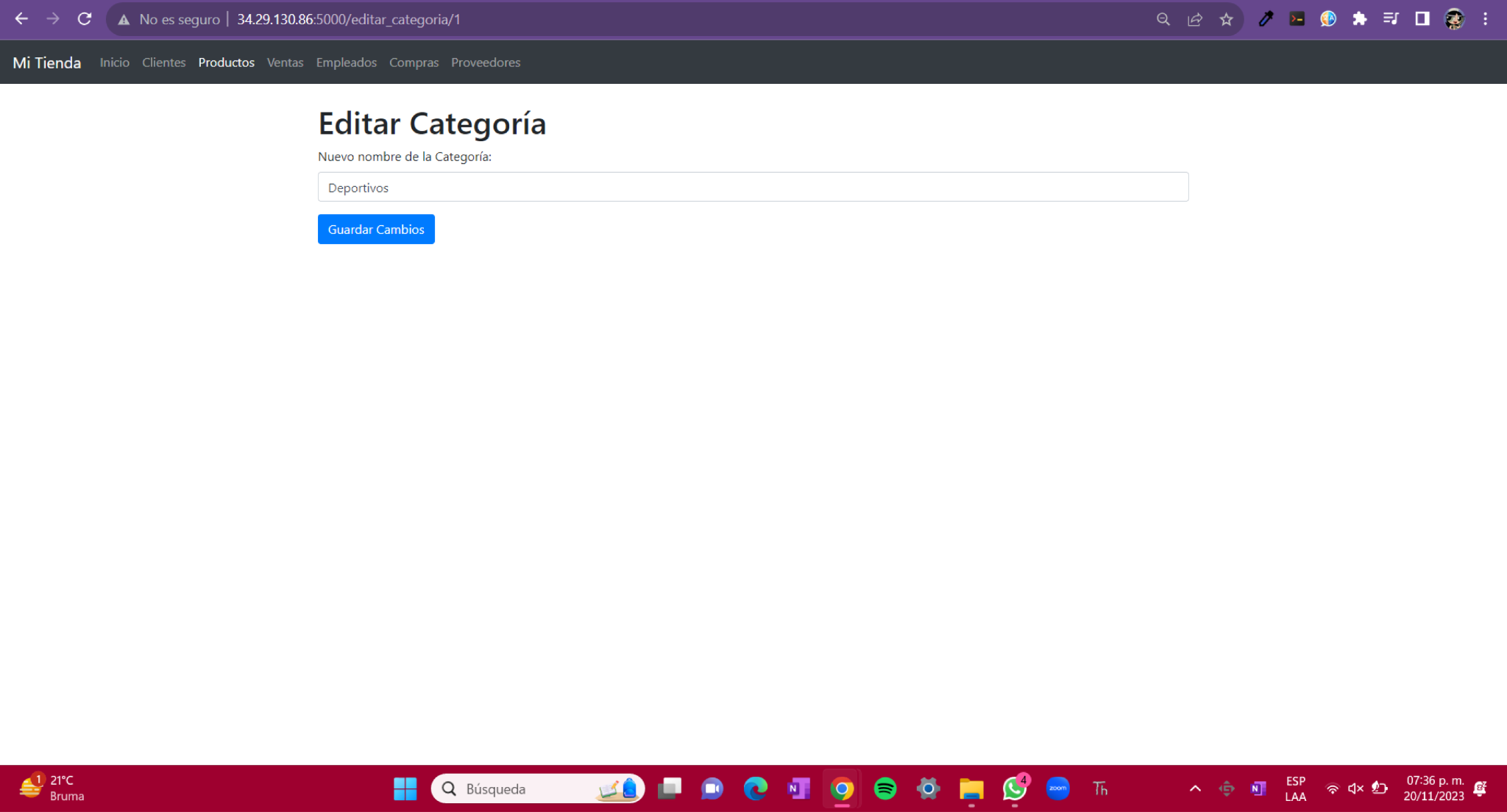


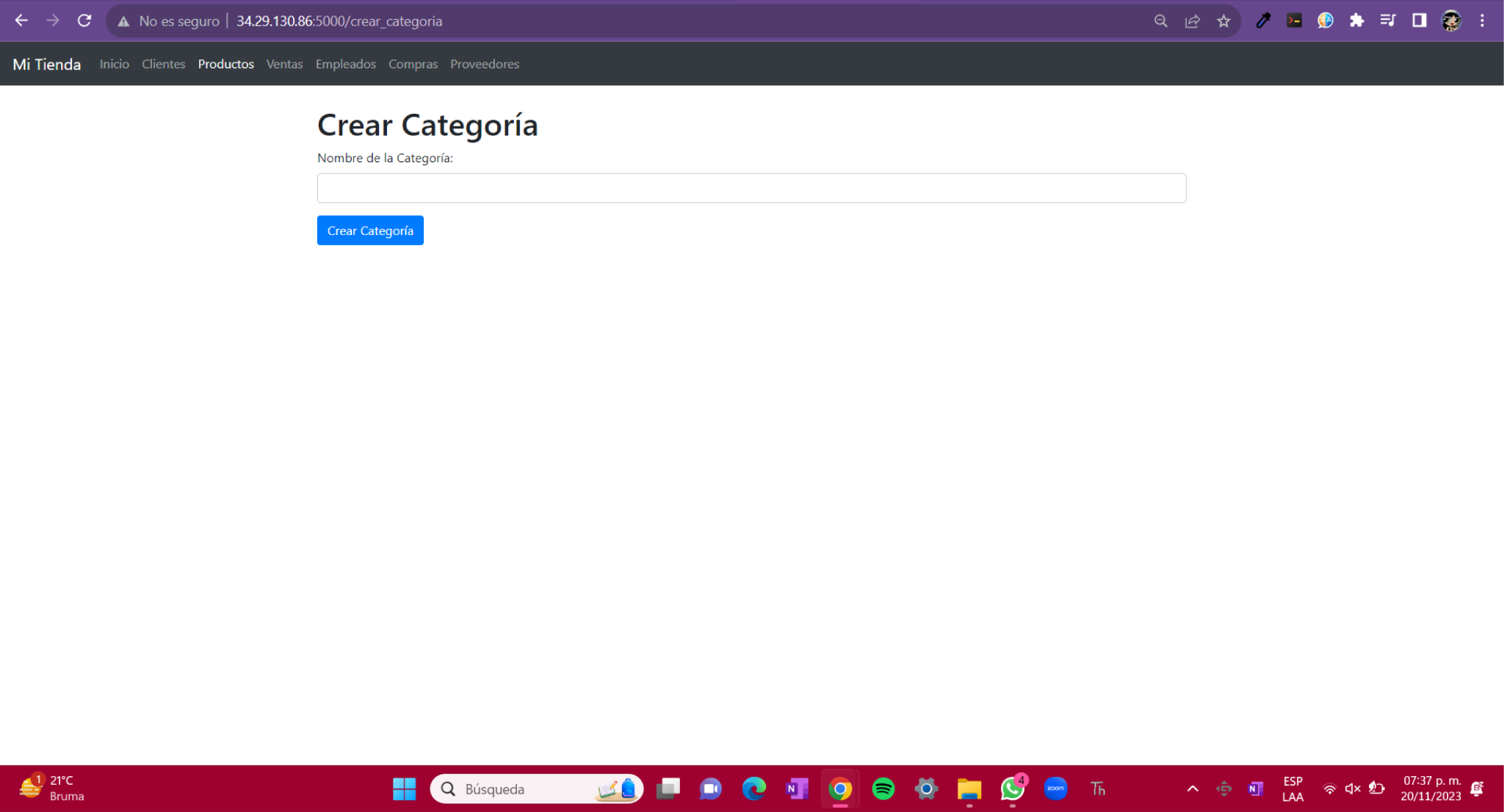












**Conclusiones**

En conclusión, la realización de este proyecto trajo consigo varios retos pero también aprendizajes que es lo importante dentro de todo. Uno de los retos más evidentes fue la longitud del proyecto, si bien este no fue tan complejo, si nos demoró mucho tiempo realizarlo ya que los requerimientos fueron varios.

Asimismo, tuvimos que investigar sobre la marcha como es que se integraban los stored procedures e integrar un inicio de sesión en nuestra app, ya que es algo que nunca habíamos visto en clase, y cuando por fin se vió, el tiempo que teníamos para hacerlo ya era algo corto.

Sin embargo, este proyecto también no permitió reforzar los conocimientos adquiridos a lo largo del curso ya que realmente este proyecto requirió que aplicaremos todo lo visto en bases de datos, procedimientos almacenados, aplicaciones flask en la plataforma de google cloud y la normalización de las tablas de la base de datos, por lo que podemos decir que somos capaces de desarrollar más proyectos de este estilo en un futuro.

Por último, queremos agregar que tras desarrollar el sistema de autogestionamiento para microempresas de la comercialización de calzado, se puede ver como gracias a nuestro proyecto, la situación de ets micronegocio mejora al ahora poder contar con una aplicación que les permite gestionar los inventarios de sus locales, a sus empleados , clientes y proveedores, saber el total de las ventas y compras realizadas, así como hacer adecuaciones a su catálogo de productos en todo momento a un precio accesible, para evitar el bajo nivel de servicio, la acumulación de stock y los altos niveles de ineficiencia.

**Video de demostración**

En este apartado se incluye el link de YouTube del video de demostración de nuestro proyecto final.

Link: <https://youtu.be/0nC5SCZv4j8>

**Referencias**

* Organización Internacional del Trabajo (OIT). (2023). *Comercio.* Recuperado el 18 de noviembre del 2023 de: <https://www.ilo.org/global/industries-and-sectors/commerce/lang--es/index.htm>
* Secretaría de Economía (2015). La industria del calzado en México. Recuperado el 18 de noviembre del 2023 de: <https://www.gob.mx/se/articulos/la-industria-del-calzado-en-mexico>
* Dussel E. Pérez L. (2020). *Nuevas condiciones, retos y oportunidades de la cadena del calzado en México post-Covid-19.* Recuperado el 18 de noviembre del 2023 de: <https://d1wqtxts1xzle7.cloudfront.net/63868283/CondicionesRetosOportunidadesPostCovid1920200708-127627-h1psn8-libre.pdf?1594265958=&response-content-disposition=inline%3B+filename%3DNuevas_condiciones_retos_y_oportunidades.pdf&Expires=1700369270&Signature=dxACyZ9xZmCMzwKwypHH-cehgkcLfEO~0bmEef~2Kcdd5gkgA92nhdKLAHRO5lufcn5Kf~EqwiBXAVjZ97QROCiUl0CgE2c7h5Iz7ck3LU~OAqKazOiZkjtw-lAb60LhaOdajUFkU3caNF0FsKz6g6NM~zPNzItnyumf2gCNl95Lku7Hh3tNeIqIB0q4ibxcK5yC5hab5Rv1oM9eLXKJVfkFZUs6BmBHjj1SCAd7vSYDWXcPvvt4dymeuxWFfP2DaeEXIJpo67kOcWMsi2piuUceeAI9zuBHgOP9cv4sugp~HsG7j5qROXTU-dIB~YyghJhXPeAwiTE-WTtoWd8Dwg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>
* González J. C. (2023). *Microempresas: Clave para el crecimiento económico y el empleo en México.* Recuperado el 18 de Noviembre del 2023 de: <https://www.mexicoemprende.org.mx/que-es-una-microempresa/>
* BBVA. (2023). *¿Qué es una microempresa?* Recuperado el 18 de noviembre del 2023 de: <https://www.bbva.mx/educacion-financiera/creditos/caracteristicas-microempresas.html>
* Soto P. & Cuervo M. (2023). *Base de Datos Avanzadas - Normalización Tablas Proyecto Final. [hoja de cálculo de Excel].* Recuperado el 21 de noviembre del 2023 de: <https://docs.google.com/spreadsheets/d/1uiRNHyZ_Mj4Mv_2X6UG4hCY-HMJTAPYHIjWm8XFbB-g/edit?usp=sharing>
* Soto P. & Cuervo M. (2023). *Video de demostración del proyecto.* Recuperado el 24 de noviembre del 2023 de: <https://youtu.be/0nC5SCZv4j8>