**Informe de Proyecto: Migración de Sistema monolítico a microservicios**

**Perfulandia SPA**

****

**Desarrollo Fullstack I**

**Docente:**

**Victor Rosendo**

**Integrante:**

**Simón Tobar**

**Fecha Entrega:**

**31/03/2025**

**Contexto:**

Perfulandia SPA es una empresa chilena que ha tenido éxito en nuestro país, desde sus orígenes en el barrio Meiggs, hasta la apertura de nuevas sucursales a lo largo del país, ofreciendo productos de calidad a precios competitivos dentro del rubro de la venta de perfumes.

La adaptación digital de Perfulandia inicialmente empezó con un sistema monolítico que permitía distintas funciones de venta, tanto para generar ventas online, como presencial en sus sucursales y todo esto manejado bajo una jerarquía administrativa para el control del inventario en una base de datos en el software propio de esta empresa.

Dicho sistema actualmente, luego de la extensión de la empresa a múltiples sucursales, ha generado una falla del sistema que presenta problemas de rendimiento y disponibilidad que ponen en riesgo las operaciones que se generan de forma diaria para la empresa, lo que ha reducido las ventas importantemente.

Los objetivos principales de este proyecto es migrar toda esta estructura monolítica a un software que utilice una estructura basada en microservicios, para mejorar el rendimiento y de esta forma también prevenir problemas futuros a la empresa si esta sigue creciendo y pueda alcanzar incluso a instalarse fuera del país para ejercer el rubro.

**Estrategia de Microservicios:**

El tipo de estrategia principal es el uso de los microservicios por funcionalidad para este proyecto, por múltiples razones:

* Servicios para el almacenamiento y manejo de bases de datos, al ser una empresa orientada a ventas, es muy importante tener el control general de stock, precios e información relevante de los empleados para generar consultas que agilicen el cálculo de utilidades por cada sucursal.
* APIs de servicios para pagos, creaciones de cuentas para usuarios del sistema, para brindar múltiples opciones a los clientes como tal, y no depender de un mismo sistema de pagos, por ejemplo, para que los usuarios tengan flexibilidad a la hora de comprar un producto, y tener información imprescindible de estos, como, por ejemplo, la dirección en casos de envío, el correo personal de estos para informar de promociones, pagos, etc. Y en caso de que uno de estos servicios falle, exista posibilidad de que los usuarios puedan escoger alternativas y no se pierda información crucial que afecte el ritmo de ventas.
* Integración entre aplicaciones del sistema, esto permitirá que la empresa como tal pueda brindar herramientas de venta que se conecten entre sí, al sistema agilizando aún más las ventas, como, por ejemplo, aplicaciones móviles para los usuarios, páginas web, etc.

Los beneficios de esta estrategia principalmente constan en la agilidad de ventas, ya que el sistema actual no permite múltiples ventas, por colas innecesariamente largas, pérdidas de datos, etc.

Además, el optar por este nuevo sistema permite que utilicemos herramientas de informática que son fáciles de manejar, son amigables a cualquier tipo de hardware, lo que no resultará en un cambio drástico de este, y también funcione ante cualquier tipo de cliente.

Dichas herramientas mencionadas anteriormente, son las siguientes:

* Lenguaje de programación (Java) trabajada en la IDE de JetBrains IntelliJ IDEA, Java permitirá un desarrollo ágil de este sistema, aplicación, claramente todo el sistema tendrá este único lenguaje y es importante tenerlo en consideración ante cualquier cambio futuro que se pueda beneficiar el sistema. El IDE de Java utilizado también permitirá crear conexiones a la base de datos y otros servicios de manera segura y eficiente.
* Lenguaje de base de datos (SQL) conectada mediante Laragon, SQL es un lenguaje universal de base de datos, que nos permitirá almacenar tablas con datos de los productos, empleados, clientes, etc., por sucursales. También programaremos consultas que, mediante estas conexiones, se podrá manejar por completo la base de datos en el software del sistema, con los permisos adecuados tanto si es un administrador, empleado de la empresa, como si también fuese un cliente, que pueda el mismo ver sus ventas, su información y cambiarla si es necesario.
* Framework SpringBoot 3 (Java), uno de los frameworks más utilizados en la industria, que tiene múltiples ventajas y principalmente se utilizará para la creación del software, SpringBoot se destaca por tener GraalVM Native Image Support, que permite compilación AOT (Ahead of Time) para construir aplicaciones que utilicen bajos recursos, utiliza versiones de Java por sobre la 17, que significa un mejor rendimiento y seguridad del sistema, tiene buena observación del sistema, que permite mejores métricas y tracking de información y principalmente Spring Web que servirá para conectar APIs al sistema de forma eficiente.

**Herramientas de Equipo:**

Actualmente, como sólo trabaja una persona en el proyecto, utilizaremos GitHub para crear los repositorios y guardar información relacionada al proyecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**Ética del Desarrollo:**

Privacidad de los Datos:

El acceso a la base de datos debe estar configurada en base a la jerarquía de los usuarios que accedan a él, por ejemplo, cada cliente debe ser capaz de ver los datos de productos, precios, poder asegurar la compra y guardarla en su cuenta, y en su misma cuenta poder modificar sus datos personales, mientras que la administración pueda manejar stock, precios, y hacer consultas de las ventas generadas en sucursales, por día, mes, etc.

Seguridad:

El sistema debe ser seguro ante cualquier ataque, o fallo que pueda presentar pérdida de información, se debe tener respaldo y acción inmediata a la resolución de cualquier fallo, al migrar a un sistema de microservicios, esto permite que se puedan manejar los problemas de seguridad de forma encapsulada mientras el resto de los servicios funciona con regularidad.

Despliegue:

El proyecto tiene un despliegue de trabajado organizado, con tareas especificadas por día y requerimientos específicos a trabajar según importancia del sistema, el retraso de cualquiera de estas tareas será informado y especificado para la reorganización de las tareas cumpliendo los plazos pertinentes.

La integridad de los datos se rige por las Leyes N° 19.628 y N° 21.719, el cual la primera garantiza el derecho a la privacidad y protección de datos personales de las personas naturales o físicas, y la siguiente regula el tratamiento de datos personales y crea la Agencia de Protección de Datos Personales, vigente a partir del 1 de diciembre del año 2026.

**Análisis de Requerimientos:**

Perfiles:

Cliente – Usuario promedio que utilizará el sistema para comprar

* Crear Usuario / Iniciar Sesión
* Buscar Productos / Agregar Productos a un Carrito
* Realizar Pedidos / Ver Historial de pedidos
* Gestionar Perfil
* Solicitar Reporte
* Dejar reseñas y evaluaciones a productos
* Aplicar cupones y descuentos

Administrador – Usuario con permisos de administración de cuentas, permisos, y revisión general del sistema

* Gestionar Usuarios
* Configurar Permisos
* Monitorear / Respaldar Sistema

Gerencia – Usuario con permisos sobre los administrativos, que administran utilidades y cálculos de ingresos en general

* Gestionar Inventario
* Gestionar Reportes
* Gestionar Sucursales
* Gestionar Pedidos

Ventas – Usuario que genera las ventas, tanto físicas como online

* Registrar Ventas
* Atender Devoluciones y reclamos
* Consultar Inventario
* Generar Facturas

Logística – Usuario que genera todo lo que tenga que ver con envíos y reabastecimiento

* Gestionar Envíos
* Optimizar Rutas de Entrega
* Actualizar estado de Pedidos
* Gestionar Proveedores

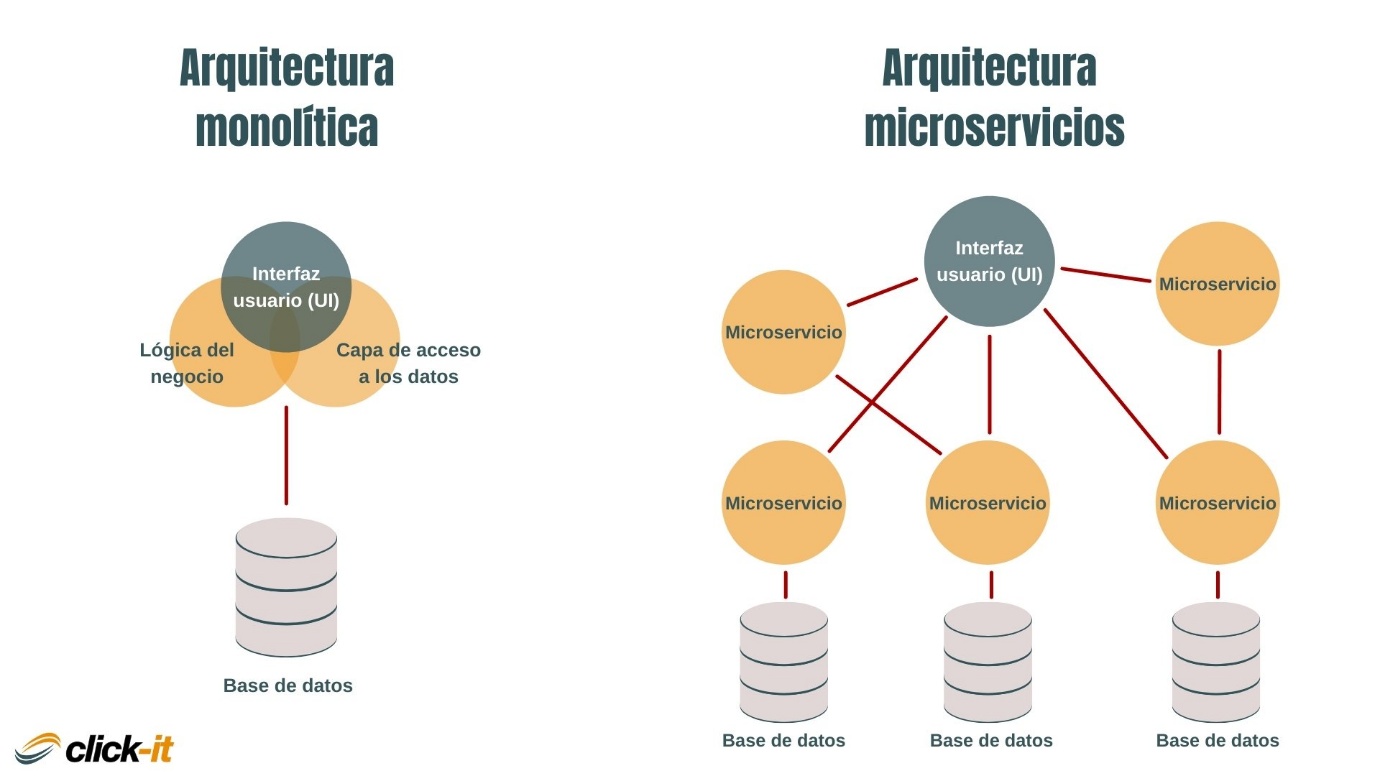
Sistema – En general, para motivos de seguridad y acceso a la información

* Acceso a la plataforma / Bases de datos
* Bloquear accesos no deseados

**Sistema actual:**

Por lo anterior, este sistema está basado en una estructura monolítica. Cuenta con múltiples fallas y cuellos de botella, debido al gran tránsito de información.

* Múltiples productos en diferentes sucursales requieren un tráfico de datos más rápido para los distintos tipos de tareas que tenga asignadas el sistema
* Las tareas que el sistema ejecuta mediante la web suelen ser más tardíos porque un mismo servicio está relacionado con todo



Dicho esto, en la figura anterior se muestra de ambas arquitecturas, la ventaja de la de microservicios es que al utilizar distintos de estos, se agiliza y se reduce el consumo de recursos, por lo que para este proyecto es crucial, además, se pueden incluir muchos más microservicios a futuro.

Diagrama

El contenido generado por IA puede ser incorrecto.**Diagrama de actores:**

**Diagrama

El contenido generado por IA puede ser incorrecto.Diagrama de Casos de uso:**

**Migración por Servicio:**

Definiremos el tipo de migración a una por servicios, por el simple motivo de que, la base de datos y el tipo de software monolítico que presenta es más simple de transportar y modificar que migrar los datos con un nuevo modelo, ya que el modelo actual de datos no presenta mucha complicación.

Por lo cual, primero se separarán los servicios bajo la misma base de datos. Luego, se considerará modificar algunos para adaptarlos a la nueva estructura.

Finalmente, al migrar los datos a la base de datos, se desarrollarán el resto de los servicios adaptados a la base.

**Consideración de compatibilidad:**

Si se tiene en cuenta que el lenguaje a desarrollar será todo exclusivamente en Java, si el código del sistema anterior es de otro lenguaje, o versiones de Java antiguas, deberá replantearse el código, y ya que SQL es un lenguaje universal, sólo se adaptaran las Querys y consultas a Java con su Framework de Spring Boot.

**Riesgos:**

Según la información de los requerimientos solicitados por la empresa, existen dos tipos de riesgos cruciales en esta migración.

La primera tiene que ver netamente con la base de datos, se evitará a toda costa la pérdida de información en la ejecución del plan. Para esto temporalmente crearemos bases de respaldo, y de la misma forma también estas se podrían implementar en el nuevo sistema, lo que implicará un mayor costo, pero mayor seguridad.

La segunda es, en temas de compatibilidad, adaptarse al hardware existente, hacer revisión de este puede que se necesite modificar algún que otro equipo, lo mismo si es que no hay un servidor físico, migrar los datos a una nube, por lo cual ambas opciones son un costo adicional.

**Planificación y Cronograma:**

Se utilizará el enfoque Scrum:

Se generarán Backlogs (Todo esto apoyado por GitHub) del producto, para tener información de las novedades, cambios y bugs arreglados.

Sprint Planning para establecer metas de tiempo para cada tarea, también con logs de los tiempos de trabajo.

Daily Scrums, reuniones diarias de menos de 15 minutos para comentar los avances, generar soluciones rápidas a problemas y cambios de ruta de ser necesario.

Hacer revisiones de cada implementación con notas y comentarios retrospectivos que apoyen el trabajo.

Por temas de tiempo y que este proyecto está organizado sólo por una persona, la Carta Gantt se omitió, pero se respetarán los tiempos e implementación del nuevo sistema.

**Conclusión:**

Este proyecto es netamente académico, por lo que el informe sirve para establecer un orden, herramientas y parámetros al momento de codificar.

Se investigó el tema de los frameworks, herramientas de programación para entender conceptos básicos pero imprescindibles en una documentación, ya que está dirigido a una empresa.

Gracias a la documentación también podemos intuir el futuro del sistema, su escalabilidad y sus mejorías. Lo bueno de este sistema es que está basado en una empresa de ventas, lo que indica que la parte más importante es el manejo de la información de lo que se ofrece, la seguridad de sus usuarios y la utilidad que esta genera. Por tanto, el futuro y mejoría del sistema depende netamente de las tecnologías futuras, todo en base a Java, cualquier informático que quiera trabajar en este sistema con la documentación adecuada, y la estructura de microservicios, puede aplicar herramientas recientes de programación con el fin de mejorar aún más el sistema.

Sin más que agregar, se despide

Simón Tobar.

**Anexo:**

Repositorio GitHub

<https://github.com/YungRaku/Project-DUOC-Fullstack>