Logotipo

Descripción generada automáticamente





Contenido

[1.](#_heading=h.23ckvvd) Introducción 3

[2.](#_heading=h.ihv636) Propósito 3

[3.](#_heading=h.32hioqz) Alcance 3

[4.](#_heading=h.1hmsyys) Equipo de Trabajo-Actores del Desarrollo 4

[5.](#_heading=h.41mghml) Recomendaciones de conformidad con esta práctica. 5

[6.](#_heading=h.2grqrue) Referencias y estándares aplicables a este documento: 5

[7.](#_heading=h.vx1227) Definiciones, acrónimos y abreviaciones. 5

[8.](#_heading=h.3fwokq0) DESCRIPCIÓN DE ARQUITECTURA: 6

[8.1](#_heading=h.1v1yuxt) VISTAS: 6

[8.2](#_heading=h.4f1mdlm) TIPOS DE VISTAS: 6

[8.3](#_heading=h.2u6wntf) Framework Conceptual 7

[8.3.1](#_heading=h.19c6y18) Descripción de la arquitectura en contexto 7

[8.3.1](#_heading=h.3tbugp1) Actores Usuarios y sus roles: 8

[8.3.2 Actividades de arquitectura en el ciclo de vida 8](#_heading=h.28h4qwu)

[8.3.3 Descripciones prácticas de arquitectura. 8](#_heading=h.nmf14n)

[8.4](#_heading=h.37m2jsg) Documentación de la arquitectura 8

[8.4.1 Identificación de los actores usuarios y sus responsabilidades 9](#_heading=h.1mrcu09)

[8.5](#_heading=h.46r0co2) Selección de puntos de vista de la arquitectura 9

[8.6](#_heading=h.2lwamvv) Vistas de la arquitectura 10

[UML-VISTA DE ESCENARIO 10](#_heading=h.111kx3o)

[UML-VISTA LOGICA 11](#_heading=h.3l18frh)

[UML-VISTA LOGICA 12](#_heading=h.206ipza)

[UML-VISTA DE PROCESO 13](#_heading=h.4k668n3)

[UML-VISTA DE DESPLIEGUE 14](#_heading=h.2zbgiuw)

[UML-VISTA FISICA 15](#_heading=h.1egqt2p)

[Usabilidad y Calidad de software 16](#_heading=h.3ygebqi)

[Usabilidad y Calidad de software 17](#_heading=h.2dlolyb)

[8.7](#_heading=h.sqyw64) Consistencia en la cantidad de vistas de la arquitectura. 18

[8.7.1](#_heading=h.3cqmetx) DESCRIPCION DE MODULOS 18

[8.7.2](#_heading=h.1rvwp1q) DESCRIPCIÓN DE COMPONENTES 18

[8.7.3](#_heading=h.4bvk7pj) DESCRIPCION DE CONECTORES 18

[9.1](#_heading=h.2r0uhxc) Acceso a fuentes de información del proyecto 19

# Introducción

Este documento presenta la arquitectura de software para el sistema de gestión de repuestos de autos, un proyecto cuyo objetivo es mejorar la eficiencia en el manejo y administración de inventarios, ventas y abastecimiento de repuestos. La implementación de este sistema permitirá gestionar de forma centralizada el inventario, agilizar el proceso de ventas y optimizar la relación con proveedores para garantizar la disponibilidad de productos en todo momento.

El sistema está diseñado para proporcionar a los usuarios una interfaz intuitiva y fácil de usar, que facilite las operaciones diarias y permita el monitoreo en tiempo real del stock, pedidos y ventas. Adicionalmente, la arquitectura del sistema está orientada a la escalabilidad y la adaptabilidad, permitiendo futuras integraciones y mejoras de acuerdo con las necesidades del negocio.

Esta documentación describe las principales decisiones de diseño y estructura los diferentes aspectos de la arquitectura en vistas que capturan tanto la funcionalidad como los elementos técnicos esenciales para su desarrollo.

# Propósito

El propósito de este documento es presentar la arquitectura de software para el sistema de gestión de repuestos de autos. La arquitectura descrita proporciona una base estructurada que facilita el desarrollo y mantenimiento del sistema, asegurando su escalabilidad y adaptabilidad a las necesidades futuras.

El documento incluye diversas vistas para ilustrar las decisiones arquitectónicas clave y asegurar que se cumplen los requisitos funcionales y no funcionales del sistema.

# Alcance

. Este documento describe el diseño arquitectónico para el sistema de gestión de repuestos de autos, que tiene como objetivo centralizar y optimizar los procesos relacionados con la gestión de productos y la relación con proveedores y clientes. El sistema buscará mejorar la eficiencia de los concesionarios, talleres y proveedores, brindando herramientas para una administración efectiva y en tiempo real de los repuestos automotrices.

El alcance del proyecto incluye:

La creación de una plataforma integral que gestione las operaciones de venta, abastecimiento, y control de inventarios de repuestos de autos.

El desarrollo de funcionalidades que permitan la integración de datos entre los actores del negocio (concesionarios, talleres, proveedores), optimizando los flujos de información y las decisiones comerciales.

La implementación de una infraestructura flexible y escalable, que permita la adaptación del sistema conforme las necesidades del negocio crezcan o cambien.

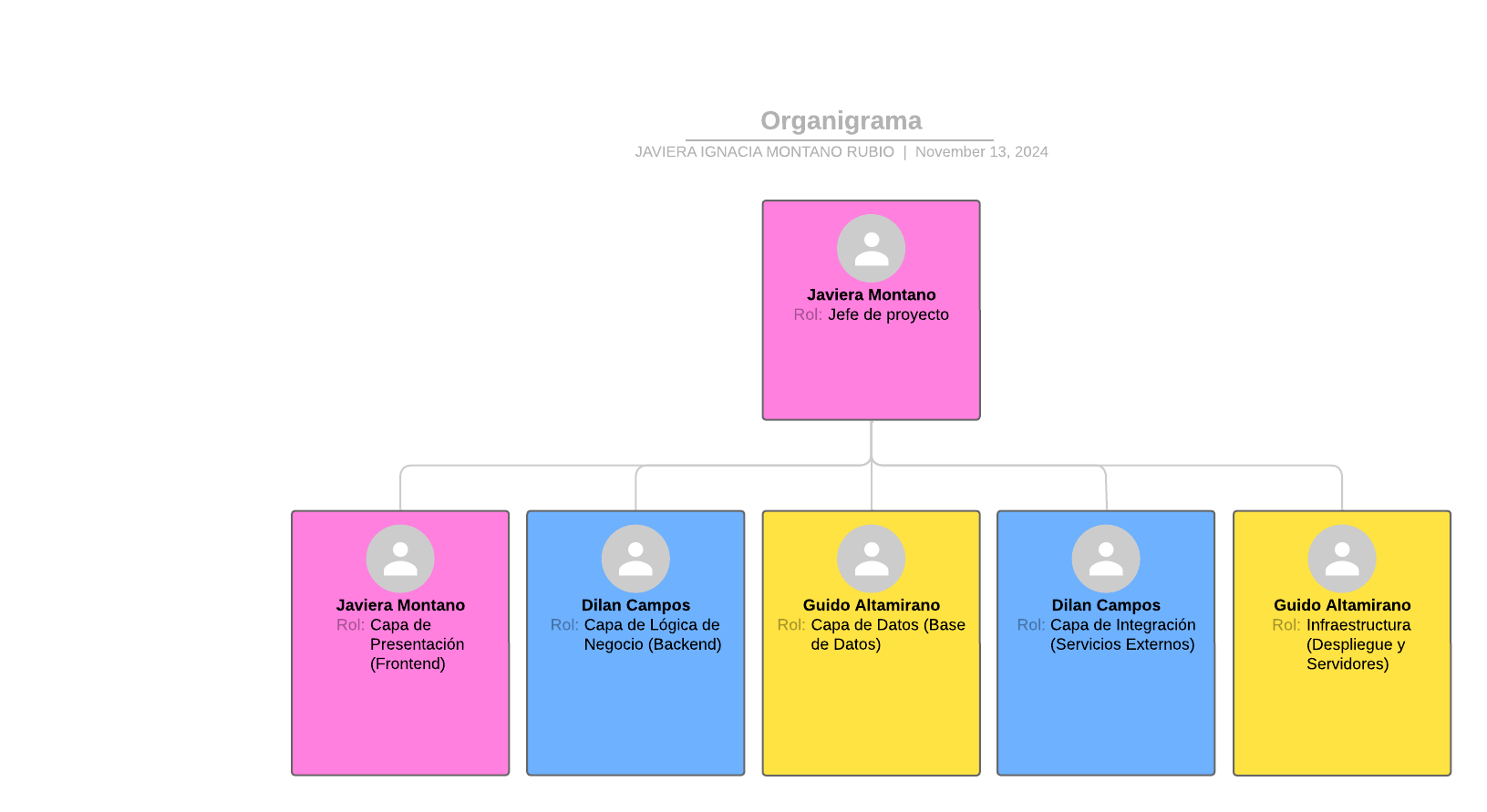
# Equipo de Trabajo-Actores del Desarrollo

Objetivo del Proyecto: Desarrollar una plataforma web para la compra y venta de repuestos de autos, que incluya un sistema de gestión de inventarios, un catálogo de productos, y funcionalidades de pago en línea e integración con proveedores de repuestos.

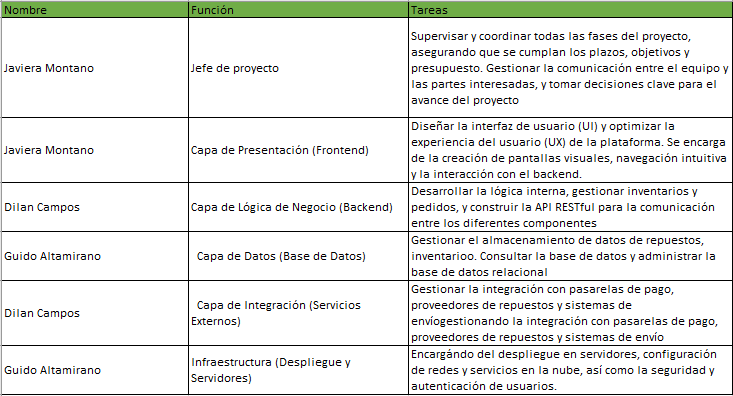
Componentes Principales del Sistema:

* Capa de Presentación (Frontend): La capa que interactúa con el usuario final a través de la interfaz de usuario (UI) y la experiencia de usuario (UX).
* Capa de Lógica de Negocio (Backend): La capa que se encarga de procesar las solicitudes del usuario, gestionar las operaciones comerciales (ventas, inventarios) y conectar con la base de datos.
* Capa de Datos (Base de Datos): La capa que almacena los datos del sistema (repuestos, inventarios, usuarios, transacciones, etc.).
* Capa de Integración (Servicios Externos): Conecta el sistema con servicios externos como pasarelas de pago, proveedores de repuestos, y sistemas de envío.
* Infraestructura (Despliegue y Servidores): La infraestructura que soporta el sistema, incluyendo servidores, servicios en la nube, y redes.

**Organigrama del equipo de proyecto que cubriría el desarrollo del proyecto.**



**Tabla de Roles y Funciones de los actores del Desarrollo:**



# Recomendaciones de conformidad con esta práctica.

Para poder interpretar de mejor manera los diagramas de este documento se recomienda entender de antemano un poco de Arquitectura de Software.

# Referencias y estándares aplicables a este documento:

- IEEE 830-1998 ST

- Architecture Tradeoff Analysis Method

- ISO 9126 -2001 Calidad del Software y Métricas de evaluación

- The 4+1 View .Kruchten – 1009

# Definiciones, acrónimos y abreviaciones.

DAS:

Documento de Arquitectura de Software

ARQUITECTURA DE SOFTWARE:

Conjunto de elementos estáticos, propios del diseño intelectual del sistema, que definen y dan forma tanto al código fuente, como al comportamiento del software en tiempo de ejecución. Naturalmente este diseño arquitectónico ha de ajustarse a las necesidades y requisitos del proyecto.

DIAGRAMA DE CASOS DE USO:

Es un tipo de diagrama UML de comportamiento y se usa frecuentemente para analizar varios sistemas. Permiten visualizar los diferentes tipos de roles en un sistema y cómo esos roles interactúan con el sistema.

DIAGRAMA DE CLASES:

Es una herramienta para comunicar el diseño de un programa que se creó para orientar objetos y que permite modelar relaciones entre diferentes entidades.

MODELO DE DATOS (BD):

Es el proceso de creación de una representación visual o esquema que define los sistemas de recopilación y administración de información de cualquier organización.

DIAGRAMA DE ACTIVIDADES:

Es una herramienta que muestra cómo los eventos de un caso de uso se relacionan entre sí o cómo una colección de casos de uso se coordinan para representar el proyecto en cuestión.

DIAGRAMA DE DESPLIEGUE:

Un diagrama de despliegue es una forma de ilustrar el hardware y el software de un sistema. Ayuda a visualizar los procesadores, los nodos y los dispositivos vinculados. En el modelado UML, estos diagramas son muy útiles ya que ayudan a describir el tiempo de ejecución de los nodos de procesamiento y especificar sus detalles para fines de construcción.

DIAGRAMA DE VISTA FISICA:

Describe el sistema desde el punto de vista de un ingeniero de sistemas. Está relacionada con la topología de componentes de software en la capa física (hardware), así como las conexiones físicas entre estos componentes.

# DESCRIPCIÓN DE ARQUITECTURA:

## VISTAS:

Describa que vistas o dimensiones según el Modelo UML 4+1 de Philippe Kruchten se deben representar y argumente su utilidad.

Es una representación de un área de interés o perspectiva del sistema en alto nivel que se crearon distintos modelos para visualizar de mejor manera todos los procesos que el software tendría, tales como, Diagrama de Casos de Uso, Diagrama de Clases, Diagrama de Despliegue, entre otros.

### TIPOS DE VISTAS:

Especificación de una convención de cómo construir y usar una vista. Deben satisfacer la capacidad de creación y análisis de una vista.

ACTORES USUARIOS RELACIONADOS

Individuo, equipo u organización con intereses relativos al sistema.

ESCENARIO:

Especifica el comportamiento y limita el interés de un área específica del sistema para uno o varios Actores Usuarios o stakeholders.

MODULO O COMPONENTE:

Cualquier elemento estructural abstracto, visible, externo, de alto nivel, analizable, que pueda constituir una funcionalidad de la solución del sistema.

ATRIBUTOS DE CALIDAD:

Un atributo de calidad es una cualidad deseable de la solución, que pueda manifestarse en forma de requerimiento no funcional, que pueda ser medible, testeable y finalmente evaluable.

## Framework Conceptual

### Descripción de la arquitectura en contexto

Este documento presenta la arquitectura como una serie de vistas basadas en la arquitectura de software del modelo 4+1 de Kruchten que tienen como propósito modelar y unificar los diversos procesos que contiene este proyecto, con el fin de que el software se vea mas atractivo e intuitivo para el usuario.

Estas vistas son:

* La vista de escenarios (Diagrama de Caso de Uso extendido)
* La vista lógica (Modelo de Clases y Modelo de Datos)
* La vista de procesos (Diagrama de actividad)
* Vista de Despliegue (Diagrama de componentes o Modelo de Capas)
* La vista física (Diagrama topológico o de servicios).

No hay ninguna vista separada de una misma implementación, descrita en este documento.

Estas vistas están hechas sobre Lenguaje de modelo unificado (UML) en su versión 2.0 desarrolladas usando la aplicación de modelamiento de software convencional. Las herramientas utilizadas para el modelamiento son las siguientes: UMLetino, ORACLE SQL DEVELOPER DATA MODELER, STARUML

### Actores Usuarios y sus roles:

Este documento representa la identificación de Actores/Usuarios Stakeholders y sus roles a partir de la interpretación de los casos de uso del Negocio asociados.

Insertar Tabla de Principales Actores por Rol o cargo que deberían ser considerados en los procesos a automatizar por el Sistema a desarrollar.

| ACTOR | ROL / CARGO | FUNCION |
| --- | --- | --- |
| Javiera Montano | Jefe de Proyecto | Planificación y ejecución del proyecto, supervisión general del desarrollo y avance. |
| Javiera Montano | Desarrolladora Frontend | Desarrollo de la interfaz de usuario (UI/UX), implementación de funcionalidades en web y móvil. |
| Guido Altamirano | Desarrollador Backend | Desarrollo y mantenimiento de la lógica del sistema, API, y servicios backend. |
| Dilan Campos | Administrador Base de Datos | Diseño y administración de la base de datos, gestión de datos relacionados con inventarios y pedidos. |
| Dilan Campos | Tester | Realización de pruebas del frontend para asegurar su funcionalidad y experiencia de usuario. |
| Guido Altamirano | Tester | Pruebas de backend, API y servicios integrados para detectar posibles errores. |
| Guido Altamirano | Responsable de Infraestructura | Gestión del despliegue del sistema, configuración de servidores, seguridad y redes. |
| Javiera Montano | Scrum Master | Gestión de los procesos ágiles (SCRUM) dentro del equipo de desarrollo. |
| N/A | Cliente Final | Compra de repuestos a través de la plataforma, consulta de disponibilidad y precios. |
| N/A | Proveedor de Repuestos | Suministro de repuestos, actualizaciones de inventarios y precios. |
| N/A | Administrador del Sistema | Gestión general del sistema y los usuarios, supervisión de las operaciones y transacciones. |

### 8.3.2 Actividades de arquitectura en el ciclo de vida

El proyecto comienza con la captura de requisitos del sistema, donde nos enfocamos en el análisis de las necesidades del cliente y en la integración de un sistema que permita la gestión eficiente de repuestos y pedidos. A partir de estos requerimientos, nos orientamos a un enfoque de Ciclo Iterativo Incremental, lo que nos permite asegurar la calidad y funcionalidad del sistema en cada fase del proyecto, garantizando que el producto final sea efectivo y libre de errores. En cada iteración, se validarán los avances con el cliente, lo que asegura que el producto final cumpla con las expectativas de los usuarios.

### 8.3.3 Descripciones prácticas de arquitectura.

Comenzamos con la captura de requisitos a través de entrevistas y análisis del negocio, seguido por la creación de historias de usuario para definir los casos de uso principales del sistema. A continuación, nos centramos en los requerimientos funcionales del sistema, los cuales fueron representados en el Modelo de Casos de Uso. Luego, se desarrolló un Diagrama de Clases que define las entidades y los métodos que interactúan dentro del sistema.

Con base en los requerimientos, creamos un Modelo de Datos que define cómo se gestionan y almacenan los datos de repuestos, inventarios y pedidos en la base de datos. Posteriormente, se elaboró un Diagrama de Despliegue que describe cómo se implementará el software en el entorno de producción y qué recursos serán necesarios para su correcto funcionamiento. Además, se diseñó un Diagrama de Actividades para representar los flujos de trabajo y la interacción del usuario con el sistema en distintas fases. Por último, se desarrolló un Diagrama de Secuencias, que muestra cómo los diversos procesos del sistema se comunican e interactúan entre sí.

## Documentación de la arquitectura

Nuestra metodología se basa en el ciclo de vida del desarrollo de software, que incluye las siguientes fases: Análisis, Diseño, Desarrollo, Pruebas y Mantenimiento. En cada una de estas fases, se documentan los avances y se llevarán a cabo las pruebas necesarias para garantizar la calidad del sistema. La documentación detallada de cada fase del proyecto ayudará a asegurar que se cumplan los objetivos establecidos y se optimice el tiempo de desarrollo.

### 8.4.1 Identificación de los actores usuarios y sus responsabilidades

| ACTOR/USUARIO | CARGO/FUNCIÓN | NOMBRE ESCENARIO/ PROCESO | ATRIBUTOS EN EL SISTEMA |
| --- | --- | --- | --- |
| Administrador/Vendedor | Gestiona el sistema, la base de datos y realiza ventas | 1. Administrar pedidos y proveedores.  2. Gestionar la base de datos.  3. Procesar ventas.  4. Gestionar clientes. | Administrador/Vendedor |
| Cliente | Realiza pedidos y consulta repuestos | 1. Realizar pedidos.  2. Consultar inventarios. | Cliente |
| Proveedor | Gestiona repuestos y entregas | 1. Gestionar inventarios y entregas. | Acceso a inventarios y registros de entregas. |
| Gestor de Inventario | Controla los niveles de stock | 1. Controlar el stock.  2. Asegurar la disponibilidad de repuestos. | Gestor de Inventario |
| Transportista | Realiza las entregas de repuestos | 1. Gestionar entregas y rutas. | Acceso a la información de envíos y actualizaciones de estado. |

## Selección de puntos de vista de la arquitectura

| VISTAS | UML | Cantidad |
| --- | --- | --- |
| Escenario Principal | Diagrama de Caso de uso | 1 |
| Vista Lógica | Diagrama de Clases  Diagrama de Comunicación  Diagrama de Secuencia | 1  1  1 |
| Vista de Proceso | Diagrama de Actividad | 1 |
| Vista Física | Diagrama de Despliegue | 1 |
| Vista de Despliegue | Diagrama de componentes  Diagrama de paquetes | 1  1 |

## Vistas de la arquitectura

| UML-VISTA DE ESCENARIO |
| --- |
| Diagrama de Caso de uso |
|  |

| UML-VISTA LOGICA |
| --- |
| [Diagrama de Clases] |
|  |

| UML-VISTA LOGICA |
| --- |
| Diagrama de Comunicación |
| Sistema cliente    Sistema Admin |

| UML-VISTA LOGICA |
| --- |
| Diagrama de Secuencias |
| Diagrama de secuencia para Cliente    Diagrama de secuencia para Admin    Diagrama de secuencia para Api Webpay |

| UML-VISTA DE PROCESO |
| --- |
| DIAGRAMA DE ACTIVIDAD |
|  |

| UML-VISTA FISICA |
| --- |
| DIAGRAMA DE DESPLIEGUE |
|  |

| UML-VISTA DE DESPLIEGUE |
| --- |
| DIAGRAM DE COMPONENTES |
|  |

| UML-VISTA DE DESPLIEGUE |
| --- |
| DIAGRAM DE PAQUETES |
|  |

| Usabilidad y Calidad de software |
| --- |
| Vistas principales del sistema |
|  |

## Consistencia en la cantidad de vistas de la arquitectura.

### **DESCRIPCIÓN DE MÓDULOS** **(DASHBOARD)**

| Nombre del módulo | Descripción | Componentes incluidos |
| --- | --- | --- |
| Gestión de Pedidos | Módulo para gestionar pedidos de clientes, incluyendo la revisión de inventario, confirmación de pedido y seguimiento de entrega. | Formulario de creación de pedido |
| Inventario y Stock | Módulo para supervisar niveles de inventario, controlar entradas y salidas de productos, y realizar ajustes según disponibilidad. | Panel de control de inventario |
| Evaluación de Proveedores | Módulo para gestionar la relación con proveedores, realizar pedidos de productos y revisar el cumplimiento de entregas. | Formulario de evaluación de proveedores |
| Reportes de Ventas e Inventario | Módulo para generar reportes sobre ventas, inventario y pedidos, permitiendo la visualización de tendencias y estados de stock. | Generador de reportes |
| Gestión de Usuarios | Módulo para administrar la información de usuarios del sistema, incluyendo clientes y acceso a funciones según roles. | Formulario de registro de usuario |

### **DESCRIPCIÓN DE COMPONENTES**

| Nombre del Componente | Descripción | Componentes relacionado |
| --- | --- | --- |
| Formulario de Pedido | Registra los pedidos de repuestos de los clientes. | Consultar inventario, registrar detalles de pedido. |
| Formulario de Cliente | Captura los datos del cliente, como contacto y vehículo. | Consultar historial de compras. |
| Formulario de Inventario | Gestiona el stock de repuestos, actualizando las existencias. | Actualización de stock, consulta de repuestos disponibles. |
| Formulario de Venta | Procesa las ventas de repuestos y actualiza el inventario. | Registro de venta, actualización de inventario. |
| Formulario de Facturación | Genera la factura para las ventas realizadas. | Generación de factura, registro de pago. |
| Evaluación de Proveedores | Registra y evalúa la calidad de los proveedores de repuestos. | Consultar proveedores, actualizar inventario. |
| Evaluación de Proveedores | Registra y evalúa la calidad de los proveedores de repuestos. | Consultar proveedores, actualizar inventario.Formulario de Entregas |

### DESCRIPCION DE CONECTORES

* + - 1. **Arquitectura lógica**:

Desde un dispositivo con navegador Chrome o una aplicación móvil (capa del cliente), el usuario realiza una solicitud al servidor web o directamente a los servicios de Firebase mediante una API (usando HTTP/HTTPS). Firebase (Capa de datos) gestiona las respuestas y permite el acceso a datos en tiempo real, como el inventario de repuestos y el historial de pedidos.

* + - 1. **Performances**

Los criterios de rendimiento para los componentes del sistema incluyen alta capacidad de respuesta en la gestión de pedidos y consulta de inventario. La plataforma integra estos componentes en una sola solución para garantizar eficiencia en la gestión de ventas y entregas.

Componentes de Firebase Database:

* **Firebase Realtime Database** o **Firestore**: ofrece almacenamiento en la nube con sincronización en tiempo real, ideal para la gestión de inventario y pedidos en tiempo real.
* **Almacenamiento escalable**: Firebase ajusta automáticamente el espacio y los recursos según la demanda, sin necesidad de configurar núcleos o memoria.
* **Optimización de rendimiento**: diseñada para manejar múltiples solicitudes y grandes volúmenes de datos de inventario de manera eficiente.

Componentes del servicio web:

* **Instancia:** l3v2
* **Núcleos:** 8
* **RAM:** 32 GB
* **Almacenamiento:** 1 TB

Este servicio web optimiza el acceso rápido al sistema, proporcionando un rendimiento adecuado para consultas y transacciones de pedidos.

* + - 1. **Detalles de la implementación**

El sistema se implementará en una aplicación web que utiliza Firebase como backend para el almacenamiento de datos y autenticación de usuarios. Firebase ofrece una infraestructura en la nube con almacenamiento seguro y sincronización en tiempo real, ideal para gestionar el inventario y los pedidos de repuestos de manera accesible y escalable.

* **Firebase Realtime Database o Firestore** para almacenamiento en tiempo real
* **Firebase Authentication** para gestión segura de usuarios
* **Hosting de Firebase** para desplegar la aplicación web y ofrecer alta disponibilidad y rendimiento.
  + - 1. **Criterios de calidad**

1. **Pruebas unitarias** para verificar el correcto funcionamiento de los diferentes componentes del sistema de gestión de inventario y pedidos.
2. **Pruebas end-to-end (E2E)** para garantizar que el sistema de gestión de pedidos funcione correctamente, desde el pedido hasta la actualización del inventario en Firebase.
   * + 1. **Lenguajes y plataformas**

* Firebase: Gestionar la base de datos del sistema de inventario y los pedidos de repuestos sin necesidad de configurar un servidor complejo
* HTML: Lenguaje de etiquetas, usado para crear páginas web.
* CSS: Lenguaje usado para dar estructura a la página WEB.
* JavaScript: Lenguaje para programar en la Página Web.
* Node.js: Backend eficiente para gestionar solicitudes y APIs en tiempo real.
* Angular: Framework frontend para una interfaz de usuario dinámica y organizada.
* Ionic: Framework para crear una app multiplataforma o páginas Web.