

**Universidad ORT Uruguay**

**Facultad de Ingeniería**

**Obligatorio Arquitectura de Software**

Entregado como requisito para la obtención del título de  
Ingeniero en Sistemas

Link a repositorios de github:

[https://github.com/IngSoft-AR/255163\\_260956](https://github.com/IngSoft-AR/255163_260956)

[https://github.com/IngSoft-AR/front-255163\\_260956](https://github.com/IngSoft-AR/front-255163_260956)

Ana Gutman – 260956

Federica Cabrera – 255163

# **DOCUMENTO DE DESCRIPCIÓN DE ARQUITECTURA**

## ***PLATAFORMA YAGANÍ***

**25/11/2024**

***Federica Cabrera, Ana Gutman***

## ÍNDICE

<b>1. Introducción</b>	<b>4</b>
1.1. Propósito	4
1.2. Alcance	4
<b>2. Antecedentes</b>	<b>4</b>
2.1. Propósito del sistema	4
Objetivos Principales	4
2.2. Requerimientos significativos de arquitectura	5
2.2.1. Resumen de Requerimientos Funcionales	5
2.2.2. Resumen de Requerimientos de Atributos de Calidad	7
<b>3. Documentación de la arquitectura</b>	<b>9</b>
3.1. Diagrama de Contexto	9
3.2. Vistas de Módulos	10
3.2.1. Vista de Descomposición	10
3.2.1.1. Representación primaria	10
3.2.1.2. Catálogo de elementos:	10
3.2.1.3. Decisiones de diseño:	11
3.2.2. Vista de Uso	12
3.2.2.1. Representación primaria	12
3.2.2.2. Catálogo de elementos	13
3.2.2.3. Decisiones de diseño	13
3.2.3. Vista de Layers	14
3.2.3.1. Representación primaria	14
3.2.3.2. Catálogo de elementos	15
3.2.3.3. Decisiones de diseño	15
3.2.3. Vistas de Componentes y conectores	16
3.3.1. Representación primaria	16
3.3.2. Catálogo de elementos	17
3.3.3. Interfaces	18
3.3.4. Comportamiento	20
3.3.5. Relacion con elementos lógicos	21
3.3.6. Decisiones de diseño	21
3.4. Vistas de Asignación	23
3.4.1. Vista de Despliegue	24
3.4.1.1. Representación primaria	24
3.5. Reporte de pruebas de carga	24
<b>ANEXO</b>	<b>26</b>
1) Evidencia de las Pruebas de carga	26
2) Capturas de los prototipos en funcionamiento:	27
3) Capturas de la incorporación de herramientas digitales externas:	31

## 1. Introducción

### 1.1. Propósito

Este documento describe la arquitectura diseñada para la plataforma YAGNI, desarrollada como solución tecnológica para la empresa gastronómica SOLID. El objetivo principal de este documento es proporcionar una visión detallada y estructurada de las decisiones de arquitectura tomadas, las vistas relevantes del sistema y cómo estas se alinean con los requerimientos funcionales y no funcionales especificados.

El diseño de la arquitectura busca garantizar que el sistema cumpla con los estándares de calidad, rendimiento y seguridad exigidos por el cliente, al mismo tiempo que se asegura la simplicidad, la escalabilidad y la mantenibilidad del sistema en el tiempo.

### 1.2. Alcance

El alcance de este documento incluye la descripción de la arquitectura diseñada para la plataforma YAGNI, abarcando sus módulos principales, las interacciones entre componentes, las decisiones de diseño tomadas y las vistas de arquitectura necesarias para cumplir con los requerimientos del proyecto. Este documento cubre tanto las funcionalidades del sistema como los atributos de calidad críticos.

## 2. Antecedentes

### 2.1. Propósito del sistema

El sistema YAGNI está diseñado para coordinar y optimizar las operaciones clave de SOLID, una empresa gastronómica. Su propósito es ofrecer una plataforma centralizada que integre pedidos de clientes, gestión de inventarios, logística de transporte y operación de refrigeradores inteligentes. Esto se traduce en una experiencia fluida y confiable para los clientes y una gestión eficiente para la empresa.

#### Objetivos Principales

1. **Optimizar procesos internos** como la producción, transporte y almacenamiento de productos, reduciendo errores y tiempos de espera.
2. **Garantizar la confiabilidad** del sistema con actualizaciones en tiempo real, claves OTP seguras para acceso a refrigeradores y notificaciones instantáneas entre módulos.
3. **Asegurar la escalabilidad** para soportar incrementos futuros en la cantidad de usuarios, pedidos y operaciones logísticas.
4. **Mejorar la experiencia del cliente** mediante una interfaz intuitiva que permita verificar el stock en tiempo real y retirar productos de refrigeradores inteligentes de forma segura.

## **Usuarios principales del sistema:**

- **Clientes:** Realizan pedidos y consultan el stock en tiempo real. Retiran sus pedidos con apoyo de la app de refrigeradores inteligentes
- **Administradores:** Gestionan productos, precios, usuarios, locales, cocinas y camionetas a través del BackOffice, también acceden a reportes y listados de pedidos, productos, movimientos y existencias.
- **Encargados de cocina:** Preparan productos con base en las órdenes generadas por la plataforma.
- **Conductores de logística:** Transportan productos a los locales con apoyo de la app de logística y reponen stock en los refrigeradores con apoyo de la app de refrigeradores inteligentes
- **Encargados de locales:** Supervisan la correcta operación de los refrigeradores inteligentes y el retiro de pedidos por parte de los clientes.

## **2.2. Requerimientos significativos de arquitectura**

Los requerimientos de arquitectura se derivan tanto de las funcionalidades clave como de los atributos de calidad necesarios para garantizar el desempeño esperado de la plataforma.

### **2.2.1. Resumen de Requerimientos Funcionales**

ID Requerimiento	Requerimiento	Descripción	Actor
RF 1	Gestión de usuarios y roles	Permite crear, editar, eliminar usuarios y asignar roles con permisos específicos, asegurando autenticación segura.	Administrador
RF 2	Administración de productos	Gestión de productos incluyendo creación, edición, eliminación, definición de precios e ingredientes asociados.	Administrador
RF 3	Visualización de pedidos	Proporciona una vista detallada de los pedidos realizados, mostrando estados como "iniciado", "completo" o "incompleto".	Administrador, Cliente
RF 4	Consulta de stock en tiempo real	Permite consultar el stock actualizado para cada refrigerador, local y cocina, reflejando movimientos recientes.	Administrador, Cliente
RF 5	Gestión de transporte	Organiza y monitorea el transporte de productos desde las cocinas a los locales.	Logística

RF 6	Operación de refrigeradores inteligentes	Controla apertura mediante OTP y actualiza automáticamente el stock tras operaciones de ingreso o retiro.	Cliente, Logística, Local
RF 7	Generación de pedidos	Incluye selección de productos, método de pago y local para retiro con validación de pago	Cliente
RF 8	Preparación de productos en cocinas	Muestra a las cocinas qué productos deben preparar y notifica automáticamente su disponibilidad.	Cocina
RF 9	Reportes detallados	Genera reportes sobre pedidos, ventas, existencias y tiempos de procesamiento	Administrador
RF 10	Notificación de lotes listos	Envía notificaciones a Logística cuando un lote está listo para ser transportado	Cocina, Logística
RF 11	Rastreo de productos en tránsito	Monitorea el estado de los productos en cada etapa del transporte hacia los locales	Logística
RF 12	Registro de clientes	Permite el registro con credenciales seguras, incluyendo métodos de pago asociados	Cliente
RF 13	Compra de productos	Ofrece un flujo optimizado para hacer pedido, retirar pedido y enfrentarse a errores de stock como marcar pedido incompleto.	Cliente
RF 14	Retiro automatizado de pedidos	Incluye selección de local, asignación de OTP, apertura de refrigeradores y actualización de inventario	Cliente
RF 15	Gestión de locales y refrigeradores	Configuración de refrigeradores y gestión de locales para definir qué productos almacenan y los permisos asociados	Administrador
RF 16	ABM de camiones y rutas	Administración de vehículos y planificación de rutas de transporte.	Logística
RF 17	Alarmas de refrigeradores	Permite activar alarmas en caso de intentos no autorizados o problemas con el stock	Local, Cliente

RF 18	Recuperación de credenciales	Integración con proveedor externo para autenticación y recuperación de contraseñas	Cliente, Usuario
RF 19	Protección contra MitM	Implementación de medidas para proteger los datos sensibles durante su transmisión	Cliente, Logística
RF 20	Simulación de sistemas externos	Provisión de API y simuladores para refrigeradores, logística y pasarela de pago	Administrador, Desarrollador
RF 21	Configuración de parámetros iniciales	Permite establecer parámetros como ventanas de preparación en la inicialización	Administrador
RF 22	Tolerancia a fallos	Garantiza que las funcionalidades críticas como compra y retiro sigan operando pese a fallos en otros módulos	Administrador, Cliente
RF 23	Monitoreo de permisos	Registra todos los accesos al sistema, asegurando que las acciones sean realizadas por actores autorizados	Administrador

## 2.2.2. Resumen de Requerimientos de Atributos de Calidad

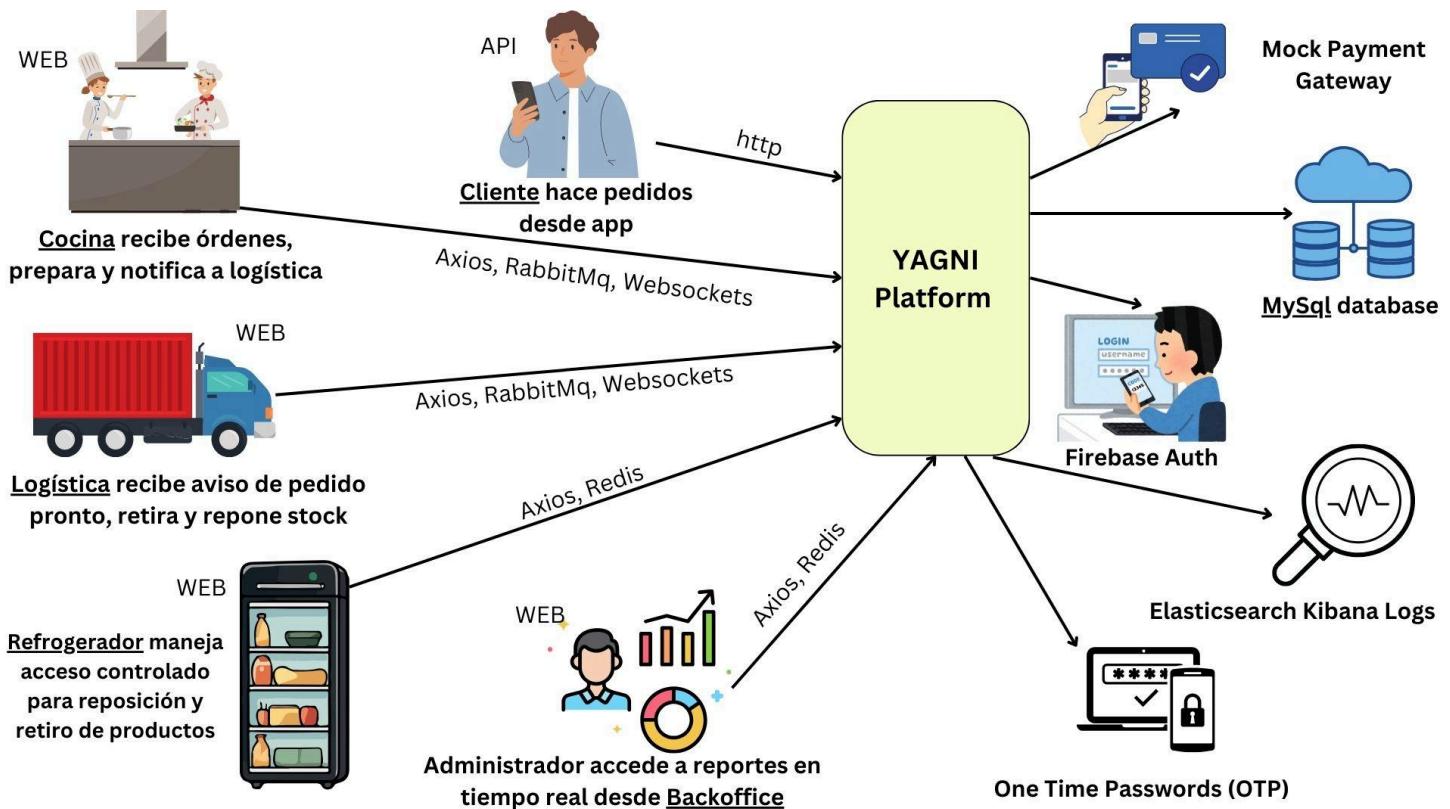
ID Requerimiento	ID Requerimiento de Calidad o Restricción	Descripción
RF 4	AC1 Confiabilidad	La actualización del stock debe reflejar movimientos en menos de 1 minuto.
RF 5, RF 6	AC2 Rendimiento	Las notificaciones entre Cocina y Logística deben procesarse en menos de 2 segundos.
RF 6	AC3 Seguridad	Las claves OTP para la apertura de refrigeradores deben expirar tras 30 segundos y ser únicas.
RF 3, RF 5	AC4 Escalabilidad	La plataforma debe soportar hasta 500,000 usuarios simultáneos.

RF 7, RF 9	AC5 Usabilidad	La interfaz de las aplicaciones debe ser intuitiva, facilitando la comprensión de flujos
RF 9	AC6 Tiempos de Reportes	Los reportes generados en el módulo Administrador deben ser procesados en menos de 500 ms.
RF 8	AC7 Trazabilidad	Todas las operaciones realizadas desde Cocina, Logística y Refrigeradores deben registrarse en logs detallados.
RF 5, RF 6	AC8 Disponibilidad	Las aplicaciones críticas deben garantizar una disponibilidad mínima del 99.9%.
RF 6, RF 9	AC9 Rendimiento	El tiempo entre la solicitud del cliente y la entrega de datos críticos debe ser menor a 1 segundo
RF 8	AC10 Integridad	Los datos relacionados con pedidos, stock y logs deben ser consistentes en todo momento
RF 20, RF 27	AC11 Seguridad de Acceso	Ninguna credencial de usuario debe almacenarse en texto plano; protegerse contra ataques MitM.
RF 28	AC12 Robustez	El sistema debe ser capaz de operar las funciones de compra y retiro, incluso si fallan otras funcionalidades.
RF 25	AC13 Extensibilidad	La arquitectura debe permitir agregar pasarelas de pago con el menor impacto posible en el código existente.
RF 15, RF 16	AC14 Autenticación	Verificación robusta de credenciales y medios de

		pago antes de confirmar transacciones
RF 9, RF 11	AC15 Tiempo Real	La información del inventario debe reflejar cambios realizados en menos de 60 segundos
RF 7	AC16 Auditabilidad	Todas las acciones de usuarios y procesos deben quedar registradas para su auditoría posterior

### 3. Documentación de la arquitectura

#### 3.1. Diagrama de Contexto



## 3.2. Vistas de Módulos

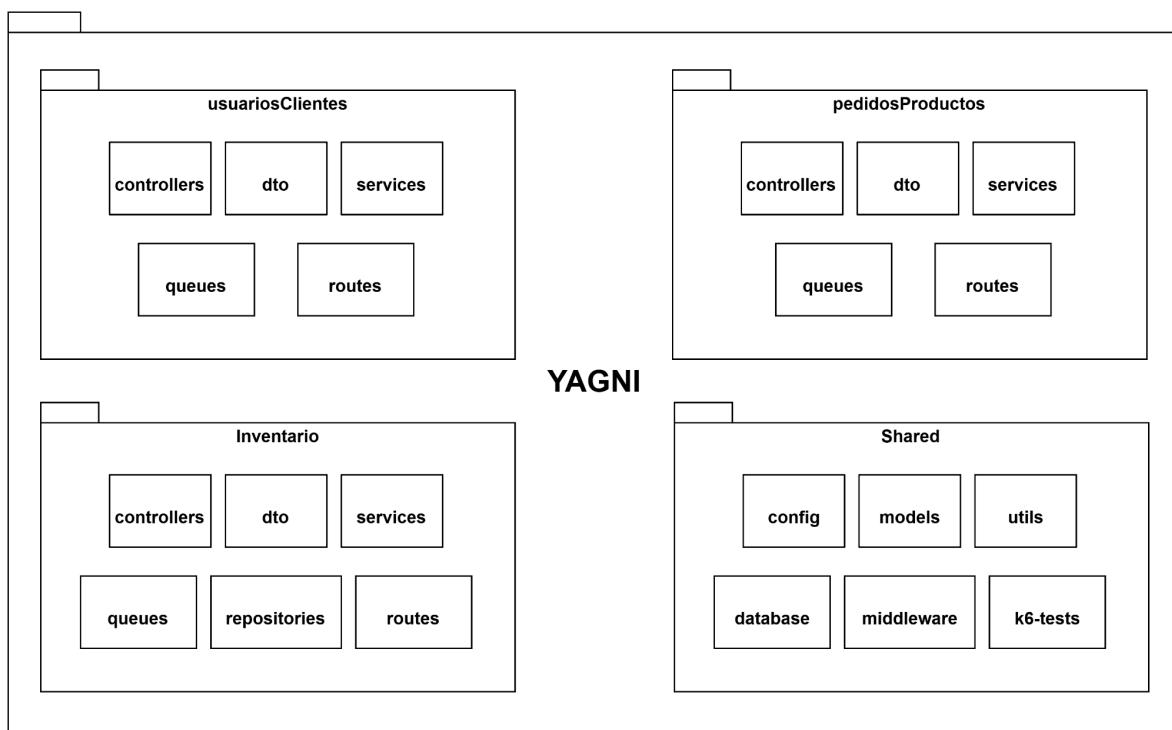
### 3.2.1. Vista de Descomposición

#### 3.2.1.1. Representación primaria

La arquitectura de nuestro sistema está organizada en tres módulos principales, alineados con las funcionalidades y dominios clave:

- 1) **UsuariosClientes:** Gestión de usuarios, autenticación y roles.
- 2) **PedidosProductos:** Procesamiento de pedidos y administración del catálogo de productos.
- 3) **Inventario:** Manejo de logística, existencias y cocinas inteligentes.

A continuación, se presenta el diagrama de descomposición del sistema:



#### 3.2.1.2. Catálogo de elementos:

A continuación se detallan las responsabilidades de cada módulo:

Módulo	Responsabilidad
<b>UsuariosClientes</b>	Gestión de usuarios y clientes, incluyendo autenticación, manejo de perfiles y roles. Ejemplo: validar credenciales, asignar roles administrativos o de cliente, y permitir acceso basado en permisos.

<b>PedidosProductos</b>	Procesamiento de pedidos, administración del catálogo de productos y cálculo de totales. Ejemplo: manejo de órdenes generadas por clientes y cálculo de hora de retiro.
<b>Inventario</b>	Manejo de inventario y logística, control de stock, administración de refrigeradores inteligentes y gestión de flujos logísticos. Ejemplo: monitorear la disponibilidad de productos en refrigeradores en tiempo real y coordinar la entrega desde las cocinas.
<b>Shared</b>	Contiene configuraciones compartidas y utilidades generales del sistema. Ejemplo: manejo de middlewares como autenticación y manejo de errores, conexión a la base de datos (MySQL) y configuraciones de Firebase para autenticación.

### 3.2.1.3. Decisiones de diseño:

#### **Agrupación Modular Simple**

El sistema se dividió en tres módulos principales: UsuariosClientes, PedidosProductos e Inventario. Además, se añadió un módulo Shared que centraliza configuraciones y utilidades comunes. Esta división se realizó con el objetivo de reflejar los dominios funcionales principales de la plataforma, manteniendo una organización clara y lógica.

La decisión sigue el principio KISS (Keep It Simple, Stupid), minimizando la complejidad inicial del sistema. Esto asegura que cada módulo pueda abordar su propio dominio funcional de manera independiente y clara, facilitando futuras expansiones o escalabilidad sin reestructurar todo el sistema.

#### **Centralización de Funciones Compartidas (Shared)**

Se creó un módulo dedicado para consolidar elementos comunes, como configuraciones, middlewares y utilidades compartidas. Este módulo actúa como un recurso transversal para todos los demás módulos, promoviendo consistencia y reutilización de código.

Al centralizar estas funcionalidades, se facilita el mantenimiento del sistema. Cualquier cambio o ajuste que afecte múltiples módulos puede realizarse en un solo lugar, reduciendo duplicaciones y mejorando la eficiencia del desarrollo.

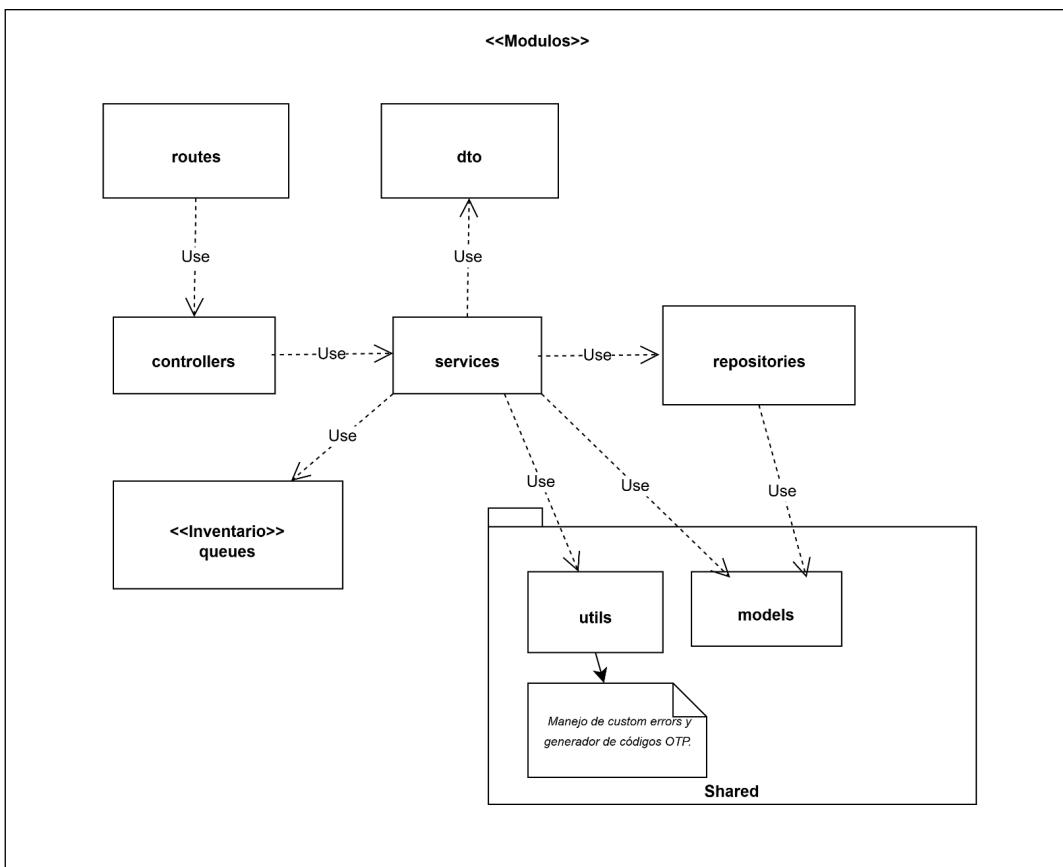
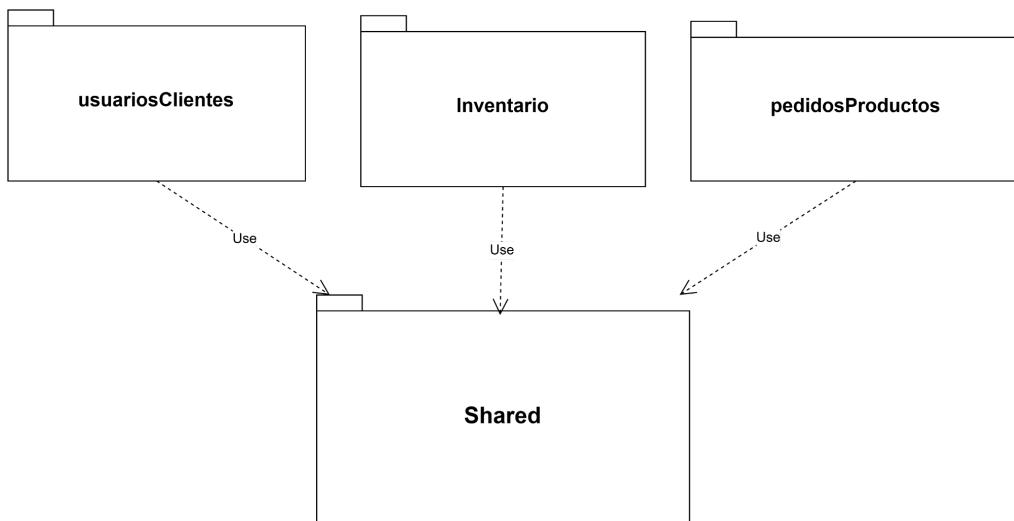
**ADR Referente:** [ADR001: Modularización del backend](#)

**ADR Referente:** [ADR003: Base de Datos Compartida](#)

**ADR Referente:** [ADR0012: Patrón DTO](#)

## 3.2.2. Vista de Uso

### 3.2.2.1. Representación primaria



### 3.2.2.2. Catálogo de elementos

Elemento	Responsabilidad
<b>Routes</b>	Define los endpoints HTTP y organiza las rutas para el acceso a las funcionalidades del módulo.
<b>Controllers</b>	Procesan las solicitudes HTTP, validan datos de entrada y delegan operaciones a los servicios.
<b>Services</b>	Implementan la lógica de negocio, coordinan procesos complejos y gestionan las reglas del sistema.
<b>Repositories</b>	Manejan la interacción con la base de datos para consultas y persistencia de datos
<b>DTO</b>	Estandarizan la transferencia de datos entre las capas para asegurar consistencia y claridad
<b>Queues</b>	Gestionan tareas asíncronas, como las notificaciones o las actualizaciones en tiempo real
<b>Shared</b>	Proporciona configuraciones comunes, middlewares, utilidades y modelos compartidos por todos los módulos

### 3.2.2.3. Decisiones de diseño

#### **Estándares en la Organización Interna**

El flujo interno de cada módulo en el sistema YAGNI está diseñado siguiendo un patrón claro y estructurado que asegura la cohesión y el desacoplamiento. Cada módulo comienza su interacción a través de las rutas (routes), que definen los puntos de entrada al sistema mediante los endpoints HTTP. Estos endpoints son gestionados por los controladores (controllers), los cuales se encargan de validar las solicitudes y delegarlas a los servicios (services). Los servicios son el núcleo de la lógica del negocio, implementando reglas específicas y coordinando operaciones complejas.

Los servicios interactúan con los repositorios (repositories), que son responsables de acceder directamente a la base de datos para consultas y operaciones de persistencia. Durante este flujo, los objetos de transferencia de datos (DTO) aseguran que la información intercambiada entre las capas sea consistente y siga un formato estándar. Además, cada módulo puede emplear colas (queues) para manejar tareas asíncronas, como notificaciones o actualizaciones en tiempo real.

El módulo compartido (Shared) actúa como una base común, proporcionando utilidades esenciales, configuraciones globales y middlewares. Estas herramientas, como el manejo de errores personalizados y la generación de OTP, permiten una implementación DRY (Don't Repeat Yourself) y aseguran que todos los módulos interactúen de manera uniforme con el entorno compartido.

Este enfoque asegura un diseño limpio y desacoplado, donde cada componente tiene una responsabilidad específica. Esto facilita la extensibilidad y permite que futuros cambios en una capa no afecten las demás.

### Uso de DTOs para la Transferencia de Datos

Estandarizamos los datos entre capas mediante DTOs. Ya que los DTOs aseguran consistencia en la información transferida y simplifican la validación de datos, reduciendo errores. Esto responde al atributo de calidad AC5 (Usabilidad), haciendo el sistema más predecible y fácil de entender para desarrolladores.

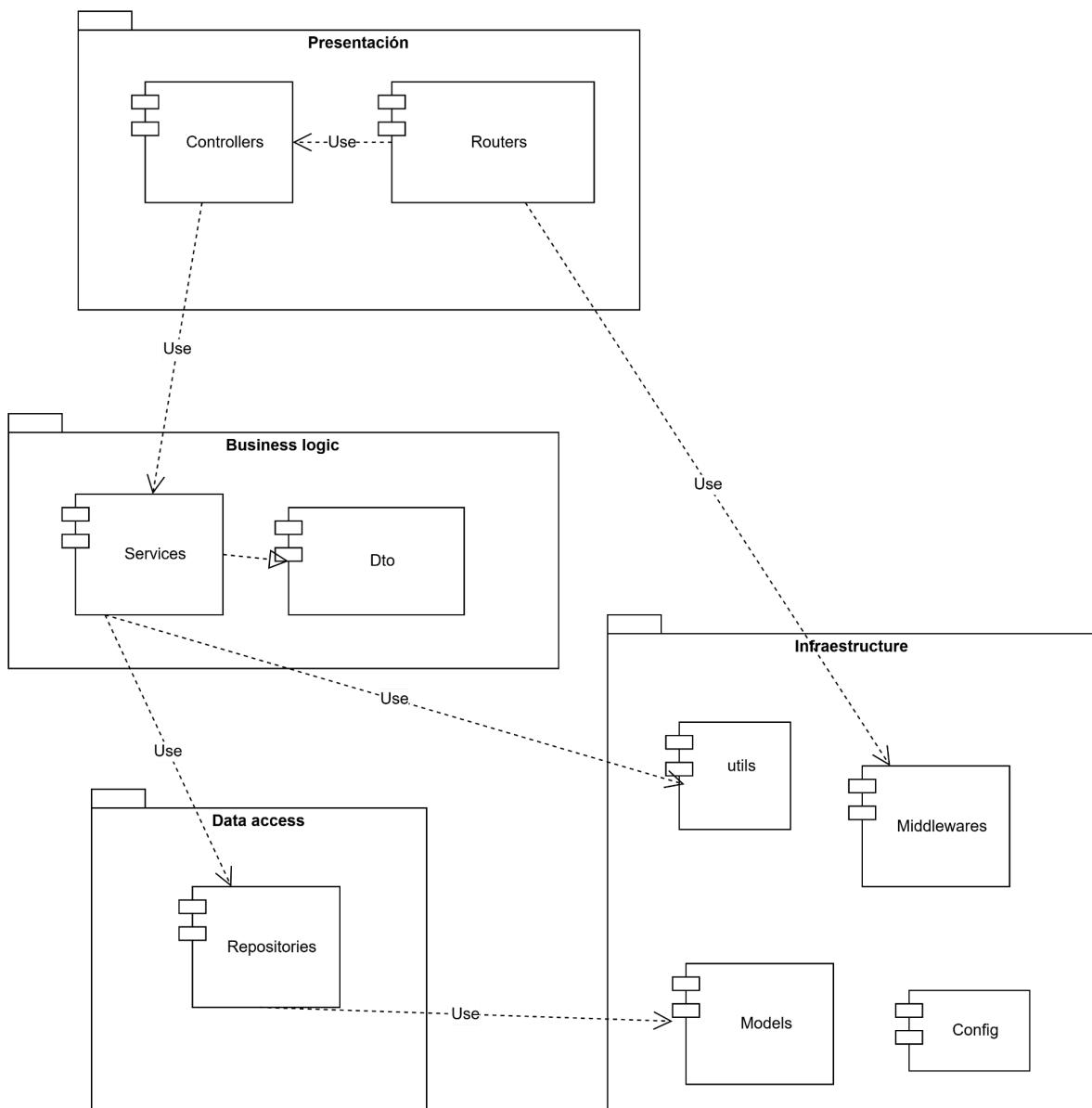
**ADR Referente:** [ADR001: Modularización del backend](#)

**ADR Referente:** [ADR003: Base de Datos Compartida](#)

**ADR Referente:** [ADR0012: Patrón DTO](#)

#### 3.2.3. Vista de Layers

##### 3.2.3.1. Representación primaria



### 3.2.3.2. Catálogo de elementos

Capa	Responsabilidad
<b>Web API</b>	Proporciona puntos de entrada para las solicitudes HTTP, gestionadas por routes y controllers.
<b>Business Logic</b>	Contiene la lógica de negocio y reglas específicas del sistema, gestionada por los services
<b>Data Access</b>	Gestiona el acceso a la base de datos, asegurando consistencia y eficiencia mediante repositories y DTOs.
<b>Shared Infrastructure</b>	Proporciona configuraciones comunes, middlewares, modelos y utilidades compartidas.

### 3.2.3.3. Decisiones de diseño

En la Vista de Layers, se optó por una separación estricta en tres niveles: Web API, Business Logic y Data Access. Esta decisión responde a la necesidad de mantener la modularidad y el desacoplamiento entre las responsabilidades principales del sistema.

La capa **Web API** actúa como punto de entrada para todas las interacciones externas, manejando las solicitudes HTTP mediante controladores y rutas bien definidas. Esto asegura que la lógica del negocio esté aislada de las interacciones del usuario.

La capa **Business Logic** implementa las reglas de negocio y coordina las operaciones complejas. Al centralizar esta funcionalidad, se garantiza que la lógica principal sea reutilizable y fácilmente extensible sin afectar las capas externas.

La capa **Data Access** se encarga de la persistencia de datos, utilizando repositorios y DTOs para estandarizar las operaciones con la base de datos. Este diseño asegura la consistencia y facilita la integración de cambios futuros en la estructura de datos.

Por último, el módulo **Shared Infrastructure** complementa las tres capas principales proporcionando middlewares, modelos y configuraciones compartidas. Este enfoque minimiza la duplicación de código y asegura la uniformidad en la interacción entre capas.

Estas decisiones, basadas en el patrón de arquitectura en capas, permiten que el sistema cumpla con atributos de calidad como la escalabilidad (AC4) y la trazabilidad (AC7), mientras optimizan la mantenibilidad y el desarrollo iterativo del sistema.

**ADR Referente:** [ADR001: Modularización del backend](#)

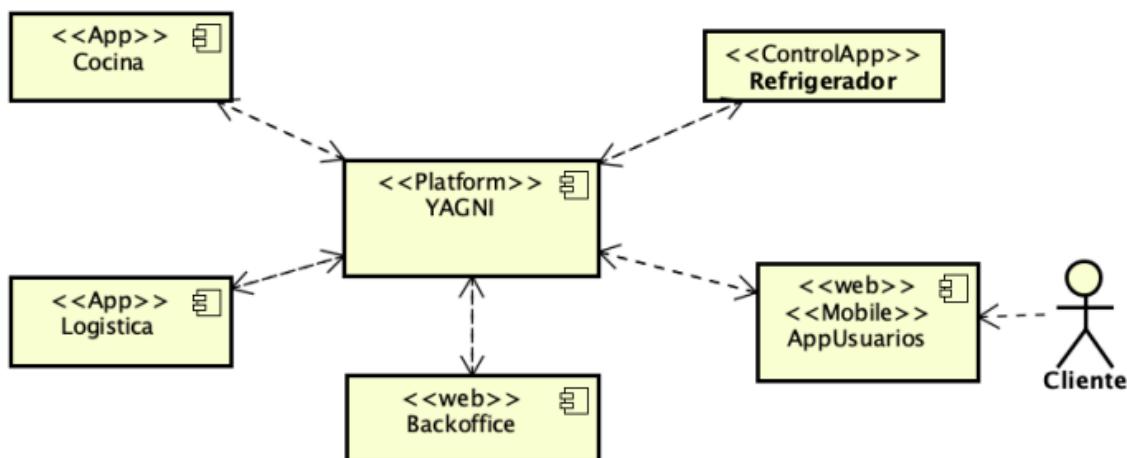
**ADR Referente:** [ADR003: Base de Datos Compartida](#)

**ADR Referente:** [ADR0012: Patrón DTO](#)

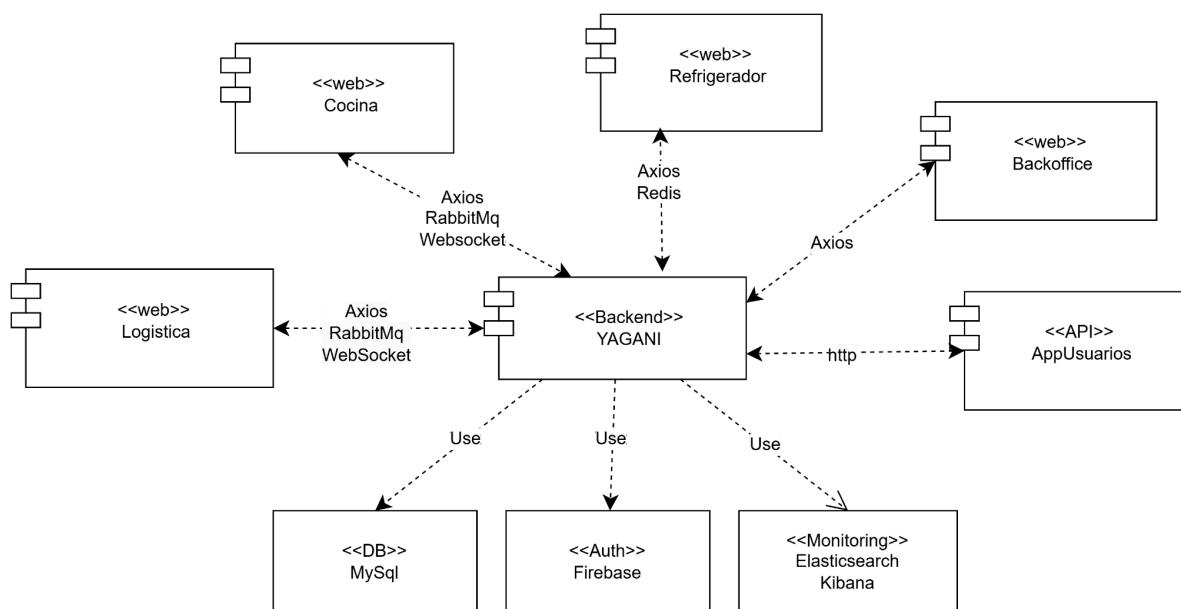
### 3.3. Vistas de Componentes y conectores

#### 3.3.1. Representación primaria

En la letra del obligatorio se expone el siguiente diagrama de contexto, para representar la a la plataforma YAGNI



En base al diseño anterior y al alcance de los requerimientos del obligatorio, nuestra primera versión del sistema con prototipos para los modulos Cocina, Refrigerador, Logística y Backoffice, se puede representar con el siguiente diagrama de contexto:



### 3.3.2. Catálogo de elementos

Componente/Conektor	Tipo	Descripción
<b>YAGANI</b>	Backend	Nodo central que gestiona la lógica del sistema, coordinando la interacción entre los módulos y servicios externos.
<b>Cocina</b>	Web	Módulo que gestiona las operaciones de preparación de alimentos, comunicación en tiempo real con logística y refrigeradores.
<b>Logística</b>	Web	Módulo encargado de la planificación y seguimiento del transporte de pedidos desde cocinas a refrigeradores.
<b>Refrigerador</b>	Web	Módulo que administra refrigeradores inteligentes, incluyendo acceso mediante OTP y actualizaciones de stock.
<b>Backoffice</b>	Web	Interfaz para administradores que permite gestionar productos, pedidos y stock.
<b>AppUsuarios</b>	API	Aplicación móvil para clientes que permite realizar pedidos, pagos y consultar estado de los pedidos
<b>MySQL</b>	Base de Datos	Base de datos relacional que almacena información estructurada sobre pedidos, usuarios, y stock.
<b>Firebase</b>	Autenticación	Servicio externo utilizado para la autenticación segura de usuarios en la plataforma.
<b>Elasticsearch/Kibana</b>	Monitoreo	Herramientas para centralizar logs y monitorear el estado y rendimiento del sistema.
<b>RabbitMQ</b>	Middleware	Sistema de mensajería utilizado para tareas asíncronas y comunicación entre módulos como Cocina y Logística
<b>Redis</b>	Middleware	Utilizado para el manejo de claves OTP temporales y caching para mejorar el rendimiento.
<b>Axios</b>	Conektor HTTP	Biblioteca de cliente HTTP utilizada para la comunicación síncrona entre módulos y servicios.
<b>WebSocket</b>	Conektor Tiempo Real	Protocolo para establecer comunicaciones bidireccionales en tiempo real entre el backend y los módulos.

### 3.3.3. Interfaces

#### Interfaces del Backend

**Interfaz:** API de inventario

**Componente que lo provee:** Backend - Módulo Inventory

Servicio	Descripción
Gestión de Camionetas	Permite listar, crear, editar y eliminar camionetas, además de consultar información detallada
Administración de Cocinas	Proporciona endpoints para gestionar las cocinas, incluyendo su creación, modificación y eliminación
Consultas de Inventory	Incluye la funcionalidad de consulta del estado de productos específicos en el inventory
Gestión de Locales	Permite la creación, actualización, eliminación y consulta de información sobre locales físicos
Control de Lotes	Maneja los lotes de productos, incluyendo su creación, modificación, y seguimiento
Gestión de Refrigeradores	Administra refrigeradores inteligentes, acceso mediante OTP, y actualizaciones de stock

**Interfaz:** API de Pedidos y Productos

**Componente que lo provee:** Backend - Módulo PedidosProductos

Servicio	Descripción
Gestión de Pedidos	Permite listar, crear, modificar, y consultar pedidos por ID o cliente, además de marcar estados como incompletos.
Administración de Productos	Proporciona endpoints para agregar, modificar, eliminar, y consultar productos dentro del sistema
Simulación de Pasarela de Pagos	Implementa una pasarela simulada que permite pruebas de flujos de pago, con respuestas aleatorias

**Interfaz:** API de Usuarios y Clientes

**Componente que lo provee:** Backend - Módulo UsuariosCientes

Servicio	Descripción
Gestión de Usuarios	Permite la creación, modificación, eliminación y consulta de usuarios dentro del sistema

Gestión de Clientes	Administra los clientes registrados, incluyendo sus métodos de pago y actualización de perfiles
Autenticación	Incluye login, validación de usuarios, y manejo de sesiones

## Interfaces con Servicios Externos

**Interfaz:** Conexión con Redis

**Componente que lo provee:** Redis Middleware

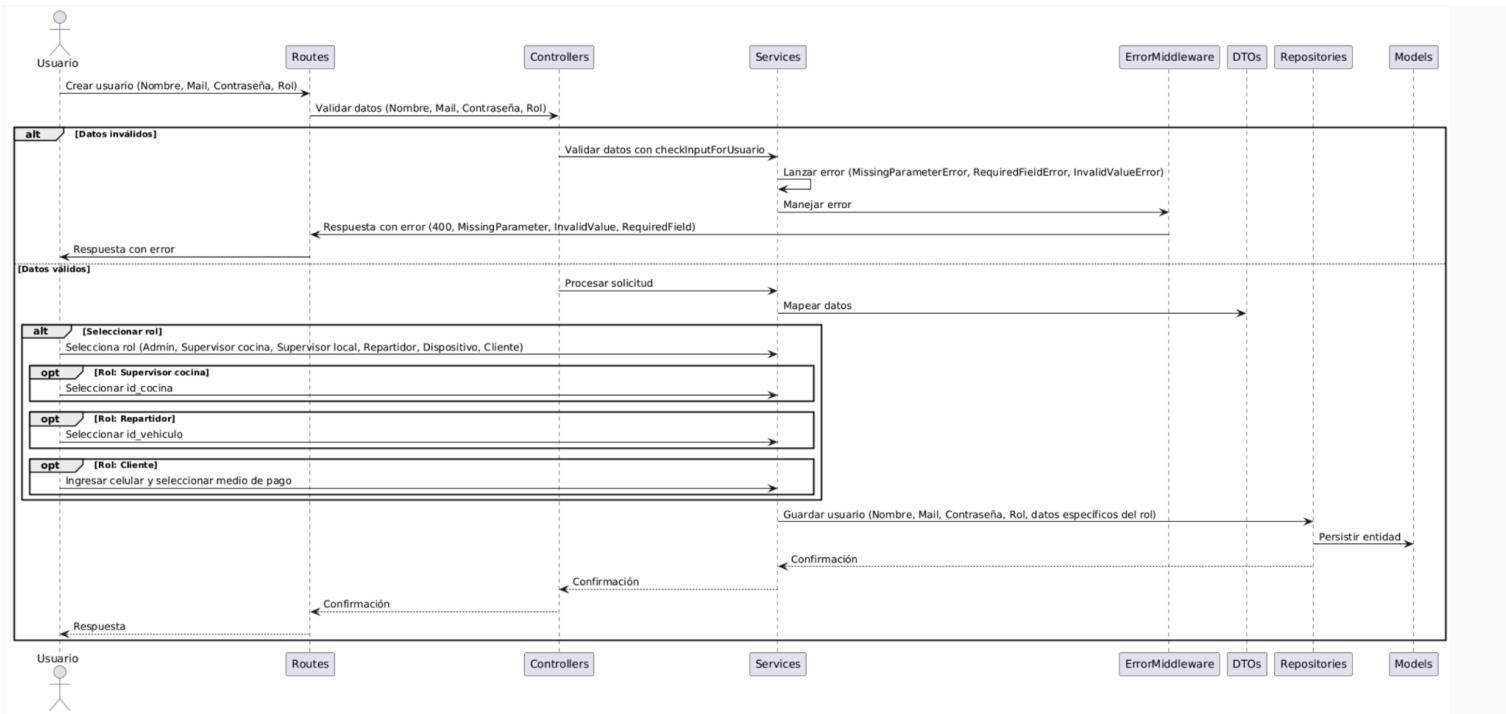
Servicio	Descripción
Generación de OTPs	Gestiona la creación y validación de claves OTP temporales con un tiempo de vida de 30 segundos
Caching de Datos	Mejora el rendimiento al almacenar datos de acceso frecuente en memoria

**Interfaz:** Conexión con RabbitMQ

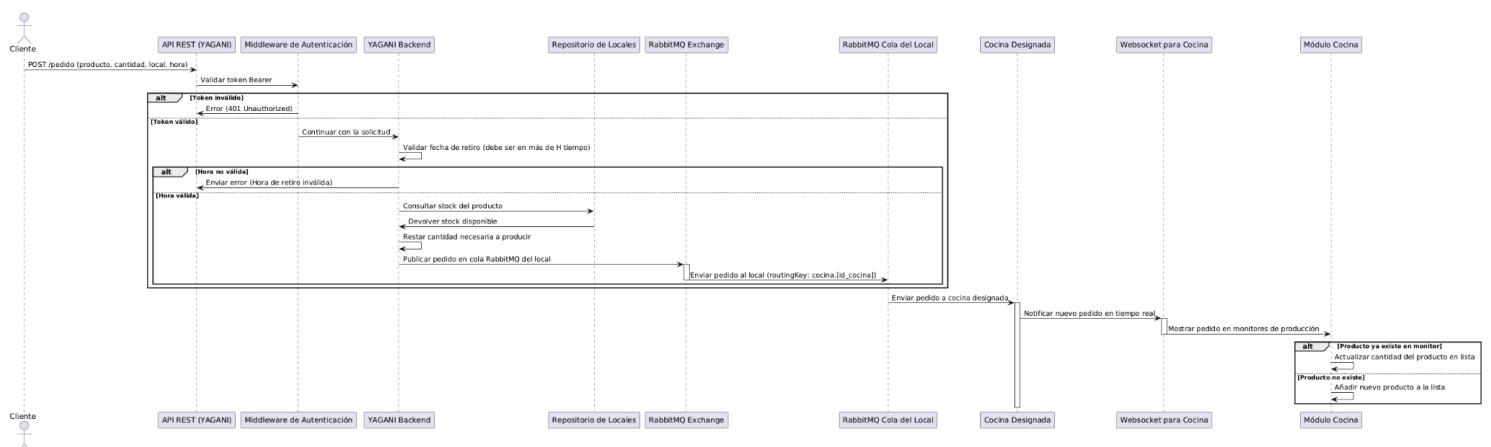
**Componente que lo provee:** Middleware RabbitMQ

Servicio	Descripción
Notificaciones Asincrónicas	Permite el intercambio de mensajes entre módulos como Cocina y Logística en tiempo real
Balanceo de Tareas	Distribuye dinámicamente tareas entre consumidores utilizando el patrón Competing Consumers

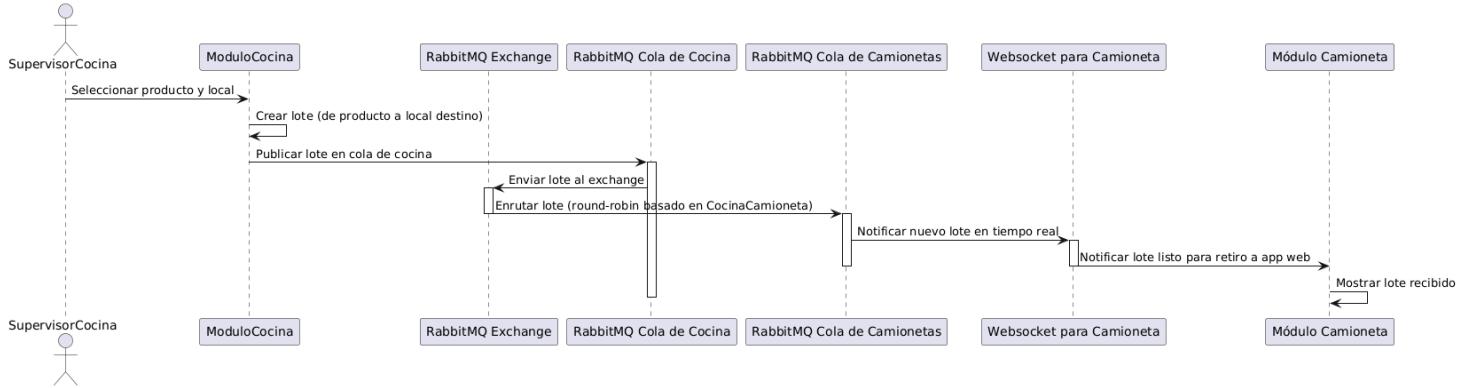
### 3.3.4. Comportamiento



Caso 1: Ingreso de usuario al sistema



Caso 2: Cliente ingresa un pedido al sistema



Caso 3: Supervisor cocina ingresa finalización de un lote al sistema

### 3.3.5. Relacion con elementos lógicos

Componente	Paquetes
Backend	Controladores, Servicios, Repositorios, Rutas (Routes)
Inventario	Controladores, Servicios, Repositorios
PedidosProductos	Controladores, Servicios, Repositorios, Simulador de Pasarela de Pagos
UsuariosClientes	Controladores, Servicios, Repositorios, Autenticación
Shared	Configuración, Middlewares (Autenticación, Logs, Errores), Modelos, Utilidades (como OTPs)
Cocina	Controladores, Servicios, Comunicación con RabbitMQ y WebSockets
Logística	Controladores, Servicios, Comunicación con RabbitMQ y WebSockets
Refrigerador	Controladores, Servicios, Comunicación con Redis y OTPs

### 3.3.6. Decisiones de diseño

#### Redis para OTPs Temporales:

**Decisión:** Redis fue seleccionado como el sistema para gestionar las claves OTP (One-Time Password) necesarias para la apertura de los refrigeradores inteligentes. Este sistema permite generar, almacenar y validar OTPs con un tiempo de vida (TTL) configurado en 30 segundos. Redis, conocido por su velocidad y capacidad para manejar claves temporales, asegura que cada OTP sea único y que su gestión sea eficiente y segura.

**Justificación:** La necesidad de un sistema confiable y rápido para manejar claves temporales está directamente vinculada al requerimiento funcional de operar los refrigeradores de manera controlada (RF 6) y al atributo de calidad de seguridad (AC3). Redis permite garantizar tiempos de respuesta inferiores al segundo, algo crítico para la experiencia del usuario y la confiabilidad del sistema. Además, la capacidad de Redis para eliminar automáticamente las claves vencidas simplifica significativamente la implementación.

**ADR Referente:** [ADR005 – Redis para OTPs Temporales](#).

### Logs Centralizados con Elasticsearch y Kibana

**Decisión:** Para centralizar el monitoreo del sistema, se integraron Elasticsearch y Kibana como herramientas principales para gestionar y analizar los logs operativos. Estos logs incluyen trazabilidad de operaciones, detección de errores y métricas de rendimiento en tiempo real. Este enfoque asegura que todas las interacciones relevantes queden documentadas de forma accesible y auditável.

**Justificación:** La trazabilidad (AC7) y la necesidad de generar reportes precisos (RF 9) exigen un sistema que no solo almacene logs, sino que permita analizarlos de manera efectiva. Elasticsearch proporciona una base de datos potente para almacenar grandes volúmenes de datos, mientras que Kibana ofrece visualizaciones intuitivas para detectar problemas de rendimiento y monitorear el estado del sistema. Esto asegura la confiabilidad operativa y la capacidad de respuesta ante incidentes.

**ADR Referente:** [ADR006 – Logs Centralizados con Elasticsearch y Kibana](#).

### RabbitMQ para Comunicación Asincrónica

**Decisión:** RabbitMQ fue elegido como middleware para la comunicación asincrónica entre módulos como Cocina y Logística, utilizando el patrón **Pub/Sub list-based**. Esto permite que los eventos críticos, como el estado de un pedido o la disponibilidad de un lote, se notifiquen entre módulos de manera rápida y desacoplada.

**Justificación:** El sistema requiere una latencia mínima en las notificaciones para cumplir con el atributo de rendimiento (AC2) y los requerimientos funcionales de comunicación entre módulos (RF 5, RF 6). RabbitMQ asegura que las notificaciones sean procesadas en menos de dos segundos, incluso con alta carga. Su diseño escalable y su capacidad de implementar patrones de suscripción lo convierten en una solución ideal para sistemas distribuidos como YAGNI.

**ADR Referente:** [ADR009 – Patrón Pub/Sub con RabbitMQ](#).

### Middleware Centralizado para Manejo de Errores

**Decisión:** Se implementó un middleware dedicado para manejar de manera uniforme los errores en toda la plataforma. Este middleware intercepta las excepciones generadas en los controladores y genera respuestas estandarizadas que incluyen códigos de estado y mensajes claros para los usuarios.

**Justificación:** La consistencia en las respuestas de la API es fundamental para la experiencia del usuario y el mantenimiento del sistema. Al centralizar la lógica de manejo de errores, se evita duplicar código en los controladores y se facilita la implementación de futuras mejoras. Este diseño responde al atributo de usabilidad (AC5) y asegura que los errores se gestionen de manera predecible y eficiente.

**ADR Referente:** [ADR008 – Middleware Centralizado para Manejo de Errores](#)

### **Base de Datos Compartida**

**Decisión:** El sistema utiliza una base de datos MySQL compartida por los módulos principales: UsuariosClientes, PedidosProductos e Inventario. Esto permite centralizar la información crítica del sistema y garantizar la consistencia en las operaciones transaccionales.

**Justificación:** La arquitectura centralizada asegura que las transacciones entre módulos sean consistentes y confiables, cumpliendo con los atributos de calidad de confiabilidad (AC1) y escalabilidad inicial (AC4). Además, simplifica la administración de datos al eliminar la necesidad de sincronización entre bases de datos separadas, lo que reduce la complejidad del sistema en esta etapa inicial.

**ADR Referente:** [ADR003 – Base de Datos Compartida](#).

### **WebSocket para Comunicación en Tiempo Real**

**Decisión:** Se utilizó WebSocket para implementar comunicaciones bidireccionales en tiempo real entre módulos como Cocina y Logística. Esto permite que eventos críticos, como actualizaciones de pedidos, se propaguen instantáneamente a los usuarios.

Justificación: Para cumplir con los requerimientos de rendimiento (AC2) y usabilidad (AC5), era necesario un sistema que manejara actualizaciones en tiempo real. WebSocket es ideal para estos casos, ya que reduce la latencia y el consumo de ancho de banda en comparación con soluciones basadas en polling o HTTP. Esto asegura una experiencia fluida e interactiva para los usuarios.

**ADR Referente:** [ADR0015 - Implementación de WebSockets para la Gestión de Notificaciones](#)

### **Mock Payment Gateway**

**Decisión:** Se implementó una pasarela de pagos simulada para facilitar las pruebas iniciales de los flujos de pago sin depender de integraciones con servicios externos. Este simulador introduce respuestas y tiempos de espera aleatorios para replicar escenarios reales.

**Justificación:** La simulación de la pasarela de pagos permite validar los flujos de negocio relacionados con los pagos (RF 7) sin incurrir en costos de integración en esta etapa de desarrollo. Además, al replicar posibles errores y variaciones de tiempo, se asegura que el sistema pueda manejar estas situaciones de manera robusta.

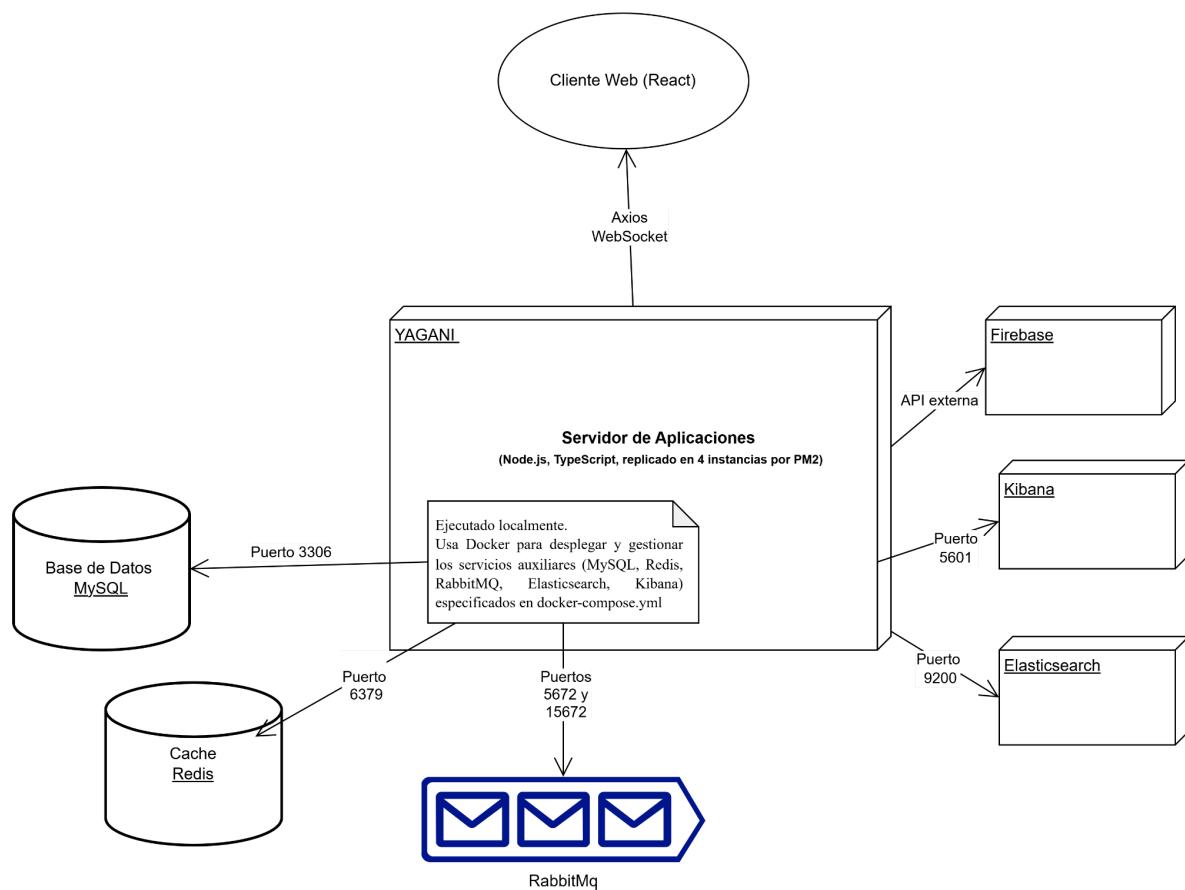
**ADR Referente:** [ADR004 – Mock Payment Gateway](#).

## 3.4. Vistas de Asignación

Esta sección describe cómo los elementos de software de YAGANI se asignan a su entorno operativo. Se incluyen vistas de despliegue, instalación y las decisiones de diseño que sustentan estas asignaciones. Estas vistas son esenciales para comprender las configuraciones físicas y lógicas del sistema y garantizar que se cumplan los requerimientos funcionales (RF) y no funcionales (RNF).

### 3.4.1. Vista de Despliegue

#### 3.4.1.1. Representación primaria



## 3.5. Reporte de pruebas de carga

En el [anexo](#) se adjunta una foto con la información del hardware de la computadora en al que se corrieron las pruebas de carga y los resultados de las mismas en modo normal. Lamentablemente, los mismos no lograron cumplir con los estandares establecidos en la letra.

Como estrategia de performance para intentar que mejore el tiempo de ejecución en las condiciones establecidas, se incorporó del uso pm2, gestor de procesos de node.js, para replicar el sistema en 4 instancias (junto con más modificaciones que se justifican en el [ADR0011- PM2 para la Gestión de Procesos en Producción](#)). Sin embargo, al correr denuevo los tests en el nuevo entorno optimizado, solamente se obtuvieron mejoras marginales en los tiempos de respuesta.

A continuación se muestran los mejores resultados que obtuvimos por la implementación de pm2, que fue para el requerimiento 7:

Modo normal sin pm2:

```
execution: local
  script: tests/listarExistencias.js
  output: -

scenarios: (100.00%) 1 scenario, 3000 max VUs, 2m30s max duration (incl. graceful stop):
  * default: Up to 3000 looping VUs for 2m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

  ✓ status is 200
  X response time < 500ms
    ↴ 6% - ✓ 5333 / X 80030

  http_req_connecting..... avg=244.19μs min=63... med=63... max=2913μs p(50)=63... p(95)=63...
  X http_req_duration.....: avg=3.2s min=1.51ms med=3.67s max=6.33s p(90)=4.92s p(95)=5.26s
    ↴ expected response=true | avg=3.2s min=1.51ms med=3.67s max=6.33s p(90)=4.92s p(95)=5.26s
```

Modo normal con pm2:

```
execution: local
  script: tests/listarExistencias.js
  output: -

scenarios: (100.00%) 1 scenario, 3000 max VUs, 2m30s max duration (incl. graceful stop):
  * default: Up to 3000 looping VUs for 2m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

  ✓ status is 200
  X response time < 500ms
    ↴ 6% - ✓ 8855 / X 80030

  http_req_connecting..... avg=244.19μs min=63... med=63... max=2913μs p(50)=63... p(95)=63...
  X http_req_duration.....: avg=2.2s min=1.11ms med=1.67s max=2.33s p(90)=2.92s p(95)=1.26s
    ↴ expected response=true | avg=3.2s min=1.51ms med=3.67s max=6.33s p(90)=4.92s p(95)=5.26s
```

Como se puede ver los resultados de duración siguen estando por arriba del doble de los 500ms esperados.

Los scripts para probar cada funcionalidad se encuentran en src/shared/k6-tests/tests. Tienen la configuración para probar el modo de carga normal, y debajo comentado el código para probar el modo de sobrecarga. De este modo, el mismo se puede aplicar para probar el sistema si eventualmente se toman las medidas necesarias para alcanzar los estandares de tiempo esperados en las condiciones de sobrecarga.

## ANEXO

### 1) Evidencia de las Pruebas de carga

Elemento	Valor
Nombre del SO	Microsoft Windows 11 Home
Versión	10.0.22631 compilación 22631
Descripción adicional del SO	No disponible
Fabricante del SO	Microsoft Corporation
Nombre del sistema	COMPUANA
Fabricante del sistema	ASUSTeK COMPUTER INC.
Modelo del sistema	Vivobook_ASUSLaptop K6502HC_K6502HC
Tipo de sistema	PC basado en x64
SKU del sistema	
Procesador	11th Gen Intel(R) Core(TM) i9-11900H @ 2.50GHz, 2496 Mhz, 8 procesadores...
Versión y fecha de BIOS	American Megatrends International, LLC. K6502HC.301, 18/08/2022
Versión de SMBIOS	3.3
Versión de controladora integrada	3.01
Modo de BIOS	UEFI
Fabricante de la placa base	ASUSTeK COMPUTER INC.
Producto de placa base	K6502HC
Versión de la placa base	1.0
Rol de plataforma	Móvil
Estado de arranque seguro	Activada
Configuración de PCR7	Se necesita elevación de privilegios para ver
Directorio de Windows	C:\WINDOWS
Directorio del sistema	C:\WINDOWS\system32
Dispositivo de arranque	\Device\HarddiskVolume1
Configuración regional	España
Capa de abstracción de hardware	Versión = "10.0.22621.2506"
Nombre de usuario	CompuAna\agutm
Zona horaria	Hora estándar de Montevideo
Memoria física instalada (RAM)	16,0 GB
Memoria física total	15,7 GB
Memoria física disponible	1,19 GB
Memoria virtual total	38,7 GB
Memoria virtual disponible	8,73 GB
Espacio de archivo de paginación	23,0 GB

#### Requerimientos 7, 8 9 y 10:

Los tiempos promedios en desplegar los listados solicitados no deben superar los 500 ms con el sistema operando en modo normal y un segundo operando en modo sobrecargado.

```

execution: local
script: tests/listarExistencias.js
output: -

scenarios: (100.00%) 1 scenario, 3000 max VUs, 2m30s max duration (incl. graceful stop):
  * default: Up to 3000 looping VUs for 2m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

✓ status is 200
✗ response time < 500ms
↳ 6% - ✓ 5333 / ✗ 80030

checks.....: 53.12% 90696 out of 170726
data_received.....: 60 MB 498 kB/s
data_sent.....: 31 MB 255 kB/s
✗ errors.....: 100.00% 80030 out of 80030
  http_req_blocked.....: avg=29.27µs min=0s med=0s max=31.34ms p(90)=0s p(95)=0s
  http_req_connecting.....: avg=24.45µs min=0s med=0s max=29.9ms p(90)=0s p(95)=0s
✗ http_req_duration.....: avg=3.2s min=1.51ms med=3.67s max=6.33s p(90)=4.92s p(95)=5.26s
  { expected_response:true }...: avg=3.2s min=1.51ms med=3.67s max=6.33s p(90)=4.92s p(95)=5.26s
  http_req_failed.....: 0.00% 0 out of 85363
  http_req_receiving.....: avg=87.6µs min=0s med=0s max=5.38ms p(90)=520.1µs p(95)=531.4µs
  http_req_sending.....: avg=16.99µs min=0s med=0s max=3.49ms p(90)=0s p(95)=0s
  http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
  http_req_waiting.....: avg=3.2s min=1.51ms med=3.67s max=6.33s p(90)=4.92s p(95)=5.26s
  http_reqs.....: 85363 711.327039/s
iteration_duration.....: avg=3.2s min=2.03ms med=3.67s max=6.33s p(90)=4.92s p(95)=5.26s
iterations.....: 85363 711.327039/s
vus.....: 14 min=14 max=3000
vus_max.....: 3000 min=3000 max=3000

running (2m00.0s), 0000/3000 VUs, 85363 complete and 0 interrupted iterations
default ✓ [=====] 0000/3000 VUs 2m0s
ERROR[0120] thresholds on metrics 'errors, http_req_duration' have been crossed

```

```

✓ status is 200
✗ response time < 500ms
↳ 1% - ✓ 484 / ✗ 28529

checks.....: 50.83% 29497 out of 58026
data_received.....: 9.9 MB 82 kB/s
data_sent.....: 12 MB 97 kB/s
✗ errors.....: 100.00% 28529 out of 28529
  http_req_blocked.....: avg=75.17µs min=0s med=0s max=6.22ms p(90)=510.49µs p(95)=527.23µs
  http_req_connecting.....: avg=65.9µs min=0s med=0s max=3.27ms p(90)=503.49µs p(95)=524.29µs
✗ http_req_duration.....: avg=9.61s min=47.78ms med=12.03s max=15.1s p(90)=14.14s p(95)=14.47s
  { expected_response:true }...: avg=9.61s min=47.78ms med=12.03s max=15.1s p(90)=14.14s p(95)=14.47s
  http_req_failed.....: 0.00% 0 out of 29013
  http_req_receiving.....: avg=84.84µs min=0s med=0s max=20.88ms p(90)=520.59µs p(95)=528.9µs
  http_req_sending.....: avg=29.25µs min=0s med=0s max=2.81ms p(90)=0s p(95)=505.6µs
  http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
  http_req_waiting.....: avg=9.61s min=47.78ms med=12.03s max=15.1s p(90)=14.14s p(95)=14.47s
  http_reqs.....: 29013 241.728815/s
iteration_duration.....: avg=9.61s min=47.78ms med=12.03s max=15.1s p(90)=14.14s p(95)=14.47s
iterations.....: 29013 241.728815/s
vus.....: 25 min=25 max=3000
vus_max.....: 3000 min=3000 max=3000

running (2m00.0s), 0000/3000 VUs, 29013 complete and 0 interrupted iterations
default ✓ [=====] 0000/3000 VUs 2m0s
ERROR[0120] thresholds on metrics 'errors, http_req_duration' have been crossed

```

```

✓ status is 200
✗ response time < 500ms
↳ 0% - ✓ 77 / X 20335

checks.....: 50.18% 20489 out of 40824
data_received.....: 7.7 MB 64 kB/s
data_sent.....: 6.5 MB 54 kB/s
✗ errors.....: 100.00% 20335 out of 20335
    http_req_blocked.....: avg=192.61µs min=0s med=0s max=28.37ms p(90)=1.02ms p(95)=1.3ms
    http_req_connecting.....: avg=168.79µs min=0s med=0s max=13.45ms p(90)=890.98µs p(95)=1.15ms
✗ http_req_duration.....: avg=13.99s min=181.77ms med=16.66s max=18.49s p(90)=17.92s p(95)=18.24s
    { expected_response:true }.....: avg=13.99s min=181.77ms med=16.66s max=18.49s p(90)=17.92s p(95)=18.24s
    http_req_failed.....: 0.00% 0 out of 20412
    http_req_receiving.....: avg=149.58µs min=0s med=0s max=11.85ms p(90)=528.49µs p(95)=556µs
    http_req_sending.....: avg=35.39µs min=0s med=0s max=8.14ms p(90)=0s p(95)=510.4µs
    http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
    http_req_waiting.....: avg=13.99s min=180.85ms med=16.66s max=18.49s p(90)=17.92s p(95)=18.24s
    http_reqs.....: 20412 169.96864/s
iteration_duration.....: avg=13.99s min=182.79ms med=16.66s max=18.49s p(90)=17.92s p(95)=18.24s
iterations.....: 20412 169.96864/s
vus.....: 57 min=57 max=3000
vus_max.....: 3000 min=3000 max=3000

running (2m00.1s), 0000/3000 VUs, 20412 complete and 0 interrupted iterations
default ✓ [=====] 0000/3000 VUs 2m0s
ERRO[0120] thresholds on metrics 'errors, http_req_duration' have been crossed

```

```

execution: local
script: tests/listarPedidosEstado.js
output: -

scenarios: (100.00%) 1 scenario, 3000 max VUs, 2m30s max duration (incl. graceful stop):
* default: Up to 3000 looping VUs for 2m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

✓ status is 200
✗ meets response time
↳ 2% - ✓ 933 / X 39648

checks.....: 51.14% 41514 out of 81162
data_received.....: 14 MB 115 kB/s
data_sent.....: 16 MB 136 kB/s
✗ errors.....: 100.00% 39648 out of 39648
    http_req_blocked.....: avg=58.94µs min=0s med=0s max=5.35ms p(90)=0s p(95)=527.1µs
    http_req_connecting.....: avg=51.92µs min=0s med=0s max=3.92ms p(90)=0s p(95)=523.1µs
✗ http_req_duration.....: avg=6.81s min=9.29ms med=8.36s max=9.07s p(90)=8.81s p(95)=8.93s
    { expected_response:true }.....: avg=6.81s min=9.29ms med=8.36s max=9.07s p(90)=8.81s p(95)=8.93s
    http_req_failed.....: 0.00% 0 out of 40581
    http_req_receiving.....: avg=86µs min=0s med=0s max=4.73ms p(90)=516.5µs p(95)=528.8µs
    http_req_sending.....: avg=22.72µs min=0s med=0s max=3.51ms p(90)=0s p(95)=56.4µs
    http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
    http_req_waiting.....: avg=6.81s min=9.29ms med=8.36s max=9.07s p(90)=8.81s p(95)=8.93s
    http_reqs.....: 40581 338.152418/s
iteration_duration.....: avg=6.81s min=9.8ms med=8.36s max=9.07s p(90)=8.81s p(95)=8.94s
iterations.....: 40581 338.152418/s
vus.....: 22 min=22 max=3000
vus_max.....: 3000 min=3000 max=3000

running (2m00.0s), 0000/3000 VUs, 40581 complete and 0 interrupted iterations
default ✓ [=====] 0000/3000 VUs 2m0s
ERRO[0120] thresholds on metrics 'errors, http_req_duration' have been crossed

```

## Requerimiento 14:

El tiempo entre que el conductor selecciona el refrigerador y recibe la clave enviada por YAGNI no debe superar a un segundo bajo cualquier condición de carga en el sistema

```
execution: local
script: createOTP.js
output: -

scenarios: (100.00%) 1 scenario, 3000 max VUs, 2m30s max duration (incl. graceful stop):
  * default: Up to 3000 looping VUs for 2m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

✓ response status was 200
X meets response time
  ↳ 11% - ✓ 5100 / X 41136

checks.....: 55.51% 51336 out of 92472
data_received.....: 17 MB 137 kB/s
data_sent.....: 16 MB 130 kB/s
X errors.....: 100.00% 41136 out of 41136
  http_req_blocked.....: avg=44.16µs min=0s med=0s max=4.68ms p(90)=0s p(95)=505.52µs
  http_req_connecting.....: avg=39.86µs min=0s med=0s max=4.2ms p(90)=0s p(95)=503.7µs
  X http_req_duration.....: avg=4.97s min=45.97ms med=6.2s max=7.56s p(90)=7.24s p(95)=7.36s
    { expected_response:true }...: avg=4.97s min=45.97ms med=6.2s max=7.56s p(90)=7.24s p(95)=7.36s
  http_req_failed.....: 0.00% 0 out of 46236
  http_req_receiving.....: avg=132.13µs min=0s med=0s max=15.29ms p(90)=132.39µs p(95)=524.9µs
  http_req_sending.....: avg=7.99µs min=0s med=0s max=2.21ms p(90)=0s p(95)=0s
  http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
  http_req_waiting.....: avg=4.97s min=40.17ms med=6.2s max=7.56s p(90)=7.24s p(95)=7.36s
  http_reqs.....: 46236 383.202257/s
iteration_duration.....: avg=5.97s min=1.04s med=7.2s max=8.56s p(90)=8.24s p(95)=8.36s
iterations.....: 46236 383.202257/s
vus.....: 65 min=65 max=3000
vus_max.....: 3000 min=3000 max=3000

running (2m00.7s), 0000/3000 VUs, 46236 complete and 0 interrupted iterations
default ✓ [=====] 0000/3000 VUs 2m0s
ERRO[0121] thresholds on metrics 'errors, http_req_duration' have been crossed
```

#### Requerimiento 16:

El tiempo máximo de respuesta entre que el usuario confirma el pedido y que YAGNI le responde no debe superar los 600 ms. con el sistema operando en modo normal y un segundo operando en modo sobrecargado.

```

execution: local
script: createPedido.js
output: -

scenarios: (100.00%) 1 scenario, 3000 max VUs, 2m30s max duration (incl. graceful stop):
  * default: Up to 3000 looping VUs for 2m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

✓ status is 201
✗ response time < 600ms
  ↳ 0% - ✓ 127 / ✗ 18389

checks.....: 50.34% 18643 out of 37032
data_received.....: 12 MB 102 kB/s
data_sent.....: 12 MB 100 kB/s
✗ errors.....: 100.00% 18389 out of 18389
  http_req_blocked.....: avg=171.36µs min=0s med=0s max=6.01ms p(90)=812.65µs p(95)=1.05ms
  http_req_connecting.....: avg=150.59µs min=0s med=0s max=6.01ms p(90)=580.34µs p(95)=1.04ms
✗ http_req_duration.....: avg=15.43s min=43.97ms med=18.01s max=21.15s p(90)=20.62s p(95)=20.85s
  { expected_response:true }.....: avg=15.43s min=43.97ms med=18.01s max=21.15s p(90)=20.62s p(95)=20.85s
  http_req_failed.....: 0.00% 0 out of 18516
  http_req_receiving.....: avg=126.72µs min=0s med=0s max=6.42ms p(90)=524.9µs p(95)=535.29µs
  http_req_sending.....: avg=42.17µs min=0s med=0s max=5.24ms p(90)=0s p(95)=518.22µs
  http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
  http_req_waiting.....: avg=15.43s min=43.97ms med=18.01s max=21.15s p(90)=20.62s p(95)=20.85s
  http_reqs.....: 18516 154.269832/s
iteration_duration.....: avg=15.43s min=43.97ms med=18.01s max=21.15s p(90)=20.62s p(95)=20.85s
iterations.....: 18516 154.269832/s
vus.....: 46 min=46 max=3000
vus_max.....: 3000 min=3000 max=3000

running (2m00.0s), 0000/3000 VUs, 18516 complete and 0 interrupted iterations
default ✓ [=====] 0000/3000 VUs 2m0s
ERROR[0120] thresholds on metrics 'errors, http_req_duration' have been crossed

```

## 2) Capturas de los prototipos en funcionamiento:

Como un supervisor cocina visualiza los productos prontos y puede armar lotes notificando al repartidor que ya están prontos para ser retirados:

### Módulo Cocina

#### Rol de usuario actual: Supervisor Cocina

Monitores: Visualización de productos a preparar

##### Cocina ID: 1

###### Local 1

- Producto ID: 11, Cantidad: 40 Armar Lote
- Producto ID: 1, Cantidad: 199996680 Armar Lote
- Producto ID: 2, Cantidad: 20 Armar Lote
- Producto ID: 4, Cantidad: 40 Armar Lote
- Producto ID: 3, Cantidad: 100 Armar Lote

Como un repartidor visualiza los lotes disponibles para retirar y los marca indicando que los retiró:

## Módulo Logística

### Rol de usuario actual: Repartidor

Repartidor: Gestión de lotes para tu camioneta

#### Camioneta ID: 1

- Cocina origen: 1 - Local destino: 1. Lote ID: 6, Producto ID: 1, Cantidad: 10
- Cocina origen: 1 - Local destino: 1. Lote ID: 2, Producto ID: 1, Cantidad: 10
- Cocina origen: 1 - Local destino: 1. Lote ID: 11, Producto ID: 1, Cantidad: 10
- Cocina origen: 1 - Local destino: 1. Lote ID: 12, Producto ID: 1, Cantidad: 10
- Cocina origen: 1 - Local destino: 1. Lote ID: 1, Producto ID: 1, Cantidad: 10

[Levantar lotes seleccionados](#)

Como un administrador puede ver el listado de pedidos incluyendo su estado y tiempo transcurrido desde que se hizo el pedido hasta que se completó. En tiempo real.

[ESTADO DE PRODUCTO](#) [PEDIDOS POR ESTADO](#) [COCINA](#) [REFRIGERADORES](#) [EXISTENCIAS POR](#)

#### Listado de Pedidos

1

- **Pedido del cliente con ID 1: Completo**

Tiempo transcurrido: 47 minutos

- **Pedido del cliente con ID 1: Iniciado**

Como un administrador puede hacer seguimiento en tiempo real de un producto

[ESTADO DE PRODUCTO](#)[PEDIDOS POR ESTADO](#)[COCINA](#)[REFRIGERADORES](#)

## Estado del Producto en Tiempo Real

5

[Iniciar Seguimiento](#)[Detener Seguimiento](#)

Estado: En Refrigerador

Como un administrador puede controlar y tener acceso a todos los refrigeradores de los distintos locales:

[PEDIDOS](#) [ESTADO DE PRODUCTO](#) [PEDIDOS POR ESTADO](#) [COCINA](#) [REFRIGERADORES](#) [EXISTENCIAS POR PRODUCTO](#)

### Administración de Refrigeradores

#### Selecciona un Local

- Local ID: 1 - Nombre: Local 1 [Seleccionar](#)
- Local ID: 2 - Nombre: Local 2 [Seleccionar](#)
- Local ID: 3 - Nombre: Local 3 [Seleccionar](#)
- Local ID: 4 - Nombre: Local 4 [Seleccionar](#)
- Local ID: 5 - Nombre: Local 5 [Seleccionar](#)

Como un administrador puede controlar y tener acceso a las distintas cocinas:

[PEDIDOS](#) [ESTADO DE PRODUCTO](#) [PEDIDOS POR ESTADO](#) [COCINA](#) [REFRIGERADORES](#) [EXISTENCIAS POR PRODUCTO](#)

### Administración de Cocinas

#### Lista de Cocinas

- Cocina ID: 1 - Direccion: Calle 1, Ciudad 1 [Seleccionar](#)
- Cocina ID: 2 - Direccion: Calle 2, Ciudad 2 [Seleccionar](#)
- Cocina ID: 3 - Direccion: Calle 3, Ciudad 3 [Seleccionar](#)
- Cocina ID: 4 - Direccion: Calle 4, Ciudad 4 [Seleccionar](#)
- Cocina ID: 5 - Direccion: Calle 5, Ciudad 5 [Seleccionar](#)

Panel general de un administrador:

## Seleccione el módulo al que desea ingresar:

Rol de usuario actual: Admin

[MÓDULO ADMINISTRADOR](#)[MÓDULO REFRIGERADORES](#)[MÓDULO LOGÍSTICA](#)[MÓDULO COCINA](#)[CERRAR SESIÓN](#)

Como un cliente o un repartidor que quiere interactuar con un refrigerador inteligente recibe su One Time Password para verificar su identidad:

<input type="checkbox"/>			<b>fedecabreradoglio</b> 100	Tu OTP para Refrigerador - Tu OTP es: 371460
<input type="checkbox"/>			<b>fedecabreradoglio</b> 100	Tu OTP para Refrigerador - Tu OTP es: 674964
<input type="checkbox"/>			<b>fedecabreradoglio</b> 100	Tu OTP para Refrigerador - Tu OTP es: 468129
<input type="checkbox"/>			<b>fedecabreradoglio</b> 100	Tu OTP para Refrigerador - Tu OTP es: 249729
<input type="checkbox"/>			<b>fedecabreradoglio</b> 100	<b>Tu OTP para Refrigerador</b> - Tu OTP es: 687976

Tu OTP para Refrigerador ➔ Recibidos x



**fedecabreradoglio@gmail.com**  
para mí ▾

dom, 24 nov, 1:16 a.m. (hace 1 día)



Traducir al español x

Tu OTP es: 468129



**fedecabreradoglio@gmail.com**  
para mí ▾

dom, 24 nov, 1:16 a.m. (hace 1 día)



Traducir al español x

Tu OTP es: 465039



**fedecabreradoglio@gmail.com**  
para mí ▾

dom, 24 nov, 1:16 a.m. (hace 1 día)



Traducir al español x

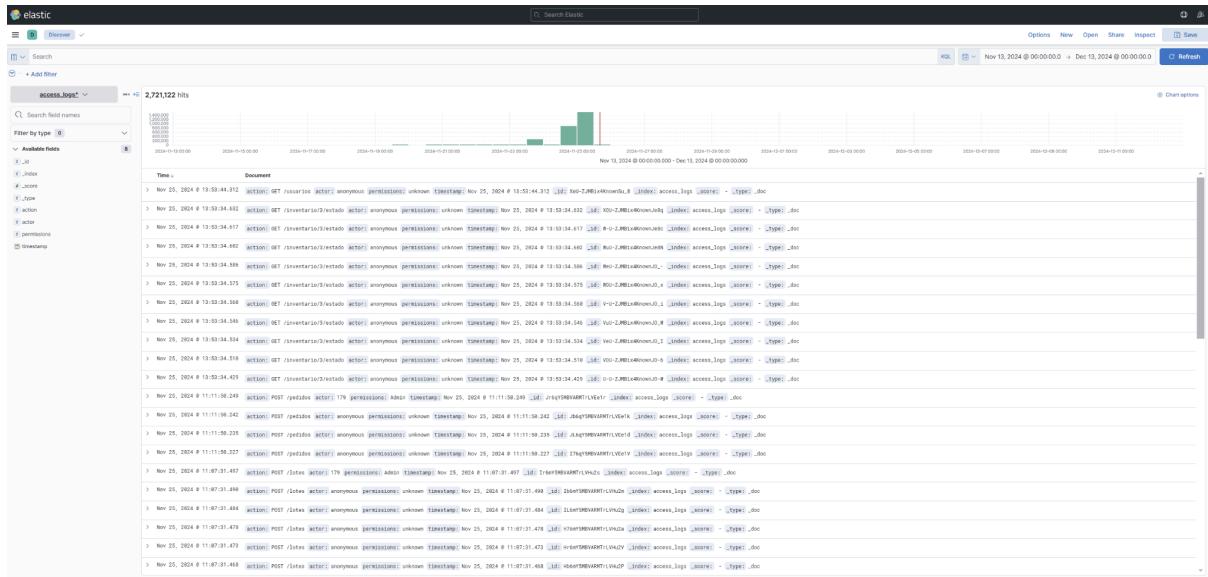
Tu OTP es: 261471

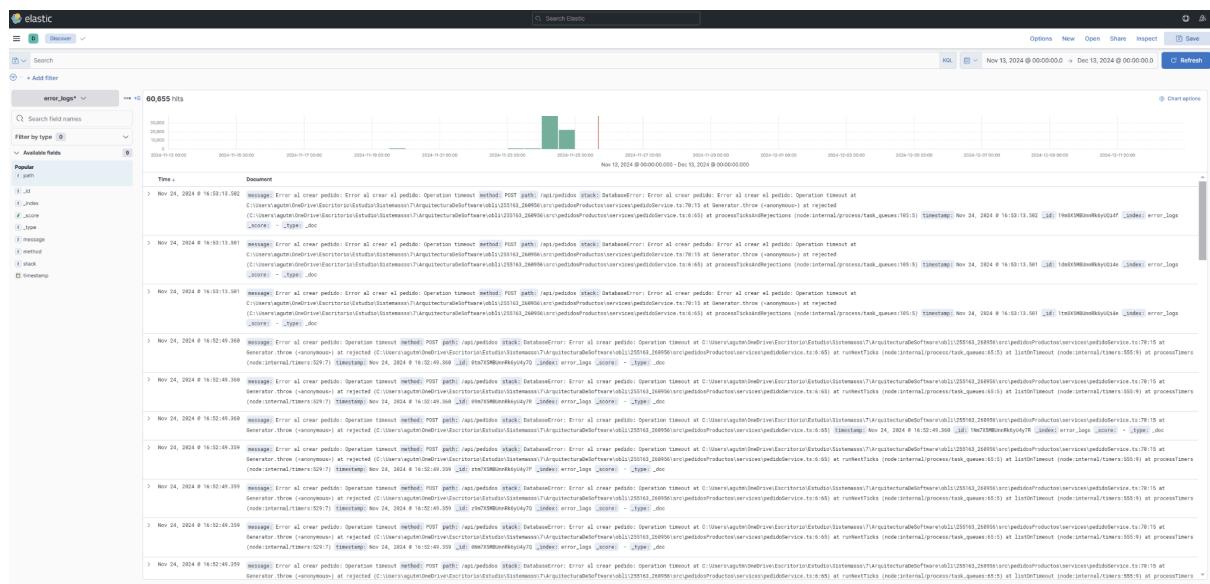
### **3) Capturas de la incorporación de herramientas digitales externas:**

Cómo visualizamos los usuarios registrados en nuestra plataforma mediante firebase:

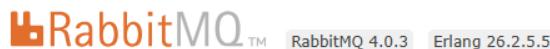
Yaganí	Authentication					
	Usuarios	Método de acceso	Plantillas	Uso	Configuración	Extensions
	<a href="#">Authentication</a>					
	<a href="#">Users</a>					
	Buscar por dirección de correo electrónico, número de teléfono o UID de usuario					
	<a href="#">Agregar usuario</a>					
Identificador	Proveedores	Fecha de creación	↓	Fecha de acceso	UID de usuario	
camioneta nueva@gmail.com		25 nov 2024		25 nov 2024	B6PTIwchSb0Rw6AqhbfmCx...	
cocina nueva@gmail.com		25 nov 2024		25 nov 2024	hSy4KNHS2ue9n1G08LSWRc...	
admin nuevo@gmail.com		25 nov 2024		25 nov 2024	XO4nObiuEPYaf5z2t0whwxZh...	
cliente nuevo@gmail.com		25 nov 2024		25 nov 2024	i3RKSnXOUNm0SEUKtLEfIp...	
admin@gmail.com		25 nov 2024		25 nov 2024	z1ZdXEYvHzd86CA81Bz0los...	
nuevo@mail.com		25 nov 2024		25 nov 2024	0enVV2gx3x6bzbsqoGSy0B...	
email@gmail.com		25 nov 2024		25 nov 2024	saWU04wWNkIWxnhXlZv4b...	
cliente uno@gmail.com		25 nov 2024		25 nov 2024	0FDd2hEW0MXN2fCLOYSR1...	
cli1@mail.com		25 nov 2024		25 nov 2024	bi88KOuh9PjHPckdP4QylgFP...	
cocina1@mail.com		25 nov 2024		25 nov 2024	RyGn7lnkHPU45xHbn8lIUaY...	
rep05@gmail.com		24 nov 2024		24 nov 2024	3v2vdrTWCmRg6uhLSuq43m...	
repartidor05@gmail.com		24 nov 2024		24 nov 2024	TKwV76v9wBP2yAoZ7mlCWx...	
cocina05@gmail.com		24 nov 2024		24 nov 2024	5Xa0UBqo40ZjqqtY4EaSuc...	
admin05@gmail.com		24 nov 2024		24 nov 2024	lQqrqrFemZhj5GfwKBFYFgJr...	
cliente05@gmail.com		24 nov 2024		24 nov 2024	10xLsRWWdsSzN6BnQKnL9...	
supcoocina@gmail.com		24 nov 2024		24 nov 2024	FDxJLzGnglTGGSBt7dP9Ez7...	

Como visualizamos los logs del sistema mediante Elasticsearch Kibana:





Mediante la interfaz de RabbitMQ vemos el exchange lotes y pedidos:



Overview    Connections    Channels    **Exchanges**    Queues and Streams    Admin

## Exchanges

▼ All exchanges (9)

Pagination

Page **1** **▼** of 1 - Filter:   Regex ?

Virtual host	Name	Type	Features	Message rate in	Message rate out	+/-
/	<b>(AMQP default)</b>	direct	D			
/	<b>amq.direct</b>	direct	D			
/	<b>amq.fanout</b>	fanout	D			
/	<b>amq.headers</b>	headers	D			
/	<b>amq.match</b>	headers	D			
/	<b>amq.rabbitmq.trace</b>	topic	D I			
/	<b>amq.topic</b>	topic	D			
/	<b>exchange_lotes</b>	direct	D			
/	<b>exchange_pedidos</b>	direct	D			

► Add a new exchange

## Exchange lotes:

RabbitMQ™ RabbitMQ 4.0.3 Erlang 26.2.5.5

Overview Connections Channels Exchanges Queues and Streams Admin

### Exchange: exchange\_lotes

Overview

Message rates last minute ?

Currently idle

Details

Type	direct
Features	durable: true
Policy	

Bindings

This exchange

↓

To	Routing key	Arguments	Bind
cola_camioneta_1	camioneta.1		Bind
cola_camioneta_10	camioneta.10		Bind
cola_camioneta_2	camioneta.2		Bind
cola_camioneta_3	camioneta.3		Bind
cola_camioneta_4	camioneta.4		Bind
cola_camioneta_5	camioneta.5		Bind
cola_camioneta_6	camioneta.6		Bind
cola_camioneta_7	camioneta.7		Bind
cola_camioneta_8	camioneta.8		Bind
cola_camioneta_9	camioneta.9		Bind

Add binding from this exchange

To queue :  \*

Routing key:

Arguments:  =  String

Bind

▶ Publish message

▶ Delete this exchange

## Exchange: exchange\_pedidos

### Overview

Message rates [last minute](#) ?

Currently idle

#### Details

Type	direct
Features	durable: true
Policy	

### Bindings

This exchange



To	Routing key	Arguments	Actions
cola_cocina_1	cocina.1		<a href="#">Unbind</a>
cola_cocina_10	cocina.10		<a href="#">Unbind</a>
cola_cocina_11	cocina.11		<a href="#">Unbind</a>
cola_cocina_12	cocina.12		<a href="#">Unbind</a>
cola_cocina_13	cocina.13		<a href="#">Unbind</a>
cola_cocina_14	cocina.14		<a href="#">Unbind</a>
cola_cocina_15	cocina.15		<a href="#">Unbind</a>
cola_cocina_16	cocina.16		<a href="#">Unbind</a>
cola_cocina_17	cocina.17		<a href="#">Unbind</a>
cola_cocina_18	cocina.18		<a href="#">Unbind</a>
cola_cocina_19	cocina.19		<a href="#">Unbind</a>
cola_cocina_2	cocina.2		<a href="#">Unbind</a>
cola_cocina_20	cocina.20		<a href="#">Unbind</a>
cola_cocina_21	cocina.21		<a href="#">Unbind</a>
cola_cocina_22	cocina.22		<a href="#">Unbind</a>
cola_cocina_23	cocina.23		<a href="#">Unbind</a>
cola_cocina_24	cocina.24		<a href="#">Unbind</a>
cola_cocina_25	cocina.25		<a href="#">Unbind</a>
cola_cocina_26	cocina.26		<a href="#">Unbind</a>