URJC

DAS-P1-2023

Candel Casado Víctor
Esteban Martín Adrián
López Corchado Manuel
Gómez López Daniel
Soriano Aragón Adrián /
a.soriano.2021@alumnos.urjc.es

URJC, ETSII, Móstoles
Diseño y Arquitectura del Software

Contenido

1. Iteracion Semana 1

1.1. Resumen Iteracion 1

1.2. Roles del Equipo

Arquitecto Software Senior (ASS)	Arquitecto Software Junior (ASJ)	Arquitecto Software Cognitivos (ASC)
Manuel López	**Adrián Esteban**	**Adrián Soriano**
Víctor Candel		**Daniel Gómez**

ID	Nombre	Descripción
RF1	Migración a Microservicios	Migrar la arquitectura monolítica a una basada en microservicios. Este requisito es el objetivo principal del proyecto y afecta a todos los demás componentes del sistema.
RF2	Componente Gateway	Acceder a los datos de la empresa mediante un componente Gateway. Este componente actuará como intermediario entre los clientes y los servicios.
RF3	Gestión de Reparto y Rutas	Gestionar el reparto y las rutas de los camiones. Este componente se encargará de asignar los pedidos a las flotas de transporte, calcular las rutas óptimas de los camiones. (Plantearse si va a haber diferentes tipos de rutas y de camiones)
RF4	Pasarela de Pagos Externa	Realizar pagos mediante una pasarela externa. Este componente se integrará con una pasarela de pago externa que proporcionará otra empresa.
RF5	Acceso a Datos Personales de Clientes	Acceder a los datos personales de los clientes. Este componente permitirá consultar y modificar los datos personales de los clientes.
RF6	Realización de Pedidos	Realizar pedidos de los productos. Este componente permitirá a los clientes realizar pedidos de los productos disponibles.
RF7	Reporte de Incidencias	Reportar incidencias en el reparto. Este componente permitirá reportar a los gestores de las rutas cualquier tipo de incidencia que ocurra durante el reparto.
RF8	Estadísticas	Proporcionar estadísticas sobre el estado de los pedidos y los camiones. Este componente proporcionará información valiosa sobre el estado de los pedidos y la situación en tiempo real de los camiones.

1.3 Requisitos Funcionales del Sistema

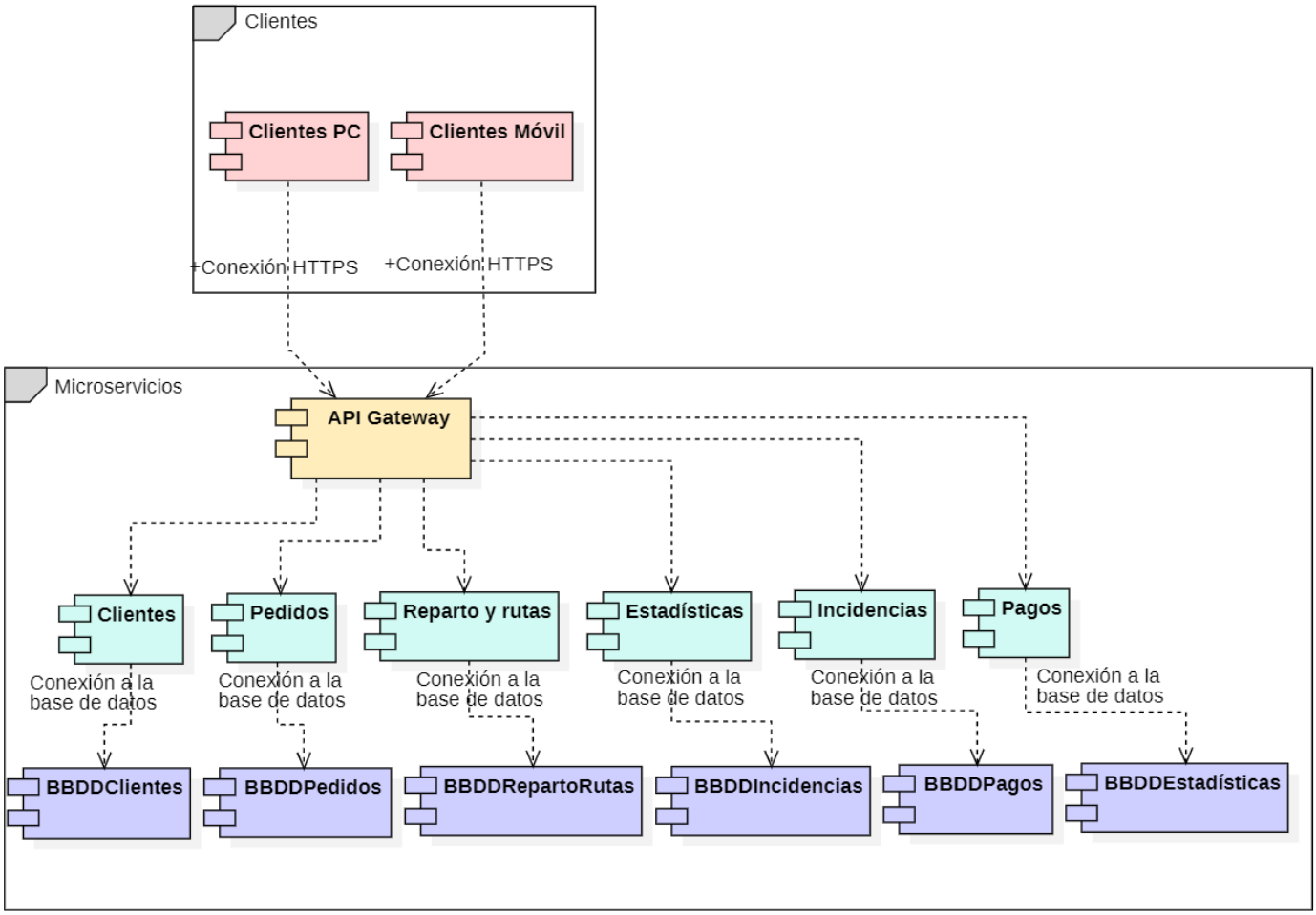
1.4 MADR Selección Arquitectura.

00002 - selección estilo arquitectura

ID	ADD - 0001
FECHA	22/10/2023
AUTORES	Manuel López Corchado y Víctor Candel Casado
ESTADO	**ACEPTADA**
PLANTEAMIENTO PROBLEMA	Se necesita encontrar una arquitectura de software que sea lo más compatible posible con una aplicación de una compañía de productos que permita gestionar clientes, pedidos, reparto y rutas, estadísticas, incidencias y pagos.
OPCIONES CONSIDERADAS	¤ **0002-1-Arquitectura Microservicios** ¤ **0002-2-Arquitectura Monolítica** ¤ **0002-3-Arquitectura Orientada a Servicios** ¤ **0002-4-Arquitectura por capas**
VENTAJAS	¤ **0002-1** Estabilidad. Modularidad. Código Reutilizable. Agilidad en cambios. Aplicación independiente. Menor riesgo. ¤ **0002-2** Simplicidad. Despliegue sencillo. Seguridad. ¤ **0002-3** Reutilización de los componentes Agilidad empresarial. ¤ **0002-4** Separación clara de responsabilidades. Simplicidad. Bajo coste.
INCONVENIENTES	¤ **0002-1** Alto consumo de memoria. Dificultad en la realización de pruebas. Gestión complicada por el número de componentes. Aislamiento, dificulta la depuración. Coste de implantación alto. ¤ **0002-2** Escalabilidad limitada, ineficiente al crecer Dependencia del servidor ¤ **0002-3** Vulnerabilidad a ataques HTML y XML Gran consumo de recursos ¤ **0002-4** Los cambios pueden requerir desplegar todas las capas. Bajo rendimiento y escalabilidad complicada.

DECISIÓN FINAL	Trabajo redundante entre capas. 0002-1-Arquitectura Microservicios. La arquitectura de microservicios brinda más ventajas que las demás según los requerimientos del cliente y valoración del equipo, sus las desventajas las consideramos asumibles.

1.5. Diagrama de Componentes de la arquitectura



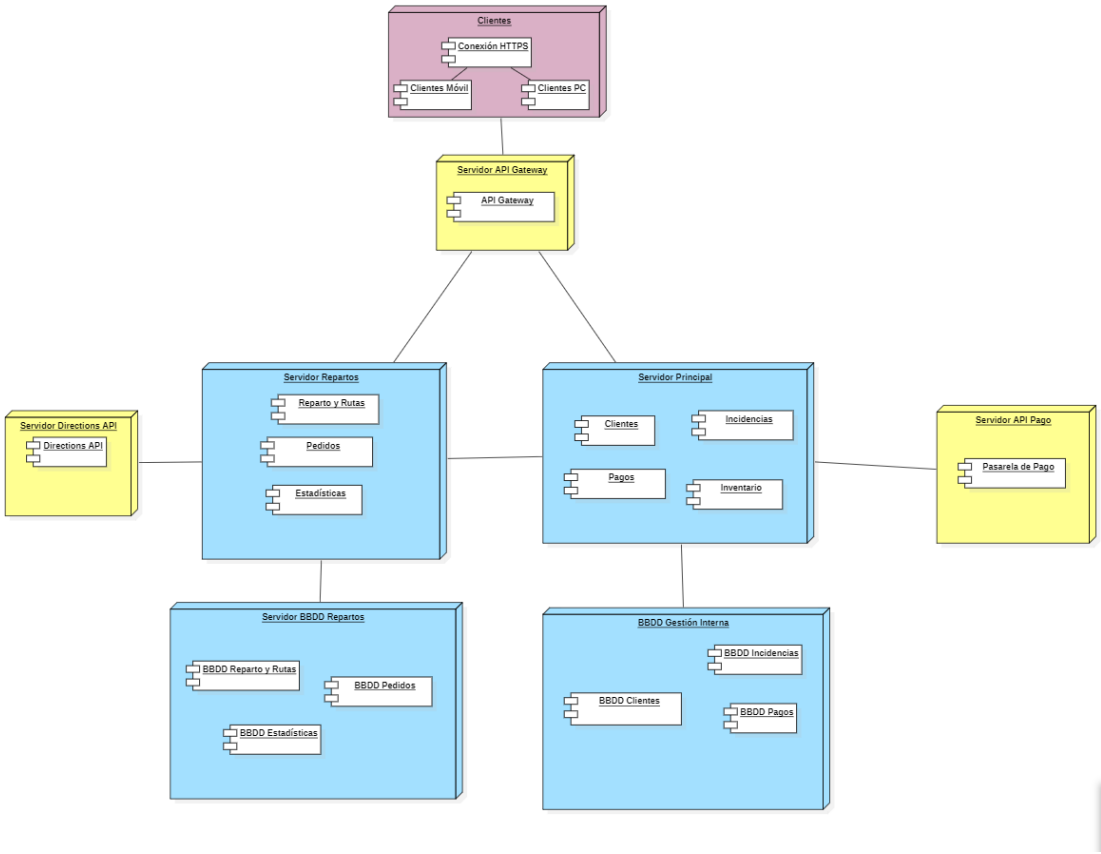
1.6 Tiempos estimados Iteracion 1.

Week	Iteration	Time in ADD (ASS)	Reflection Time (ASS-ASC)	Time in refined ADD (ASS)	Design Time (ASJ)
1	1	30	25	20	30
2	2				
3	3				
4	4				

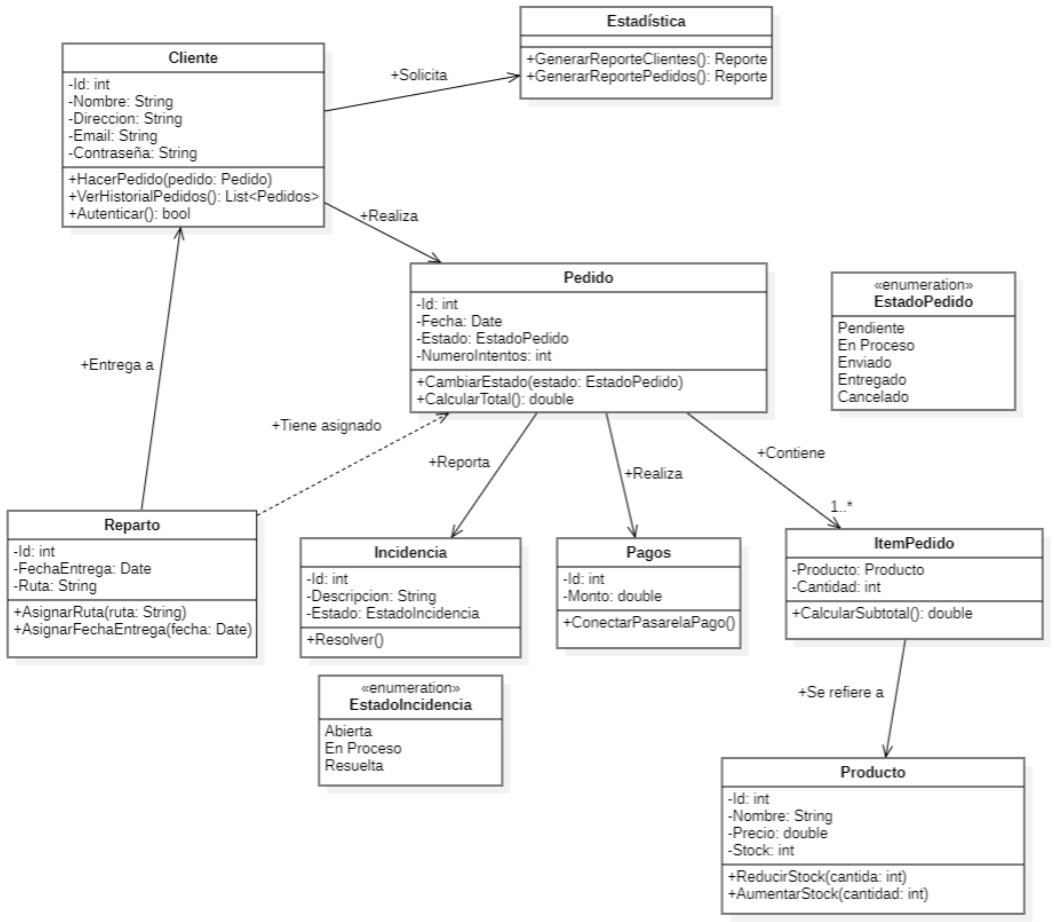
2. Segunda iteración

2.1 Resumen Iteracion 11.

2.2. Diagrama de Despliegue de la Arquitectura.



2.3 Diagrama de Clases



2.4 MADR

2.4.2. Selección API repartos

00004 – selección api gateway	
ID	ADD - 0004
FECHA	28/10/2023
AUTORES	Manuel López Corchado y Víctor Candel Casado
ESTADO	**ACEPTADA**
PLANTEAMIENTO PROBLEMA	Es esencial contar con una API de gestión de logística para coordinar la distribución de entregas a domicilio. Esta API debe estar diseñada para asistir a los conductores mediante el uso de un sistema GPS.
OPCIONES CONSIDERADAS	¤ 0004-1-Google Maps Platform ¤ 0004-2-Here Location Services ¤ 0004-3-MapBox
VENTAJAS	¤ **Google Maps Platform:** Amplia cobertura geográfica, extensa base de datos de ubicaciones de todo el mundo que facilitan la navegación en cualquier lugar. Actualizaciones en tiempo real con ajuste de rutas automático. ¤ **Here Location Services:** Optimización de rutas avanzado. Geocodificación precisa. Disponibilidad de mapas personalizados. ¤ **MapBox:** Flexibilidad estética. Datos de tráfico en tiempo real. Integrable en dispositivos móviles.
	¤ **Google Maps Platform:** Empleo de bases de datos preconfiguradas que ocasionan una carga excesiva en la arquitectura. Reducción considerable del desempeño cuando se acumulan numerosas rutas en el gateway. Ciertas funciones son restringidas tras un pago. ¤ **Here Location Services:** Compleja configuración inicial. La optimización de rutas lleva más tiempo que las otras propuestas. Servicios extra de un coste elevado. ¤ **MapBox:** Limitaciones de cobertura geográfica. Menos opciones de personalización.
DECISIÓN FINAL	Google Maps Platform es nuestra elección debido a que sus múltiples ventajas opacan a las de sus competidores. La multiplataforma, la estimación de tiempo y distancia precisas son lo que nos ha llevado a tomar esta decisión.

2.4.3. Selección repartos SKU

00005 – selección Google Maps Platform SKU	
ID	ADD - 0005
FECHA	28/10/2023
AUTORES	Manuel López Corchado y Víctor Candel Casado
ESTADO	**ACEPTADA**
PLANTEAMIENTO PROBLEMA	Habiendo tomado Google Maps Platform como nuestra API de geolocalización, hemos de decidir que versión elegir en base a nuestras necesidades.
OPCIONES CONSIDERADAS	¤ 0005-1-Directions ¤ 0005-2-Advanced Directions
VENTAJAS	¤ **Directions:** Menor coste económico. ¤ **Advanced Directions:** Mejor optimización. Geocodificación precisa. Hasta 25 puntos de parada por ruta.
	¤ **Directions:** Hasta 10 puntos de parada en cada ruta. Menos preciso y optimizado. ¤ **Advanced Directions:** Bastante costoso.
DECISIÓN FINAL	Advanced Directions es nuestra elección, debido a que aunque el precio sea más elevado ofrece más ventajas que la versión básica y consideramos que vale la pena gastar más para optimizar el servicio de entrega, la eficiencia beneficiará a la empresa y será una inversión para el ahorro de tiempo y recursos.

2.4.4. Selección API Gateway

00006 – selección api gateway	
ID	ADD - 0006
FECHA	29/10/2023
AUTORES	Manuel López Corchado y Víctor Candel Casado
ESTADO	**ACEPTADA**
PLANTEAMIENTO PROBLEMA	En este proyecto, se necesita una API Gateway para poder mejorar la seguridad, la escalabilidad y la recopilación de datos mediante protocolos HTTP/Rest.
OPCIONES CONSIDERADAS	¤ **0006-1-Kong Gateway** ¤ **0006-2-APISIX**
VENTAJAS	¤ **Kong Gateway:** Facilita la administración Compatible con Lua y Go entre otros Buen rendimiento ¤ **APISIX:** Se aloja en la nube Excelente rendimiento. Admite complementos de desarrollo en varios lenguajes.
	¤ **Kong Gateway:** Empleo de bases de datos preconfiguradas que ocasionan una carga excesiva en la arquitectura. Reducción considerable del desempeño cuando se acumulan numerosas rutas en el gateway. Ciertas funciones son restringidas tras un pago. ¤ **APISIX:** Escasa documentación debido a su reciente salida al mercado. Requiere de supervisión constante para un funcionamiento óptimo.
DECISIÓN FINAL	APISIX tiene un rendimiento superior a sus rivales, además de una comunidad más activa actualmente. Esto junto con su compatibilidad con numerosos lenguajes de programación hacen de esta API la mejor opción.

2.4.5. Selección API BDD

	00007 – selección patrón de bases de datos
ID	ADD - 0007
FECHA	29/10/2023
AUTORES	Manuel López Corchado y Víctor Candel Casado
ESTADO	**ACEPTADA**
PLANTEAMIENTO PROBLEMA	Para poder tener una arquitectura basada en microservicios, necesitamos disponer de múltiples bases de datos. Es por esto que planeamos establecer un patrón de bases de datos.
OPCIONES CONSIDERADAS	¤ **0007-1-CQRS Comand Query Responsibility** ¤ **0007-2-Database per Service**
VENTAJAS	¤ **CQRS:** Rendimiento y escalabilidad al separar las operaciones de lectura y actualización. Gestión de seguridad eficaz. Sistema flexible a evoluciones. ¤ **Database per Service:** Desacoplamiento Reutilización de negocio y separación de responsabilidades.
	¤ **CQRS:** La implementación puede agregar complejidad adicional al sistema. El aprendizaje de nuevos miembros del equipo sobre este patrón puede ser lento. ¤ **Database per Service:** La separación de las bases de datos puede ser un inconveniente ya que el servidor consume muchos recursos. La complejidad de mantenimiento y gestión es mayor.
DECISIÓN FINAL	Database per service es nuestra elección debido a que se adecua a nuestros objetivos de microservicios al suplir el aislamiento entre los distintos sectores del servicio.

2.5 Tiempos estimados Iteracion 11.

Week	Iteration	Time in ADD (ASS)	Reflection Time (ASS-ASC)	Time in refined ADD (ASS)	Design Time (ASJ)
1	1	30	25	20	30
2	2	45	20	20	50
3	3				
4	4				