SISTEMA DE DELIVERY ‘RLXDFOOD’ PARA RESTAURANTES DE LA CIUDAD DE POPAYÁN.

DEFINICIÓN DE LA ARQUITECTURA.

Autores:

Juan David Muñoz Pasquel
Whalen Stiven Caicedo Obando

INGENIERÍA DE SOFTWARE II

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

PROGRAMA DE INGENIERÍA DE SISTEMAS

2021

Tabla de contenido

1. Resumen

Una de las actividades con mayor demanda, aceptación y oportunidad de crecimiento que ha traído consigo la expansión de las tecnologías de la información a nivel global, es el Delivery (Servicio a través del cual un restaurante reparte sus platos a domicilio) a través de las plataformas digitales; algunos ejemplos son las plataformas "IFood", "Rappi", "Uber eats" entre otras muchas aplicaciones existentes que brindan la opción a los restaurantes de tomar parte en la denominada transformación digital. RlxdFood surge como una nueva opción de Delivery para los restaurantes de la ciudad de Popayán, esta es la aplicación que se va a construir como solución y aporte a la opción de Delivery de los distintos restaurantes, partiendo de una necesidad común y unos requerimientos específicos; en este documento se plantea la descomposición de la arquitectura (teniendo en cuenta y/o priorizando algunas de las vistas más importantes para la descomposición y el entendimiento del sistema)

2. Propósito

El modelado y la descomposición del sistema de Delivery 'RlxdFood' para restaurantes en Popayán se han realizado en diferentes documentos; hasta el momento se cuenta con tres de ellos: **Documento de Requerimientos, Documento de Descomposición de arquitectura y Documento de presentación,** pensados para los diferentes tipos de público que pudieran necesitar la documentación respectiva a sus diferentes áreas competentes. El propósito de este documento es exponer la descomposición realizada de la arquitectura, priorizando algunas de las vistas e incluso, omitiendo algunas vistas que más adelante, en siguientes versiones de los documentos, podrían aparecer.

3. Representación de la arquitectura

En la figura 1 se muestra un esquema básico general de la arquitectura del sistema de Delivery "RlxdFood" para restaurantes de Popayán. En este esquema, se evidencia que, hasta el momento, se cuenta con ocho subsistemas, los cuales serán base para el funcionamiento de la aplicación, cumpliendo – cada uno de ellos – un rol específico y una función determinada. Como primera instancia de la descomposición del sistema, se tocarán algunos de ellos (Sistema de gestión de restaurantes, Sistema de gestión de componentes, Sistema de gestión de usuarios, Sistemas de gestión de platos) considerados como los subsistemas que brindarán valor al cliente desde el primer momento.

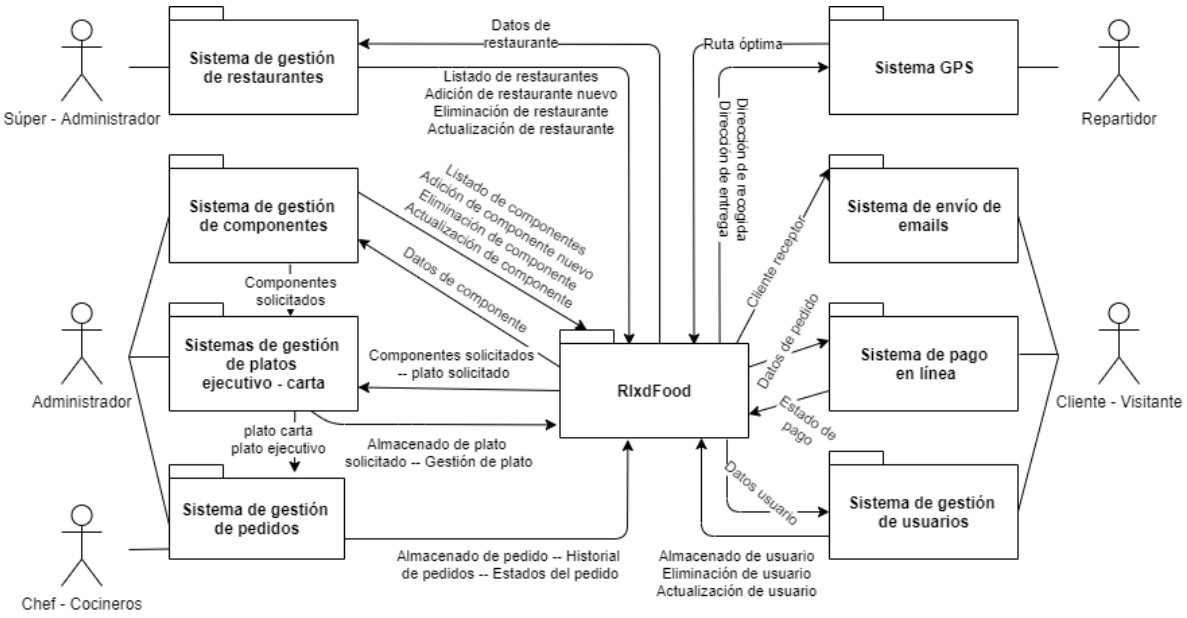


Figura 1 Esquema básico general del sistema

A continuación, se denotan las vistas que se trabajarán a lo largo de este documento, como parte de la primera versión de descomposición de la arquitectura.

a. **Vista de requerimientos:** La vista de requerimiento es casi obligatoria, puesto que, conforme a los requerimientos que se tengan, se tomarán las decisiones correspondientes acerca de qué patrones arquitecturales se utilizarán en base a los atributos priorizados. Es importante tener en cuenta que este documento es de descomposición arquitectural, por ende, esta vista indicará única y exclusivamente los requerimientos de importancia arquitectural.
b. **Vista lógica:** Al hablar de subsistemas, la vista lógica se hace presente casi de inmediato, puesto que esta vista permitirá conocer una descomposición de los módulos, contenedores y componentes que presente el subsistema que se esté analizando. En este documento se hará un análisis de los diferentes subsistemas que se tocarán como parte de la primera descomposición de la arquitectura.
c. **Vista de implementación:** Una vez se tiene claridad sobre los módulos, contenedores y componentes de los diferentes subsistemas, es posible tener una idea mucho más clara del funcionamiento de estos y su relación con el sistema en general, a través de la vista de implementación, donde se hará posible tener una idea de las dependencias necesarias a nivel de código.

4. Vista de requerimientos

Este documento tiene un propósito de descomposición de la arquitectura, por ende, sólo se indicarán, en esta vista, algunos de los requerimientos, atributos de calidad y escenarios relevantes a la hora de tomar decisiones arquitecturales, partiendo de la existencia de atributos priorizados, historias épicas y de usuario que los respalden y una modificación de la red de atributos.[1]

1. Ver documento de requerimientos – Link en la sección de anexos

4.1. Historias épicas de importancia arquitectural

Prioridad	Código	Historia épica	Contexto
1	HE01	Yo **como** administrador del restaurante **necesito** registrar los componentes del almuerzo ejecutivo **para** posteriormente ofrecer a través de la plataforma el almuerzo de un día determinado.	Un componente del almuerzo ejecutivo debe tener un id, nombre, tipo (entrada, principio, proteína o bebida). Ejemplos de componentes son: Sopa de verduras, sopa de carantanta, frijoles, lentejas, pollo frito, pollo sudado, limonada, jugo de tomate etc.
2	HE02	Yo **como** administrador del restaurante **necesito** registrar el almuerzo ejecutivo del día en el sistema **para** que los clientes sepan que se ofrece como entrada, principio, proteína y bebida.	El restaurante puede ofrecer uno o más componentes por entrada, principio, proteína y bebida. Por ejemplo, un cliente puede elegir de principio: frijoles, arvejas o lentejas; de entrada: Sopa de verduras, sopa de maíz o frutas. Cuando algún componente se agote, el administrador lo puede quitar de la oferta para que los clientes no lo soliciten más. Se debe además poder seleccionar y agregar una imagen del almuerzo ejecutivo. Además, se puede agregar una ligera 2 descripción del plato ejecutivo que motivará al

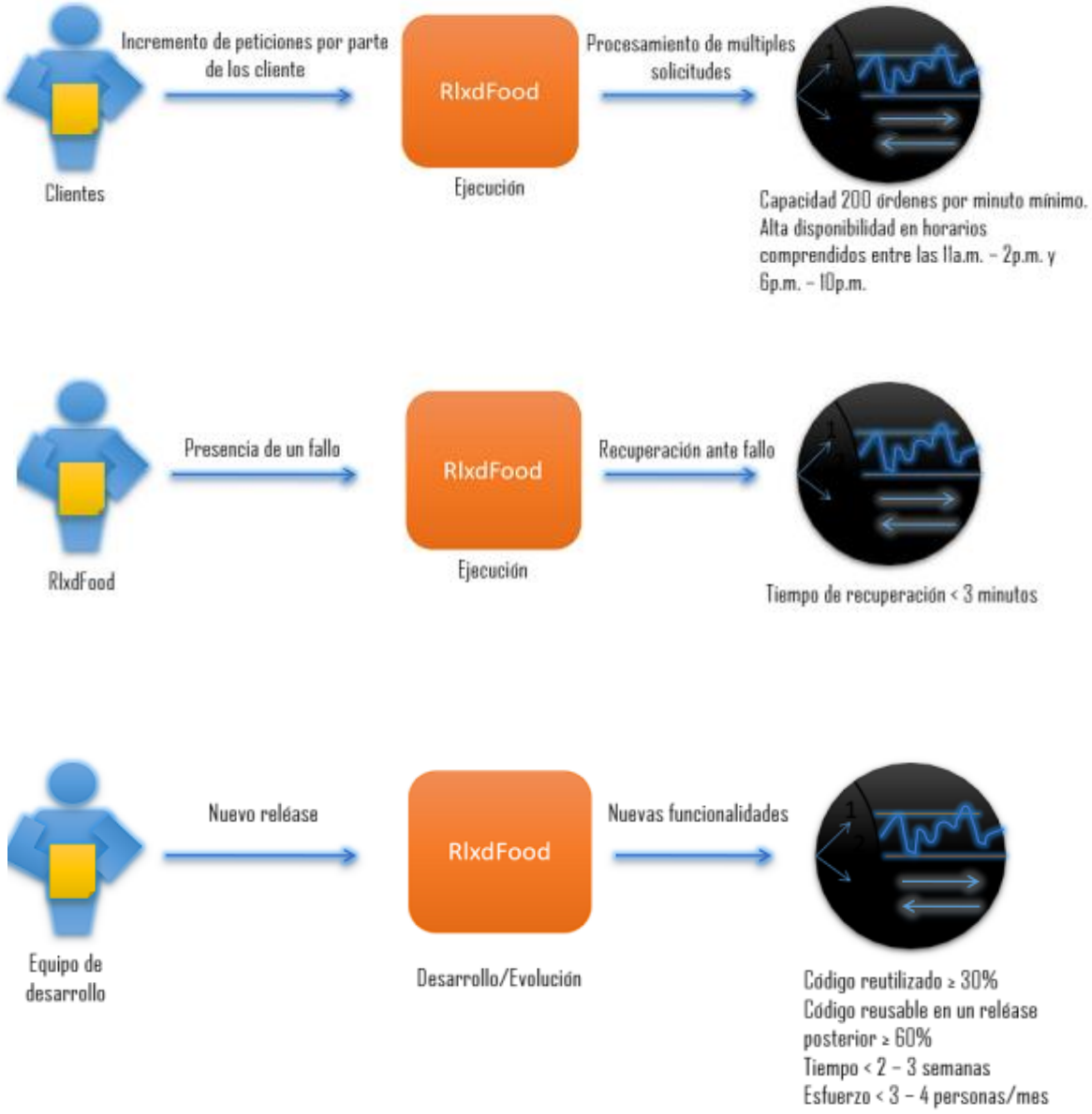
			cliente a realizar el pedido.
3	HE03	Yo **como** administrador del restaurante **necesito** registrar los platos a la carta que ofrece el restaurante **para** que los clientes puedan conocerlos y pedir a través del sistema.	Cada plato a la carta debe tener un nombre, una descripción y una imagen. Ejemplos de plato especiales son: pechugas rellenas con tocino y queso, bandeja paisa, italian sausage and peppers, cheesy broccoli stuffed chicken breast, etc.
9	HE09	Yo **como** súper – administrador de la plataforma **necesito** registrar un nuevo restaurante **para** que el restaurante pueda posteriormente ofrecer sus platos y atender pedidos a través de la plataforma.	Un restaurante tiene un nit, nombre, eslogan, propietario, dirección, teléfonos, ciudad, administrador y fotografía. El listado de restaurantes se debe mostrar en una tabla que permita filtrar de acuerdo a su nombre, nit o el nombre del administrador.

Tabla 1Requerimientos de importancia arquitectural

4.2. Atributos de calidad de importancia arquitectural

Teniendo en cuenta los requerimientos que se presentan para el proyecto (especificados en el documento de requerimientos) se tiene la priorización de los siguientes atributos de calidad: **Usabilidad, Eficiencia de desempeño, Adecuación funcional, Fiabilidad, Mantenibilidad, Portabilidad;** de los cuales, se encontró una mayor necesidad e inclinación hacia los siguientes: **Eficiencia de desempeño, Adecuación funcional, Mantenibilidad, Fiabilidad.** Esto permite poner atención especial en algunos atributos y posponer algunos otros, como lo son la Usabilidad y la Portabilidad. En las siguientes secciones (específicamente en la sección "Resumen de las decisiones de arquitectura") se aclarará qué importancia tiene esta preselección a nivel arquitectural; por ahora, se debe tener en cuenta que el proyecto tiene como prioridad los seis atributos anteriormente mencionados y, de ellos, cuatro son de mayor importancia.

4.3. Escenarios de calidad de importancia arquitectural



5. Resumen de las decisiones de arquitectura

Tomando como base los atributos que se han priorizado (mencionados en la sección 5.2 de este documento) y de acuerdo a los requerimientos (aquellos seleccionados como de importancia arquitectural – Mencionados en la sección 5.1 de este documento) es posible presentar las siguientes decisiones – arquitecturalmente hablando.

- **Estilo arquitectónico:** Los requerimientos mostrados de importancia arquitectural indican que cada restaurante tiene un tipo de plato (hasta el momento pueden ser platos ejecutivos o platos a la carta) que con el pasar del tiempo, puede escalar (ya no serían solamente platos ejecutivos y platos a la carta, sino que puede incluirse comida rápida, por ejemplo), esto permitirá que muchos restaurantes más se hagan aliados a la plataforma. Uno de los puntos

fuertes de los microservicios es, precisamente, el favorecer la escalabilidad, seguido de brindar facilidad en el mantenimiento de cada subsistema, además de permitir la reutilización de código a la hora de un nuevo reléase, favoreciendo así uno de los atributos priorizados (Mantenibilidad). Por otro lado, una de las grandes ventajas de los microservicios es que permite especializar muy bien ciertas tareas (algo que puede ayudarnos a la hora de, por ejemplo, crear un nuevo tipo de plato – es posible tener microservicios que se especialicen en la gestión de cada tipo de plato) favoreciendo otro de los atributos de calidad priorizados anteriormente (Adecuación funcional). En contra parte, los microservicios presentan una gran desventaja y es que presentan debilidad a nivel de performance, pudiendo vulnerar uno de los atributos de calidad priorizados. Sin embargo, esta desventaja es mínima en comparación a lo que puede ofrecernos este estilo arquitectónico y, para tratar de disminuir su impacto (tanto como se pueda) y no vulnerar en gran medida el atributo de eficiencia de desempeño, es posible utilizar técnicas como los son los balanceadores de carga. A pesar de todo lo mencionado, la decisión que se ha tomado es optar por una arquitectura basada en servicios, en donde se comparte una única base de datos para todos los subsistemas; en esta medida, se aprovecha lo mejor de microservicios y lo mejor de una arquitectura en monolito.

- **Lenguaje de programación:** Se utilizará el lenguaje de programación Java EE, debido a que el grupo de desarrollo del proyecto cuenta con experiencia y un grado de experticia en el mismo. Además de esto, Java EE tiene varias especificaciones de API, como JDBC (brindando una alternativa para la conexión a base de datos), e-mail (necesaria para la interfaz y la interacción del repartidor). Por último, pero no menos importante, Java se ejecuta en una máquina virtual, brindando la opción de trabajarlo en múltiples plataformas.
- **Framework:** Para este desarrollo se hará uso de Spring Boot, framework partidario de SpringFramework. Este framework facilita tanto la tarea de desarrollo como la tarea de despliegue en un servidor. Spring Boot se enfoca en facilitar las tareas que no son de desarrollo, para que cada desarrollador se encargue y se enfoque únicamente en la tarea de desarrollar, ganando tiempo y esfuerzo.
- **Tipo de aplicación:** Lo adecuado para una aplicación de esta categoría, serían aplicaciones web, sin embargo, el equipo de desarrollo cuenta con falta de experiencia sobre las mismas y sobre aplicaciones Android. Por lo mismo, se toma la inclinación hacia aplicación de escritorio, con una posibilidad de complementarse muy bien, en un futuro y en versiones posteriores, con páginas y aplicaciones web, además de aplicaciones para smartphone también.

6. Modelo de contexto

En la figura 2[2] se muestra, como punto referencial, la aplicación que se va a construir “RlxdFood”, la cual se presenta en relación con seis actores principales y tres sistemas adicionales o de soporte a la aplicación. Los actores principales que se han identificado hasta el momento son: Chef – cocineros, repartidor, visitante, cliente, administrador del restaurante y súper administrador. Por otro lado, los sistemas que actúan en conjunto a la aplicación principal “RlxdFood” son: GPS system, E-mail system, online payment system (por el momento no se ha tomado decisión sobre el sistema específico o el api de terceros que se utilizará). Cada actor y sistema juega un rol importante dentro de la aplicación, los cuales se ven a continuación.

2. Diagrama disponible en:
https://drive.google.com/file/d/12G9ttxV_cWGGWpPNu5ZZOt9sB2iiDNbY/view?usp=sharing

- **Chef – cocineros:** Encargados de preparar los platos conforme a lo solicitado en el pedido. A nivel de la plataforma, quienes se encarguen de este rol tendrán la tarea de actualizar los estados del pedido “Recibido” – “En preparación” y “En preparación” – “Enviado”. Además de poder seleccionar el repartidor que hará la entrega.

- **Repartidor:** Persona encargada de recoger el pedido solicitado y transportarlo hacia la dirección estipulada. Dentro de la plataforma, esta persona también tiene la tarea de cambiar los estados del pedido “Enviado” – “En camino” y “En camino” – “Entregado” y, además de cambiar el estado del pedido como tal, también deberá cambiar su estado “Disponible” – “Ocupado” y “Ocupado” – “Disponible” para poder ser seleccionado para realizar entregas.

- **Visitante:** Un visitante es todo aquel que ingresa a la plataforma, puede mirar los restaurantes listados, los platos tanto ejecutivos como a la carta que ofrecen los diferentes restaurantes sin ningún tipo de inicio de sesión. Si un visitante desea realizar un pedido, debe registrarse en la plataforma y pasaría a ser llamado “Cliente”.

- **Cliente:** Un cliente es todo aquel que se ha registrado satisfactoriamente en la plataforma, puede realizar pedidos, observar el estado de su pedido y el tiempo estimado que tardará en ser entregado, además de las diferentes opciones de pago como lo son: tarjeta de crédito, PSE y efectivo.

- **Administrador del restaurante:** Un administrador del restaurante es quien se encarga de la gestión de los diferentes platos. La gestión puede incluir registro, actualización, eliminado de un componente de un plato ejecutivo, registro y actualización del plato ejecutivo como tal, registro de platos a la carta, entre otros.

- **Súper administrador:** Es quien se encarga de brindar soporte técnico a la aplicación (mantenimiento y solución de bugs) además, este súper administrador tiene como tarea la gestión de restaurantes, lo que incluye el registro, la actualización, eliminado, entre otros.

- **GPS System:** Sistema que brindará apoyo a la aplicación en el seguimiento del pedido en tiempo real. En adición, también proporcionará una ruta al repartidor desde el restaurante hasta la dirección estipulada para entregar los diferentes pedidos.

- **E – mail System:** El sistema de E – mail brindará apoyo a la aplicación con el envío de mensajes, notificaciones a los diferentes actores y dependiendo de las diferentes situaciones.

- **Online payment System:** El sistema de pago en línea será utilizado como medio para realizar los pagos de los diferentes pedidos que no se den en modalidad de “pago con dinero en efectivo

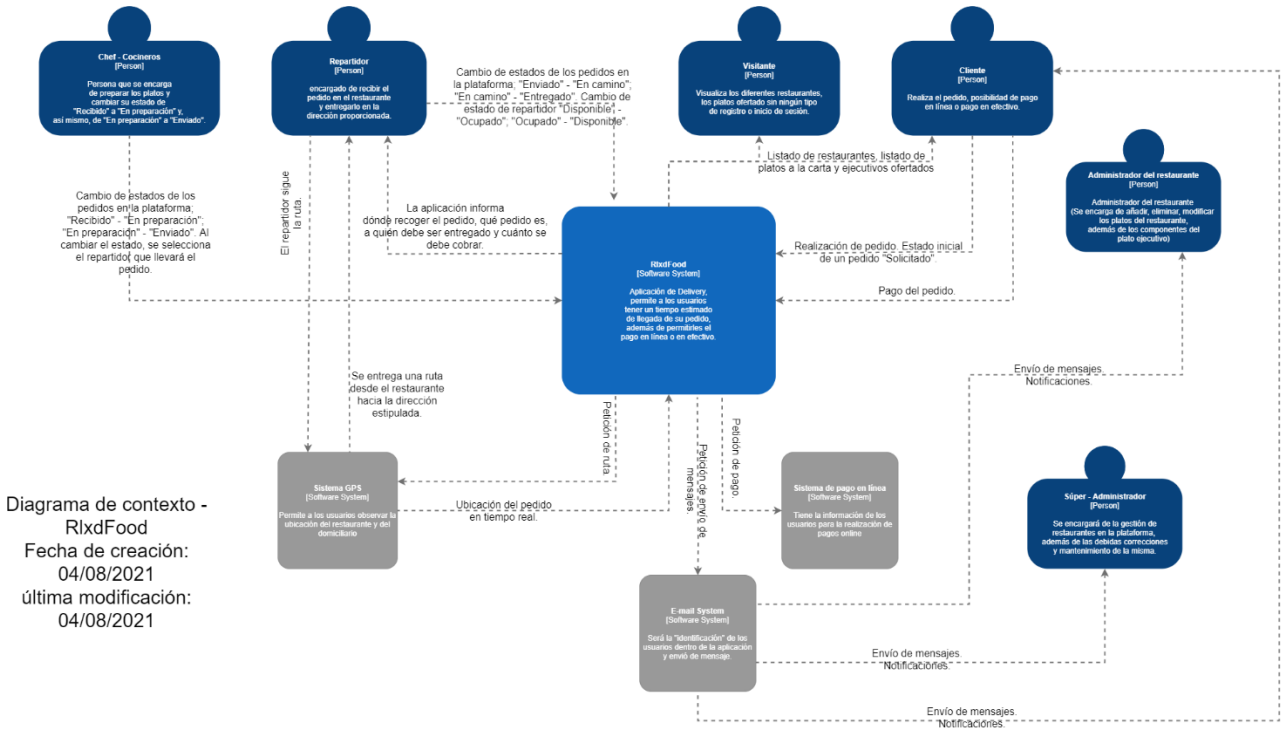


Figura 2 Modelo de contexto de la aplicación RlxdFood

7. Vista lógica

7.1. Vista de contenedores y componentes

Realizando un "Zoom" sobre RlxdFood (aplicación central a la que se hará descomposición de la arquitectura) en el diagrama de contexto, entra en juego el diagrama de contenedores[3], siendo RlxdFood el contenedor que estamos viendo, este cuenta con:

- **Base de datos:** Como se indicó anteriormente, se tiene una única base de datos para todos los subsistemas, aprovechando lo mejor tanto de microservicios como de la arquitectura de monolito.
- **Aplicación de escritorio:** La aplicación será el centro del desarrollo de este proyecto, pues esta aplicación será la que tenga interacción con los diferentes actores del sistema.
- **API'S de conexión:** Las Apis de conexión brindan la opción de consumir servicios de terceros que serán necesarios para una mejor experiencia por parte de algunos de los diferentes actores en el sistema. Hasta el momento se han contemplado 3 Apis de conexión, Api de conexión a servicio de E-mail, Api de conexión a servicio GPS, Api de conexión a servicio de pago en línea. Sin embargo, a medida que la aplicación crezca y los diferentes requerimientos de los Stakeholders evolucionen, podrían necesitarse algunas otras.

3. Diagrama disponible en:
https://drive.google.com/file/d/1rYT_ZSc9KqlqmpmlZ83gdq_QZG9FB_kG/view?usp=sharing

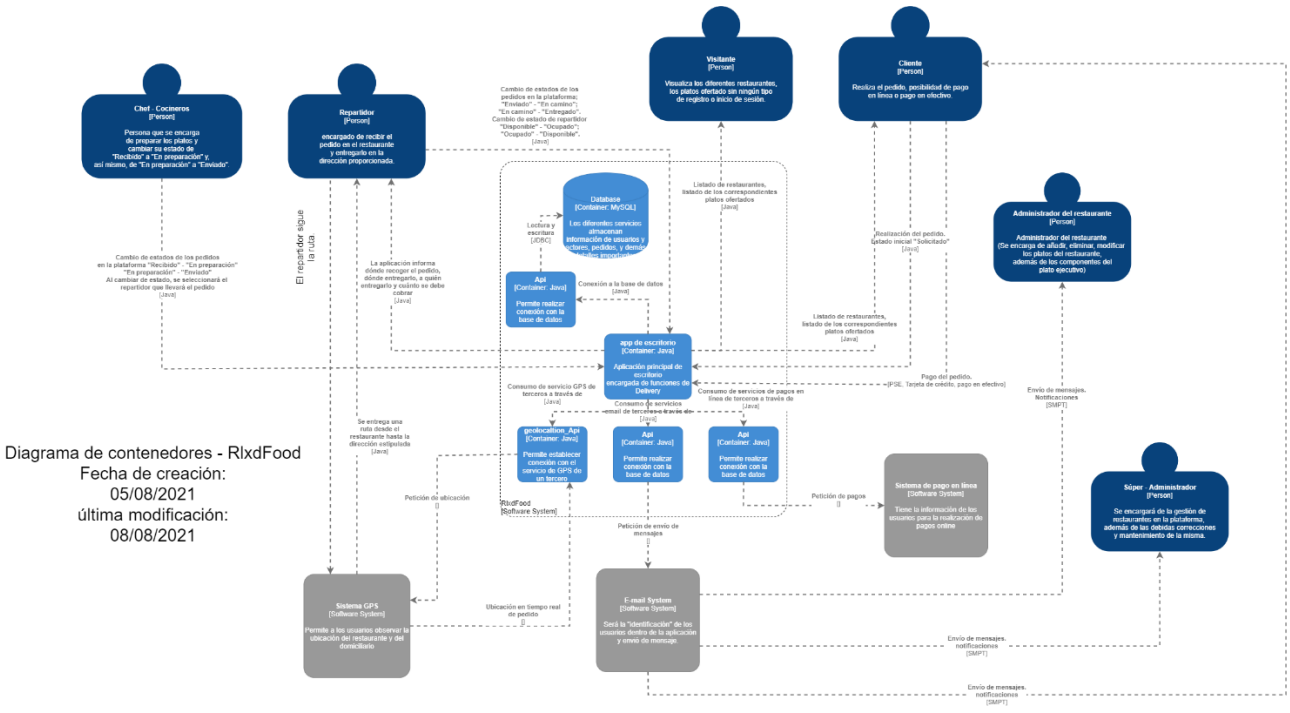


Figura 3 Modelo de contenedores de la aplicación RlxdFood

Posterior a una ampliación del modelo de contexto para determinar los contenedores que podría tener el sistema, se tiene que uno de los contenedores es "Aplicación de escritorio", el cual es el contenedor principal y en el que se basa el desarrollo del proyecto, pues desde donde se verá el valor que se le aporte al cliente.

Realizando un "Zoom" en el contenedor de "Aplicación de escritorio" se observan los diferentes componentes que actuarán para dar vida a este contenedor. En la figura 4[4] se evidencian los componentes mencionados. Nótese que cada uno de estos componentes responden a los subsistemas planteados en el diagrama de descripción básica general del sistema (Figura 1). Ahora bien, los componentes de Registro y Login hacen parte de un servicio más grande que corresponde al subsistema planteado en el diagrama de descripción básica general del sistema como "Sistema Gestión de usuario"; además de esto, el diagrama de la figura 4 muestra un componente de gestión de componentes, mismo que brindará alguno de los aspectos necesarios en otros componentes como lo es "Sistema de gestión de platos" (Figura 1). Es importante resaltar que en como primera instancia, se pensó en que todas las posibilidades de plato (plato ejecutivo, plato a la carta, en adelante – plato de comida rápida, por ejemplo) formaran parte de un solo subsistema "Gestión de platos". Sin embargo, se tomó la determinación de apuntar un poco más por la escalabilidad y se separó en diferentes microservicios cada plato que se tenga (si posteriormente se crea un nuevo tipo de plato, será un microservicio adicional), a pesar de esto, todos los microservicios de tipo de plato, formarán parte de un servicio más grande determinado como "Gestión de platos" dando como válido el componente que se tiene establecido tanto en el diagrama de la figura 1 como en el diagrama de la figura 4

4. Diagrama disponible en:
https://drive.google.com/file/d/1ARPTh7ZWDu7MmjACByjsEvGm5M-7N8Oe/view?usp=sharing

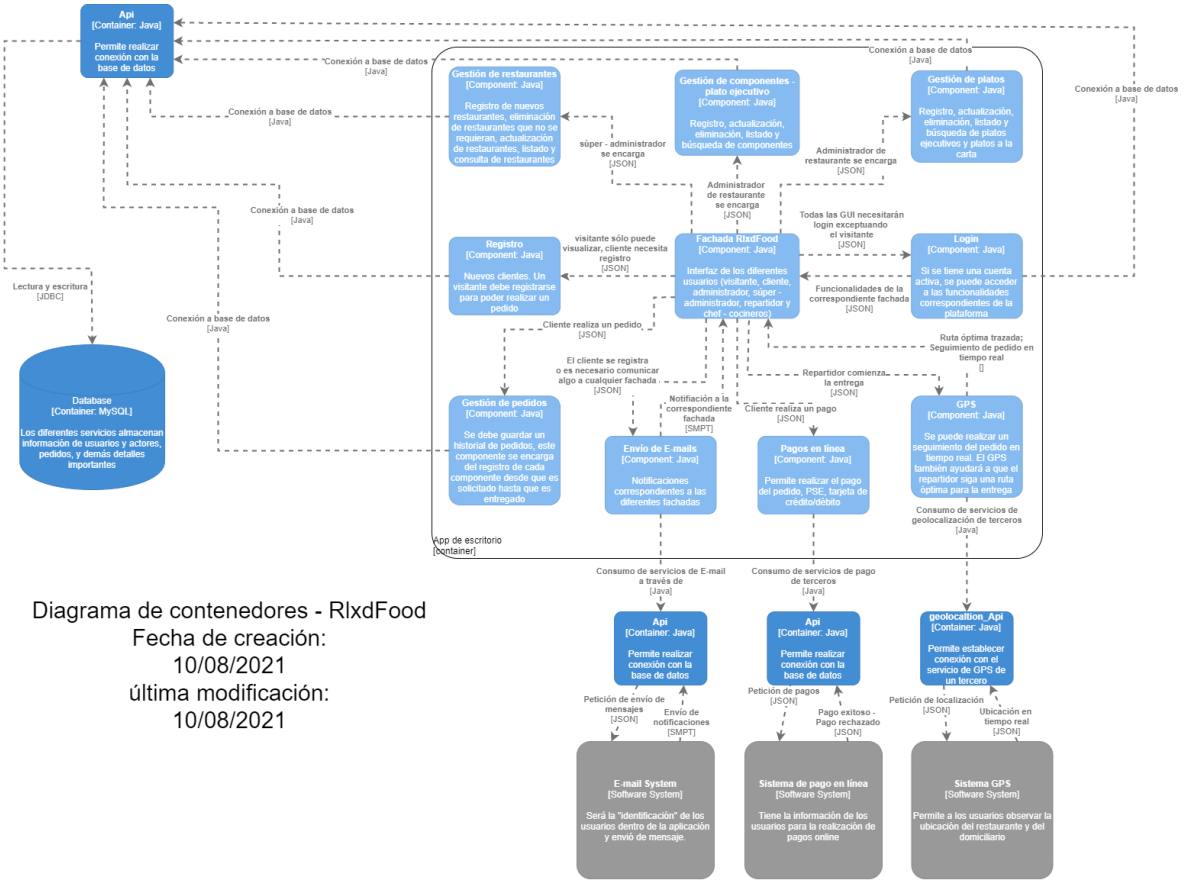


Diagrama de contenedores - RIxdFood
Fecha de creación:
10/08/2021
última modificación:
10/08/2021

Figura 4 Modelo de componentes de la aplicación RIxdFood

7.2. Vista de módulos

En los apartados anteriores, se ha determinado un gran desglose de la aplicación, el contexto, los contenedores, los componentes y algunos puntos base que darán soporte a la aplicación. Como un paso más profundo de descomposición, se procede a realizar la vista de módulos, donde se muestran los paquetes en los que se dividen los subsistemas. Es necesario hacer claridad que la mayoría de los subsistemas (al menos los que se tienen hasta el momento) cumplen con la misma estructura, la cual denota el diagrama de “Módulos – Packages generales a los subsistemas de la aplicación” (Figura 5). A pesar de esto, si habrá subsistemas que necesitarán acceso a más paquetes, como se evidencia en el diagrama de “Módulos – Packages del subsistema restaurant-handler-service” (Figura 6).

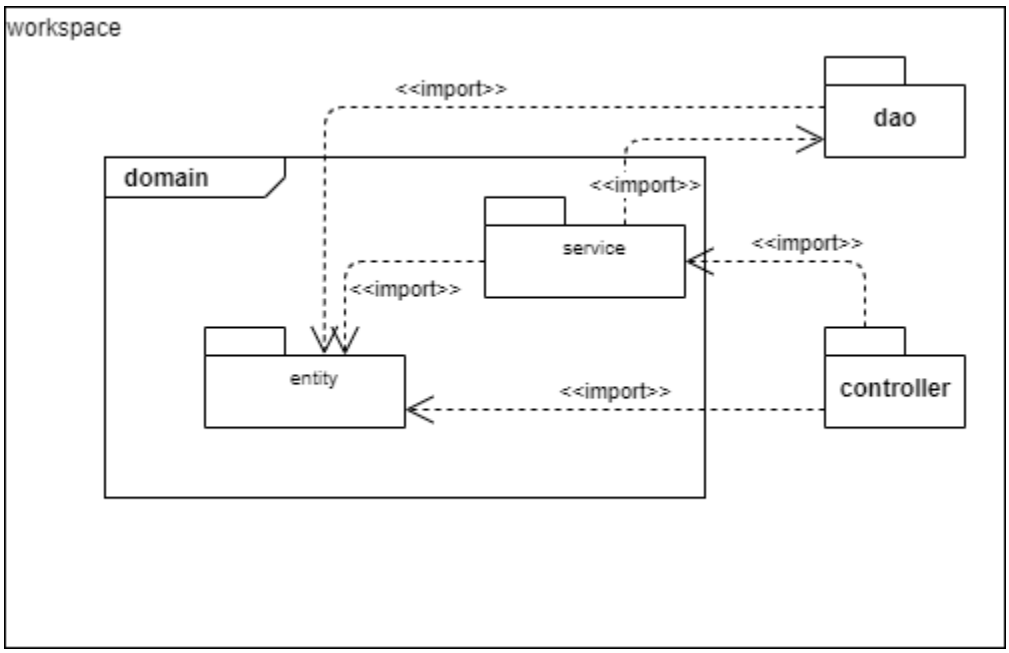


Figura 5 Módulos - Packages generales a los subsistemas de la aplicación RlxdFood

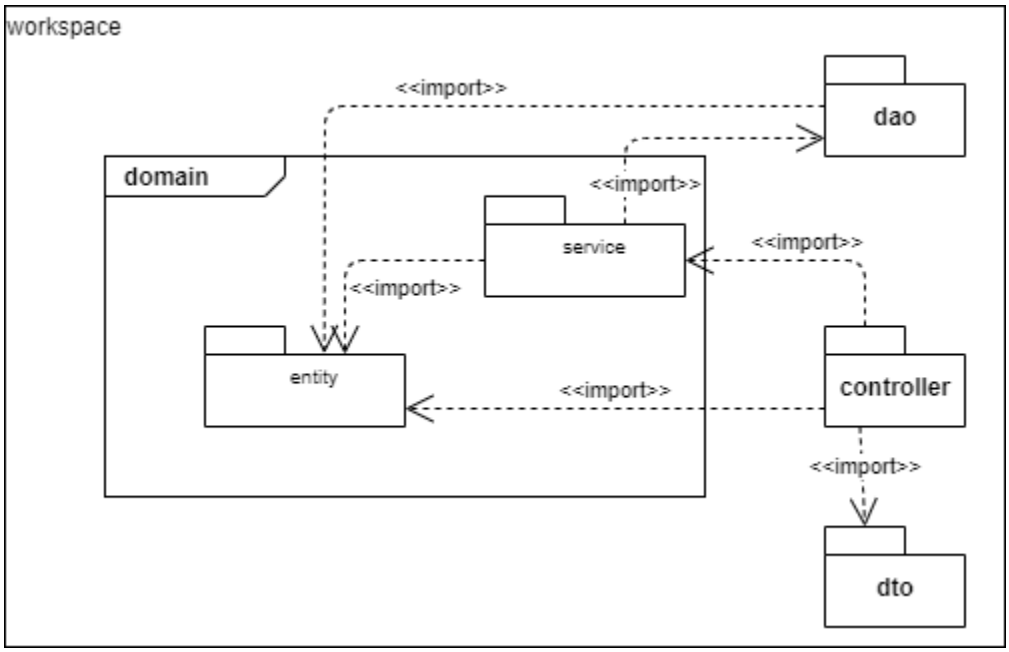


Figura 6 Módulos - Packages del subsistema restaurant-handler-service

7.3. Vista de componentes y conectores

Se muestran, a continuación, los componentes que harán parte de una primera versión de entrega, estos componentes son los que se considera entregarán valor al cliente desde el inicio.

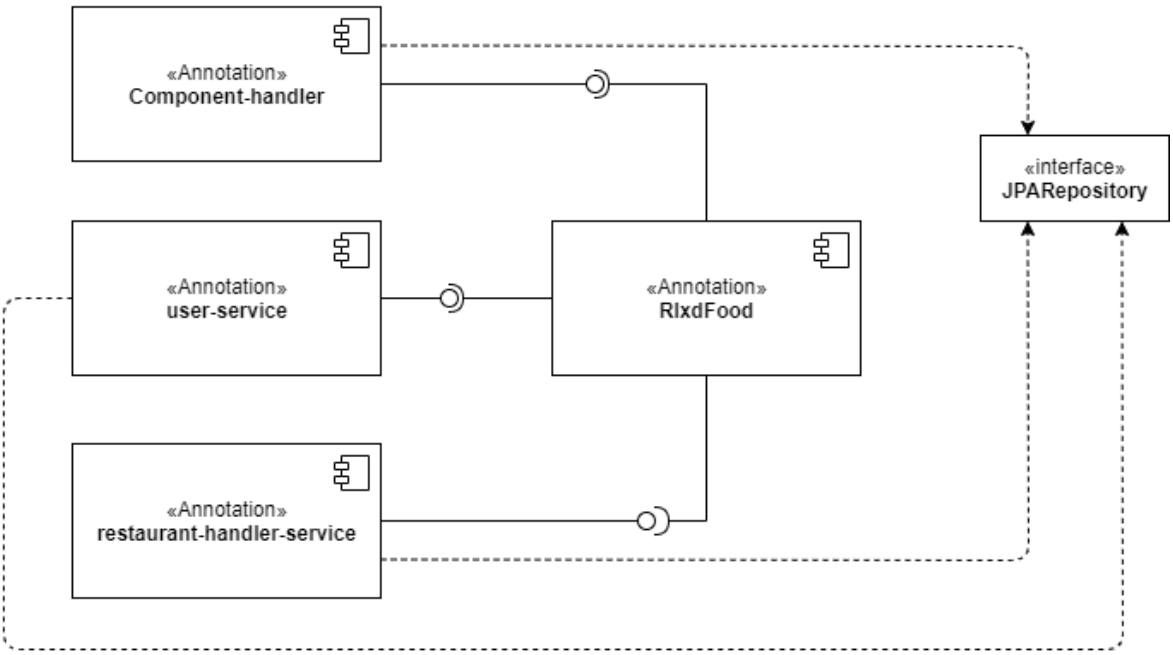


Figura 7 Diagrama de contenedores - 1era versión

8. Vista de implementación

En la figura 8 se muestra el diagrama de clases y la implementación del subsistema components – handler. Los demás subsistemas (al menos los que se tiene hasta el momento) siguen el mismo esquema, exceptuando el subsistema restaurant-handler-service, el cual utiliza un paquete adicional denominado dto.

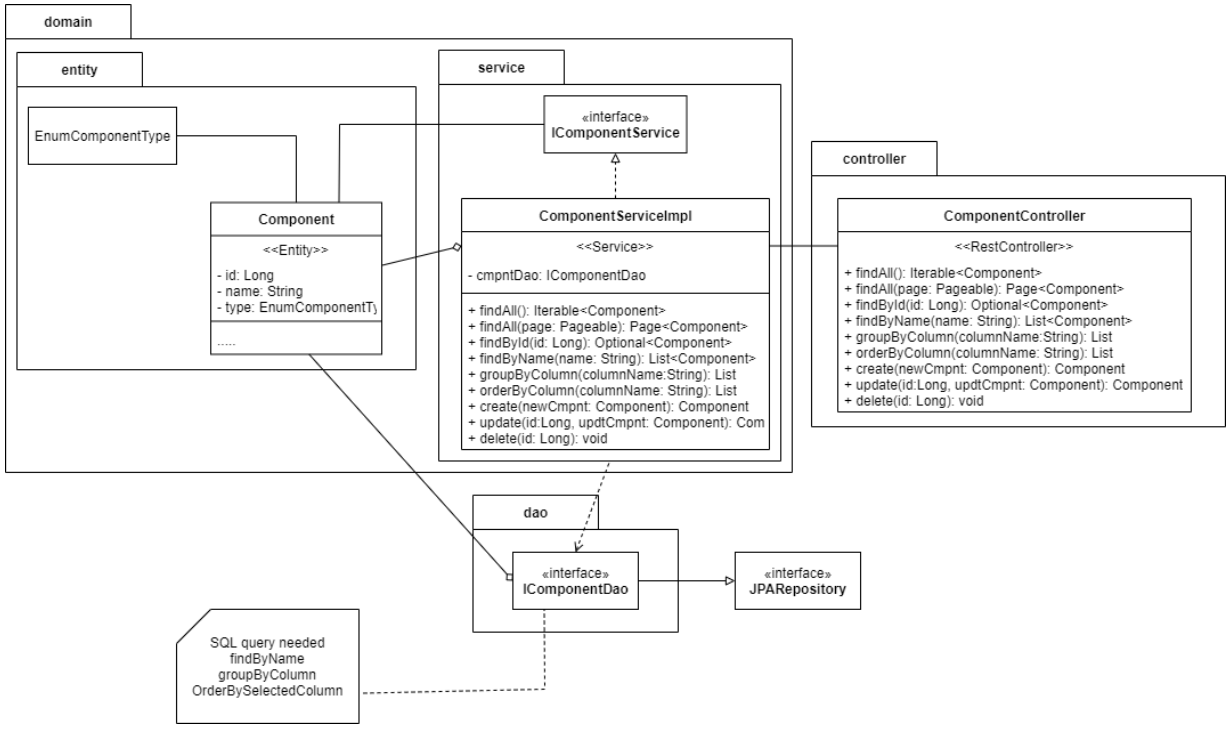


Figura 8 Diagrama de clases del subsistema components-handler

9. Anexos

Se anexa el enlace a Jira, donde se organizaron las tareas pendientes; se terminó al Sprint a la fecha de sustentación del proyecto (08/09/2021) donde aún se tenían tareas pendientes por lo que posiblemente se vea afectado el Burn Down Chart; estas tareas fueron terminadas posterior a la realización de la sustentación. Se comparte el link del repositorio de GitHub

- https://braquelsoft.atlassian.net/jira/software/projects/BBQLSOFT/boards/1
- https://github.com/JDMPasquel/RlxdFood_Microservices/blob/main/Documentacion_RlxdFood-requerimientos.pdf
- https://github.com/JDMPasquel/RlxdFood_Microservices.git