

# **Sistemas Basados en Conocimiento**

**Autor:** Jonathan Andres Rosero Soto

## **Binnarium Pizza Chatbot**

### **1. Objetivo general**

Desarrollar un chatbot para una pizzería con el propósito de automatizar el proceso de ventas mediante el uso de una ontología.

### **2. Objetivos específicos**

- Diseñar un flujo de procesos para la venta de pizzas
- Desarrollar un chatbot mediante la herramienta Telegram
- Instanciar un proceso para ordenar pizzas mediante la escritura de lenguaje natural.

### **3. Alcance**

#### **3.1. Declaración de problema**

El problema de los sistemas de pedidos de pizzas por llamada telefónica afecta a las personas que gustan de la pizza. El impacto del problema es la pérdida de clientes por una falta de atención al cliente. Una solución eficaz sería incluir un sistema de pedido de pizzas que recomiende pizzas con ingredientes al gusto de cada cliente mediante un chatbot.

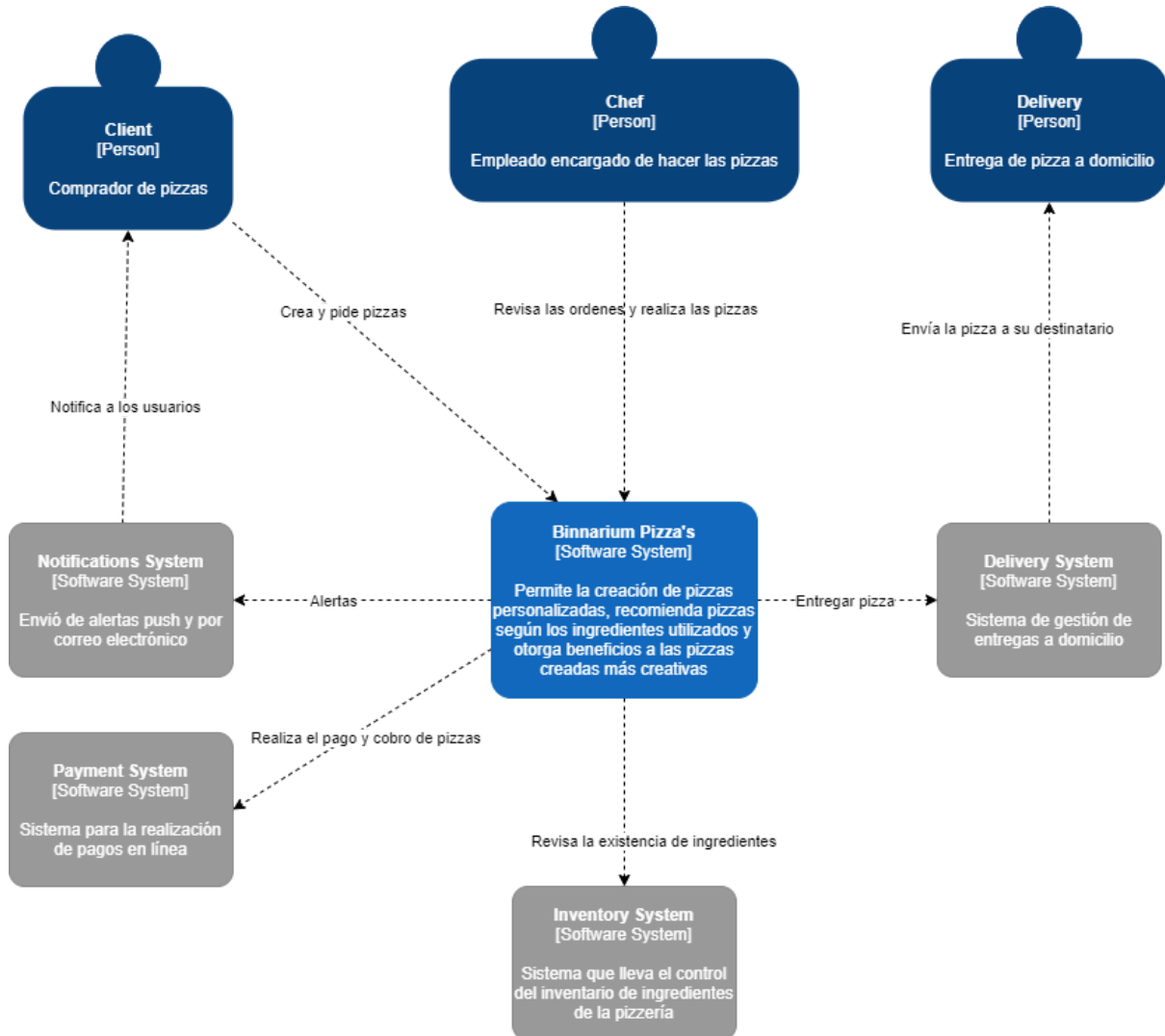
#### **3.2. Declaración de posición de producto:**

Para las personas que gustan de comprar y consumir pizzas . Binnarium Pizzas es un sistema de compra de pizzas que permite a los clientes personalizar su pedido de pizza, recomendarle una pizza según sus gustos y realizar un pedido. A diferencia de los pedidos por llamada telefónica, nuestro sistema se desarrolla en un chatbot con múltiples opciones y una experiencia centrada en la facilidad del cliente.

## 4. Propuesta

### 4.1. Modelo C4

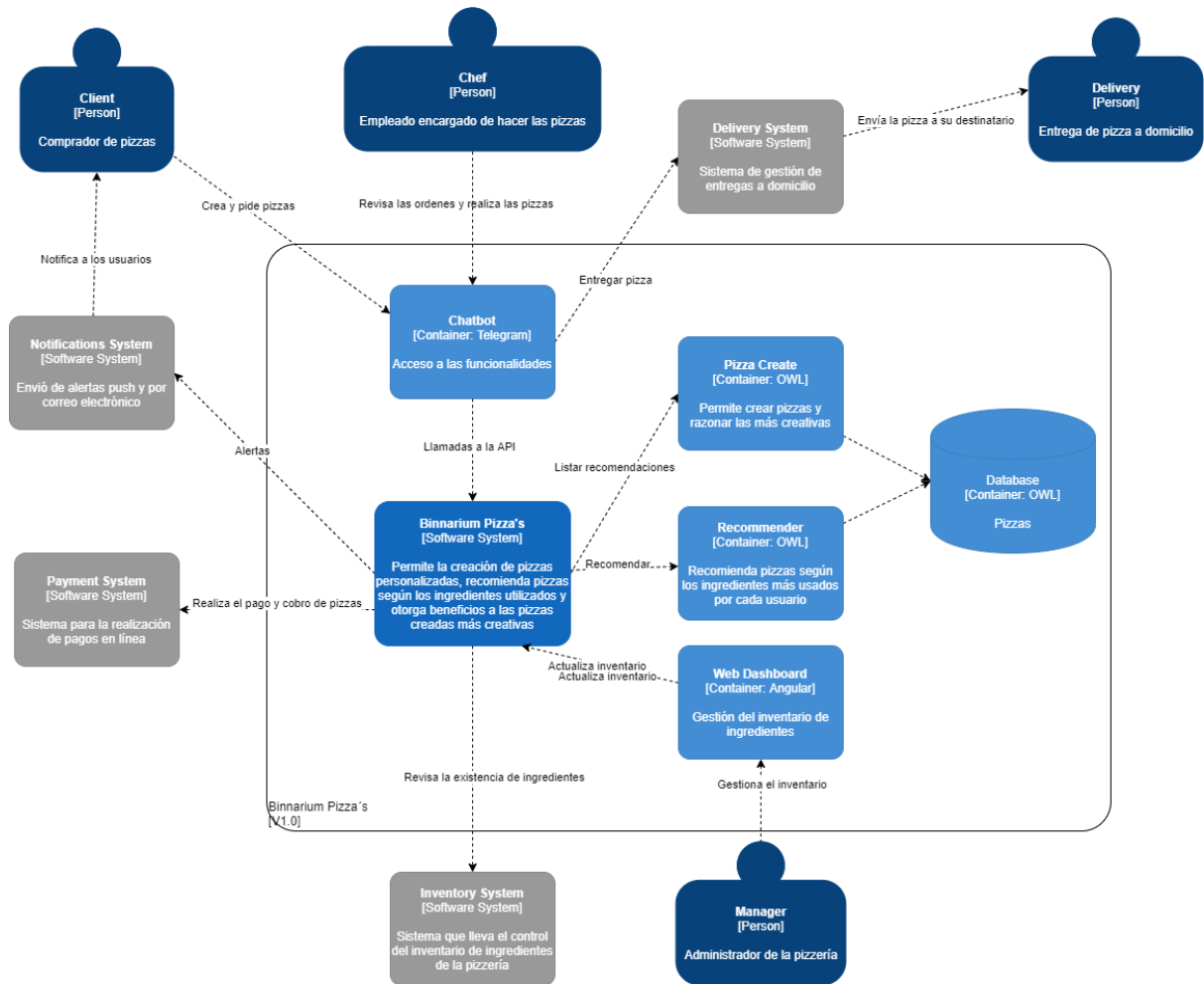
#### 4.1.1. Nivel 1: Contexto



Binnarium Pizza's es un sistema de creación de pizzas personalizadas, esto lo hace conectándose a un sistema de inventario de ingredientes, de pagos, notificaciones y entregas. También permite mostrar las creaciones de otros usuarios y compartirlas, las más creativas de la semana reciben incentivos mediante descuentos u otros beneficios.



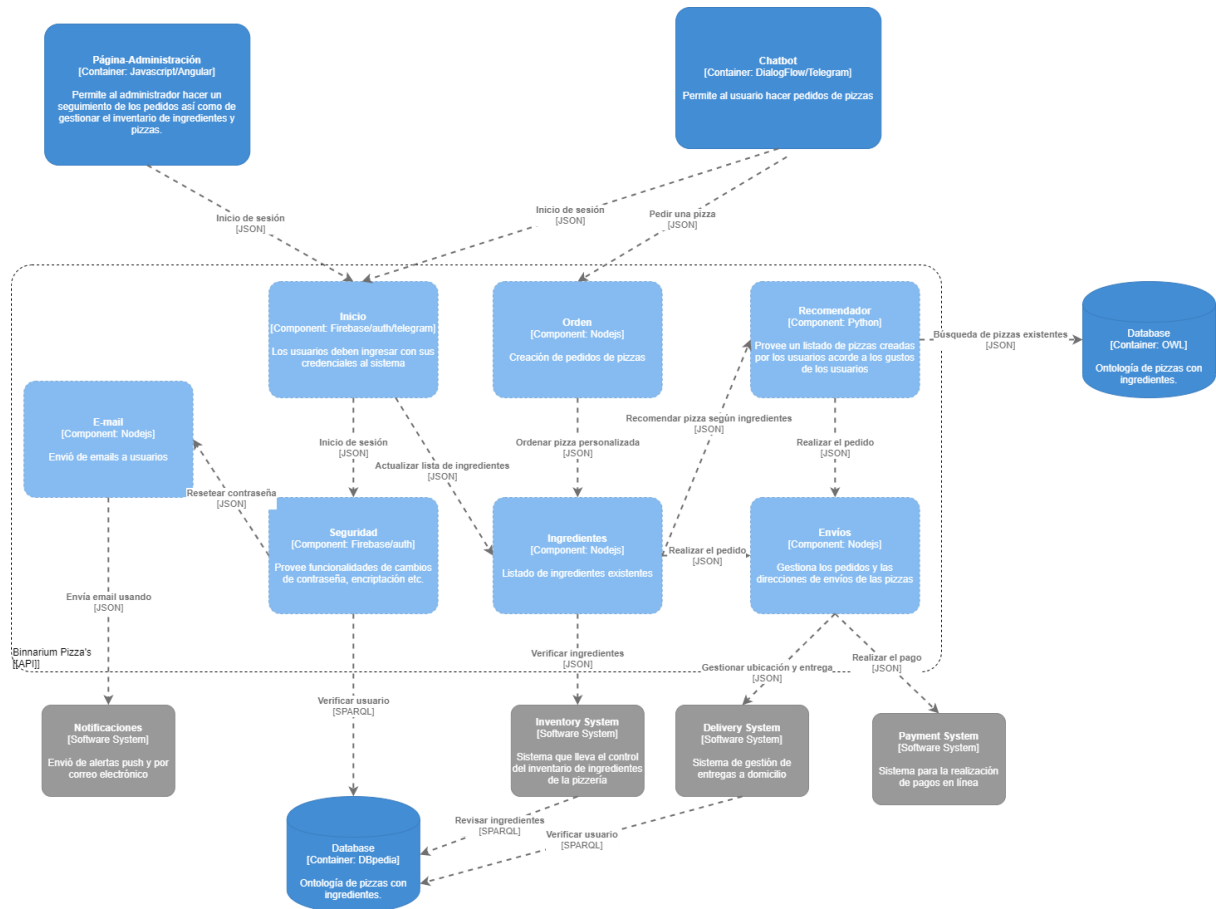
#### 4.1.2. Nivel 2: Contenedores



El sistema se conectará mediante un chatbot al cliente el cual podrá pedir su pizza de las recomendaciones ofrecidas por el sistema, las creaciones más creativas de la semana o crear una pizza por ingredientes personalizada. Toda la información se quedará guardada en una ontología OWL, la cual permitirá realizar inferencias de los gustos de los clientes.



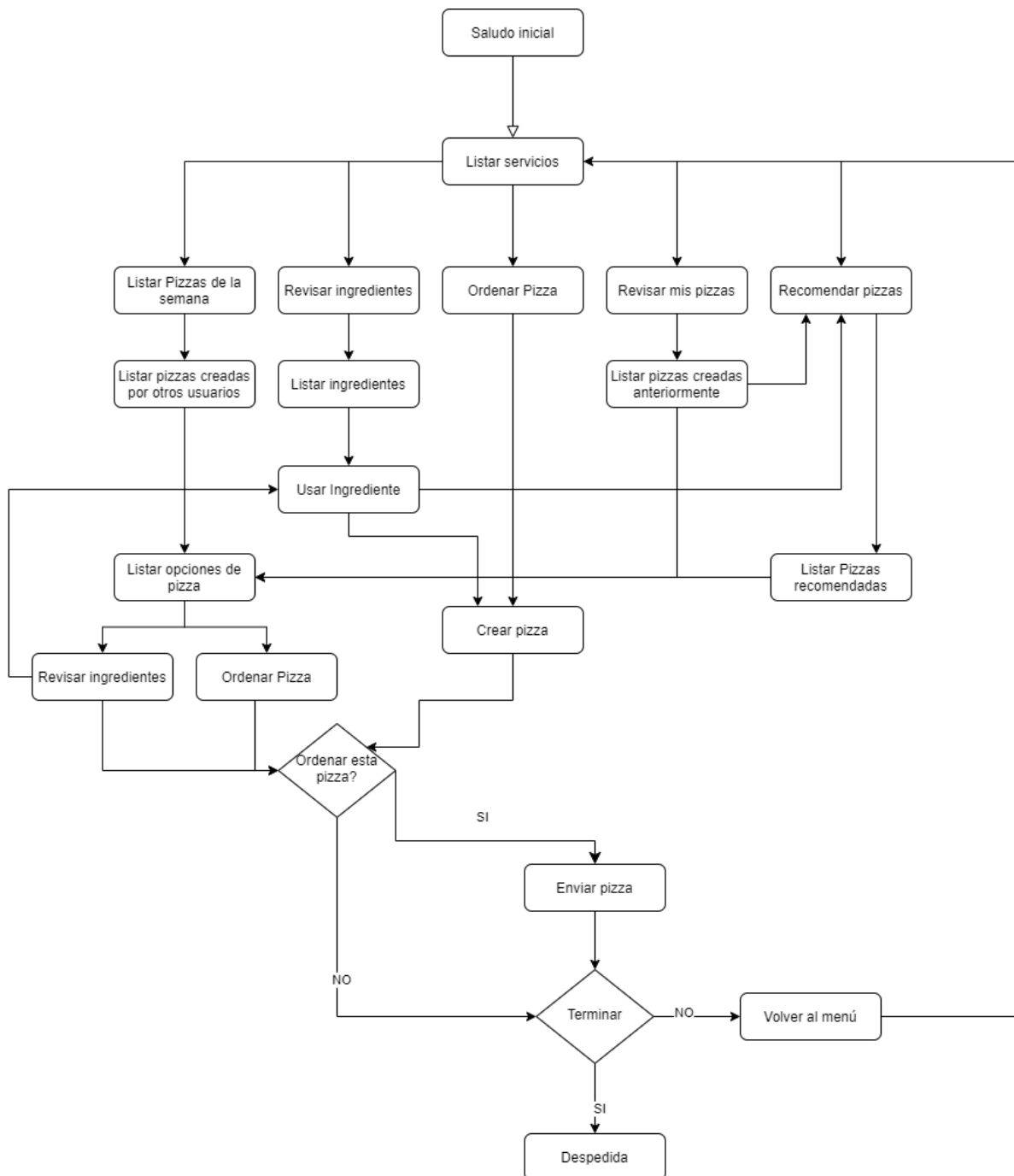
### 4.1.3. Nivel 3: Componentes



Existen dos aplicativos desde los cuales los usuarios podrán acceder la página web para el administrador y el chatbot para los clientes. Mediante la página web el administrador podrá realizar el seguimiento de los envíos así como la gestión del inventario de ingredientes y pizzas; mientras que en el chatbot los clientes harán los pedidos de pizzas que pueden ser personalizados o una recomendación según los ingredientes que más gusten, este sistema de recomendación se conectará tanto a la base de datos de una ontología propia como a una base de datos de DBpedia para expandir la cantidad de pizzas que se pueden ofertar.



## 4.2. Diagrama de flujo del Chatbot



## 5. Implementación

### 5.1. Herramientas Utilizadas

#### 5.1.1. Protégé

Protégé es un software Open Source, que permite la creación y edición de ontologías, esto nos permite crear aplicaciones basadas en ontologías tanto simples como complejas.

### 5.1.2. *Spring*

Spring es un framework para el lenguaje java que nos facilita la creación de servicios rest, mediante un conjunto de bibliotecas de terceros.

### 5.1.3. *Apache Jena Fuseki*

Es un servidor de SPARQL la cual cuenta con bibliotecas disponibles para usar tanto en java como en python para realizar consultas a las ontologías.

### 5.1.4. *Telegram*

Es una plataforma de mensajería instantánea para el envío de varios archivos y comunicación en masa, también cuenta con sistemas de creación de chatbots mediante el chatbot padre.

### 5.1.5. *DBpedia*

Es una plataforma para extraer información estructurada de Wikimedia. La información se asemeja a un grafo de conocimiento abierto (OKG) que está disponible para todos en la web.

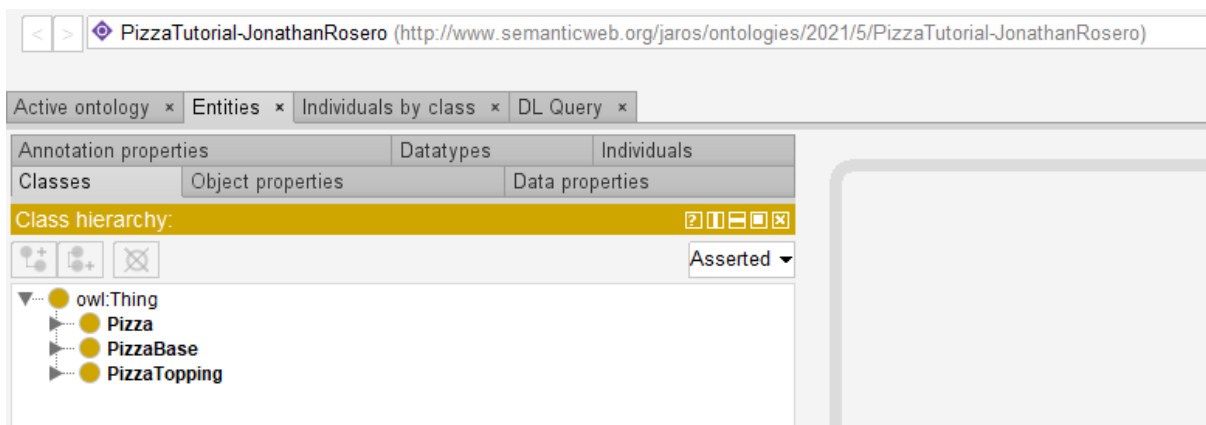
### 5.1.6. *Dialogflow*

Es una plataforma de comprensión de lenguaje natural que se utiliza para diseñar e integrar una interfaz de usuario conversacional en aplicaciones móviles.

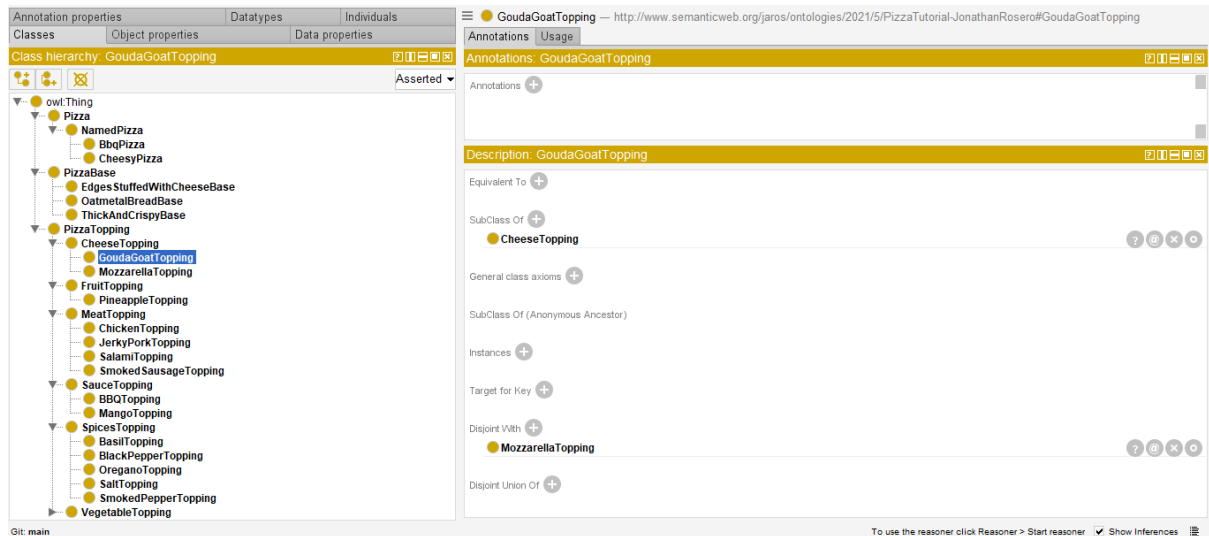
## 5.2. Desarrollo

### 5.2.1. *Ontología*

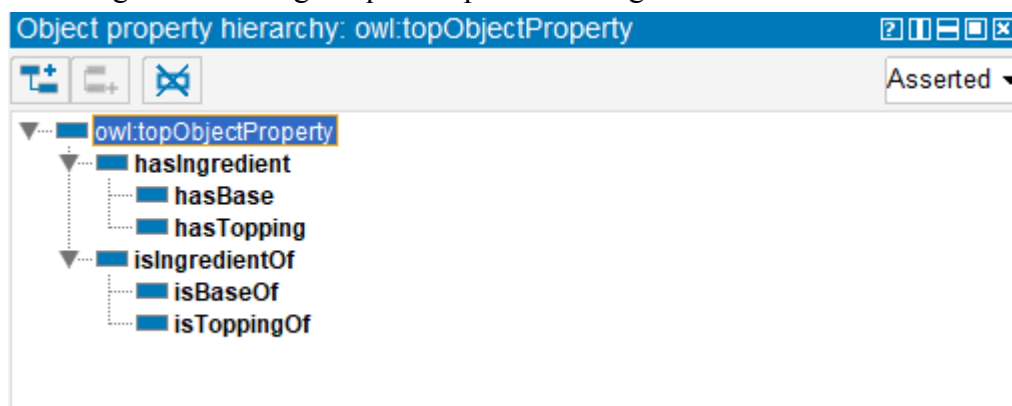
Lo primero que se desarrolló para esta primera fase es una ontología en Protégé que contenga las principales pizzas e ingredientes, con dichos ingredientes los usuarios podrán crear nuevas pizzas



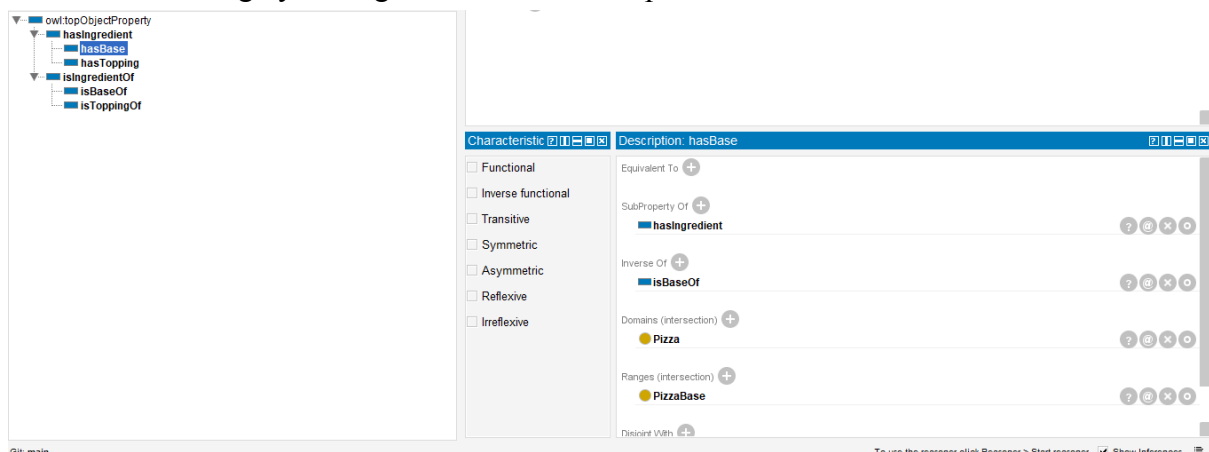
Para esta ontología de pizzas se han creado tres clases: las Pizzas, las Bases y los Aderezos. A su vez cada clase se divide en subclases que permite tener mayor control de la ontología, por ejemplo el Queso Gouda es una subclase de Queso que a su vez es una subclase de Aderezos.



Una vez creados las clases con sus subclases se procedió a determinar propiedades que tienen las clases y que por la misma razón heredan las subclases, para este caso se creó la subclase “tiene ingrediente” al igual que su opuesto “es ingrediente de”.



Una vez determinadas las relaciones se les añaden los sujetos y predicados de cada relación, al haber colocado como inversos “tiene ingrediente” de “es ingrediente de” solo necesitamos colocar los dominios y rangos a un conjunto de propiedades, ya que el inverso tomará al dominio como rango y al rango como dominio respecto a su inverso.



Con esto se tendría lista la ontología para poder ser usada mediante un servicio y consumida por cualquier usuario.

### 5.2.2. *Servicio REST*

Para la creación de un servicio que exponga la información de la ontología de manera que pueda ser usada por los usuarios, se creó un servicio REST mediante el framework SPRING BOOT.

Lo primero que se realizó es la importación de las bibliotecas de Apache Jena, al servicio de conexión con Jena le enviamos la ontología anteriormente creada.

```
public JenaService(){
}

/**
 * Loads OntModel
 * @return
 * @throws IOException
 */
OntModel OpenConnectOWL() throws IOException {
    if(!loaded){
        Resource resource = resourceLoader.getResource("classpath:PizzaTutorial-JonathanRosero.owl");
        OntModel model = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM_RULE_INF);
        InputStream input = resource.getInputStream();
        this.pizzaOntology = (OntModel) model.read(input, "");
        this.loaded = true;
    }
    return this.pizzaOntology;
}
```

Una vez conectados a la ontología levantamos el servicio de Fuseki el cual nos permitirá crear sentencias SPARQL con los que verificaremos accesos a la ontología; con este servicio podremos enviar sentencias SPARQL y que nos devuelva en formato JSON, con el fin de facilitar la lectura al servicio de exposición de datos. Mediante este servicio también podremos conectar al endpoint de DBpedia al cual haremos consultas en formato SPARQL y los expondremos mediante el servicio REST.





```
String getJSONStringResult(String raw) throws IOException {
    String str = "";
    try {
        Query query = QueryFactory.create(raw);
        QueryExecution queryExec = QueryExecutionFactory.create(query, OpenConnectOWL());
        ResultSet rs = queryExec.execSelect();

        if (rs.hasNext()) {
            ByteArrayOutputStream go = new ByteArrayOutputStream();
            ResultSetFormatter.outputAsJSON((OutputStream) go, rs);
            str = new String(go.toByteArray(), charsetName: "UTF-8");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return str;
}
```

A continuación se desarrolló el sistema de envío de sentencias SPARQL hacía el servicio de Jena.

```
@GetMapping(produces = MediaType.APPLICATION_JSON_VALUE)
public String getRecommendation(){
    String query = "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>"+
        "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>"+
        "PREFIX pizza:<http://www.semanticweb.org/jaros/ontologies/2021/5/PizzaTutorial-JonathanRosero#>"+
        "SELECT * WHERE { ?Toppings rdfs:subClassOf pizza:PizzaTopping }";

    return showToppingService.getRecommendation(query);
}
```

Una vez enviado la sentencia SPARQL al servicio de Jena, el resultado es tomado por el servicio encargado de mostrar la ontología. La información es servida mediante un endpoint en formato JSON.



```
@Service
public class ShowToppingServiceImpl implements ShowToppingService {

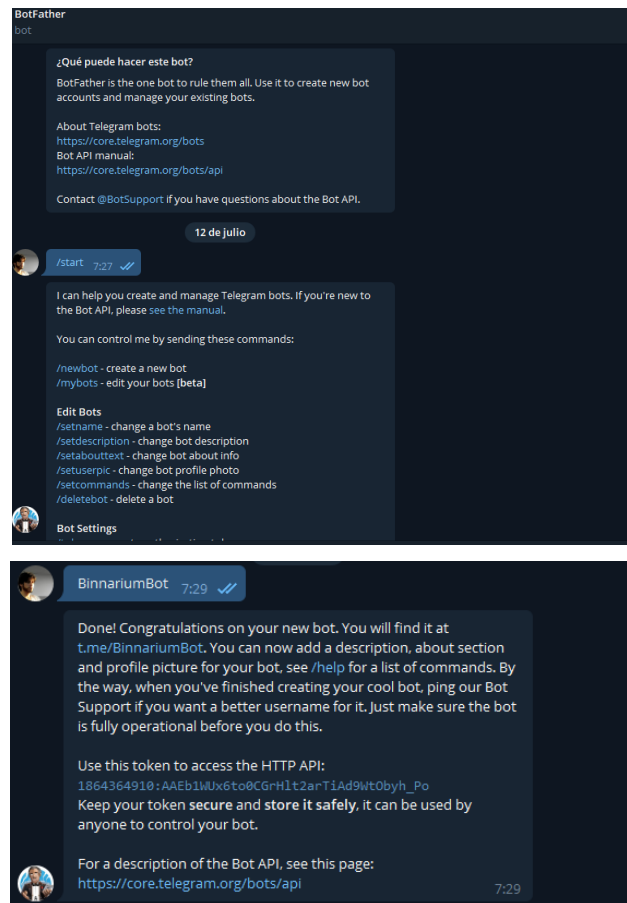
    @Autowired
    JenaService jenaService;

    @Override
    public String getRecommendation(String userPreferences) {
        String result = "";
        try {
            result = jenaService.getJSONStringResult(userPreferences);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return result;
    }

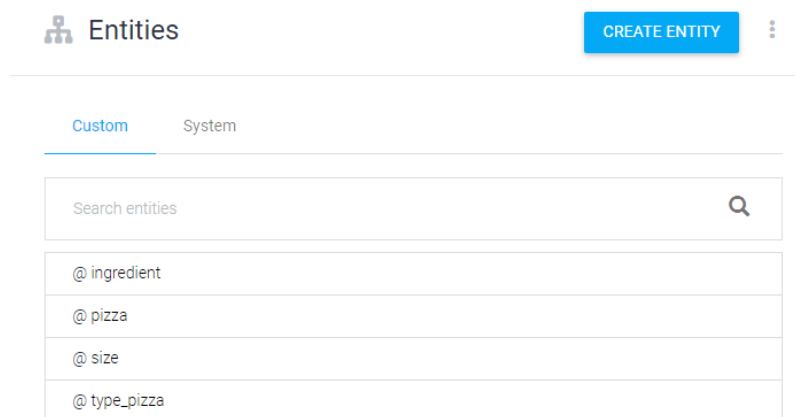
    @Override
    public ResultSet execQuery(String query) throws IOException {
        return jenaService.execSparQL(query);
    }
}
```

### 5.2.3. Chatbot

Para la creación del chatbot se utilizó el servicio de creación de chatbots de Telegram mediante el BotFather, este servicio nos crea un bot mediante comandos en este caso /newbot, ingresamos el nombre e identificador de nuestro bot y nos devolverá el token para poder conectarlo a un servicio externo o desarrollar uno propio.



Una vez creado el bot se procede a conectar en este caso con Dialogflow, el cual nos ayudará con la interpretación de texto y el aprendizaje automatizado. En dialogflow deberemos crear entidades las cuales serán las palabras claves con las que el chatbot trabajará para hacer el pedido de las pizzas.



Después de crear las entidades se procede a escribir intenciones que son frases iniciales o palabras que los usuarios escribirán o darán clic, estas intenciones son las que nos permitirán crear el flujo de conversación.

Cada intención está compuesta por:

- El contexto: son valores de parámetros que son muy usados durante las conversaciones
- Eventos: entradas programadas que detonan el intento
- Frases de entrenamiento: frases o palabras escritas por el usuario que son detectadas por el chatbot y crean una respuesta

Training phrases ⓘ Search training phrases 🔍 ^

” Add user expression

” enviame una taco pizza

” deme una pizza mexicana

” quiero una pizza carbonara

” quiero una pizza frutos del mar de tamaño pequeña

” quiero ordenar una pizza bbq personal

” ordenar pizza tamaño familiar

” ordenar pizza bbq

” ordenar

” quiero ordenar una pizza

- Parámetros: es la instanciación de las entidades, estos parámetros pueden venir desde las frases de entrenamiento o como parte del contexto.
- Respuestas: son las respuesta que dará el chatbot según las entradas que haya tenido, pueden existir varias respuestas de varios tipos distintos, desde una imagen hasta un texto completo e incluso enviar variables como el nombre del usuario datos, etc provistos por Telegram.

Responses ⓘ ^

DEFAULT TELEGRAM +

📘 Responses from this tab will be sent to the Telegram integration.

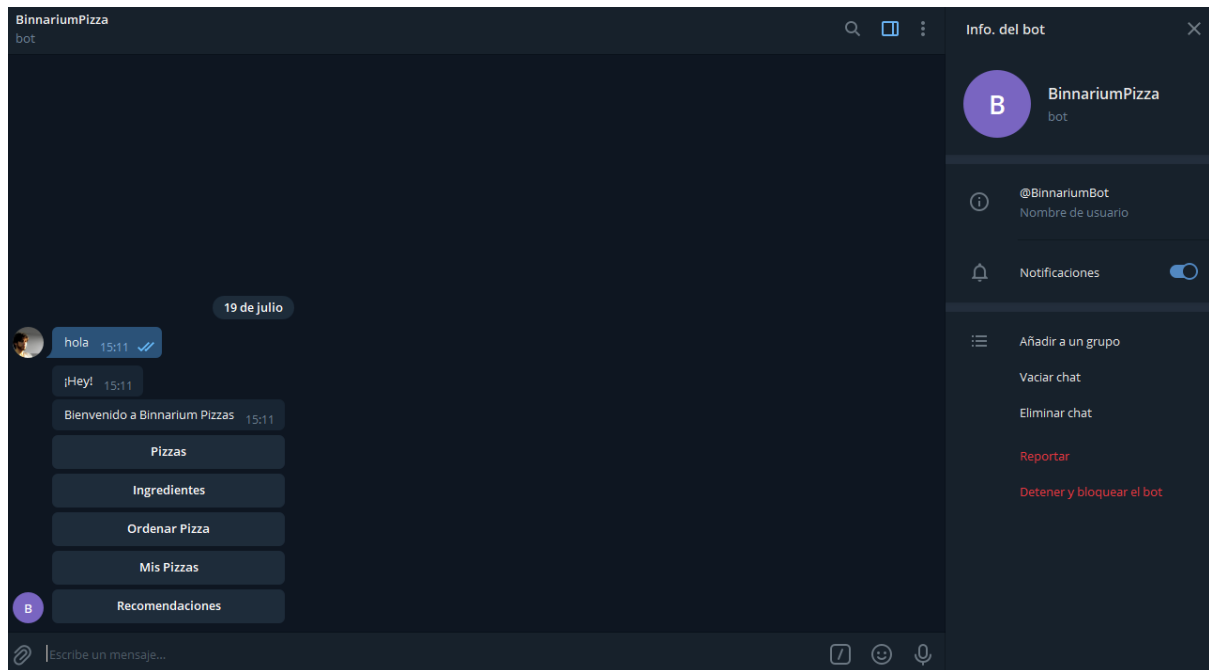
Use responses from the DEFAULT tab as the first responses. ☒

Text Response 🗑

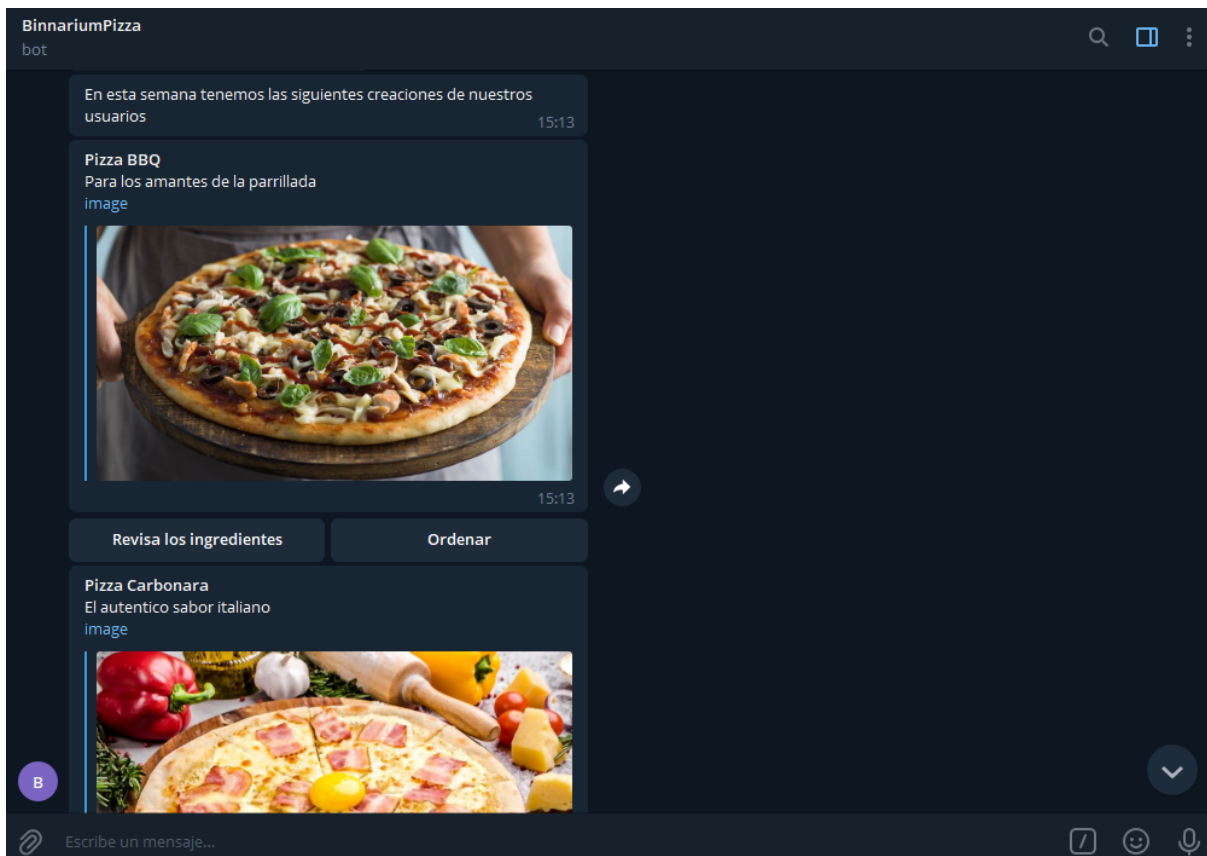
1 Gracias ya le enviaremos su pizza \$type\_pizza de tamaño \$size

2 Enter a text response variant 📄

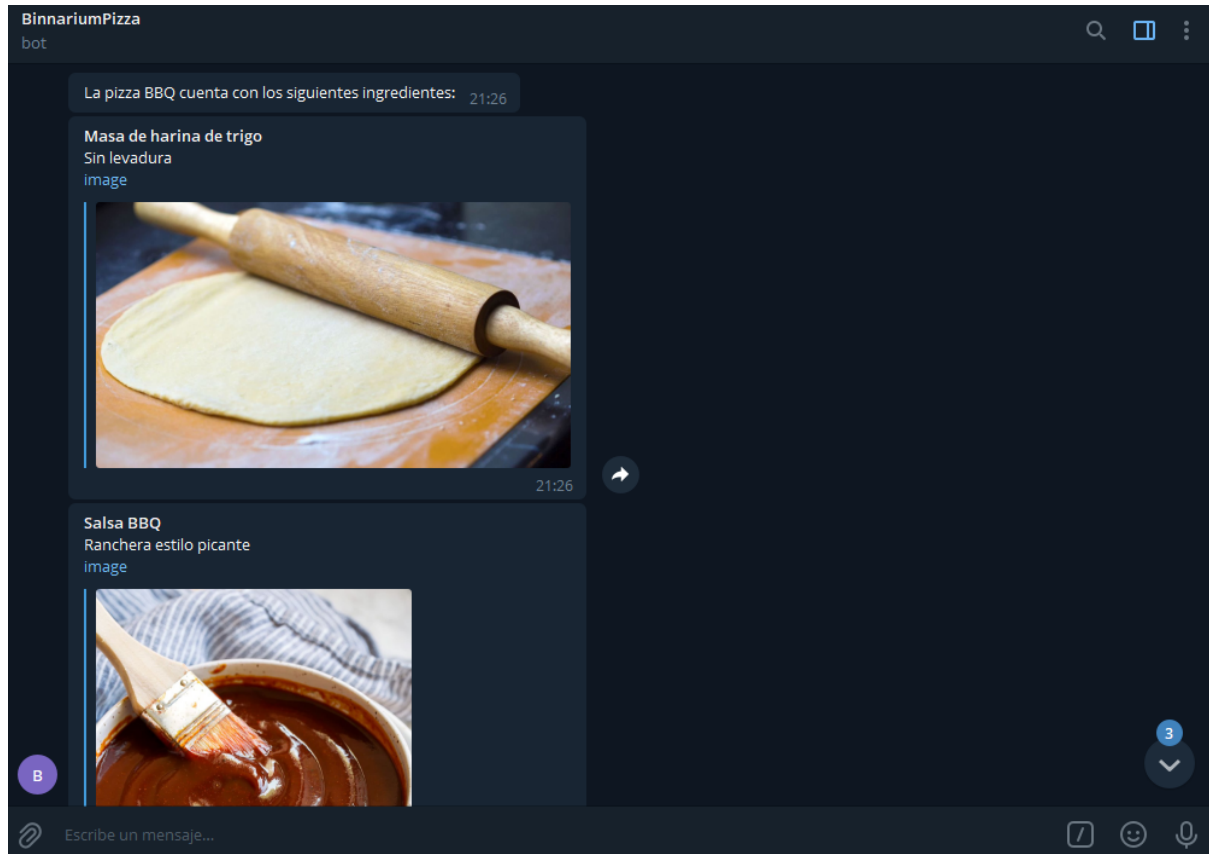
### 5.3. Resultados



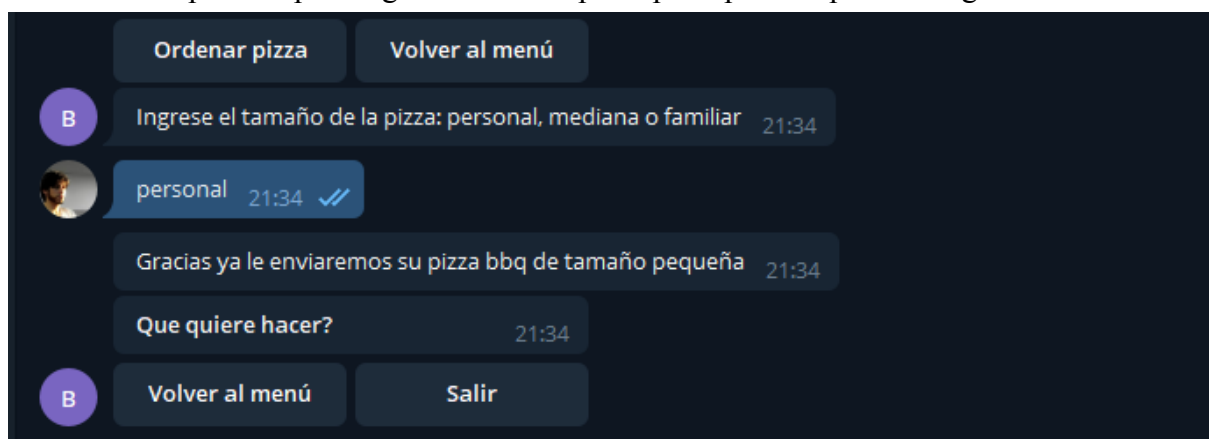
El chatbot siempre empezará con un saludo cordial y mostrando un menú con todos los servicios otorgados, el usuario podrá tipear lo que desea o directamente dar clic en cada opción.



Al dar clic en pizzas el chatbot mostrará las pizzas de la semana creadas por otros usuarios, mostrándonos las opciones disponibles para cada pizza, entre ellas: Revisar los ingredientes de la pizza u ordenarla directamente. Al final tendremos posibilidad de regresar al menú u ordenar una pizza.

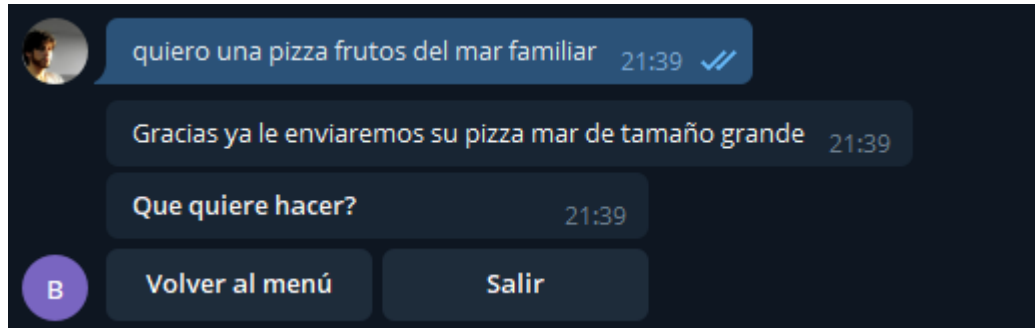


Si revisamos los ingredientes nos mostrará la base y toppings de la pizza escogida, al final se muestran las opciones para regresar al menú principal o pedir la pizza escogida.

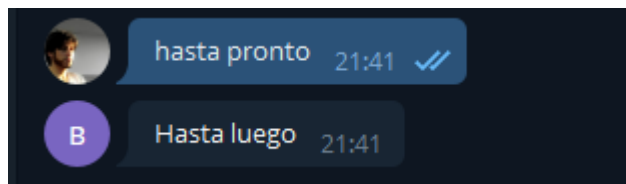


Al hacer el pedido de la pizza siempre necesitará que se ingrese el tipo de pizza y el tamaño en caso de que no se escojan una de las dos opciones el bot pedirá que se ingrese mediante un mensaje. Al enviar la orden se puede regresar al menú o terminar la conversación.

Gracias a Dialog Flow el bot puede entender lenguaje natural de tal manera que se pueda hacer pedidos sin necesidad de usar el menú. También puede realizar las acciones del menú escribiendo en el chat.



Para terminar la conversación el usuario podrá escribir salir o algún sinónimo o en su defecto podrá escoger la opción de salir al realizar un pedido o desde el menú.



## 6. Conclusiones

El uso de ontologías para la integración de datos tiene algunas ventajas sobre los sistemas basados en palabras clave. Las ontologías proveen un vocabulario compartido común (conceptos) para representar la información incluida en los documentos (contenidos). Además las ontologías permiten definir relaciones entre los conceptos (roles). Tanto conceptos como roles pueden ser usados para realizar consultas más complejas y recuperar de forma precisa la información en la que el usuario está interesado.

Una ontología especifica una conceptualización o una forma de ver el mundo, por lo que cada ontología incorpora un punto de vista. Además, una ontología contiene definiciones que proveen del vocabulario para referirse a un dominio. Estas definiciones dependen del lenguaje utilizado para describirlas. Todas las conceptualizaciones (definiciones, categorizaciones, jerarquías, propiedades, herencia, etc.) de una ontología pueden ser procesables por máquina.