



---

# PIZZAGRAPH

---

Sistemas de Información de Gestión y Business Intelligence



**Sergio Rubio Martín**  
**[srubim00@estudiantes.unileon.es](mailto:srubim00@estudiantes.unileon.es)**

**18 DE DICIEMBRE DE 2019**



## **ÍNDICE:**

- 1. Introducción (pg. 2)**
- 2. Objetivos (pg. 2 - 5)**
- 3. Herramientas (pg. 6 - 9)**
- 4. Trabajo (pg. 10 - 35)**
- 5. DAFO (pg. 36 - 37)**
- 6. Líneas de futuro (pg. 38 - 39)**
- 7. Bibliografía (pg. 39)**
- 8. Lecciones Aprendidas (pg. 40)**

# 1. Introducción:

Un sistema de recomendación es aquel que filtra de entre varias opciones disponibles y muestra los resultados más óptimos a los usuarios de forma rápida y accesible.

La aplicación web que he diseñado, la he llamado PizzaGraph y recomienda a los usuarios las mejores pizzas que mejor concuerden con la definición de pizza deliciosa que tendrá el usuario en su cabeza. Toda recomendación tendrá como resultado una pizza de un establecimiento de León.

Las recomendaciones que realiza PizzaGraph son recomendaciones para distintos ámbitos concretos, o lo que es lo mismo, son recomendaciones basándose en la finalidad que busca el cliente. Un claro ejemplo puede ser recomendar pizzas que contengan un ingrediente en concreto, aunque PizzaGraph ofrece más formas de recomendación que se detallarán a lo largo de esta memoria. Cabe destacar que las pizzerías que se muestran son el top 10 de León.

El enlace a Github con toda el código y su documentación es el **(1)** de la bibliografía.

# 2. Objetivos:

El objetivo principal de este trabajo es hacer que PizzaGraph sea una aplicación de recomendación de pizzas, y por consiguiente de pizzerías, que satisfagan la necesidad del usuario en un tiempo corto y de forma eficiente. Todas estas recomendaciones serán a nivel provincial, es decir, solo se recomendarán pizzerías que se encuentren en León de forma que las recomendaciones estarán más centradas evitando posibles errores en la recomendación.

He elegido León como lugar de despliegue no solo porque sea la ciudad en la que vivo sino más bien por la gran gastronomía y sus costumbres que tiene. Un ingrediente principal de León es la cecina y la cecina entre otros, forma parte de los ingredientes que se utilizan a la hora de hacer algunas pizzas. También la costumbre en León es la de tapeo por el centro o por lugares considerados céntricos como puede ser la zona donde se encuentra el edificio Europa. Aunque sorprenda, hay lugares que dan de tapa raciones de pizza caseras. También el estilo de vida del leonés, por regla general, es que los fines de semana salen a cenar con la familia, amigos o con otras personas y no tiene por qué la persona ir a los establecimientos, sino que también puede pedir a domicilio.

De esto se puede obtener 2 ideas principales:

- El mercado que pide pizza, ya de por sí grande, se duplica cuando se añade el factor de que la comida se puede llevar a domicilio.
- La comida al poder ser entregada a domicilio, el usuario o la persona valorará la marca de la pizzería no por su local o empleados, sino que valorará la calidad de la comida que recibe y degusta.

En estos gráficos se puede ver qué pesa más en una pizzería a la hora de ganarse una reputación en el mercado:

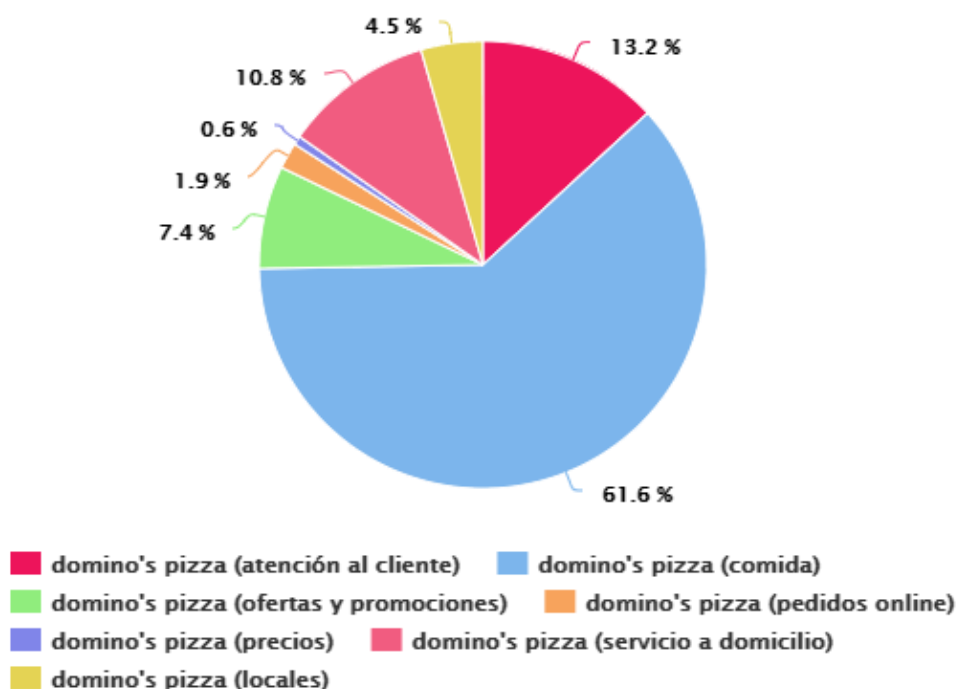


Ilustración 1 - Reputación de Domino's Pizza (2)

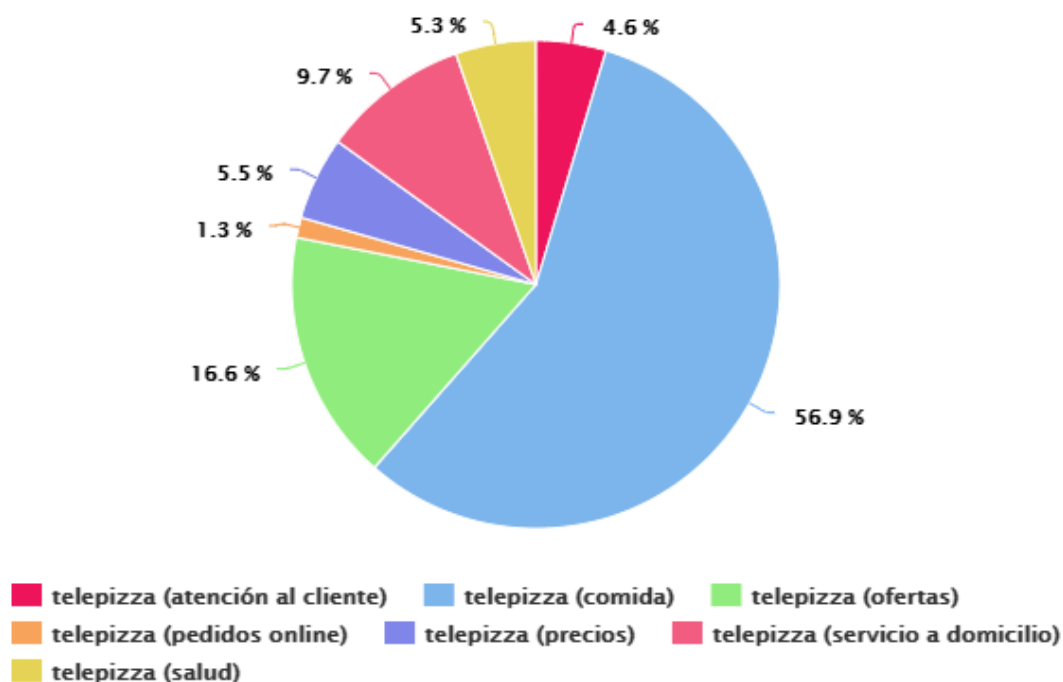


Ilustración 2 - Reputación de Telepizza (2)

Mi aplicación web no solo tiene el objetivo de recomendar a través de una única opción, es decir, permite recomendar pizzas de forma flexible. Veamos todas las funciones que tiene PizzaGraph:

- **Mi Agenda:** esta función permite al usuario ver todas las puntuaciones que ha realizado con anterioridad de forma que tenga en mente si ya ha probado una pizza en un tiempo pasado y además conocer qué nota le ha dado, pudiendo mantener un control o registro de todo lo que ha valorado.
- **Recomendación por Ingredientes:** esta función permite al usuario realizar una búsqueda exhaustiva en un tiempo eficiente de las pizzas que contienen un determinado ingrediente que el usuario ha seleccionado con anterioridad. Estos ingredientes se muestran al usuario en orden alfabético, de forma que solo con pulsar la primera letra del ingrediente le redireccionará al grupo de ingredientes que empiecen por dicha letra siendo así una aplicación web accesible para el usuario, es decir, haciéndole la vida más fácil.

- **Puntuar Pizza:** permite al usuario realizar una puntuación o valoración basada en su experiencia de la pizza que haya degustado anteriormente. El número con los que podrá puntuar el usuario será con números naturales y definidos entre el 0 y el 10, ambos incluidos.
- **Pedir Aleatoria:** esta función recomienda una pizza aleatoria, dentro del catálogo, al usuario, de forma que pueda experimentar con nuevas pizzas y descubrir un nuevo mundo de sabores o volver a degustar de las mismas pizzas que para los usuarios son manjares porque ya las ha probado anteriormente. Normalmente al final esta opción es utilizada por gente aventurera que busca experimentar. Cabe destacar que si por un casual la pizza aleatoria que sale, el usuario ya la ha probado anteriormente y no le ha gustado o por el contrario contiene un ingrediente el cual produce reacciones alérgicas a la persona, el usuario podrá pedir los datos de una recomendación de una pizza aleatoria de nuevo infinitas veces.
- **Recomendación por Valoraciones:** esta es la función más importante de la aplicación web ya que permite al usuario obtener una recomendación teniendo en cuenta, no solo sus propias valoraciones, sino también las valoraciones de todos los usuarios que estén registrados en la aplicación de PizzaGraph. Como resultado final mostrará a los usuarios las pizzas que hayan sido puntuadas anteriormente con la puntuación media obtenida. Para aumentar aún más la accesibilidad de la aplicación web he añadido dos opciones dentro de esta función las cuales son ordenar los resultados obtenidos de mayor a menor o de menor a mayor.

## 3. Herramientas:

### I. Github:

Es un control de versiones, o lo que es lo mismo, es un sistema de gestión de versiones de código, donde subo todo mi código que compone la aplicación web de forma que se puede utilizar desde cualquier equipo. También funciona como una red social de desarrolladores que permite trabajar en colaboración con otros desarrolladores y poder realizar seguimientos del trabajo.

Se podría considerar el escaparate de todo programador donde cada uno sube todo lo conseguido hasta el momento. **(3)**

Mi Github como mencioné al principio de la memoria está como primera referencia en la bibliografía. En mi caso particular para este trabajo he subido esta memoria, la presentación powerpoint y PDF de mi aplicación y también he subido el código de la aplicación dividiendo el Front-End y el Back-End en dos carpetas separadas.

Para configurar Github en mi equipo he seguido los pasos que muestra la referencia **(4)** de la bibliografía.



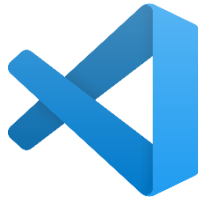
*Ilustración 3 - Github*

### II. Visual Studio Code:

Es un editor de código que pone a disposición de los desarrolladores funciones como autocompletar el cierre de un paréntesis, bucles y además permite instalar una cantidad de plugins que son beneficiosos para los desarrolladores facilitando labores complicadas como puede ser realizar algunas configuraciones delicadas. Este programa también ofrece un servicio de depuración que permite al

programador realizar búsquedas de errores del código evitando que éste tenga que malgastar tiempo innecesario. **(5)**

En el transcurso de mi trabajo no he tenido gran problema con este editor, es más, me ha parecido francamente accesible, es decir, todas las funciones que he necesitado las he encontrado fácilmente sin perder mucho tiempo.



*Ilustración 4 - Visual Studio Code*

### III. Neo4j:

Es un software de libre uso destinado a crear y gestionar bases de datos orientadas a grafos. Al ser orientada en grafos permite al desarrollador crear una base de datos flexible y eficiente a la hora de realizar consultas, modificaciones o creaciones. En Neo4j la información está almacenada en nodos y los nodos pueden estar relacionados con otros nodos. Cada nodo tiene unas propiedades o atributos que muestran información del nodo. **(6)**

Para realizar búsquedas, Neo4j utiliza un algoritmo especial de grafos como son la búsqueda en profundidad y la búsqueda en anchura.

Estas son algunas ventajas que tiene Neo4j respecto a otras bases de datos en las cuales se utiliza el modelo de tablas **(7)**:

- La velocidad de búsqueda depende solo del número de relaciones concretas, no del conjunto de datos, mejorando así el rendimiento y la eficiencia en la búsqueda descartando de primera mano datos innecesarios.
- Muestra los resultados en tiempo real.
- Presentación intuitiva y resumida de las relaciones.
- Estructura flexible y ágil.





Ilustración 5 - Neo4j

## IV. Node.js:

Node.js es un entorno de JavaScript que permite montar la parte del Back-End, o lo que es lo mismo, el servidor que da respuestas a las peticiones que realiza el usuario en el Front-End. Node soporta protocolos como TCP, DNS y HTTP haciendo así que sirvan para el despliegue de aplicaciones web.

El punto fuerte de esta tecnología es que tiene mucha capacidad de mantener muchas conexiones abiertas y esperando a las peticiones. **(8)**



Ilustración 6 – Nodejs

## V. Vuetify:

Vuetify es un framework que combina la potencia del popular VueJs con la estética de Material Design. Este framework pone a disposición de los desarrolladores varios componentes personalizables para usar directamente en la creación del Front-End de la aplicación web. No solo permite cambiar el aspecto de los componentes sino también su comportamiento. Como resultado de utilizar Vuetify es que se obtiene un Front-End vistoso, agradable a la vista y además funcional de una manera sencilla y en tiempo aceptable. **(9)**



Ilustración 7 - Vuetify



## VI. Gimp:

Gimp es un software libre de edición de imágenes y dibujos. He tenido que utilizar esta tecnología para la creación y personalización del logo de PizzaGraph que se puede ver siempre en la parte superior izquierda de cada página.

Sobre todo, Gimp me ha sido muy útil a la hora de quitar el fondo de la imagen por que todo archivo en Gimp empieza con un fondo blanco. **(10)**



*Ilustración 8 – Gimp*

## 4. Trabajo:

Mi aplicación web consta de un Front-End que es lo que permite al usuario interactuar con la aplicación y el Back-End que es el que responde las peticiones que el usuario hace a través del Front-End. Este sería el esquema:

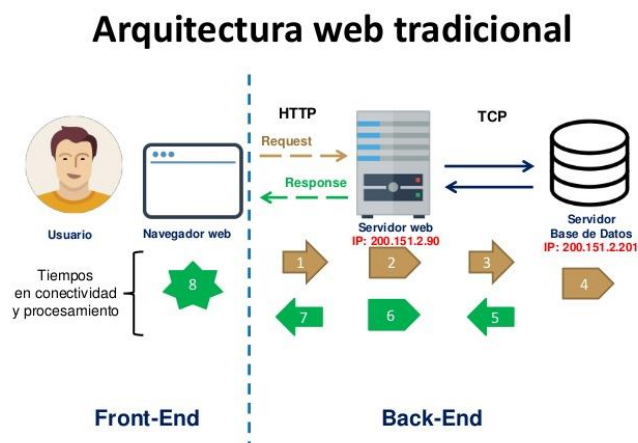


Ilustración 9 - Arquitectura Web

Para mantener todas las diferentes ideas separadas, explicaré una acción que puede realizar el usuario visualmente, luego como internamente el código realiza la acción y posteriormente explicaré como el Back-End responde al Front-End para realizar la acción. La aplicación tiene un carrusell que va cambiando las imágenes de la página principal, en la cual se pueden anunciar las pizzerías:

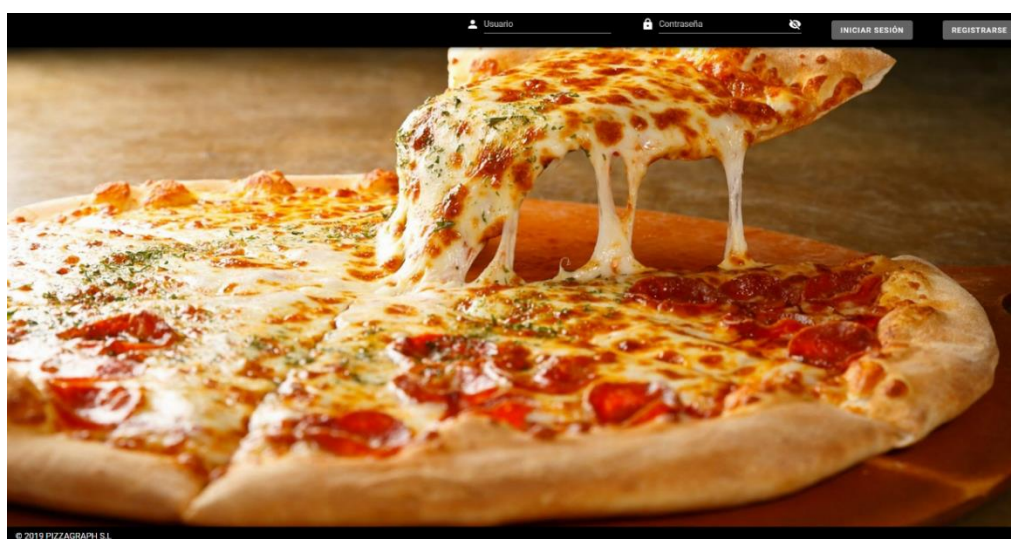


Ilustración 10 - Fondo 2



Ilustración 11 - Fondo 3

### ❖ Loguearse:

En primer lugar, el usuario podrá visualizar la ventana de bienvenida donde permite al usuario registrarse o loguearse.



Ilustración 12 - Fondo 1

En caso de querer loguearse porque ya tiene una cuenta simplemente tendrá que introducir su usuario y contraseña en los dos campos que se encuentra en la parte superior. También tiene la opción de ocultar contraseña o mostrar para confirmar que la ha introducido bien.

PizzaGraph cuenta con un sistema que advierte a los usuarios cuando su contraseña o usuario han sido introducidos de forma errónea.



*Ilustración 13 - Error en Login*

El código que realiza la solicitud de validación es el siguiente:



```
login() {
  this.array=[];
  this.recomendandoVo=false;
  this.recomendandoResto=false;
  this.recomendandoIngre=false;
  this.puntuando=false;
  this.aleatorio= false;
  this.recomendandoGente=false;
  this.ordenadoArray=false;
  if (this.user !== null && this.password !== null) {
    var data={user:this.user,password:this.password};

    this.$http.post('http://localhost:3000/login', data).then(response => {

      if(response.body.message==false){
        this.snackbarUsuario=true;
      }
      else{
        this.bool_login = true;
        this.usuario=response.body.usuario;
        var color= document.getElementById("inspire");
        color.style.backgroundColor="#FFF3E0";
      }

    }, response => {
      alert(JSON.stringify(response.body));
    });
  }
},
```

Ilustración 14 - Front-End Login

Como se puede observar se establecen todas las variables que permiten ver las subventanas a false para que no se vean y comprobando que están todos los datos bien, realiza una petición al Back-End. El cual responde de la siguiente manera:

```
app.post("/login", function (req, res) {
  var user = req.body.user;
  var password = req.body.password;

  var query = "MATCH (n:Persona) WHERE n.usuario='" + user + "' AND n.password='" + password + "' return n";

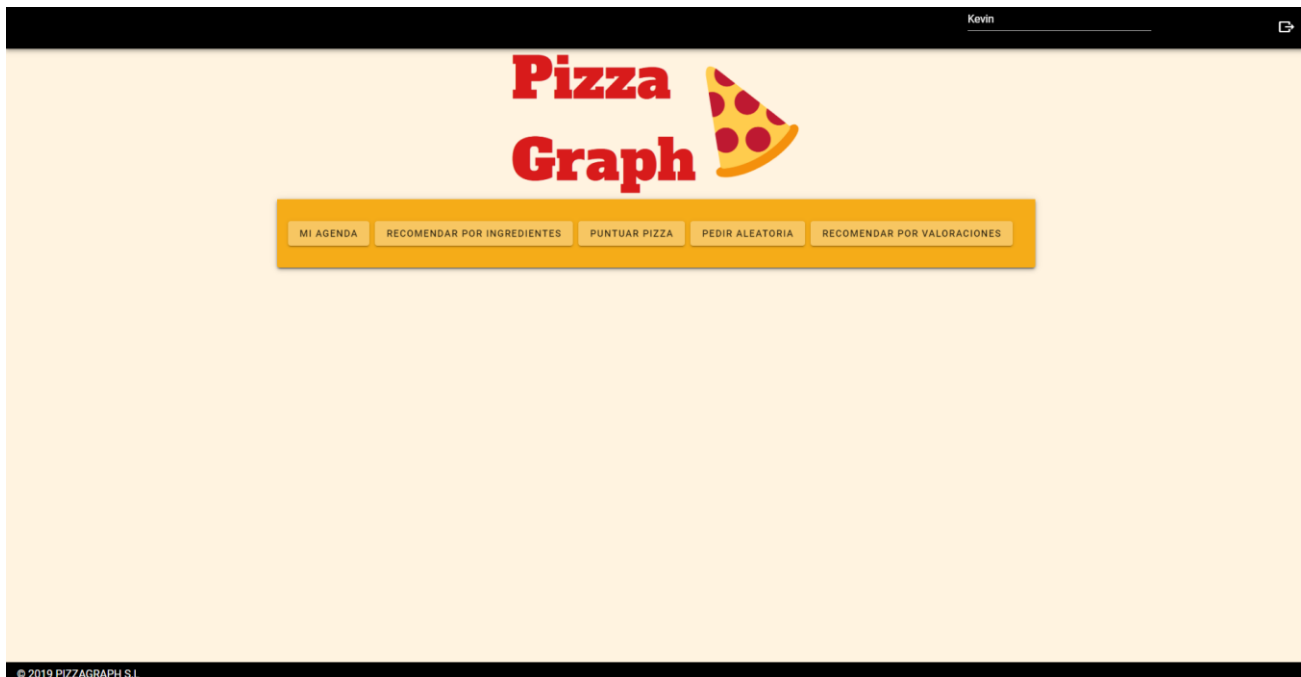
  const resultPromise = session.run(query);
  resultPromise.then(result => {
    session.close();

    if (result.records.length == 0) {
      res.send({ message: false })
    }
    else {
      var record = result.records[0].get(0).properties.usuario;
      res.send(record);
    }
  })
})
```

Ilustración 15 - Back-End Login

La sentencia busca en la base de datos un nodo de tipo persona cuyo usuario y contraseña coincidan con las introducidas. Si no encuentra ningún resultado devuelve un mensaje al Front-End diciendo que no lo ha encontrado. En caso de que lo haya encontrado le responde con el usuario del nodo encontrado.

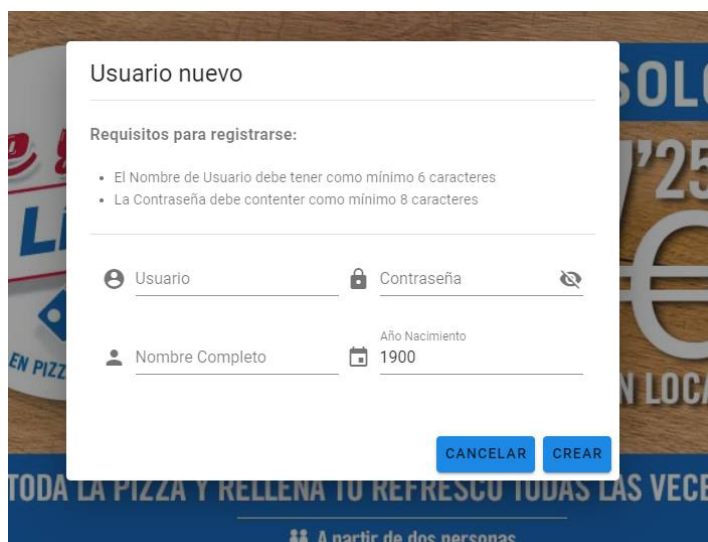
Una vez el usuario se haya logueado le aparece la siguiente ventana al usuario:



*Ilustración 16 - Menú principal*

## ❖ Registrarse:

Si por el contrario el usuario no dispone de una cuenta siempre podrá registrarse dándole al botón de registrarse en la pantalla de inicio. Los datos que se pedirán serán el nombre, un usuario, una contraseña y una fecha de nacimiento. El usuario haciendo clic en crear tendría su nueva cuenta disponible y se loguearía de forma automática.



Usuario nuevo

Requisitos para registrarse:

- El Nombre de Usuario debe tener como mínimo 6 caracteres
- La Contraseña debe contener como mínimo 8 caracteres

Usuario
  Contraseña

Nombre Completo
  Año Nacimiento 1900

A partir de dos personas

Ilustración 17 – Registrarse

El código que realiza la solicitud de validación es el siguiente:

```
registrar() {
    var form = new Object();
    form.nombre=this.nombrecompleto_registrar;
    form.usuario = this.nombre_registrar;
    form.password = this.pass_registrar;
    form.nacimiento=this.nacimiento_registrar;

    this.$http.post('http://localhost:3000/usuarios', form).then(() => {
        this.user = this.nombre_registrar;
        this.password = this.pass_registrar;
        this.bool_login = true;
        var color= document.getElementById("inspire");
        color.style.backgroundColor="#FFF3E0";
        this.cerrar_registrar();
    }, response => {
        alert(response.body);
    });
},
```

Ilustración 18 - Registrarse Front-End



En primer lugar, crea un objeto Json que rellena sus propiedades con los datos introducidos por el usuario, luego envía el objeto al Back-End para crear su respectivo nodo en la base de datos. El Back-End responde de la siguiente manera:

```
app.post("/usuarios", function (req, res) {
  var user = req.body.usuario;
  var nombre = req.body.nombre;
  var password = req.body.password;
  var nacimiento = req.body.nacimiento;
  var query = "CREATE (n:Persona{nombre:'" + nombre + "', nacimiento:" + nacimiento + ", usuario:'" + user + "', password:'" + password + "'})";

  const resultPromise = session.run(query);
  resultPromise.then(result => {
    session.close();
    res.send({ message: true })
  })
})
```

*Ilustración 19 - Registrarse Back-End*

La query permite crear directamente el nodo con los datos de la persona, pero si introduce un usuario ya existente este le denegará la creación y tendrá que utilizar otro usuario no existente.

## ❖ Mi Agenda:

El usuario podrá ver todas las valoraciones que ha realizado él anteriormente con ese usuario de forma que puede llevar un registro rápido y sencillo evitando el tener que utilizar la memoria o llevar en un bloc de notas toda la información.

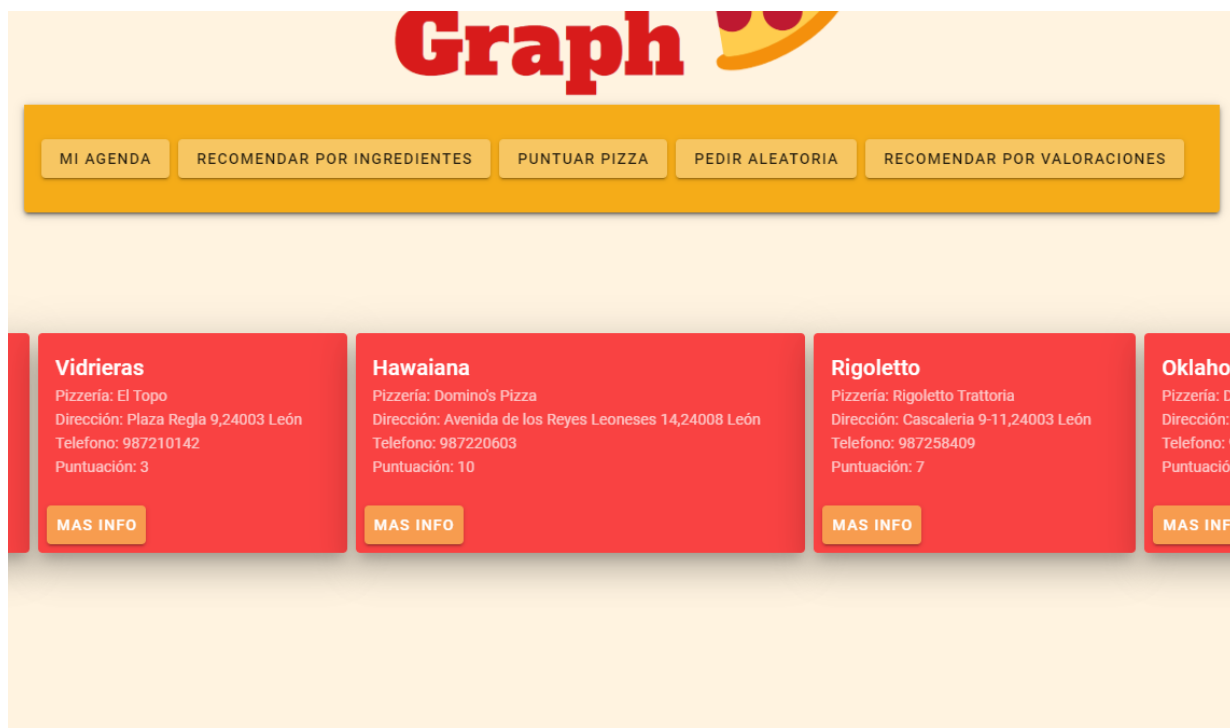


Ilustración 20 - Mi Agenda

El código que realiza la solicitud de visualizar la agenda es el siguiente:

```
recomendarMio(){
  this.recomendandoGente=false;
  this.array=[];
  var data={user:this.user};
  this.recomendandoYo=true;
  this.recomendandoResto=false;
  this.recomendandoIngre=false;
  this.puntuando=false;
  this.aleatorio= false;
  this.ordenadoArray=false;

  this.$http.post('http://localhost:3000/mio', data).then(response => {
    this.array=response.body;
  }, response => {
    alert(JSON.stringify(response.body));
  });
},
```

Ilustración 21 - Mi Agenda Front-End

Realiza una petición directa al Back-End de forma que solo le pasa el usuario como argumento para hacer la búsqueda en la base de datos. El Back-End responde de la siguiente manera:

```
app.post("/mio", function (req, res) {
  var user = req.body.user;
  var session = driver.session();
  var query = "MATCH (n:Persona{usuario:'" + user + "'})-[v:VALORA]->(p:Pizza)<-[:OFRECE]-(e:Pizzeria) return p,v.valoración,e";

  var array = [];
  var objeto;

  // Run a Cypher statement, reading the result in a streaming manner as records arrive:
  session
    .run(query)
    .subscribe({
      onNext: function (record) {

        objeto = (record.get(0).properties);
        objeto.puntuacion = record.get(1).low;
        objeto.pizzeria = record.get(2).properties.nombre;
        objeto.direccion = record.get(2).properties.direccion;
        objeto.telefono = record.get(2).properties.telefono;

        array.push(objeto);

      },
      onCompleted: function () {
        session.close();
        res.send(array);
      },
      onError: function (error) {
        console.log(error);
      }
    });
});
```

*Ilustración 22 - Mi Agenda Back-End*

La query utilizada permite realizar, en la base de datos, una búsqueda de un nodo persona que tenga como usuario el usuario que ha solicitado la búsqueda de forma que obtienen todas las relaciones de tipo VALORA haciendo que finalmente se obtengan las pizzas valoradas por ese usuario. El Back-End devuelve un array o conjunto de elementos que contienen los datos de la pizza.

## ❖ Recomendar por ingredientes:

El usuario podrá realizar una búsqueda filtrando por un ingrediente o bien porque solo come pizzas que contengan el ingrediente que más le gusta o bien porque ha visto una pizza que le parece deliciosa y comprueba que no tenga un ingrediente al cual es alérgico.

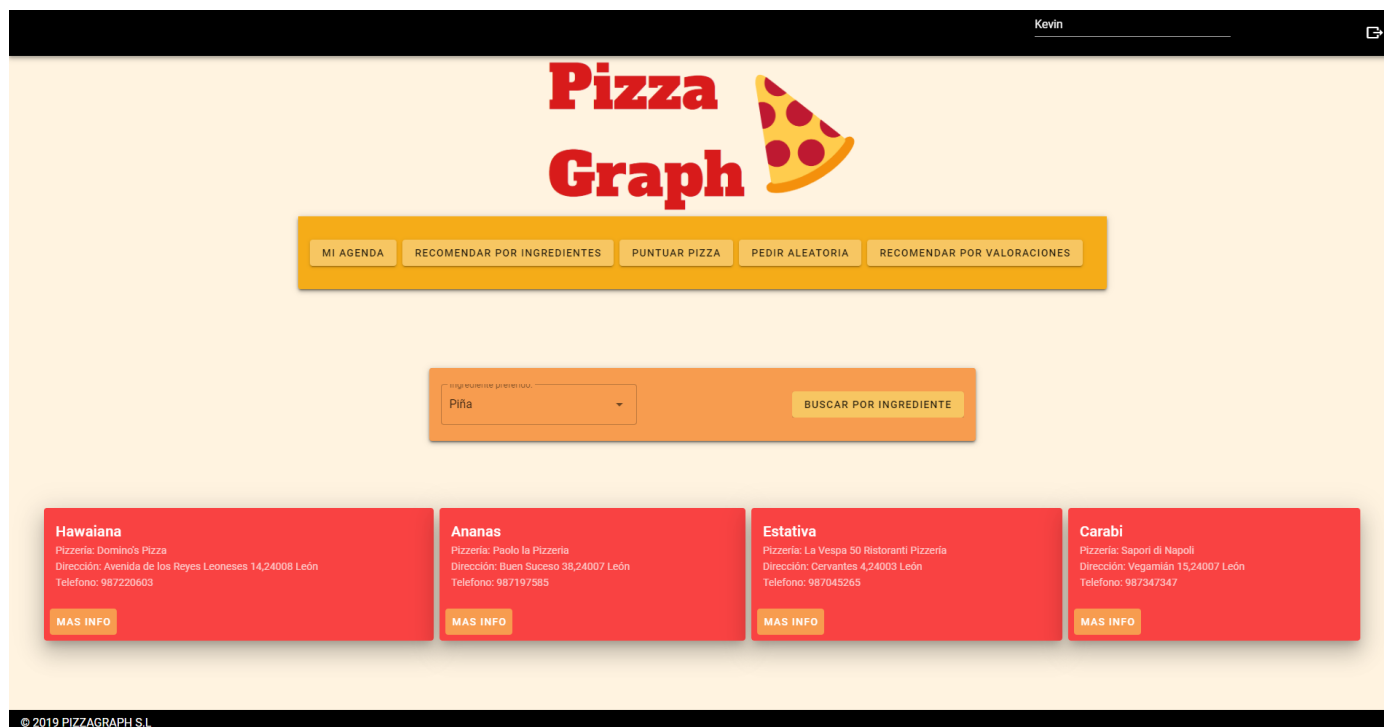


Ilustración 23 - Recomendar por Ingredientes

El código que realiza la solicitud de buscar por ingredientes es el siguiente:

```
recomendarIngre(){
  this.array=[];
  this.recomendandoYo=false;
  this.recomendandoIngre=true;
  this.puntuando=false;
  this.aleatorio= false;
  this.recomendandoGente=false;
  this.ordenadoArray=false;

  var data={user:this.user, ingrediente: this.e2, alergia: this.alergia};

  this.recomendandoResto=false;

  this.$http.post('http://localhost:3000/ingrediente', data).then(response => {
    this.array=response.body;
  }, response => {
    alert(JSON.stringify(response.body));
  });
},
```

*Ilustración 24 - Recomendar por Ingredientes Front-End*

Realiza una petición directa al Back-End donde pasa como argumentos el usuario y el ingrediente a buscar. El Back-End responde de la siguiente manera:



```
app.post("/ingrediente", function (req, res) {
  var user = req.body.user;
  var ingrediente = req.body.ingrediente;
  var session = driver.session();
  var query = "MATCH (n:Pizzeria)-[:OFRECE]->(p:Pizza)-[:CONTIENE]->(i:Ingrediente) WHERE i.nombre='" + ingrediente + "' return p,n";

  var array = [];
  var objeto;

  // Run a Cypher statement, reading the result in a streaming manner as records arrive:
  session
    .run(query)
    .subscribe({
      onNext: function (record) {
        objeto = (record.get(0).properties);
        objeto.pizzeria = record.get(1).properties.nombre;
        objeto.direccion = record.get(1).properties.direccion;
        objeto.telefono = record.get(1).properties.telefono;

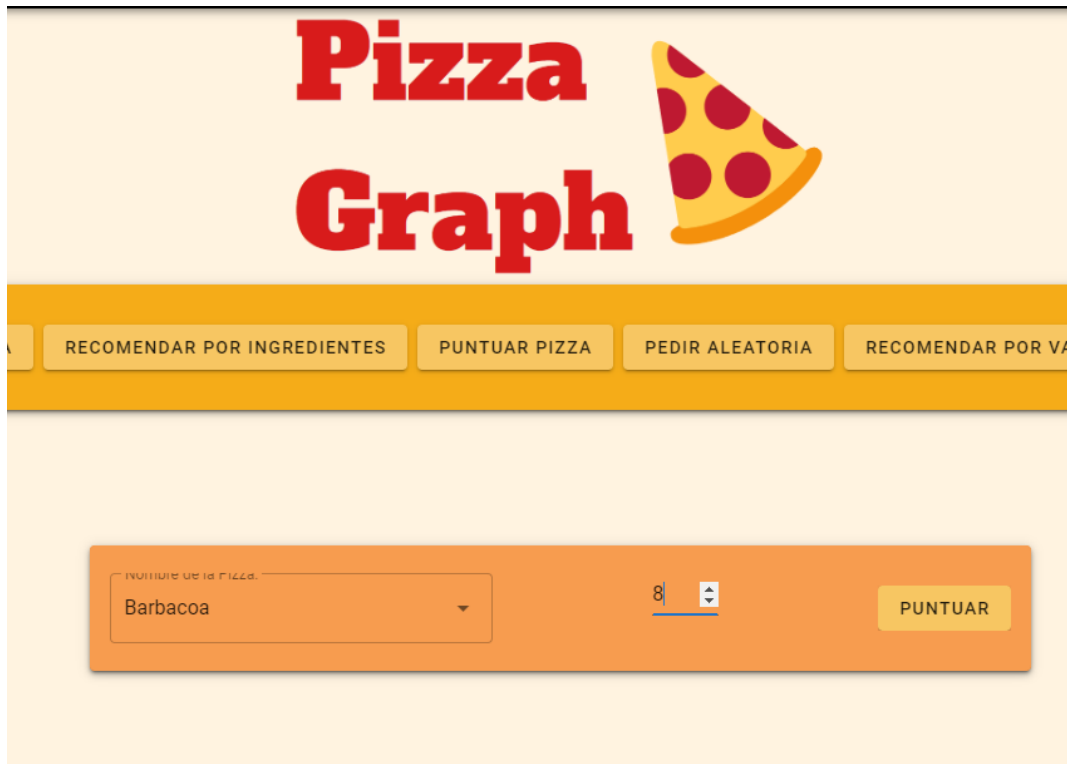
        array.push(objeto);
      },
      onCompleted: function () {
        session.close();
        res.send(array);
      },
      onError: function (error) {
        console.log(error);
      }
    });
});
```

*Ilustración 25 - Recomendar por ingredientes Back-End*

La query busca las pizzas que contienen el ingrediente pasado como argumento y devuelve la pizza y la pizzería donde se vende para añadir parte de esa información en el resultado final.

## ❖ Puntuar Pizza:

El usuario podrá puntuar una pizza que haya degustado anteriormente de forma que la valoración quedará almacenada en su perfil.



The screenshot shows the 'Puntuar Pizza' (Rate Pizza) interface. At the top, the 'Pizza Graph' logo is displayed. Below it is a navigation bar with four buttons: 'RECOMENDAR POR INGREDIENTES', 'PUNTUAR PIZZA' (which is highlighted), 'PEDIR ALEATORIA', and 'RECOMENDAR POR VA'. The main area contains a form with a dropdown menu labeled 'Nombre de la pizza:' with 'Barbacoa' selected. To the right of the dropdown is a text input field containing the number '8'. To the right of the input field is a 'PUNTUAR' button.

Ilustración 26 - Puntuar Pizza

Si el usuario ha metido una puntuación con decimales o dejado algún campo en blanco, se le mostrará en la ventana una advertencia de que debe rellenarlos de forma correcta.

The screenshot shows the Pizza Graph application interface. At the top, a red error message box states: "Puntuación= (min 0 - max 10) y debe escoger una pizza" with a "CLOSE" button. Below this, the "Pizza Graph" logo is displayed. A navigation bar contains four buttons: "RECOMENDAR POR INGREDIENTES", "PUNTUAR PIZZA", "PEDIR ALEATORIA", and "RECOMENDAR POR". The "PUNTUAR PIZZA" button is active. Below the navigation bar, a form for rating a pizza is shown. It includes a dropdown menu labeled "Nombre de la Pizza:" with "Barbacoa" selected, a text input field for the rating containing "8.9", and a "PUNTUAR" button.

Ilustración 27 - Error en Puntuar Pizza

En cambio, si todo ha ido bien se le mostrará un mensaje de que todo ha sido guardado correctamente:

The screenshot shows the Pizza Graph application interface after a successful rating. A green success message box at the top states: "Su puntuación se ha realizado correctamente" with a "CLOSE" button. Below this, the "Pizza Graph" logo is displayed. The navigation bar and the rating form are visible, but the rating input field is now empty.

Ilustración 28 - Puntuar Pizza Correcto



El código que realiza la solicitud de puntuar una pizza es el siguiente:

```
puntuar() {
    this.array=[];

    this.recomendandoYo=false;
    this.recomendandoResto=false;
    this.recomendandoIngre=false;
    this.aleatorio=false;
    this.ordenadoArray=false;
    this.recomendandoGente=false;
    var pas=true;
    var cadena=this.puntuacion.toString();
    var i=0;

    for(i=0;i<cadena.length;i++){
        if(cadena.charAt(i)=='.' || cadena.charAt(i)==',' ){
            pas=false;
            i=cadena.length
        }
    }

    if(this.puntuacion<=10 && this.puntuacion>=0 && this.e1!=undefined && pas==true){
        var data={user:this.user,pizza:this.e1,puntuacion:this.puntuacion};
        this.$http.post('http://localhost:3000/puntuar', data).then(response => {
            if(response.body.message==true){
                this.snackbarPizza=true;
            }

            }, response => {
                alert(JSON.stringify(response.body));
            });
    }else{
        this.snackbarPuntuar=true;
    }
},
```

Ilustración 29 - Puntuar Pizza Front-End

Como se puede ver se comprueba que las puntuaciones no tengan decimales ya sea utilizando punto o coma. Posteriormente si todo está completo de forma correcta se enviará la petición al Back-End. El Back-End responde de la siguiente manera:

```
app.post("/puntuar", function (req, res) {
  var user = req.body.user;
  var pizza = req.body.pizza;
  var puntuacion = req.body.puntuacion;

  var query = " MATCH (p:Persona)-[v:VALORA]->(e:Pizza) WHERE p.usuario = '" + user + "' AND e.nombre='" + pizza + "' AND exists(v.valoración) RETURN v.valoración"

  var array = [];
  var session = driver.session();
  var pizza;
  var tiene = false;

  // Run a Cypher statement, reading the result in a streaming manner as records arrive:
  session
    .run(query)
    .subscribe({
      onNext: function (record) {
        if (record.length != 0) {
          tiene = true;
        }
      },
      onCompleted: function () {
        if (tiene) {
          query = "MATCH(p:Persona)-[v:VALORA]->(e:Pizza) WHERE p.usuario='" + user + "' AND e.nombre='" + pizza + "' SET v.valoración=" + puntuacion;
        }
        else {
          query = "START n=node(*) , m=node(*) where n.usuario = '" + user + "' and m.nombre = '" + pizza + "' create (n)-[:VALORA{valoración:" + puntuacion + "}]>(m)";
        }
        const resultPromise = session.run(query);
        resultPromise.then(result => {
          session.close();
          res.send({ message: true })
        })
      },
      onError: function (error) {
        console.log(error);
      }
    });
});
```

Ilustración 30 - Puntuar Pizza Back-End

Como se puede observar la primera query comprueba que el usuario haya valorado con anterioridad la pizza que recibe como argumento. En caso de que sea afirmativo buscará de nuevo la relación y cambiará la propiedad PUNTUACIÓN de la relación VALORA con la nueva puntuación a guardar.

En caso de que no haya puntuado esa pizza el usuario, se cambia la siguiente query de forma que crea la relación entre el usuario y la pizza y le asigna la nueva puntuación.

## ❖ Pedir Aleatoria:

El usuario podrá solicitar una recomendación de una pizza de forma completamente aleatoria ya bien sea porque el usuario que dejar al azar su próxima pizza o bien porque quiere saber cuál se le recomendará.

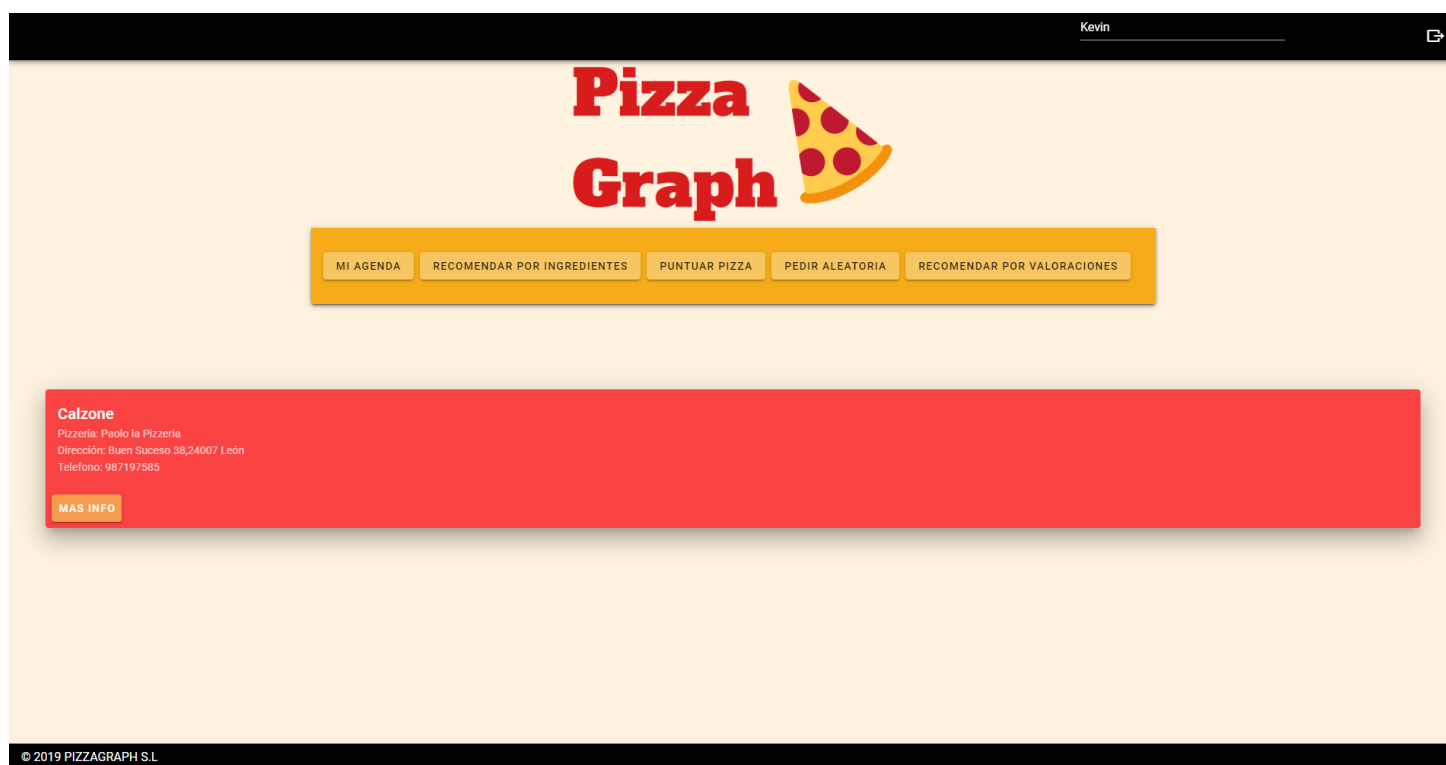


Ilustración 31 - Pedir Aleatoria

El código que realiza la solicitud de puntuar una pizza es el siguiente:



```
pedirAleatoria() {  
  
    //var data={user:this.user,password:this.password};  
    this.array=[];  
    var data={user:this.user,password:this.password};  
    this.recomendandoYo=false;  
    this.recomendandoResto=false;  
    this.recomendandoIngre=false;  
    this.puntuando=false;  
    this.recomendandoGente=false;  
    this.aleatorio= true;  
    this.ordenadoArray=false;  
  
    this.$http.post('http://localhost:3000/aleatoria', data).then(response => {  
        this.array.push(response.body);  
  
    }, response => {  
        alert(JSON.stringify(response.body));  
    });  
  
},
```

*Ilustración 32 - Pedir Aleatoria Front-End*

El Front-End realiza la petición al Back-End de una pizza aleatoria pasándole como argumento el usuario. El Back-End responde de la siguiente manera:

```
app.post("/aleatoria", function (req, res) {

  var query = "MATCH (n:Pizza) return n";
  var array = [];
  var session = driver.session();
  var pizza;

  // Run a Cypher statement, reading the result in a streaming manner as records arrive:
  session
    .run(query)
    .subscribe({
      onNext: function (record) {

        array.push(record.get(0).properties);
      },
      onCompleted: function () {

        var pos = Math.floor(Math.random() * array.length);

        pizza = array[pos];

        query = "MATCH (p:Pizzeria)-[:OFRECE]->(n:Pizza) WHERE n.nombre='" + pizza.nombre + "' return p";
        session
          .run(query)
          .subscribe({
            onNext: function (record2) {

              array = [];
              array.push(record2.get(0).properties);
            },
            onCompleted: function () {

              session.close();
              pizza.pizzeria = array[0].nombre;
              pizza.direccion = array[0].direccion;
              pizza.telefono = array[0].telefono;
              res.send(pizza);
            },
            onError: function (error) {
              console.log(error);
            }
          });
      },
      onError: function (error) { ...
      }
    });
});
```

*Ilustración 33 - Pedir Aleatoria Back-End*

La query que se utiliza en este caso simplemente realiza una búsqueda en la base de datos para obtener todas las pizzas del catálogo y posteriormente devuelve una de las pizzas obtenidas de forma aleatoria. También recoge tras utilizar otra query los datos de la pizzería donde ofrecen esa pizza aleatoria.

## ❖ Recomendar por valoraciones:

El usuario podrá obtener una recomendación basándose en la nota media que reciben todas las pizzas teniendo en cuenta que las medias aritméticas serán obtenidas utilizando los datos de las recomendaciones de todos los usuarios de la aplicación. Permite ordenar los resultados de mayor a menor nota media y viceversa:



Ilustración 34 - Recomendador por Orden

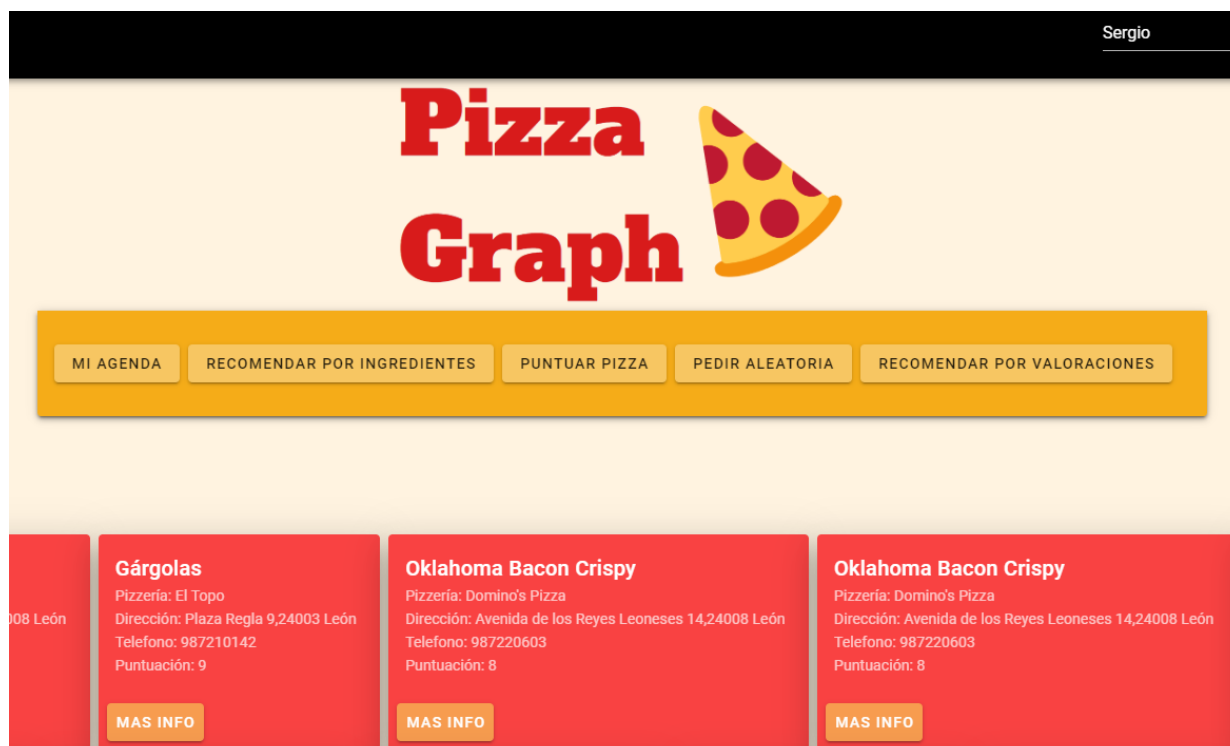


Ilustración 35 - Recomendar por Valoraciones

El código que realiza la solicitud de puntuar una pizza es el siguiente:

```
recomendarGente() {
  this.array=[];
  var data={user:this.user};
  this.recomendandoYo=false;
  this.recomendandoIngre=false;
  this.aleatorio=false;
  this.puntuando=false;
  this.recomendandoResto=true;
  this.ordenadoArray=false;
  this.recomendandoGente=false;

  this.$http.post('http://localhost:3000/gente', data).then(response => {
    this.array=response.body;
  }, response => {
    alert(JSON.stringify(response.body));
  });
},
```

Ilustración 36 - Recomendar por Valoraciones Front-End

Se puede ver que se realiza una petición al Back-End para solicitar las recomendaciones basadas en la media aritmética de las valoraciones. El Back-End responde de la siguiente manera:

```
app.post("/gente", function (req, res) {
  var user = req.body.user;
  var session = driver.session();
  var query = "MATCH (n:Persona)-[v:VALORA]->(p:Pizza)<-[:OFRECE]-(e:Pizzeria) return p,v.valoración,e";

  var array = [];
  var objeto;

  // Run a Cypher statement, reading the result in a streaming manner as records arrive:
  session
    .run(query)
    .subscribe({
      onNext: function (record) {
        objeto = (record.get(0).properties);
        objeto.puntuacion = record.get(1).low;
        objeto.pizzeria = record.get(2).properties.nombre;
        objeto.direccion = record.get(2).properties.direccion;
        objeto.telefono = record.get(2).properties.telefono;

        array.push(objeto);
      },
      onCompleted: function () {
        session.close();
        res.send(array);
      },
      onError: function (error) {
        console.log(error);
      }
    });
});
```

Ilustración 37 - Recomendar por Valoraciones Back-End

En la query realizo la búsqueda necesaria que me retorna la pizza valorada, su puntuación y la pizzería donde se ofrece. Esto se hace hasta obtener todas las valoraciones que los usuarios han realizado a las pizzas. Posteriormente se devuelve al Front-End el conjunto de valoraciones.



Como todavía hay que realizar la media aritmética de cada pizza he creado este método que me permite ordenarlos de mayor a menor o viceversa, al deseo del usuario, a través del método de la burbuja.

```
ordenarArray(par){
    var i=0;
    var j=0;
    var aux;
    var q={};
    this.definitivo=[];
    var pun=0.0;
    var veces=0;
    for( i=0;i<this.array.length;i++){
        if(JSON.stringify(q)=='{}'){

            q=this.array[i];
            pun=this.array[i].puntuacion;
            veces+=1;
        }
        else{

            if(q.nombre==this.array[i].nombre){
                pun+=this.array[i].puntuacion;
                veces+=1;
            }
            else{
                q.puntuacion=(pun/veces);
                this.definitivo.push(q);
                pun=0.0;
                veces=0;
                q={};
            }
        }
    }
    if(par==1){

        for( i=0;i<this.definitivo.length;i++){
            for( j=0;j<(this.definitivo.length-1);j++){
                if(this.definitivo[j].puntuacion<this.definitivo[j+1].puntuacion){
                    aux=this.definitivo[j];
                    this.definitivo[j]=this.definitivo[j+1];
                    this.definitivo[j+1]=aux;
                }
            }
        }
    }
}
```

Ilustración 38 - Algoritmo Ordenar Recomendaciones parte A

```

else{
    for (i=0; i<this.definitivo.length;i++) {
        for (j=0;j<(this.definitivo.length-1); j++) {
            if(this.definitivo[j].puntuacion > this.definitivo[j+1].puntuacion){
                aux= this.definitivo[j];
                this.definitivo[j] = this.definitivo[j+1];
                this.definitivo[j+1] = aux;
            }
        }
    }

    this.recomendandoYo=false;
    this.recomendandoIngre=false;
    this.aleatorio=false;
    this.ordenadoArray=false;
    this.recomendandoGente=true;
    this.recomendandoResto=false;
},

```

Ilustración 39 - Algoritmo Ordenar Recomendaciones parte B

## ❖ Ámbito General:

Como se puede ver, cada pizza que se muestra al usuario tiene un botón de “MAS INFO” que permite al usuario ver TODOS los ingredientes que componen la pizza.



Ilustración 40 - MAS INFO



Ilustración 41 - Mostrar Ingredientes

El código que realiza la solicitud de pedir los ingredientes de una pizza es el siguiente:

```

pedirIngredientes(ite){
    var data={user:this.user, pizza: ite.nombre};
    this.arrayIngredientes=[];
    this.d_ingredientes=true;

    this.$http.post('http://localhost:3000/item', data).then(response => {
        this.arrayIngredientes=response.body;
    }, response => {
        alert(JSON.stringify(response.body));
    });
},

```

Ilustración 42 - Mostrar Ingredientes Front-End

Realiza una petición al Back-End solicitando que le devuelva el resto de ingredientes que componen la pizza que es pasada como argumento. El Back-End responde de la siguiente manera:



```
app.post("/item", function (req, res) {
  var user = req.body.user;
  var pizza = req.body.pizza;
  var session = driver.session();
  var query = "MATCH (p:Pizza)-[:CONTIENE]->(i:Ingrediente) WHERE p.nombre='" + pizza + "' return i";

  var array = [];

  // Run a Cypher statement, reading the result in a streaming manner as records arrive:
  session
    .run(query)
    .subscribe({
      onNext: function (record) {
        array.push(record.get(0).properties.nombre);
      },
      onCompleted: function () {
        session.close();
        res.send(array);
      },
      onError: function (error) {
        console.log(error);
      }
    });
});
```

*Ilustración 43 - Mostrar Ingredientes Back-End*

Se realiza una búsqueda en la base de datos que devuelva los ingredientes que componen la pizza pasada como argumento. Cuando acaba de recorrer todos los ingredientes estos se guardan en un array o vector y es pasado al Front-End para que se muestren.

## 5. DAFO:



Ahora explicaré más detalladamente el DAFO:

### ❖ Fortalezas:

- Es escalable: Permite la evolución o mejora de PizzaGraph en varios ámbitos, siendo el ámbito principal la recomendación sencilla de pizzas.
- Recomendación gratuita: la aplicación web estará totalmente gratis a disposición de todas las personas y se sufragará implementando sistemas de anuncios. Cabe la

posibilidad de que en un futuro se puede añadir una opción de usuario VIP por 1,99€/mes donde no se le mostrará publicidad.

- Recomendaciones personalizadas: el usuario podrá obtener una recomendación de distintas formas (por ingredientes, valoraciones, etc).
- Fácil sistema de atención al cliente: permite una comunicación directa del usuario con el servicio de soporte en caso de que haya surgido un problema en una puntuación, recomendación o en el tratamiento de sus datos.
- Privacidad de los datos: los datos de los usuarios estarán totalmente seguros evitando brechas de seguridad. Los datos estarán encriptados.
- Sistemas de recordatorios: permite al usuario llevar la cuenta de todas las pizzas que ha degustado y de las puntuaciones que ha realizado previamente a nivel particular.

#### ❖ Oportunidades:

- No existe competencia directa: no existe una empresa que explote el hueco en el mercado de recomendar solo pizzas.

#### ❖ Transformación de debilidades en fortalezas:

- Tiempo de desarrollo corto: se puede convertir fácilmente en una fortaleza dedicando más tiempo en el futuro para perfeccionar la aplicación web.
- Recomendaciones básicas: esta debilidad subyace de la anterior, pero utilizando el tiempo que tendré en el futuro podré implementar una red neuronal que permite recomendaciones más específicas y personalizadas a cada persona.

#### ❖ Transformación de amenazas en oportunidades:

- Páginas web recomendadoras de restaurantes: estas páginas no se centran en un tipo de comida o tipo de restaurante en específico, sino que son más de ámbito general, de forma que no muestran las puntuaciones de cada plato de restaurantes. PizzaGraph, en este caso, recomienda y permite puntuar, no la pizzería, sino cada pizza que hacen.

## 6. Líneas de futuro:

PizzaGraph es una aplicación web que puede evolucionar de múltiples maneras. Estas serían algunas líneas de futuro que se pueden llegar a dar:

- En la versión inicial el usuario puede realizar una búsqueda por un único ingrediente al mismo tiempo. Por lo que en un futuro pienso en que el usuario debería poder llegar a combinar varios ingredientes para realizar una búsqueda y al mismo tiempo permitiendo una mayor personalización de la búsqueda. Para llegar a conseguir esto, creo que la mejor opción sería cambiar el selector de ingredientes por un conjunto de checkbox que permitan al usuario marcar todos los ingredientes que quiere que tenga su pizza y así poder buscar dicha pizza. Los checkbox en mi opinión, se debería mostrar en una ventana nueva saliente para evitar colapsar la página principal con tantos elementos.
- Otra línea de futuro sería implementar un nuevo método de recomendación de pizza que no se basase solo en las medias aritméticas de las valoraciones de todos los usuarios sino implementar una red neuronal de varias capas en las que las valoraciones de todos los usuarios, los ingredientes de las pizzas y su localización por aproximación tuviesen un peso importante en las ecuaciones. Para ello haría falta entrenar una red neuronal, lo cual lleva mucho tiempo, y además preocuparse de no alcanzar el “overfitting”.
- A mayores en un futuro se pretende abarcar un mayor rango de pizzerías que hay en León y posteriormente que todas las pizzerías puedan utilizar pizzas con mismos nombres, pero con su estilo propio, pudiendo así, como por ejemplo comparar la Hawaiana del Domino’s Pizza con la de Telepizza.
- También se pretende que la aplicación web no solo contenga pizzerías de león sino pizzerías, en primer lugar, de todo castilla y León y posteriormente ir aumentando poco a poco a otras comunidades autónomas hasta abarcar España entera.
- Se pretende a largo plazo implementar un sistema que permita leer un QR que contendrá el ticket de la pizzería haciendo posible que solo puedas puntuar las pizzas que están dentro del pedido y solo permitiendo al usuario puntuar dichas pizzas a lo sumo en 24 horas, sino perderá la oportunidad de puntuarlas.

- Sacar la aplicación móvil de PizzaGraph para aumentar la accesibilidad al usuario.
- Como última línea de futuro, también me gustaría que PizzaGraph se convirtiera en la base de otra aplicación web o móvil que recomendase platos y por consiguiente restaurantes de todo tipo a nivel nacional.

## 7. Bibliografía:

- 1: [https://github.com/srubim00/Business\\_Intelligence](https://github.com/srubim00/Business_Intelligence)
- 2: Telepizza VS Domino's Pizza: reputación | Blog de Oraquo. (2019). Retrieved 13 December 2019, from <https://www.oraquo.com/blog/telepizza-vs-dominos-imagen-de-marca/>
- 3: PNG, G. (2019). GitHub logo PNG. Retrieved 13 December 2019, from <http://pngimg.com/download/73352>
- 4: La guía básica de Git y Github para principiantes. (2019). Retrieved 13 December 2019, from <https://medium.com/@sthefany/primeros-pasos-con-github-7d5e0769158c>
- 5: Code, V. (2019). Visual Studio Code - Code Editing. Redefined. Retrieved 13 December 2019, from <https://code.visualstudio.com/>
- 6: Neo4j Graph Platform – The Leader in Graph Databases. (2019). Retrieved 13 December 2019, from <https://neo4j.com/>
- 7: Técnicas, C., & database, G. (2019). Graph database. Retrieved 13 December 2019, from <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/graph-database/>
- 8: Node.js, F. (2019). Node.js. Retrieved 13 December 2019, from <https://nodejs.org/es/>
- 9: Why Vuetify? — Vuetify.js. (2019). Retrieved 13 December 2019, from <https://vuetifyjs.com/es-MX/introduction/why-vuetify>
- 10: GIMP. (2019). Retrieved 13 December 2019, from <https://www.gimp.org/>



## 8. Lecciones aprendidas:

De aquí saco 2 lecciones principales:

- 1) Neo4j es el futuro ya que es un tipo de base de datos que permite casi cualquier tipo de interacción y permite al desarrollador una mayor flexibilidad que no tienen otros tipos de base de datos no basados en grafos.
- 2) La inteligencia artificial ya no es solo el futuro, es el presente y el futuro y tiene un gran numero de campos de desarrollo como la detección de fraudes en la red, recomendaciones y clasificaciones entre otras más cosas. Por lo que merece la pena investigar estos campos.