**FoodView**

Proyecto de Aula

Edward S. Mijares P. y Juan C. Mondragón O.

Patrones de Diseño Orientado a Objetos, Facultad de Ingeniería, Ingeniería de Sistemas, Universidad Cooperativa de Colombia, Campus Villavicencio

Ing. Nestor Eduardo Suat Rojas

MSC. en Ingeniería de Sistemas y Computación

Meta – Villavicencio

Mayo de 2024

Tabla de Contenido

[Introducción 4](#_Toc167435842)

[1 Planteamiento del Problema 5](#_Toc167435843)

[1.1 Descripción del problema 5](#_Toc167435844)

[1.2 Objetivos 6](#_Toc167435845)

[1.2.1 Objetivo General 6](#_Toc167435846)

[1.2.2 Objetivos Específicos 6](#_Toc167435847)

[1.3 Justificación 7](#_Toc167435848)

[2 Marco de Referencia 8](#_Toc167435849)

[2.1 Marco teórico 8](#_Toc167435850)

[2.2 Estado del arte 22](#_Toc167435851)

[2.3 Marco legal 28](#_Toc167435852)

[3 Metodología Ingenieril 30](#_Toc167435853)

[4 Desarrollo Ingenieril 36](#_Toc167435854)

[4.1 Requerimientos 36](#_Toc167435855)

[4.1.1 Requerimientos Funcionales 36](#_Toc167435856)

[4.1.2 Requerimientos No Funcionales 37](#_Toc167435857)

[4.2 Modelado 38](#_Toc167435858)

[4.3 Descripción técnica del sistema 40](#_Toc167435859)

[4.4 Diseño de interfaz 43](#_Toc167435860)

[5 Análisis Económico 44](#_Toc167435861)

[6 Conclusiones 45](#_Toc167435862)

[7 Referencias 46](#_Toc167435863)

# Introducción

En el presente informe, se presentará el desarrollo del proyecto FoodView, cuya finalidad es atender la necesidad de compartir experiencias culinarias en sitios de gastronomía en general, todo esto a través de reseñas principalmente. El desarrollo de la aplicación se llevó a cabo en el lenguaje Java, utilizando el IDE NetBeans. Esta aplicación se centra en ofrecer una experiencia completa para los usuarios, permitiéndoles gestionar sus propias preferencias y actividades relacionadas con la gastronomía, así como interactuar con otros usuarios de la aplicación. El proyecto FoodView se enmarca dentro del curso de Base de Datos, destacando la importancia de una adecuada gestión y organización de la información gastronómica y de las interacciones de los usuarios.

# Planteamiento del Problema

## Descripción del problema

La problemática que se está abordando con el proyecto FoodView radica en la necesidad de los usuarios de compartir y gestionar sus experiencias culinarias de manera efectiva en un entorno digital. Ante la diversidad y cantidad de establecimientos gastronómicos disponibles, los usuarios enfrentan dificultades para encontrar información relevante y confiable sobre restaurantes, así como para compartir sus propias opiniones y recomendaciones con otros usuarios.

En este sentido, la falta de una aplicación especializada para la gestión de experiencias culinarias dificulta la toma de decisiones de los usuarios al momento de elegir un lugar para disfrutar de la comida. Además, la ausencia de un espacio donde puedan interactuar con otros usuarios y compartir sus gustos y preferencias es super importante, llegando a ser una limitante.

Por lo tanto, FoodView busca resolver esta problemática proporcionando una aplicación que permita a los usuarios explorar, evaluar y compartir experiencias culinarias de manera sencilla y efectiva, ofreciendo herramientas para la gestión de preferencias, la interacción con otros usuarios y el acceso a información detallada sobre establecimientos, buscando facilitar la toma de decisiones de los usuarios al momento de elegir un lugar para disfrutar de la comida.

## Objetivos

### Objetivo General

Desarrollar una aplicación para la gestión y compartición de experiencias culinarias gastronómicas.

### Objetivos Específicos

Analizar los conceptos teóricos fundamentales de todo lo relacionado con base de datos, con el fin de evaluar su aplicabilidad y beneficios en el diseño y desarrollo del proyecto FoodView.

Desarrollar una Base de Datos en el lenguaje PostgreSQL, específicamente pgAdmin.

Realizar la conexión de la aplicación con el entorno de la base de datos en PostgreSQL.

## Justificación

La justificación del proyecto FoodView radica en la necesidad de los usuarios de contar con una aplicación especializada que les permita gestionar y compartir sus experiencias culinarias en un entorno digital. La diversidad y la abundancia de establecimientos gastronómicos dificultan a los usuarios encontrar información confiable y relevante sobre restaurantes, así como interactuar con otros usuarios para compartir opiniones y recomendaciones. Esta problemática afecta la experiencia del usuario y la toma de decisiones al momento de elegir un lugar para disfrutar de la comida. Por lo tanto, FoodView se propone como una solución que no solo facilita la los aspectos mencionados anteriormente, sino que también promueve la interacción social y proporciona herramientas para gestionar preferencias de manera eficiente. La metodología de investigación se centrará en el análisis de las necesidades y preferencias de los usuarios, así como en la evaluación de plataformas similares y el desarrollo de prototipos iterativos. Se espera que este proyecto beneficie a la sociedad al mejorar la experiencia gastronómica en línea, fomentar la participación comunitaria y promover una cultura de compartir conocimientos culinarios.

# Marco de Referencia

## Marco teórico

**PostgreSQL – Definición**

PostgreSQL, comúnmente conocido como "Post-GRES", es una base de datos de código abierto reconocida por su fiabilidad, flexibilidad y conformidad con los estándares técnicos abiertos. A diferencia de otros sistemas de gestión de bases de datos relacionales (RDBMS), PostgreSQL soporta tanto tipos de datos relacionales como no relacionales, lo que la convierte en una de las bases de datos más versátiles, estables y maduras disponibles en la actualidad. (S., 2022)

El origen de PostgreSQL se remonta a 1986 como una evolución de INGRES, un proyecto de base de datos relacional SQL de código abierto iniciado a principios de la década de 1970. La nueva versión, inicialmente denominada POSTGRES, fue creada por Michael Stonebraker, un destacado profesor de informática en la Universidad de Berkeley. En 1994, el proyecto incorporó soporte para SQL y, poco después, se renombró a PostgreSQL, reflejando su compatibilidad con este lenguaje de consulta.

En la actualidad, PostgreSQL sigue avanzando gracias a la dedicación de un equipo internacional de desarrolladores y colaboradores que trabajan constantemente para mejorar esta base de datos de código abierto y mantenerla gratuita. La comunidad de PostgreSQL se esfuerza por introducir nuevas funcionalidades, optimizar el rendimiento y garantizar la seguridad, consolidando así su posición como una opción robusta y confiable para la gestión de datos en diversos entornos. (S., 2022)

**¿Por qué usar PostgreSQL?**

En el actual panorama digital, caracterizado por el rápido avance tecnológico, mantener sistemas de bases de datos dinámicos es fundamental. PostgreSQL destaca por su capacidad de expansión y versatilidad, permitiendo soportar una amplia variedad de casos de uso especializados gracias a su poderoso ecosistema de extensiones, que abarca desde tipos de datos de series de tiempo hasta análisis geoespaciales.

El diseño accesible y versátil de PostgreSQL la convierte en una solución "todo en uno" para muchas empresas que buscan formas rentables y eficientes de mejorar sus sistemas de gestión de bases de datos. Como base de datos de código abierto, PostgreSQL está libre de restricciones de licencia y del riesgo de bloqueo por parte de proveedores, lo que reduce significativamente el riesgo de implementación excesiva. Tanto desarrolladores expertos como empresas conscientes de las limitaciones de los sistemas de bases de datos tradicionales apoyan firmemente PostgreSQL. Estos profesionales trabajan arduamente para ofrecer un sistema de gestión de bases de datos relacional de primera clase, probado y comprobado en diversos entornos.

PostgreSQL es una opción preferida para quienes buscan una base de datos robusta, flexible y económica que pueda adaptarse rápidamente a las necesidades cambiantes del mercado y soportar una amplia gama de aplicaciones modernas. (S., 2022)

**Consultas en PostgreSQL**

Existen dos tipos de comandos SQL:

Los que permiten crear y definir nuevas bases de datos, campos e índices.

* CREATE Utilizado para crear nuevas tablas, campos e índices
* DROP Empleado para eliminar tablas e índices.
* ALTER Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos. (Sarría, s.f.)

Los que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.

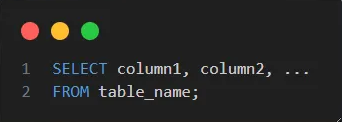
* SELECT Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.
* INSERT Utilizado para cargar lotes de datos en la base de datos en una única operación.
* UPDATE Utilizado para modificar los valores de los campos y registros especificados.
* DELETE Utilizado para eliminar registros de una tabla de una base de datos. (Sarría, s.f.)

A continuación, una breve descripción de algunos comandos y ejemplos para poder entender su aplicabilidad y función:

**Paso 1: Comando SELECT**

El comando SELECT se utiliza para recuperar datos de una o más tablas de una base de datos. La sintaxis básica de una consulta SELECT es la siguiente:

Ilustración 1 - Consulta SELECT



column1, column2, ...: Son los nombres de las columnas que deseamos seleccionar de la tabla.

table\_name: Es el nombre de la tabla de la cual queremos recuperar los datos.

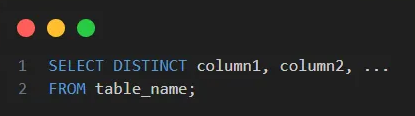
**Paso 2: La Cláusula DISTINCT**

La cláusula DISTINCT se utiliza para eliminar filas duplicadas de los resultados de una consulta. Esto significa que devuelve solo valores únicos en una columna o en una combinación de columnas.

Sintaxis

La sintaxis básica de una consulta con la cláusula DISTINCT es la siguiente:

Ilustración 2 - Clausula DISTINC



column1, column2, ...: Son los nombres de las columnas de las cuales queremos obtener valores únicos.

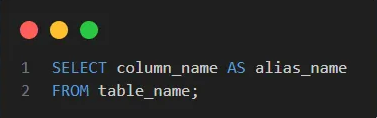
table\_name: Es el nombre de la tabla de la cual queremos recuperar los datos.

**Paso 3: Alias en Consultas SQL**

En SQL, los alias se utilizan para asignar un nombre temporal a una columna o a una tabla en una consulta. Esto puede hacer que las consultas sean más legibles y comprensibles, especialmente cuando se trabaja con múltiples tablas o columnas con nombres largos.

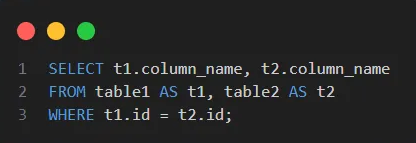
Para asignar un alias a una columna, simplemente escribimos el nombre de la columna seguido de la palabra clave AS y el alias deseado. Por ejemplo:

Ilustración 3 - AS (Alias)



Ahora, para las tablas también se les puede asignar un alias de la siguiente manera:

Ilustración 4 AS(Alias) en Tablas



Escribimos el nombre de la tabla seguido de la palabra clave AS y el alias deseado.

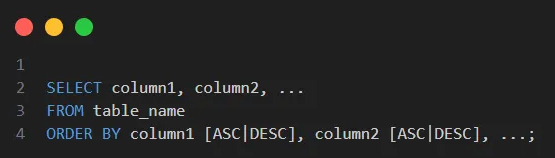
**Paso 4: Ordenando Resultados con ORDER BY**

En SQL, la cláusula ORDER BY se utiliza para ordenar los resultados de una consulta basándose en una o más columnas. Esto es útil cuando queremos visualizar los datos de una tabla en un orden específico, como orden alfabético, numérico o cronológico.

Sintaxis de **ORDER BY**

La sintaxis básica de la cláusula ORDER BY es la siguiente:

Ilustración 5 - ORDER BY



column1, column2, etc.: Las columnas por las cuales se desea ordenar los resultados.

ASC: Orden ascendente (por defecto).

DESC: Orden descendente.

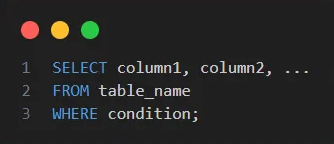
**Paso 5: Filtrando Resultados con la Cláusula WHERE**

En SQL, la cláusula WHERE se utiliza para filtrar los resultados de una consulta basándose en una condición específica. Esto nos permite seleccionar solo las filas que cumplen con ciertos criterios, lo que hace que nuestras consultas sean más precisas y útiles.

**Sintaxis de WHERE**

La sintaxis básica de la cláusula WHERE es la siguiente:

Ilustración 6 - WHERE



column1, column2, etc.: Las columnas que queremos seleccionar en la consulta.

table\_name: El nombre de la tabla de la cual queremos seleccionar datos.

condition: La condición que deben cumplir las filas para ser seleccionadas.

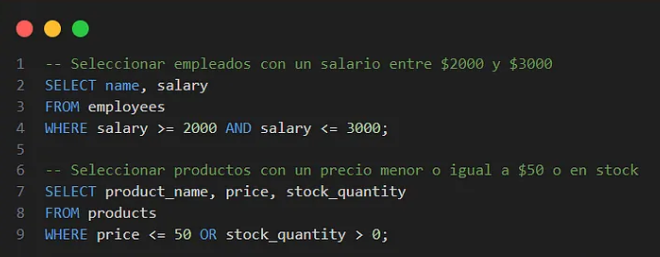
**Paso 4: Combina Condiciones con AND, OR e IS NULL**

En SQL, podemos combinar condiciones utilizando operadores lógicos como AND, OR e IS NULL. Estos operadores nos permiten construir consultas más complejas y flexibles, que pueden adaptarse a una amplia gama de situaciones.

**Uso de AND y OR**

Los operadores **AND y OR** nos permiten combinar múltiples condiciones en una sola cláusula WHERE. Aquí hay algunos ejemplos de cómo usarlos:

Ilustración 7 - Uso del AND y OR



**Operadores lógicos**

* AND Evalua dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
* OR Evalúa dos condiciones y devuelve un valor de verdar si alguna de las dos es cierta.
* NOT Devuelve el valor contrario de la expresión. (Sarría, s.f.)

**Operadores de Comparación**

* < Menor que
* > Mayor que
* <> Distinto de
* <= Menor ó Igual que
* >= Mayor ó Igual que 3
* = Igual que
* BETWEEN Utilizado para especificar un intervalo de valores o LIKE Para la comparación de una cadena de texto con una expresión regular.

**Funciones de Agregación**

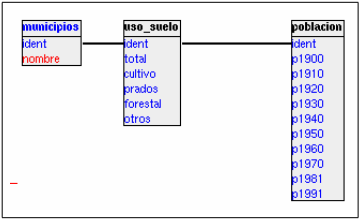
Las funciones de agregación se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

* AVG Utilizada para calcular el promedio de los valores de un campo determinado.
* COUNT Utilizada para devolver el número de registros de la selección.
* SUM Utilizada para devolver la suma de todos los valores de un campo determinado.
* MAX Utilizada para devolver el valor más alto de un campo especificado.
* MIN Utilizada para devolver el valor más bajo de un campo especificado. (Sarría, s.f.)

**Base de datos relacionales**

Es el modelo más utilizado en la actualidad. Una base de datos relacional consiste fundamentalmente en un conjunto de tablas, similares a las de una hoja de cálculo, compuestas por filas (registros) y columnas (campos). Los registros representan cada uno de los objetos descritos en la tabla, mientras que los campos representan los atributos (variables de cualquier tipo) de estos objetos. En el modelo relacional de base de datos, las tablas comparten ciertos campos entre sí. Estos campos comunes permiten establecer relaciones entre las tablas, facilitando la realización de consultas complejas. (Ilustración 8)

Ilustración 8 - Esquema de base de datos relacional



En dicha figura se muestran tres tablas con información municipal: la primera contiene los nombres de los municipios, la segunda muestra el porcentaje de diferentes usos del suelo en cada municipio, y la tercera proporciona la población de cada municipio a lo largo del siglo XX. El campo común "ident" actúa como un identificador numérico único para cada municipio. (Sarría, s.f.)

La idea fundamental de las bases de datos relacionales es la existencia de entidades (filas en una tabla) caracterizadas por atributos (columnas en la tabla). Cada tabla almacena entidades del mismo tipo y se establecen relaciones entre entidades de diferentes tipos. Los campos compartidos entre tablas permiten establecer estas relaciones. Los atributos pueden ser de unos pocos tipos simples: (Sarría, s.f.)

* Números enteros.
* Números reales.
* Cadena de caracteres de longitud variable.

Estos tipos simples se conocen como tipos atómicos y favorecen una mayor eficiencia en el manejo de la base de datos, aunque sacrifican la flexibilidad necesaria para gestionar elementos complejos del mundo real y complican la administración de datos espaciales. En general, representan un desafío para cualquier tipo de datos geométricos. Las relaciones que se establecen entre los diferentes elementos de dos tablas en una base de datos relacional pueden ser de tres tipos distintos: (Sarría, s.f.)

* Relaciones uno a uno, se establecen entre una entidad de una tabla y otra entidad de otra tabla. Un ejemplo aparece en la ilustración 8.
* Relaciones uno a varios, se establecen entre varias entidades de una tabla y una entidad de otra tabla. Un ejemplo sería una tabla de pluviómetros en la que se indicara el municipio en el que se encuentra. La relación sería entre un municipio y varios pluviómetros
* Relaciones varias a varios, se establecen entre varias entidades de cada una de las tablas. Un ejemplo sería una tabla con retenes de bomberos y otra con espacios naturales a los que cada uno debe acudir en caso de incendio.

**Diagrama de Casos de Uso**

Representar todos los tipos de objetos, relaciones y procesos con un solo diagrama sería excesivamente complejo y confuso. Por esta razón, UML utiliza 14 tipos distintos de diagramas, que se pueden clasificar en diagramas de estructura, de comportamiento y de interacción, siendo estos últimos una subcategoría de los diagramas de comportamiento.

Los diagramas de estructura se enfocan en mostrar todos los elementos de un sistema y las relaciones entre ellos. Un ejemplo común es el diagrama de clases, que permite agrupar y visualizar los elementos en jerarquías. En contraste, los diagramas de comportamiento no representan estructuras estáticas, sino que ilustran el flujo del proceso planificado o real que debería ocurrir al ejecutar un programa o software, enfatizando la dinámica.

El diagrama de casos de uso también pertenece a los diagramas de comportamiento, pero es un modelo particular porque muestra el comportamiento esperado de un sistema o software en un caso específico. A diferencia de otros diagramas de comportamiento en UML, el diagrama de casos de uso es relativamente estático, ya que solo describe acciones y objetivos sin detallar la secuencia exacta de procesos y acciones. Para representar esto último, se utilizan otros tipos de diagramas en UML, como los diagramas de actividades, que muestran los procesos cronológicamente, o los diagramas de secuencia, que ilustran el intercambio de mensajes entre los diferentes elementos de un sistema. (Digital Guide IONOS, 2020)

Elementos y estructura del diagrama de casos de uso

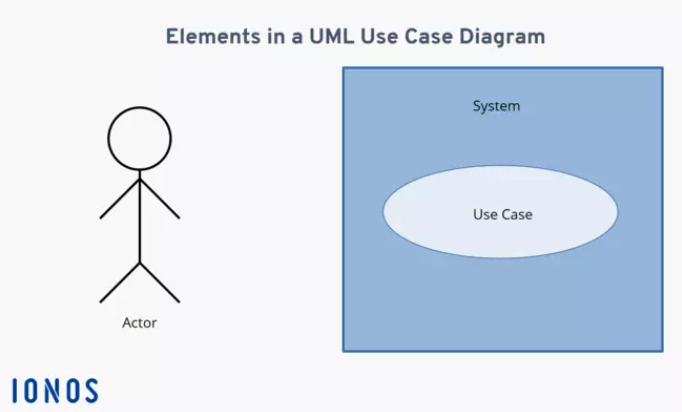
Para garantizar que el diagrama de casos de uso sea comprensible para todo el mundo de un vistazo, se utilizan elementos estandarizados para elaborarlo. En primer lugar, hay tres elementos principales:

1. Actor: tanto si es una persona, como un sistema, se representa con el dibujo de una figura humana esquemática.

2. Sistema: el sistema al que se refiere el caso de uso tiene forma de rectángulo.

3. Caso de uso: se muestra como una elipse que suele incluir un texto describiendo brevemente el proceso. (Digital Guide IONOS, 2020)

Ilustración 9 - Diagrama de casos de uso

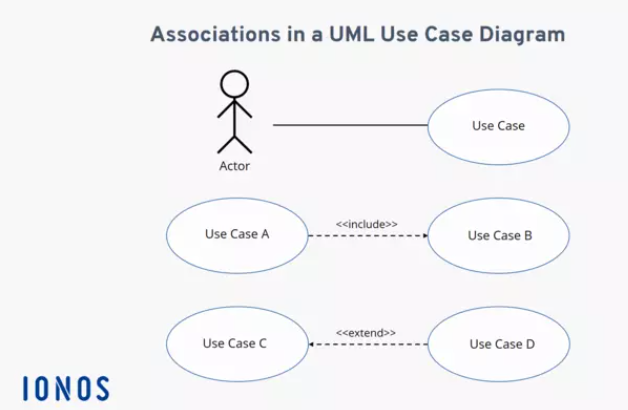


La relación entre estos elementos se representa mediante líneas de conexión llamadas asociaciones. Una línea recta entre el actor y el caso de uso indica que están relacionados. Una línea discontinua establece una relación entre diferentes casos de uso. Dado que existen dos tipos distintos de asociaciones entre casos de uso, a las líneas se les añade una palabra clave, denominada "estereotipo" en UML, que se coloca entre dos pares de paréntesis angulares. La relación de dependencia entre los casos de uso se muestra con la punta de una flecha. Se distinguen dos estereotipos:

1. Asociación <>: el caso de uso donde comienza la línea discontinua se relaciona con otro caso de uso indicado por la punta de la flecha.

2. Asociación <>: el caso de uso donde comienza la línea discontinua puede extenderse al caso de uso indicado por la punta de la flecha bajo ciertas condiciones, que no necesariamente deben cumplirse en todos los casos.

Ilustración 10 - Asociaciones UML Use Case Diagram

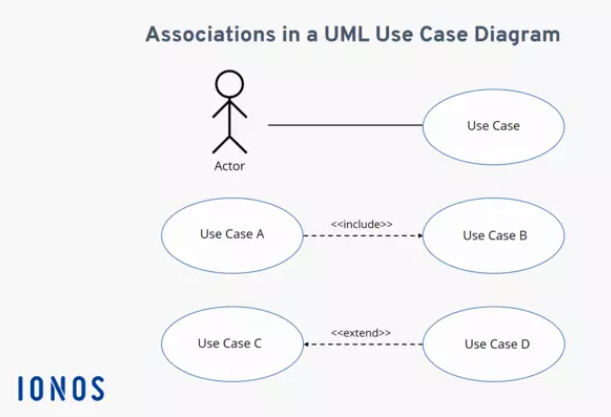


La asociación <<include>> requiere la ejecución de ambos casos de uso, mientras que la asociación <<extend>> depende de ciertas condiciones que se representan en el diagrama de casos de uso en UML como puntos de extensión. Estos puntos de extensión se muestran de dos maneras en el esquema:

Mención en la elipse del caso de uso: se describe brevemente el posible punto de extensión debajo del título del caso de uso.

Nota: partiendo del estereotipo <<extend>>, se dibuja una línea discontinua que termina en una nota (representada como un rectángulo con una esquina doblada). Esta nota incluye los títulos “Condición” y “Punto de extensión”. Detrás de “Condición” se indica, entre corchetes, la condición que debe cumplirse para que el segundo caso de uso se ejecute. Detrás de “Punto de extensión” se menciona el nombre que aparece en la elipse del caso de uso correspondiente, especificando a qué se refiere la extensión. (Digital Guide IONOS, 2020)

Ilustración 11 - Asociaciones diagrama de casos de uso



## Estado del arte

El proyecto FoodView se centra en el desarrollo de una aplicación para la gestión y compartición de experiencias culinarias gastronómicas, utilizando la base de datos PostgreSQL. Para contextualizar y justificar este proyecto, es revisar el estado del arte en relación con las bases de datos relacionales, especialmente PostgreSQL, y cómo estas tecnologías se aplican en el ámbito de las aplicaciones gastronómicas, como es el caso de nuestro proyecto.

**Desarrollo**

Bases de Datos Relacionales y PostgreSQL

Historia y Evolución:

Las bases de datos relacionales han sido el estándar en la gestión de datos debido a su estructura organizada y eficiente. PostgreSQL, en particular, se destaca por su capacidad de manejar tanto datos relacionales como no relacionales, ofreciendo una versatilidad superior en comparación con otros sistemas de gestión de bases de datos (SGBD).

PostgreSQL se originó como una evolución de INGRES en 1986, bajo la dirección de Michael Stonebraker en la Universidad de Berkeley. Inicialmente conocido como POSTGRES, el proyecto añadió soporte para SQL en 1994, momento en el cual se renombró a PostgreSQL, consolidándose como una herramienta robusta y flexible.

**Características y Ventajas:**

PostgreSQL es conocido por su conformidad con los estándares técnicos abiertos, fiabilidad y extensibilidad. Su ecosistema de extensiones permite soportar una amplia gama de aplicaciones, desde análisis geoespaciales hasta tipos de datos de series de tiempo. Este diseño flexible la convierte en una opción preferida para empresas que buscan soluciones de gestión de bases de datos sin los riesgos asociados al bloqueo de proveedores.

**Revisión de Aplicaciones Existentes:**

En el campo de las aplicaciones gastronómicas, existen varios ejemplos que demuestran cómo las bases de datos relacionales pueden ser utilizadas para gestionar grandes volúmenes de datos sobre recetas, ingredientes y experiencias culinarias. Estas aplicaciones suelen aprovechar las capacidades de consulta avanzada y la integridad referencial de las bases de datos relacionales para proporcionar una experiencia de usuario rica y detallada. Un ejemplo claro de ello es TripAdvisor.

TripAdvisor es una aplicación y sitio web que permite a los usuarios buscar y compartir experiencias gastronómicas en una amplia variedad de restaurantes y establecimientos de comida en todo el mundo. La plataforma utiliza una base de datos relacional para almacenar y gestionar información sobre millones de restaurantes, incluyendo detalles como menús, horarios de apertura, ubicación, y reseñas de usuarios. Las capacidades avanzadas de consulta de la base de datos permiten a los usuarios filtrar resultados por tipo de cocina, rango de precios, calificación y otros criterios relevantes.

Además, TripAdvisor utiliza la integridad referencial para asegurar que las reseñas y calificaciones de los usuarios estén correctamente vinculadas a los restaurantes correspondientes. Esto no solo mejora la precisión de la información disponible, sino que también facilita la personalización de la experiencia del usuario, permitiendo recomendaciones basadas en preferencias y hábitos de búsqueda previos.

El éxito de TripAdvisor demuestra cómo las bases de datos relacionales pueden ser una herramienta poderosa para gestionar y presentar información gastronómica de manera eficiente, proporcionando a los usuarios una plataforma intuitiva y confiable para descubrir y compartir experiencias culinarias.

**Desafíos y Soluciones:**

Un desafío común en estas aplicaciones es la necesidad de manejar datos complejos y no estructurados, como imágenes de platos y reseñas textuales. PostgreSQL, con su soporte para JSON y otros tipos de datos no relacionales, ofrece una solución eficaz a este problema, permitiendo una integración fluida de datos estructurados y no estructurados.

**Consultas en PostgreSQL**

Comandos y Funcionalidades:

PostgreSQL ofrece una variedad de comandos SQL que son esenciales para la gestión eficiente de bases de datos. Comandos como CREATE, DROP, ALTER, SELECT, INSERT, UPDATE y DELETE permiten crear, modificar y consultar bases de datos de manera efectiva. Además, funciones de agregación como AVG, COUNT, SUM, MAX y MIN facilitan el análisis de datos agrupados, lo cual es crucial para aplicaciones que necesitan generar reportes y estadísticas detalladas.

Ejemplos Prácticos:

La utilización de comandos avanzados como SELECT con cláusulas DISTINCT, alias y ORDER BY, junto con la cláusula WHERE para filtrado de datos, demuestra la potencia y flexibilidad de PostgreSQL en la gestión de bases de datos complejas. Estas capacidades son particularmente útiles para personalizar y optimizar las consultas en aplicaciones gastronómicas.

**Diagrama de Casos de Uso**

Los diagramas de casos de uso en UML (Unified Modeling Language) permiten representar de manera clara y comprensible las interacciones entre usuarios y el sistema, así como los diferentes procesos y escenarios que se pueden presentar en una aplicación. Para el proyecto FoodView, el uso de diagramas de casos de uso facilita la visualización de funcionalidades clave y la identificación de requisitos del sistema, asegurando que se cubran todas las necesidades del usuario final .

Se ha logrado desarrollar una aplicación para la gestión y compartición de experiencias culinarias gastronómicas, utilizando PostgreSQL como base de datos. La investigación y desarrollo llevados a cabo confirman la viabilidad y efectividad de PostgreSQL en este contexto, cumpliendo con los objetivos establecidos.

Se realizó un análisis de los conceptos fundamentales relacionados con las bases de datos, evaluando su aplicabilidad y beneficios para el proyecto FoodView. Este análisis proporcionó una base sólida para el diseño y desarrollo del sistema.

**Desarrollo de la Base de Datos en PostgreSQL:**

Se desarrolló una base de datos utilizando PostgreSQL y pgAdmin, implementando las funcionalidades necesarias para soportar la gestión de experiencias culinarias, garantizando eficiencia y robustez en el manejo de datos.

Se estableció con éxito la conexión entre la aplicación FoodView y el entorno de base de datos en PostgreSQL, permitiendo una interacción fluida y segura entre la interfaz de usuario y el almacenamiento de datos.

## Marco legal

Protección de datos y privacidad:

La Ley 1581 de 2012 define las pautas generales para la protección de datos personales en Colombia. Esta legislación es aplicable a individuos o entidades, ya sean personas naturales o jurídicas, que lleven a cabo el tratamiento de datos personales en el país. Además de establecer los derechos de los titulares de los datos, la ley también especifica las responsabilidades de aquellos encargados y responsables del tratamiento de datos, así como las consecuencias por incumplimiento. (Colombia, Colombia Potencia de la Vida, 2012)

La normativa consagra el derecho fundamental de hábeas data, el cual otorga a toda persona el derecho de conocer, actualizar y rectificar la información recolectada sobre ella en archivos y bases de datos, ya sean de entidades públicas o privadas. Los responsables del tratamiento de datos están obligados a garantizar de manera continua el ejercicio pleno y efectivo del derecho de hábeas data.

Seguridad de la información:

Por medio de la cual se modifica el Código Penal, se crea un nuevo bien jurídico tutelado - denominado "de la protección de la información y de los datos"- y se preservan integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones, entre otras disposiciones. (Colombia, Secretaría Jurídica Distrital, 2021)

En Colombia, varias leyes y normativas regulan la seguridad de la información como:

Ley 1581 de 2012 Reconoce y protege el derecho que tienen todas las personas a conocer, actualizar y rectificar las informaciones que se hayan recogido sobre ellas en bases de datos o archivos que sean susceptibles de tratamiento por entidades de naturaleza pública o privada. (Colombia, Colombia Potencial de la Vida, 2012)

# Metodología Ingenieril

Esta metodología XP, o conocida como Extreme Programming es una metodología de desarrollo que interviene dentro de las metodologías ágiles, cuyo fin es la organización, eficacia, flexibilidad y control en el momento de gestionar un proyecto, y así tener una eficiencia dentro del equipo de desarrollo. Esta metodología la formuló Kent Beck, autor del primer libro sobre este ámbito llamado «Extreme Programming Explained: Embrace Change», publicado en 1999.

Esta metodología XP es la adecuada para el desarrollo de la propuesta porque, de acuerdo con la justificación, permite mantener una comunicación constante con el cliente, lo que permitió que exista una adaptación idónea para el aporte al diseño de un sistema de monitoreo para una extrusora de elaboración de filamento.

La programación extrema consta de cinco valores que son fundamentales para tener una mayor organización y tener éxitos en la propuesta de desarrollo de nuestro proyecto y no tener inconvenientes en la hora de ponernos a desarrollar el mismo, esto contribuye a tener un ambiente de trabajo colaborativo y muy organizado.

Ilustración 12 - Valores de la metodología



**Comunicación:** la comunicación es un parte fundamental en XP para que el proyecto tenga éxito, el constante intercambio de información entre el equipo de trabajo y el cliente garantiza que el software cumpla con los requerimientos del usuario.

**Simplicidad:** este valor está relacionado con mantener la simplicidad al momento de escribir el código porque facilita la refactorización para los programadores y garantiza que el software sea escalable.

**Retroalimentación:** el cliente es parte del desarrollo del software aportando su opinión sobre los avances realizados y en la fase de pruebas verifica el funcionamiento adecuado del sistema con base en sus requerimientos.

**Coraje:** En XP, el coraje se refiere a la disposición del equipo para enfrentar los desafíos y tomar decisiones audaces en beneficio del proyecto. Esto implica estar dispuesto a experimentar, adaptarse rápidamente a los cambios y asumir riesgos calculados para lograr los objetivos establecidos.

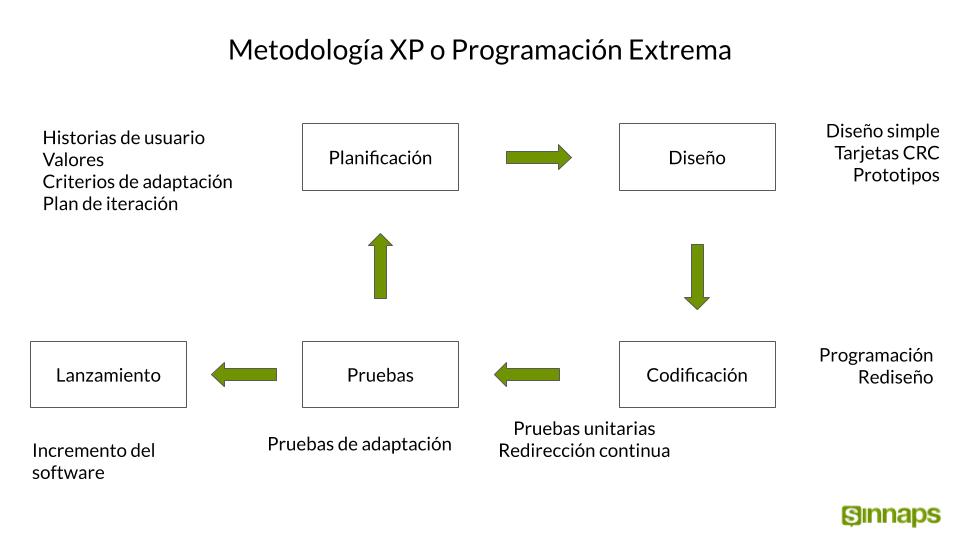
**Respeto:** Debe existir mutuo respeto entre los miembros del equipo y también hacia el cliente, solo así se logra asegurar la calidad del producto. Todos deben aportar valor al desarrollo del proyecto, aunque simplemente sea entusiasmo.

**Procesos de la Metodología XP**

Dentro de la programación extrema se utilizan varias iteraciones que se deben de cumplir y que son importantes, estas son planeación, diseño, codificación y pruebas.

La siguiente figura 2 hace referencia de cómo se deben de llevar a cabo cada una de las iteraciones y como están compuesta cada una de ellas, cabe recalcar que dentro de un proyecto que utiliza la metodología XP, implica que debe de realizar entre 10 a 15 iteraciones habitualmente.

Ilustración 13 – Fases XP



**Planificación:** en esta etapa se identifican las historias de usuario, las cuales son tarjetas donde se detallan las funcionalidades específicas del software que se va a desarrollar.

En las historias de usuarios se determinan por medio de las necesidades que el cliente tenga, y estas se dividen de acuerdo con su prioridad y se descompone en versiones.

**Diseño:** en esta etapa se realizan las programaciones, es aquí primero debemos de buscar código sencillo, y con el flujo indispensable para hacer que cumpla las funciones que tiene la historia del usuario y considerando siempre su experiencia. En esta etapa se incluye lo que es la utilización de la programación colaborativa, flexible e integral, todo esto con la finalidad de poder pasar a la fase de codificación y no tener inconvenientes en el proceso de desarrollo.

**Codificación:** aquí inicia la fase de programación. Se trabaja en parejas en el mismo ordenador con el objetivo de obtener un código de propiedad colectiva, permitiendo que el equipo avance simultáneamente y tenga conocimiento acerca del progreso del proyecto y así reducir problemas mínimos en el cumplimiento del desarrollo.

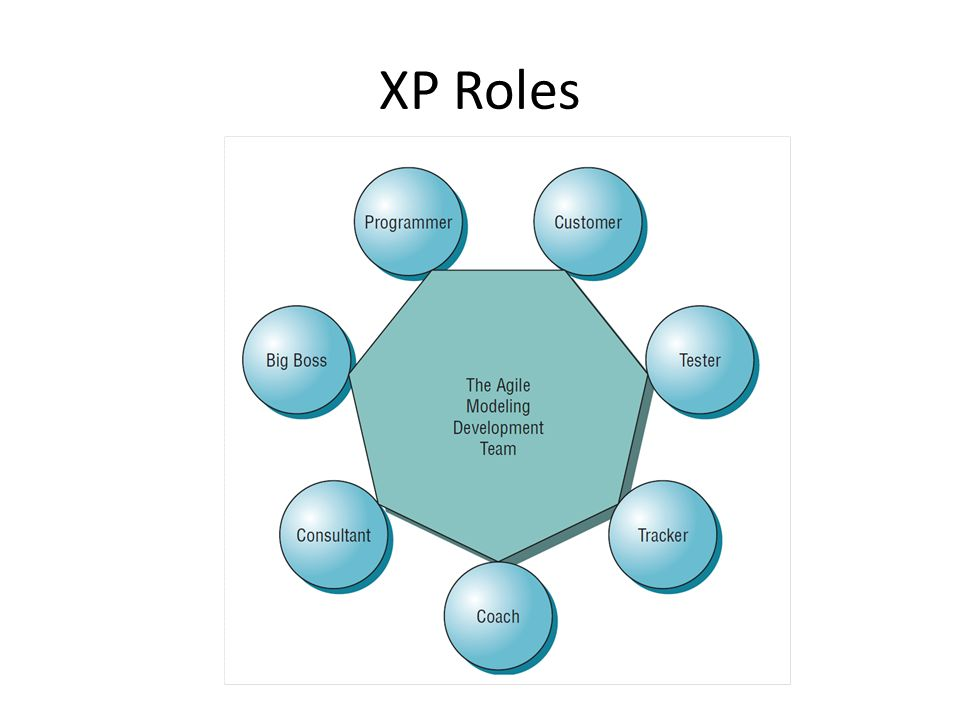
**Pruebas:** esta fase se realiza cuando el código de una función está listo, sometiéndose a pruebas unitarias continuas con la finalidad de realizar corrección de fallas de forma periódica.

XP trabaja con tiempos relativamente cortos, por lo que el control automatizado y constante es muy importante dentro de la gestión de proyectos.

**Roles XP**

Integrantes y partícipes que interviene dentro de esta metodología XP:

Ilustración 14 – Roles XP



**Programador:** es quien se encarga de escribir las pruebas unitarias, además es quien genera el código del software. La comunicación y coordinación debe ser indispensable entre todos los miembros del equipo.

**Cliente:** es la persona encargada de establecer las prioridades y marcar el proyecto, como a la vez de escribir las tarjetas denominadas historias de usuario y aprobar las pruebas funcionales que se realicen al software. Dentro del proyecto existe un solo cliente, aunque puede ser un interlocutor que representa a un sin número de personas que se verán beneficiadas por el software.

**Encargado de pruebas (Tester):** es la persona encargada de ejecutar pruebas al software de forma regular, da a conocer los resultados al equipo, además ayuda a escribir las pruebas funcionales al cliente.

**Encargado de seguimiento (Tracker):** esencial dentro de un equipo de desarrollo en XP. Además de proporcionar retroalimentación al equipo, este rol se encarga de asegurarse de que el progreso del proyecto esté alineado con los objetivos establecidos para cada iteración. El Tracker monitorea de cerca el avance del trabajo, identificando posibles desviaciones y riesgos que puedan surgir durante el proceso de desarrollo. Además, evalúa si los objetivos planteados para cada iteración son alcanzables dentro de las restricciones de tiempo y recursos disponibles. En resumen, el Tracker desempeña un papel fundamental en mantener el enfoque del equipo en las metas del proyecto y en asegurar que se cumplan los compromisos establecidos.

**Entrenador (Coach):** es el que debe conocer el proceso de la metodología XP para guiar al equipo siguiendo el proceso de forma correcta haciendo uso de las prácticas.

# Desarrollo Ingenieril

## Requerimientos

### Requerimientos Funcionales

Gestión Usuarios

CRUD de usuarios

Permitir al usuario interactuar con otros usuarios de la aplicación (foodieFriends)

Permitir al usuario administrar sus preferencias (Seleccionar, deseleccionar)

Permitir al usuario administrar sus restaurantes favoritos en el momento que quiera, (operaciones crud) y mostrarlos en su perfil para que otros usuarios interactúen

Permitir al usuario hacer reseñas a restaurantes que haya visitado, agregando una puntuación en número de estrellas y una descripción de la reseña(opcional), puede hacer muchas reseñas de un restaurante, pero la puntuación se promediará

Permitir al usuario hacer listas de restaurantes (placeList) para compartir con sus amigos (foodieFriends), la cual puede contener uno o más restaurantes

Permitir al usuario acceder al catálogo de restaurantes disponibles en la aplicación, los cuales tiene su información para que el usuario la lea.

Permitir al usuario acceder a las puntuaciones de un restaurante (Metrics) y las reseñas de otros usuarios.

Permitir al usuario observar el perfil de otro usuario

El sistema debe tener un apartado de noticias donde se mostrarán los lugares que tengan mucha más relación con el usuario por medio de las preferencias.

El sistema debe mostrar una sección de muro o publicaciones (Reseñas que hacen sus foodieFriends) y si ese usuario no tiene ningún amigo mostrará sugerencias por medio de la ubicación del usuario.

### Requerimientos No Funcionales

Base de datos

Interfaz intuitiva

Diseño responsive

Seguridad de los datos almacenados localmente

Diseño escalable

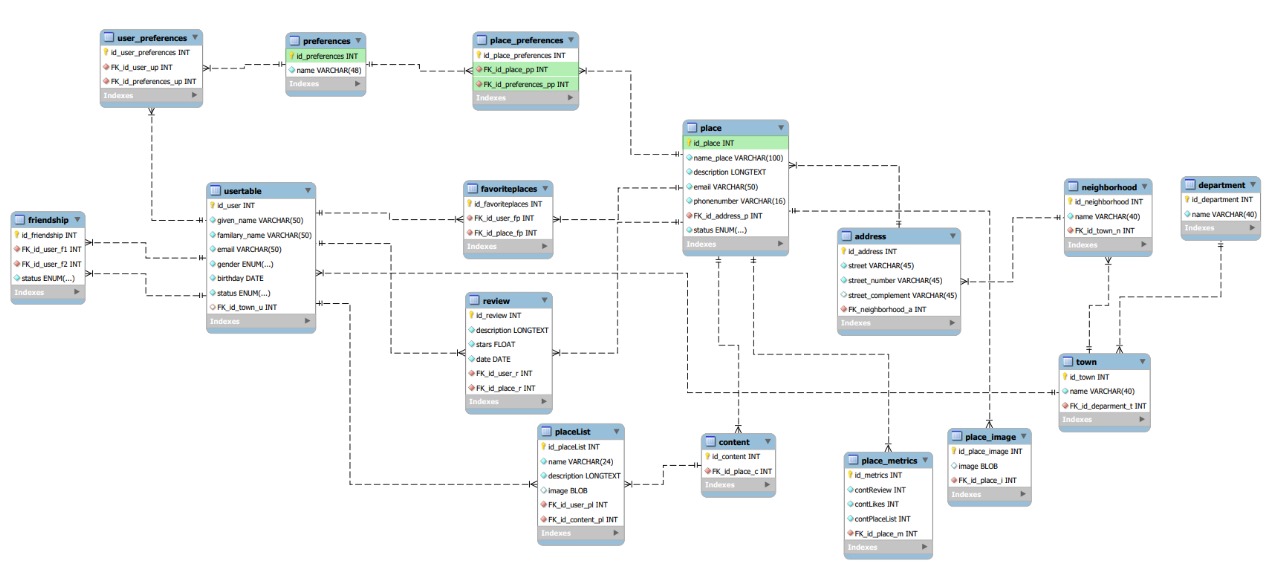
Atención de ayuda a los usuarios en caso de errores en el sistema

## Modelado

Ilustración 15 - Diagrama de Casos de Uso

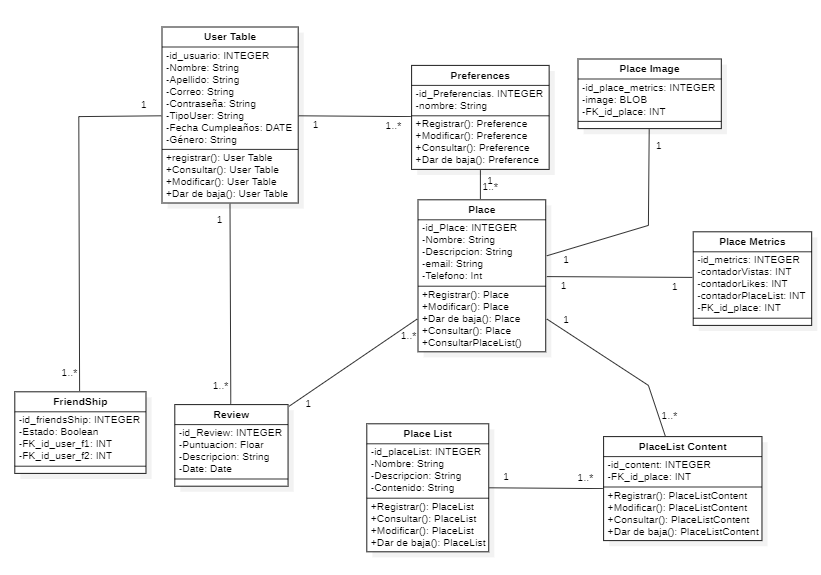


**Nota.** *Diagrama de Casos de uso, reflejando las tareas que el sistema otorga y concierne en cuanto al funcionamiento.*

Ilustración 16 - DER (Diagrama Entidad-Relación)

**Nota.** *Diagrama Entidad – Relación del proyecto FoodView, reflejando las entidades principales y su normalización en cuanto al sistema/estructura del código.*

Ilustración 17 - Diagrama de Clases



## Descripción técnica del sistema

* Modelo Vista Controlador

El patrón Modelo/Vista/Controlador (MVC) fue originalmente descrito por Trygve Reenskaug en 1979 y se introdujo como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Este patrón fue concebido para simplificar la implementación de sistemas que manejan datos múltiples y sincronizados. En el MVC, el Modelo, las Vistas y los Controladores se consideran entidades separadas, permitiendo que cualquier cambio en el Modelo se refleje automáticamente en todas las Vistas. (Gonzales & Fernández Romero, 2012)

El enfoque arquitectónico de MVC presenta varias ventajas notables, como una clara separación entre los componentes del programa, una Interfaz de Programación de Aplicaciones (API) bien definida que facilita la implementación por separado, y una conexión dinámica entre el Modelo y sus Vistas en tiempo de ejecución. Al incorporar el MVC en un diseño, las piezas del programa pueden construirse de manera independiente y luego ensamblarse durante la ejecución. Esto proporciona flexibilidad, ya que, si un componente no funciona correctamente, puede ser reemplazado sin afectar las otras partes (Romero & González, 2012).

Este enfoque contrasta con la aproximación monolítica común en muchos programas de menor complejidad, donde todos los elementos están contenidos en un único marco (Frame) con un controlador de eventos, cálculos y presentación del resultado. En este contexto, realizar cambios puede resultar no trivial, mientras que el MVC permite una mayor modularidad y mantenimiento eficiente.

**MVC en Java Swing**

Java Swing, la biblioteca de interfaces gráficas de usuario en el SDK5 de Java, es conocida por su portabilidad, simplicidad y amplio conjunto de librerías. Algunas de sus características notables incluyen la implementación de diversos componentes visuales como botones, campos de texto, tablas y barras de menús (Pantoja, 2004).

A diferencia de AWT, otra biblioteca de interfaces de usuario en Java, Swing ofrece componentes independientes de la plataforma, lo que garantiza la portabilidad de las aplicaciones en distintos entornos operativos. Además, permite la flexibilidad de cambiar entre diferentes estilos de interfaz de usuario, conocidos como "look and feels", en tiempo de ejecución. Esto posibilita que la misma aplicación tenga la apariencia de una aplicación Windows o Motif simplemente alterando el "look and feel" (Pantoja, 2004).

Swing cumple con el diseño de JavaBeans, lo que facilita la utilización de sus componentes en entornos de desarrollo integrados (IDEs) como JBuilder, Sun Forte for Java o Eclipse (Pantoja, 2004).

Una característica distintiva de Swing es su arquitectura profundamente basada en el patrón Modelo/Vista/Controlador (MVC), lo que brinda un alto grado de extensibilidad y personalización de los componentes de la biblioteca. Este enfoque facilita la modularidad y la adaptabilidad de las aplicaciones desarrolladas con Java Swing (Pantoja, 2004).

•

**Netbeans IDE**

La herramienta de desarrollo que se eligió para el uso del entorno de desarrollo NetBeans IDE. Este IDE, de código abierto bajo la licencia GPL, destaca por su capacidad para desarrollar aplicaciones con características modulares. Entre sus funcionalidades se incluyen la escritura, depuración, compilación y ejecución de programas, además de ofrecer soporte para diversos lenguajes de programación. Se destacan sus características como el autocompletado de código, herramientas para el diseño de interfaces gráficas de usuario (GUI) y la integración de marcos de trabajo para el desarrollo. NetBeans IDE es un software multiplataforma y extensible, con documentación detallada sobre las clases que conforman su API (Interfaz de Programación de Aplicaciones) (Gonzáles Mártinez & Trujillo O., 2016).

Este entorno de desarrollo se basa en la Plataforma NetBeans y se beneficia de las ventajas de un cliente enriquecido. La mayoría de las aplicaciones construidas en este entorno comparten características visuales a través de menús, barras de herramientas, barras de

progreso y otros componentes, lo que facilita el desarrollo de interfaces de usuario. La plataforma ofrece diversos elementos que simplifican la construcción de extensiones y aplicaciones con arquitectura modular. Estos incluyen la internacionalización, un editor de datos para agregar funcionalidades, la personalización de elementos de pantalla y un generador de ayuda para proyectos que se desarrolla en colaboración con APIs y componentes reutilizables (Sheyla & Gonzáles, 2013).

## Diseño de interfaz

# Análisis Económico

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Ítem | Descripción | Valor | Cantidad | Total |
|  |  |  |  |  |  |
| 1 | Tiempo de desarrollo. | Programadores | $2,000,000 | 3 | $6,000,000 | |
| 2 | Costo de herramientas. | NetBeans IDE | $0 | 1 | $0 | |
| 3 | Costo de herramientas. | PostgresSQL | $0 | 1 | $0 | |
| 4 | Costo de herramientas. | Acer Nitro 5 | $4,000,000 | 1 | $4,000,000 | |
|  |  |  |  |  |  | |
|  |  |  |  | Total: | $10,000,000 | |

# Conclusiones

Se ha desarrollado con éxito la aplicación FoodView, diseñada para la gestión y compartición de experiencias culinarias gastronómicas. La plataforma proporciona a los usuarios una herramienta completa y accesible para registrar, evaluar y compartir sus experiencias en distintos establecimientos gastronómicos, cumpliendo con el objetivo principal del proyecto.

Se ha llevado a cabo un análisis exhaustivo de los conceptos teóricos fundamentales relacionados con bases de datos. Este análisis ha demostrado ser crucial para evaluar la aplicabilidad y los beneficios de estos conceptos en el diseño y desarrollo de FoodView, asegurando una base teórica sólida que ha guiado todo el proceso de desarrollo.

Se ha desarrollado una base de datos eficiente y robusta utilizando PostgreSQL, específicamente a través de la herramienta pgAdmin. Esta base de datos ha sido diseñada para soportar todas las funcionalidades necesarias de la aplicación, garantizando un almacenamiento y gestión de datos eficaz y seguro.

La conexión entre la aplicación y el entorno de la base de datos en PostgreSQL se ha realizado con éxito. Esta integración ha permitido que FoodView interactúe de manera efectiva con la base de datos, facilitando la gestión y compartición de las experiencias culinarias de los usuarios y asegurando un rendimiento óptimo de la aplicación.

# Referencias

Colombia, C. d. (17 de Octubre de 2012). *Colombia Potencia de la Vida*. Obtenido de https://www.minambiente.gov.co/politica-de-proteccion-de-datos-personales/#:~:text=Ley%20de%20Protecci%C3%B3n%20de%20Datos,de%20naturaleza%20p%C3%BAblica%20o%20privada.

Colombia, C. d. (17 de Octubre de 2012). *Colombia Potencial de la Vida*. Obtenido de https://www.minambiente.gov.co/politica-de-proteccion-de-datos-personales/#:~:text=Ley%20de%20Protecci%C3%B3n%20de%20Datos,de%20naturaleza%20p%C3%BAblica%20o%20privada.

Colombia, C. d. (22 de Octubre de 2021). *Secretaría Jurídica Distrital*.

*Digital Guide IONOS*. (17 de Junio de 2020). Obtenido de Digital Guide IONOS: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/

Gonzales, Y. D., & Fernández Romero, Y. (12 de Junio de 2012). *Telematica*. Obtenido de https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/15

Guru, R. (2014-2024). *Refactoring.Guru*. Obtenido de https://refactoring.guru/es/design-patterns/structural-patterns

S., M. (2022). *IBM*. Obtenido de IBM: https://www.ibm.com/mx-es/topics/postgresql

Sarría, F. A. (s.f.). *Universidad de Murcia*. Obtenido de Universidad de Murcia: https://www.um.es/geograf/sigmur/sigpdf/postgresql.pdf