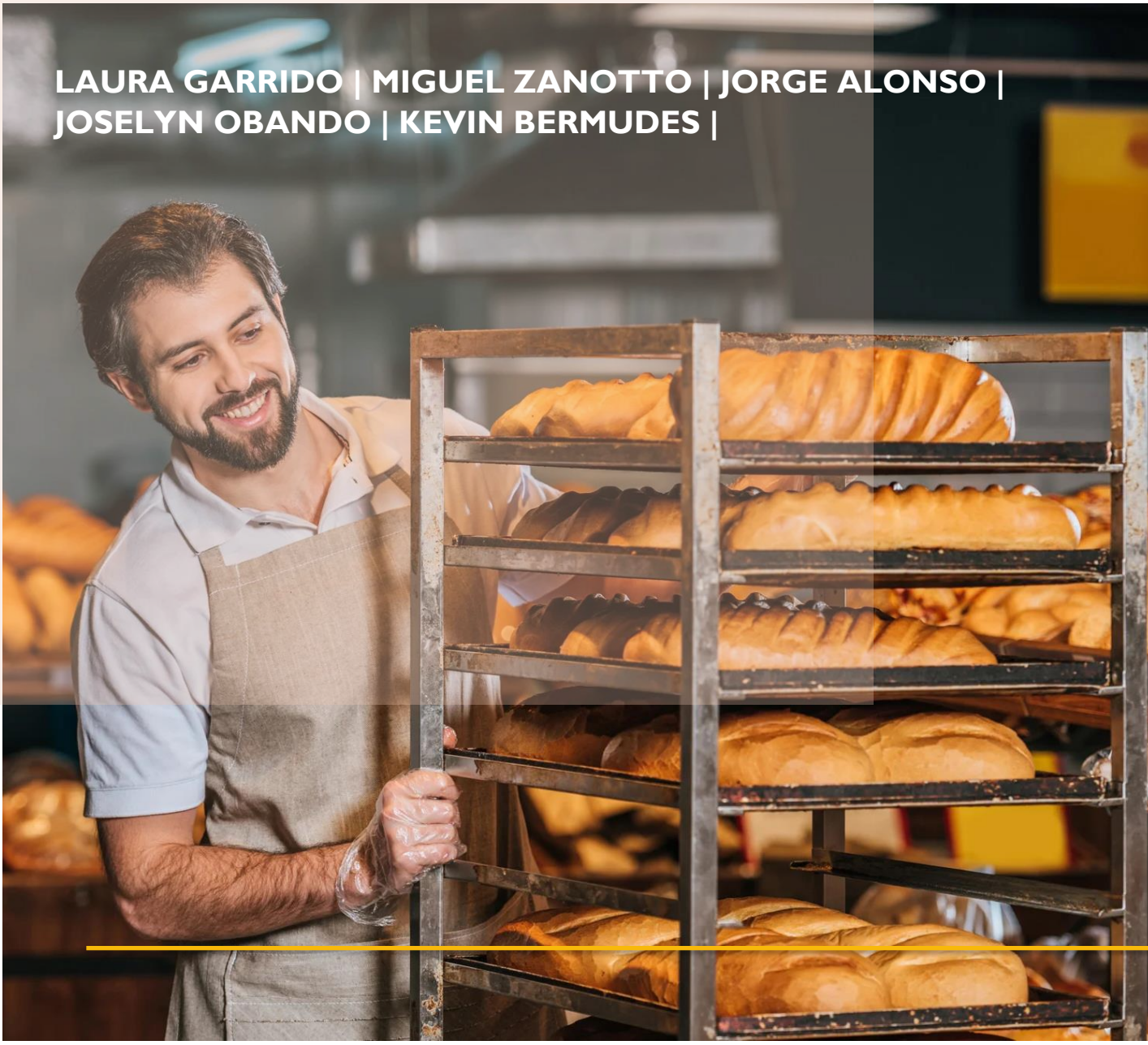
MIGA DE ORO

PROYECTO PANADERIA 2°DAW

LAURA GARRIDO | MIGUEL ZANOTTO | JORGE ALONSO | JOSELYN OBANDO | KEVIN BERMUDES |



RESUMEN

En la actualidad, las empresas necesitan una plataforma en línea para que sus clientes o proveedores conozcan sus productos y su marca. Es de gran importancia promover la empresa de manera online.

En la mayoría de los casos, es necesaria una actualización periódica de la base de datos de una empresa. Resulta de gran utilidad la existencia de un sistema o plataforma que facilite la gestión de estos datos por parte del personal no informático de las empresas e instituciones.

En el presente, Miga de Oro implementa y desarrolla una API Restful y una plataforma web para la gestión de los datos con los que trabajaría una panadería. La plataforma web, a través de una sencilla interfaz, pretende facilitar el manejo de los datos por parte del personal de la Sección de Información y Atención al Cliente y Proveedor de la empresa, de manera que sea posible añadir y modificar productos y categorías de manera sencilla.

Por otro lado, con la implementación de la API, se pretende alcanzar un mayor nivel de abstracción en el funcionamiento de la aplicación, pudiendo ser utilizada tanto por la plataforma web como de manera más técnica.

TABLA DE CONTENIDO

Introducción:
Contextulización 1
Objetivos 2
Motivación 3
Conceptos
Api restful y su mundo 4
Arquitectura actual 5
Tecnologías para el almacenamiento de datos 6
Justificación de las tecnologías elegidas 7
Validación
Validación de la Api Rest con postman 8
Validación de api rest con swagger 9
Gestion del proyecto
Resolución de incidencias en Git 10
API REST
Principales endpoints 11
Presupuesto 21

INTRODUCCIÓN:

CONTEXTULIZACIÓN

En la actualidad, la presencia de la tecnología ha transformado la vida cotidiana del día a día de las personas de manera significativa. Este cambio se ha visto aún más acelerado por la pandemia de COVID-19, que ha forzado a la sociedad y a las empresas a adoptar soluciones digitales de manera más rápida y extensa. En este contexto, las API RESTful desempeñan un papel crucial al facilitar la conectividad y la eficiencia en el desarrollo de plataformas web.

Dando que nuestro proyecto se centra en la creación de una plataforma web dedicada a una panadería, con el objetivo de proporcionar a proveedores y clientes un acceso sencillo y eficiente a la información sobre los productos disponibles de manera intuitiva. La digitalización de procesos comerciales, especialmente en el sector de alimentos y servicios, ha creado la necesidad de soluciones tecnológicas que mejoren la accesibilidad y la experiencia del usuario.

Objetivos del Proyecto:

- Desarrollar un backend sólido y seguro para la plataforma web.
- Facilitar para los proveedores y clientes la obtención de información sobre productos de manera intuitiva.
- Integrar tecnologías clave para garantizar la eficiencia y la seguridad del sistema.

Nuestro enfoque sobre el proyecto es estar en línea con la evolución actual del mercado y las expectativas de los usuarios en la era digital. Dando que tecnologías avanzadas, como las API RESTful, asegurarán la funcionalidad y seguridad robustas de la plataforma, contribuyendo así al éxito de la digitalización de una empresa.

OBJETIVOS

- *Mejora de la Gestión de Datos*: El objetivo principal de este proyecto es optimizar la gestión de los datos relacionados con los productos, categorías y clientes de nuestra panadería. La implementación de la API RESTful y la plataforma web proporcionará una forma eficiente de almacenar, actualizar y acceder a esta información de manera centralizada y segura.

- *Facilitar la Interacción con los Clientes y Proveedores*: La plataforma web ofrecerá una interfaz intuitiva que permitirá al personal de la Sección de Información y Atención al Cliente y Proveedor de nuestra empresa añadir y modificar productos y categorías de manera simple. Esta mejora en la interacción contribuirá a una experiencia más fluida y satisfactoria para nuestros clientes y socios comerciales.

- *Mayor Nivel de Abstracción y Flexibilidad*: La implementación de la API RESTful permitirá un mayor nivel de abstracción en el funcionamiento de la aplicación, lo que permitirá su uso tanto en la plataforma web de manera más técnica. Esto brindará flexibilidad para adaptarse a futuras necesidades y escenarios tecnológicos.

Beneficios Clave:

- *Eficiencia Operativa*: Al centralizar la gestión de datos a través de la API RESTful y la plataforma web, esperamos lograr una mejora significativa en la eficiencia de nuestra panadería. La automatización de procesos y la disponibilidad de información actualizada en tiempo real contribuirán a optimizar nuestras operaciones diarias.

- *Mejora en la Experiencia del Cliente*: La facilidad de uso y la capacidad de actualización ágil que ofrecerá la plataforma web tendrán un impacto directo en la experiencia del cliente. La disponibilidad de información precisa y actualizada sobre nuestros productos fortalecerán las relaciones con nuestros clientes y proveedores.

- *Adaptabilidad Tecnológica*: La implementación de una API RESTful y una plataforma web nos posicionará estratégicamente adaptarnos a futuras innovaciones tecnológicas. Permitiendo mantenernos actualizados y competitivos en un entorno empresarial en constante evolución.

MOTIVACIÓN

La motivación detrás de este proyecto radica en la necesidad de modernizar y optimizar la gestión de datos en el entorno de una panadería, (siendo estos el comercio con mayor necesidad en nuestro día a día). Reconociendo la importancia de mantener actualizada la base de datos, así como la necesidad de simplificar el manejo de la información para el personal no informático, surge la iniciativa de desarrollar una solución integral que cumpla con estos requisitos.

La API RESTful desarrolladas ofrece un alto nivel de abstracción en el funcionamiento de la aplicación, lo que permite su utilización tanto por la plataforma web como de manera más técnica. Esto garantiza una mayor flexibilidad y escalabilidad en la gestión de datos, lo que resulta crucial en un entorno empresarial dinámico.

Por otro lado, la plataforma web proporciona una interfaz sencilla y amigable que facilita la incorporación y modificación de productos y categorías. Esta herramienta está diseñada para ser intuitiva, lo que permite al personal de la Sección de Información y Proveedor de la panadería manejar eficientemente la información relevante.

Con la implementación de este proyecto, se espera una mejora significativa en la eficiencia operativa de la panadería, así como una mayor satisfacción tanto de los clientes como de los proveedores. La capacidad de gestionar de forma ágil y precisa la información relacionada con los productos y categorías se traducirá en una mejor experiencia para todos los involucrados.

En resumen, la implementación de la API RESTful y la plataforma web para la gestión de datos en la panadería representa un paso significativo hacia la modernización y optimización de los procesos internos. Estamos entusiasmados con el potencial que esta solución tiene para mejorar la eficiencia y la experiencia general de nuestros clientes y proveedores.

CONCEPTOS

API RESTFUL Y SU MUNDO

En el mundo actual, la integración de tecnologías digitales se ha convertido en un aspecto fundamental para el crecimiento y la eficiencia de las empresas. En el contexto de una panadería, la implementación de una API RESTful y su integración en la plataforma web de la empresa puede desempeñar un papel crucial en la optimización de los procesos, la mejora de la experiencia del cliente y el impulso de la presencia en línea.

¿Qué es una API RESTful? Una API RESTful es una interfaz de programación de aplicaciones que sigue los principios de la arquitectura REST (Transferencia de Estado Representacional). Permite la comunicación entre sistemas y la transferencia de datos de manera eficiente a través de internet. Su estructura flexible y su capacidad para trabajar con diferentes tipos de datos la convierten en una herramienta versátil y poderosa para la integración de sistemas.

Beneficios de Integrar una API RESTful en una Panadería:

- Automatización de procesos: La integración de una API RESTful puede automatizar tareas como la gestión de inventario, la toma de pedidos en línea y la generación de informes, lo que libera tiempo y recursos para enfocarse en otras áreas del negocio.
- Mejora de la experiencia del cliente: Al permitir la conexión entre la plataforma web y otros sistemas, se facilita la personalización de la experiencia del cliente, la implementación de programas de fidelización y la gestión eficiente de la información de los clientes.
- Optimización de la cadena de suministro: La API RESTful puede facilitar la integración con proveedores, la gestión de pedidos de materias primas y la logística, lo que contribuye a una cadena de suministro más eficiente y rentable.
- Implementación de una API RESTful en la Plataforma Web: La integración de una API RESTful en la plataforma web de una panadería requiere un enfoque estratégico y cuidadoso. Es fundamental considerar aspectos como la seguridad de los datos (de los cuales ya están implementados), la compatibilidad con sistemas existentes y la escalabilidad a largo plazo. Además, se debe garantizar una comunicación clara entre los equipos de desarrollo y los responsables del negocio para alinear los objetivos y garantizar una implementación exitosa.

ARQUITECTURA ACTUAL

Utilizáremos la arquitectura de la Arquitectura RESTFull. Del cual se divide de la siguiente manera:

Capa de presentación: Esta capa será la interfaz a través de la cual los clientes y otros sistemas interactuarán con nuestra API. Estará diseñada para ser amigable y fácil de usar, siguiendo los principios de diseño centrados en el usuario.

Capa de lógica de negocio: Aquí residirá la lógica que gobierna el comportamiento de la API. Nos aseguraremos de que las operaciones realizadas a través de la API cumplan con las reglas del negocio de la panadería, manteniendo la coherencia y la integridad de los datos.

Capa de acceso a datos: Para interactuar con la base de datos, implementaremos esta capa que gestionará todas las operaciones de lectura y escritura. Se utilizarán prácticas de seguridad sólidas para proteger la integridad de los datos.

Seguridad: La seguridad es una prioridad en nuestra arquitectura. Implementaremos medidas de autenticación y autorización robustas para proteger la API contra accesos no autorizados y ataques maliciosos.

Frontend: El frontend de la plataforma web estará basado en tecnologías modernas que permitirán una experiencia de usuario fluida e intuitiva. Nos enfocaremos en la usabilidad y la accesibilidad para garantizar la satisfacción de los clientes.

Backend: En el backend, implementaremos un sistema escalable y eficiente que se integrará sin problemas con la API RESTful. Utilizaremos buenas prácticas de desarrollo para garantizar un alto rendimiento y una fácil mantenibilidad.

Base de datos: La plataforma web se apoyará en una base de datos robusta y confiable que respaldará todas las operaciones realizadas a través de la interfaz web. Nos aseguraremos de optimizar las consultas y mantener la integridad de los datos con MongoDB y H2.

TECNOLOGÍAS PARA EL ALMACENAMIENTO DE DATOS

Los datos necesarios para el funcionamiento de una API RESTful serán gestionados desde la interfaz de la aplicación web. Para esto, se requerirán tecnologías que faciliten el almacenamiento de datos.

Como se mencionó, estos datos se componen de la información que se reciba dentro de nuestra aplicación web. En el sistema, que partirá mayoritariamente de estos datos, se solía guardar la información en archivos CSV. Sin embargo, debido a la nueva arquitectura implementada, necesitamos hacer uso de sistemas de gestión de bases de datos más confiables y fácilmente actualizables.

Se exponen dos grandes grupos de sistemas gestores de bases de datos: los relacionales (SQL) y los NoSQL. Estos ofrecerán el desarrollo de la arquitectura de una API RESTful como la nuestra de manera confiable y intuitiva.

BASE DE DATOS RELACIONALES

Las bases de datos relacionales son una parte fundamental de la infraestructura tecnológica de nuestro proyecto. Estas bases de datos se basan en el modelo relacional, que organiza los datos en tablas con filas y columnas interrelacionadas. Este enfoque proporciona una estructura sólida y coherente para almacenar y administrar la información de nuestra plataforma.

Dentro del amplio espectro de sistemas de bases de datos relacionales, nos centraremos en dos en particular: H2 y MongoDB. El sistema H2 es una base de datos relacional escrita en Java, que ofrece un rendimiento excepcional y una integración perfecta con aplicaciones Java. Por otro lado, MongoDB es una base de datos NoSQL que se destaca por su flexibilidad y capacidad para manejar grandes volúmenes de datos no estructurados.

Ahora, al considerar la aplicación de estos sistemas en nuestra API RESTful y plataforma web de panadería, es crucial reconocer su papel en el almacenamiento y recuperación eficiente de datos relacionados con pedidos, inventario, información de clientes y otros aspectos fundamentales de la operación de la panadería.

La integración de H2 y MongoDB en nuestra arquitectura de base de datos nos permitirá garantizar la seguridad, consistencia y escalabilidad necesarias para respaldar nuestras operaciones. Además, estas bases de datos nos brindarán la flexibilidad para adaptarnos a futuros requisitos y expansiones de la plataforma.

JUSTIFICACIÓN DE LAS TECNOLOGÍAS ELEGIDAS

Base de datos MongoDB y H2

Elegimos MongoDB por su capacidad para manejar grandes volúmenes de datos no estructurados, lo cual es especialmente relevante en un entorno dinámico como el de una panadería en línea. Su flexibilidad y escalabilidad nos permitirán adaptarnos a las necesidades cambiantes del negocio y de los clientes.

Por otro lado, elegimos H2 como base de datos de desarrollo debido a su ligereza y facilidad de configuración, lo que acelerará el proceso de desarrollo y pruebas.
Dando que todas estas bases lo realizaremos en memoria, tanto por H2 y Mongo DB, debido a que funcione como una base de datos perfecta con aplicaciones en Spring Boot para realizar pruebas, por tener mayor seguridad, fiabilidad, y el uso de caché para mejorar el rendimiento de las operaciones de lectura.

API

Elegimos una arquitectura RESTful porque nos permite una comunicación eficiente y escalable entre el frontend y la base de datos. Esto garantizará tiempos de respuesta óptimos y una experiencia de usuario fluida, lo cual es esencial en un entorno de comercio electrónico.

Vamos a desarrollar la API utilizando tecnologías modernas y robustas, con un enfoque en la seguridad y la modularidad para facilitar su mantenimiento y futuras expansiones.

Frontend

Vamos a desarrollar el frontend utilizando tecnologías como HTML, CSS y JavaScript, poniendo énfasis en la usabilidad, la accesibilidad y la compatibilidad con múltiples dispositivos. Buscamos una interfaz intuitiva y atractiva que refleje la identidad de la panadería y mejore la experiencia del usuario.

VALIDACIÓN

VALIDACIÓN DE LA API REST CON POSTMAN

Comenzando por la validación de los endpoints de nuestra API, hemos utilizado Postman para enviar solicitudes a cada uno de ellos y verificar las respuestas obtenidas. Esto nos ha permitido confirmar que los endpoints funcionan según lo esperado y que devuelven los datos correctos.

Validación de Categorías

Uno de los aspectos principales de nuestra plataforma es la gestión de categorías de productos. Hemos utilizado Postman para validar que las operaciones relacionadas con las categorías, como la creación, modificación y eliminación, se llevan a cabo de manera precisa y consistente.

Validación de Clientes

La interacción con los clientes es fundamental para nuestro negocio. A través de Postman, hemos validado que las operaciones de gestión de clientes, incluyendo el registro, actualización de datos y consulta, se ejecutan sin problemas y ofrecen una experiencia óptima.

Validación de Personal

El manejo eficiente del personal es esencial para el funcionamiento fluido de nuestra panadería. Con Postman, hemos verificado que las operaciones relacionadas con el personal, como la asignación de roles y la actualización de información, se realizan de manera eficaz.

Validación de Productos

La correcta gestión de los productos es un aspecto fundamental de nuestra plataforma. A través de Postman, hemos validado las operaciones de creación, actualización y eliminación de productos, garantizando su integridad y consistencia.

Validación de Proveedores

Por último, pero no menos importante, hemos utilizado Postman para validar las operaciones relacionadas con los proveedores. Esto incluye la creación de nuevos proveedores, la actualización de información y la verificación de la integración con otros aspectos del sistema.

Y lo más importante los websockets. Del cual también comprobamos dentro del postman. Siendo el más importante para mostrar al cliente una notificación por cada acción que realice en el programa.

VALIDACIÓN DE API REST CON SWAGGER

Los endpoints de nuestra API representan los diferentes puntos de acceso a los recursos y funcionalidades que ofrece nuestra plataforma. Gracias a Swagger, podemos validar cada uno de estos puntos exhaustivamente, basicamente realicando los mismos pasos de postman, asegurándonos de que cumplan con los estándares definidos y ofrezcan la funcionalidad esperada.

Categorías, Clientes, Personal, Productos y Proveedores

Cada una de estas áreas es crucial para el funcionamiento integral de nuestra plataforma. Con Swagger, podemos validar los diversos aspectos relacionados con estas entidades, desde la creación y modificación de categorías, hasta la gestión de clientes, personal, productos y proveedores. La validación exhaustiva de estos elementos es esencial para garantizar la estabilidad y seguridad de nuestra API.

Beneficios de la Validación con Swagger

La implementación de la validación de la API con Swagger conlleva una serie de beneficios significativos, como:

- Mejora la calidad y consistencia de la API.
- Facilita la comprensión y adopción de las pruebas hacia la API.
- Proporciona una documentación clara y detallada de la API.
- Permite realizar pruebas automatizadas de manera efectiva.

GESTION DEL PROYECTO

RESOLUCIÓN DE INCIDENCIAS EN GIT

La gestión de problemas en Git es fundamental en el desarrollo de software, ya que nos permite identificar, documentar y solucionar los problemas que surgen durante el proceso de desarrollo. En nuestro proyecto, es crucial gestionar los problemas en la API RESTful y la plataforma web de panadería para garantizar la estabilidad, calidad y evolución continua de nuestra aplicación.

Proceso de resolver problemas en Git

- *Identificación*: El primer paso para resolver problemas es identificarlos y documentarlos correctamente. Esto implica usar herramientas específicas de seguimiento de problemas en Git con Github y describir detalladamente cada problema que encontramos.

- *Priorización*: Una vez identificados, es muy importante priorizar los problemas según su impacto en el proyecto. Esto nos permite asignar los recursos de manera efectiva y abordar primero los problemas que afectan más gravemente el funcionamiento de la API RESTful y la plataforma web de panadería.

- *Asignación y seguimiento*: Cada problema debe ser asignado a un miembro del equipo responsable de resolverlo. Seguir de cerca el estado de los problemas garantiza que avancemos de manera proactiva en su resolución, evitando retrasos o malentendidos.

- *Resolución y validación*: Una vez que hemos abordado un problema, es muy importante hacer pruebas exhaustivas para validar la solución propuesta. Esto nos asegura que la solución sea efectiva y no cause problemas no deseados en otras partes del proyecto.

Eso junto con una reunión semanal sobre el progreso o las incidencias que surjan. Siendo de máxima importancia de las reuniones en nuestro proyecto.

API REST

PRINCIPALES ENDPOINTS

CATEGORIA:

Esta endpoint representa una categoría de productos. Cada categoría tiene un nombre (como "Pan Dulce" o "Pasteles"), un número único de identificación, y se registra automáticamente la fecha en que fue creada y la fecha de la última actualización. Además, hay un indicador que dice si la categoría está activa o no. Siendo una forma de organizar y seguir la información sobre las diferentes categorías de productos en la panadería.

EJEMPLO USO EN POSTMAN:

Añadiendo el cuerpo JSON en postman se podrá realizar las siguientes operaciones:

```
{
  "nameCategory":"Pan",
  "isActive":true
}
```

Junto con el enlace: http://localhost:8080/categoria

[No es necesario el cuerpo json en los casos de obtener categoría por id, obtener todas las categorías y eliminar categorías]

- Crear una nueva categoría (POST)
- Obtener todas las categorías (GET)
- Obtener una categoría por id (GET). Por ejemplo, si queremos ver la categoría 10 tendríamos que consultar este enlace: http://localhost:8080/categoria/10
- Actualizar una categoría por id (PUT). Siendo solo necesario el id de la categoría que queremos actualizar, en nuestro caso el 2, con un nuevo nombre. Teniendo que entrar al siguiente enlace: http://localhost:8080/categoria/2

  Y en nuestro cuerpo de json indicar el nombre nuevo que queremos, por ejemplo, Bebidas.

  ```
  {
    "nameCategory":"Bebidas"
  }
  ```

- Actualizar si una Categoría esta Activa o no(PUT). Por ejemplo, queremos cambiar esto en la primera categoría. Yendo al enlace: http://localhost:8080/categoria/1

Y cambiando el cuerpo de la manera siguiente:

```
{

  "isActive":false
}
```

- Eliminar Categoría por id (DELETE). Por ejemplo, si queremos borrar la categoría 13 tendremos que ir al enlace: http://localhost:8080/categoria/13

ERRORES COMUNES:

- **Error 404:** Primero asegúrate de que la URL sea correcta. Segundo, comprueba que no estes buscando una id que no existe en la base de datos tanto para borrar, actualizar u obtener datos de la categoría.
- **Error 400:** Verifica si estas proporcionando campos validos en el cuerpo JSON y cumples el formato establecido en los atributos del modelo de Categoría.
- **Error 409:** Este error se debe que existe ya una categoría con el nombre proporcionado. Cambie solo el nombre de la Categoría para solucionar este error o busque y actualice el nombre de la Categoría que se haya duplicado.

CLIENTE:

El endpoint Cliente representa la información de un cliente en nuestro sistema API RESTFULL. Contiene atributos como id, nombreCompleto, correo, dni, telefono, imagen, fechaCreacion, fechaActualizacion, isActive, y categoria. Cada atributo almacena detalles específicos sobre el cliente, como su identificación, nombre, información de contacto y más.

EJEMPLO USO EN POSTMAN:

Añadiendo el cuerpo JSON en postman se podrá realizar las siguientes operaciones:

```
{
   "nombreCompleto":"Jorge Alonso Cruz Vera",
   "correo":"asfaf@hotmail.com",
   "dni":"12345678A",
   "telefono":"987654321",
   "categoria":"VIP"
}
```

Junto con el enlace: http://localhost:8080/cliente

[No es necesario el cuerpo json en los casos de obtener cliente por id, obtener todos los clientes y eliminar clientes]

- Crear un nuevo cliente (POST)
- Obtener todos los clientes (GET)
- Obtener un cliente por id (GET). Por ejemplo si queremos ver el primer cliente tendríamos que consultar este enlace: http://localhost:8080/cliente/1
- Actualizar un cliente por id (PUT). Siendo solo necesario el id del cliente que queremos actualizar, en nuestro caso el 1, con un nuevo nombre. Teniendo que entrar al siguiente enlace: http://localhost:8080/cliente/1

  Y en nuestro cuerpo de json indicar el nombre nuevo que queremos, por ejemplo Luis.

  ```
  {
     "nombreCompleto":"Luis Alberto Lozano"
  }
  ```

- Actualizar si un cliente esta Activo o no(PUT). Por ejemplo queremos cambiar esto en el primer cliente. Yendo al enlace: http://localhost:8080/cliente/1

Y cambiando el cuerpo de la manera siguiente:

```
{

  "isActive":false
}
```

- Eliminar Cliente por id (DELETE). Por ejemplo, si queremos borrar el primer cliente tendremos que ir al enlace: http://localhost:8080/cliente/1

ERRORES COMUNES:

- **Error 404:** Primero asegúrate de que la URL sea correcta. Segundo, comprueba que no estes buscando una id que no existe en la base de datos tanto para borrar, actualizar u obtener datos del cliente.
- **Error 400:** Verifica si estas proporcionando campos validos en el cuerpo JSON y cumples el formato establecido en los atributos del modelo de Cliente.
- **Error 409:** Este error se debe que existe ya un cliente con el DNI proporcionado o imagen. Compruebe mediante GET cuál es el repetido y decidir si borrar o actualizar el DNI o la imagen.

CABE RECALCAR:

Nuestra dirección de cliente lo guardamos de tipo de String por las siguientes razones:

Ventajas: es más fácil y rápido de implementar, no requiere crear una tabla adicional en la base de datos, y puede ser útil si la dirección no se utiliza para búsquedas o consultas complejas.

Desventajas: puede ser complicado o ineficiente realizar búsquedas o consultas basadas en la dirección, y pueden surgir problemas de integridad si la estructura del JSON cambia o si hay inconsistencias en los datos.

Porque si lo guardamos en una tabla aparte tenemos los siguientes datos a seguir:

Ventajas: facilita consultas complejas o búsquedas específicas basadas en direcciones, asegura la coherencia y consistencia de los datos de dirección, y puede utilizarse para múltiples clientes sin duplicar información.

Desventajas: requiere crear una tabla adicional y manejar la relación entre el cliente y su dirección, y *podría llevar más tiempo implementarlo*.

Ya teniendo estas ventajas y desventajas en cuenta, hemos quedado en un acuerdo al utilizar String. Ya que será más rápido de implementar y técnicamente no lo utilizamos para búsquedas complejas.

PERSONAL:

Este endpoint modela la información de los empleados de una panadería, incluyendo su nombre, número de identificación, fechas relevantes y la sección en la que trabajan. Los campos como fechaCreacion, fechaActualizacion, e isActive son comunes para el seguimiento de auditoría y la gestión de la activación/desactivación de registros.

EJEMPLO USO EN POSTMAN:

Añadiendo el cuerpo JSON en postman se podrá realizar las siguientes operaciones:


```
{
    "dni": "03488998J",
    "nombre": "Kevin Bermudez",
    "seccion": "Repostería",
    "fechaAlta": "2023-11-16",
    "isActive": true
}
```


Junto con el enlace: http://localhost:8080/personal

[No es necesario el cuerpo json en los casos de obtener personal por id, obtener todos los personales y eliminar personal]

- Crear un nuevo personal (POST)
- Obtener todos los personales (GET)
- Obtener un personal por UUID (GET). Por ejemplo, si queremos ver el primer personal tendríamos que consultar este enlace: http://localhost:8080/personal/1
- Actualizar personal por UUID (PUT). Siendo solo necesario el UUID del personal que queremos actualizar, en nuestro caso el 1, con un nuevo nombre. Teniendo que entrar al siguiente enlace: http://localhost:8080/personal/1

  Y en nuestro cuerpo de json indicar el nombre nuevo que queremos, por ejemplo, Gary.


  ```
  {
    "nombre":"Gary Cordova"
  }
  ```

- Actualizar si un personal está Activo o no(PUT). Por ejemplo, queremos cambiar esto en el primer personal. Yendo al enlace: http://localhost:8080/personal/65d43d29-e9a7-44a6-98c2-fc21f41074d9

  Y cambiando el cuerpo de la manera siguiente:


  ```
  {
  ```


```
    "isActive":false
}
```

- Eliminar Personal por UUID (DELETE). Por ejemplo si queremos borrar el primer personal tendremos que ir al enlace: http://localhost:8080/personal/1

ERRORES COMUNES:

- **Error 404:** Primero asegúrate de que la URL sea correcta. Segundo, comprueba que no estes buscando una id que no existe en la base de datos tanto para borrar, actualizar u obtener datos del personal.
- **Error 400:** Verifica si estas proporcionando campos validos en el cuerpo JSON y cumples el formato establecido en los atributos del modelo de Personal.
- **Error 409:** Este error se debe que existe ya un Personal con el DNI proporcionado o con la misma imagen. Compruebe mediante GET cuál es el repetido y decidir si borrar o actualizar el DNI o imagen.

PRODUCTOS:

La clase Producto representa un artículo que una panadería vende. Cada producto tiene un nombre, cantidad en stock, precio, fecha de creación y actualización, una imagen (por defecto, si no se proporciona), y un indicador de si está activo o no.

Además, cada producto pertenece a una categoría específica (como "pan" o "pasteles") y es suministrado por un proveedor.

EJEMPLO USO EN POSTMAN:

Añadiendo el cuerpo JSON en postman se podrá realizar las siguientes operaciones:

```
{
   "nombre":"Pan Bimbo",
   "stock":100,
   "precio":1.87,
   "categoria":"Pan",
   "proveedor":"12345678A"
}
```

Junto con el enlace: http://localhost:8080/producto

[No es necesario el cuerpo json en los casos de obtener producto por id, obtener todos los productos y eliminar producto]

- Crear un nuevo producto (POST)
- Obtener todos los productos (GET)
- Obtener un producto por UUID (GET). Por ejemplo si queremos ver el primer producto tendríamos que consultar este enlace: http://localhost:8080/producto/550e8400-e29b-41d4-a716-446655440000
- Actualizar producto por UUID (PUT). Siendo solo necesario el UUID del producto que queremos actualizar junto con un nuevo nombre. Teniendo que entrar al siguiente enlace: http://localhost:8080/producto/a2109af4-c6e2-47ae-8f40-737226e9f25c

  Y en nuestro cuerpo de json indicar el nombre nuevo que queremos, por ejemplo, Pan Bimbolete.

  ```
  {
     "nombre":"Pan Bimbolete"
  }
  ```
- Actualizar si un producto está Activo o no(PUT). Por ejemplo, queremos cambiar esto en el primer producto. Tendremos que ir al enlace: http://localhost:8080/producto/adcbbf52-ee28-4dcb-a138-7aa2fc25aa72

Y cambiando el cuerpo de la manera siguiente:

```
{

  "isActive":false
}
```

- Eliminar Producto por UUID (DELETE). Por ejemplo si queremos borrar el primer producto tendremos que ir al enlace: http://localhost:8080/producto/a2109af4-c6e2-47ae-8f40-737226e9f25c

ERRORES COMUNES:

- **Error 404:** Primero asegúrate de que la URL sea correcta. Segundo, comprueba que no estes buscando una UUID que no existe en la base de datos tanto para borrar, actualizar o obtener datos del producto.
- **Error 400:** Verifica si estas proporcionando campos validos en el cuerpo JSON y cumples el formato establecido en los atributos del modelo de Producto.
- **Error 409:** Este error se debe que existe ya un Producto con el nombre proporcionado . Compruebe mediante GET cual es el repetido y decidir si borrar o actualizar el nombre.

PROVEEDORES:

La clase Proveedor guarda información sobre un proveedor de la panadería, como su nombre, número de identificación fiscal (NIF), número único, tipo, y si está activo. También registra cuándo se creó y actualizó por última vez.

EJEMPLO USO EN POSTMAN:

Añadiendo el cuerpo JSON en postman se podrá realizar las siguientes operaciones:


```
{
  "nif":"12451213B",
  "tipo":"Distribuidor",
  "numero":"981241531",
  "nombre":"Martin Guerro"
}
```


Junto con el enlace: http://localhost:8080/proveedores

[No es necesario el cuerpo json en los casos de obtener un proveedor por id, obtener todos los proveedores y eliminar un proveedor]

- Crear un nuevo proveedor (POST)
- Obtener todos los proveedores (GET)
- Obtener un proveedor por id (GET). Por ejemplo, si queremos ver el primer proveedor tendríamos que consultar este enlace: http://localhost:8080/proveedores/1
- Actualizar proveedor por id (PUT). Siendo solo necesario el id del proveedor que queremos actualizar junto con un nuevo nombre. Por ejemplo si queremos cambiar el nombre del primer proveedor hay entrar al siguiente enlace: http://localhost:8080/proveedores/1

  Y en nuestro cuerpo de json indicar el nombre nuevo que queremos, por ejemplo, Alonso.


```
{
  "nombre":"Alonso Federico"
}
```


- Actualizar si un proveedor esta Activo o no(PUT). Por ejemplo, queremos cambiar esto en el primer proveedor. Tendremos que ir al enlace: http://localhost:8080/proveedores/1

  Y cambiando el cuerpo de la manera siguiente:


```
{
  "isActive":false
```

}

- Eliminar Proveedor por UUID (DELETE). Por ejemplo, si queremos borrar el primer proveedor tendremos que ir al enlace: http://localhost:8080/proveedores/1

ERRORES COMUNES:

- **Error 404:** Primero asegúrate de que la URL sea correcta. Segundo, comprueba que no estes buscando una id que no existe en la base de datos tanto para borrar, actualizar u obtener datos del proveedor.
- **Error 400:** Verifica si estas proporcionando campos validos en el cuerpo JSON y cumples el formato establecido en los atributos del modelo de Proveedor.
- **Error 409:** Este error se debe que existe ya un Proveedor con el nif proporcionado . Compruebe mediante GET cuál es el repetido y decidir si borrar o actualizar el nif.

PRESUPUESTO

En primer lugar, es muy importante considerar los costos asociados con la implementación y mantenimiento del API RESTful, así como la creación de la plataforma web. Estos incluyen los miembros del equipo de desarrollo, el alojamiento web, la seguridad de la plataforma, y las pruebas necesarias para garantizar un funcionamiento óptimo.

Además, debemos tener en cuenta los recursos humanos y técnicos requeridos para la integración del API RESTful con la plataforma web, así como la capacitación del personal de la panadería para el uso adecuado de la aplicación.

Con esto ya mencionado nuestro presupuesto general sería:

- *Desarrollo de API RESTful con Spring Boot*: Entre $15,000 y $40,000.
- *Base de Datos NoSQL y SQL*: Entre $5,000 y $20,000, dependiendo de la complejidad de las bases de datos (si MongoDB o H2) y la necesidad de configuraciones avanzadas.
- *Docker y Despliegue*: Entre $5,000 y $15,000, según la complejidad de la infraestructura y la cantidad de entornos (desarrollo, prueba, producción).
- *WebSockets*: Entre $5,000 y $15,000, dependiendo de la complejidad de las funcionalidades en tiempo real.
- *Swagger y Documentación*: Entre $2,000 y $8,000.
- *Pruebas*: Entre $5,000 y $15,000, dependiendo de la cobertura de pruebas y la complejidad del sistema.
- *Seguridad*: Entre $5,000 y $15,000, dependiendo de los requisitos de seguridad específicos.
- *Mantenimiento y Soporte*: Entre $5,000 y $15,000 anuales, dependiendo de la cantidad de actualizaciones y mejoras esperadas.