**Rápido&Sabroso**

**(SAD) Software Architecture Document**

**Versión 1.0**

**Identificación de Documento**

|  |  |
| --- | --- |
| **Identificación** | SAD |
| **Proyecto** | Rápido&Sabroso |
| **Versión** | 2.3.2 |

|  |  |
| --- | --- |
| **Documento mantenido por** | Programadores |
| **Fecha de última revisión** | 30-09-2024 |
| **Fecha de próxima revisión** | 14-11-2024 |

|  |  |
| --- | --- |
| **Documento aprobado por** | Grupo de programadores |
| **Fecha de última aprobación** | 01-12-2024 |

**Historia de cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 01-08-2024 | 1.0.1 | En este primer punto nuestro equipo de trabajo nos propondremos las ideas del proyecto, además de las reuniones | Programadores página web |
| 25-11-2024 | 1.0.2 | Término del proyecto | Grupo de programadores |
|  |  |  |  |
|  |  |  |  |

**Tabla de Contenidos**

[**Índice de figuras 2**](#_heading=h.gjdgxs)

[**Introducción 3**](#_heading=h.30j0zll)

[Contexto del Problema 3](#_heading=h.1fob9te)

[**1**](#_heading=h.3znysh7) **Alcance 3**

[**2**](#_heading=h.2et92p0) **Referencias 3**

[**3**](#_heading=h.tyjcwt) **Arquitectura de Software 3**

[**4**](#_heading=h.3dy6vkm) **Objetivos y Restricciones de la Arquitectura 4**

[Objetivos de la arquitectura 4](#_heading=h.1t3h5sf)

[Restricciones de la Arquitectura 4](#_heading=h.4d34og8)

[**5**](#_heading=h.2s8eyo1) **Arquitectura Lógica 5**

[Parte Estructural 5](#_heading=h.17dp8vu)

[Parte Dinámica 5](#_heading=h.3rdcrjn)

[**6**](#_heading=h.26in1rg) **Arquitectura de Procesos 5**

[**7**](#_heading=h.lnxbz9) **Arquitectura de desarrollo 6**

[**8**](#_heading=h.35nkun2) **Arquitectura física 7**

[**9**](#_heading=h.44sinio) **Escenarios 7**

[Modelo de Casos de Uso 7](#_heading=h.2jxsxqh)

[Especificación de Casos de Uso Relevantes 7](#_heading=h.z337ya)

[Especificación de los Escenarios de Calidad Relevantes 8](#_heading=h.3j2qqm3)

[**10**](#_heading=h.1y810tw) **Tamaño y desempeño 9**

**Índice de figuras**

**Introducción**

Se requiere desarrollar una página web para la consulta y comparación de precios de comidas rápidas, permitiendo a los usuarios realizar búsquedas filtradas por tipo de comida, ubicación y rango de precios. El sistema está diseñado para facilitar a los usuarios encontrar las mejores opciones en su área, integrando características como un carrito de compras y opciones de pedidos en línea.

1. **Alcance**

El alcance del proyecto incluye:

1. Desarrollo de una página web interactiva y responsiva.
2. Implementación de funcionalidades clave como:
   * Registro y gestión de usuarios.
   * Filtros de búsqueda avanzada para comidas rápidas.
   * Visualización de los precios de las comidas en tiempo real.
   * Gestión de un carrito de compras.
3. Soporte para múltiples dispositivos (portátiles, tabletas y móviles).
4. Integración con sistemas de pago en línea.
5. **Referencias**

En este punto adjuntaremos los documentos utilizados para completar la información y desarrollo de este propio documento.

1. **Arquitectura de Software**

La arquitectura del sistema <<Rapido&Sabroso>> está representada siguiendo el enfoque del framework 4+1 y las recomendaciones del proceso unificado. Las vistas incluidas en esta versión del documento son:

* **Vista de Casos de Uso y Escenarios de Calidad**

|  |  |  |  |
| --- | --- | --- | --- |
| **Código** | **Nombre** | **Actores** | **Prioridad** |
| CU-001 | Filtros de busqueda | Usuario | Alta |
| CU-002 | Gestion de carrito de compras | Usuario | Muy Alta |
| CU-002 | Gestion de pedidos | Usuario | Alta |
| CU-004 | Visualizacion de precios en tiempo real | sistema | Alta |
| CU-005 | Registro y login | usuario y admin | Media |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Escenarios de Calidad Relevantes:**

* **ID: QS1**
  + **Nombre:** Disponibilidad: Usuarios buscando comidas.
  + **Sinopsis:** El sistema debe mostrar resultados precisos en tiempo real.
  + **Prioridad Arquitectónica:** Alta.
* **ID: QS2**
  + **Nombre:** Portabilidad: Adaptación a múltiples dispositivos.
  + **Sinopsis:** El sitio debe ser accesible desde cualquier dispositivo moderno.
  + **Prioridad Arquitectónica:** Alta.
* **ID: QS3**
  + **Nombre:** Mantenibilidad: Alertas durante la mantención del sistema.
  + **Sinopsis:** Mostrar notificaciones de mantenimiento para evitar confusiones.
  + **Prioridad Arquitectónica:** Alta.

**Vista de Metas y Restricciones**:

* Asegurar la disponibilidad de precios actualizados en todo momento.
* Lograr un tiempo de respuesta inferior a 3 segundos.
* Garantizar la seguridad en el registro y manejo de datos personales.

**Restricciones:**

* Uso obligatorio de **MySQL** como base de datos.
* Plazo de entrega limitado a **2 semanas y 4 días**
* **Vista Lógica**: La arquitectura del sistema se basa en una estructura modular. Los componentes principales son:

1. **Módulo de Usuarios:** Gestión de registro, login y perfiles.
2. **Módulo de Filtros:** Procesamiento y visualización de resultados basados en criterios.
3. **Módulo de Pedidos:** Gestión de pedidos y conexión con el carrito de compras.

* **Vista de Procesos**: El usuario realiza una búsqueda filtrada.
* El sistema consulta la base de datos para mostrar resultados.
* El usuario agrega productos al carrito de compras.
* Se realiza la gestión del pedido a través de un sistema de pago seguro.
* **Vista de Implementación.** Se implementarán los siguientes componentes:

1. Frontend: **HTML, CSS, JavaScript (React o Bootstrap)**.
2. Backend: **Django**.
3. Base de datos: **MySQL**.
4. Servidores: **Nginx o Apache**, con balanceadores de carga.
5. **Objetivos y Restricciones de la Arquitectura**

A continuación se revisan las metas y restricciones de la arquitectura.

**Objetivos de la arquitectura**

A partir de las reuniones con los stakeholders y el análisis detallado de los requerimientos, se definieron los principales objetivos arquitectónicos del sistema. Estos guían el diseño y aseguran que las necesidades del usuario y del negocio sean satisfechas de manera eficiente:

**Desempeño:**

* El sistema debe ser capaz de responder a las consultas de los usuarios en un tiempo menor a 3 segundos para búsquedas filtradas.
* La generación de reportes o análisis debe realizarse en menos de 5 minutos, incluso en escenarios de alta concurrencia.
* Soportar un mínimo de 100 usuarios simultáneos durante las horas pico sin degradación del rendimiento.

**Tolerancia a fallos:**

* La arquitectura debe garantizar que el sistema opere correctamente incluso ante fallos menores, como errores en las consultas o fallas temporales de la base de datos.
* Implementar mecanismos de recuperación automática para garantizar la disponibilidad del sistema.

**Seguridad:**

* Proteger la información personal de los usuarios, incluyendo datos de inicio de sesión y preferencias de búsqueda.
* Restringir el acceso a los datos administrativos y mantener auditorías de las actividades del sistema.

**Modificabilidad y Reuso:**

* Diseñar un sistema modular que facilite el mantenimiento y permita la incorporación de nuevas funcionalidades, como más filtros avanzados o integración con servicios externos.
* Garantizar que los componentes del sistema puedan ser reutilizados en otros proyectos relacionados.

**Operatividad:**

* Proveer una interfaz de usuario intuitiva y accesible que simplifique la interacción, incluso para personas con bajo nivel de experiencia tecnológica.
* Diseñar un sistema que sea fácil de implementar y configurar en entornos nuevos.

**Restricciones de la Arquitectura**

Existen restricciones que han sido levantadas con los stakeholders, las cuales se presentan a continuación:

**Tiempo de Construcción:**

* El sistema debe ser implementado dentro de un plazo ajustado de **2 semanas y 4 días**, lo que condiciona la elección de tecnologías y metodologías de desarrollo.
* Priorizar funcionalidades críticas en las primeras iteraciones del proyecto para asegurar entregables viables desde el principio.

**Infraestructura:**

* El sistema deberá operar en servidores replicados con balanceadores de carga para garantizar alta disponibilidad y escalabilidad.
* Usar una base de datos en clúster para soportar transacciones concurrentes sin pérdida de rendimiento.

**Componentes de Software:**

* No se contempla la adquisición de software adicional fuera del ecosistema utilizado actualmente (Django, MySQL).
* Se espera utilizar herramientas open-source para mantener los costos del proyecto bajos.

**Almacenamiento:**

* La base de datos debe contar con al menos **1 GB** de espacio disponible para registros de usuarios, datos de comidas rápidas, y reportes analíticos.
* Se debe habilitar una copia de seguridad automática para evitar la pérdida de información crítica.

**Compatibilidad:**

* El sistema debe ser completamente funcional en navegadores modernos (Chrome, Firefox, Edge, Safari) y en dispositivos móviles, tanto en Android como en iOS.

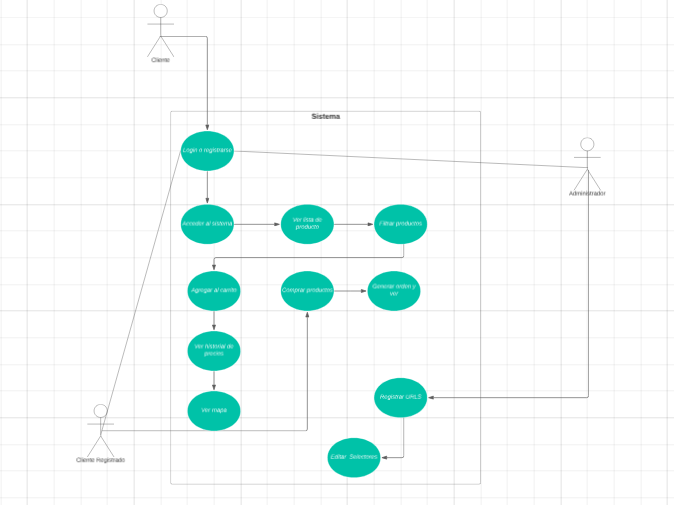
**Acceso Remoto:**

* El sistema debe garantizar conexiones seguras mediante HTTPS y sesiones autenticadas.

**Cumplimiento Regulatorio:**

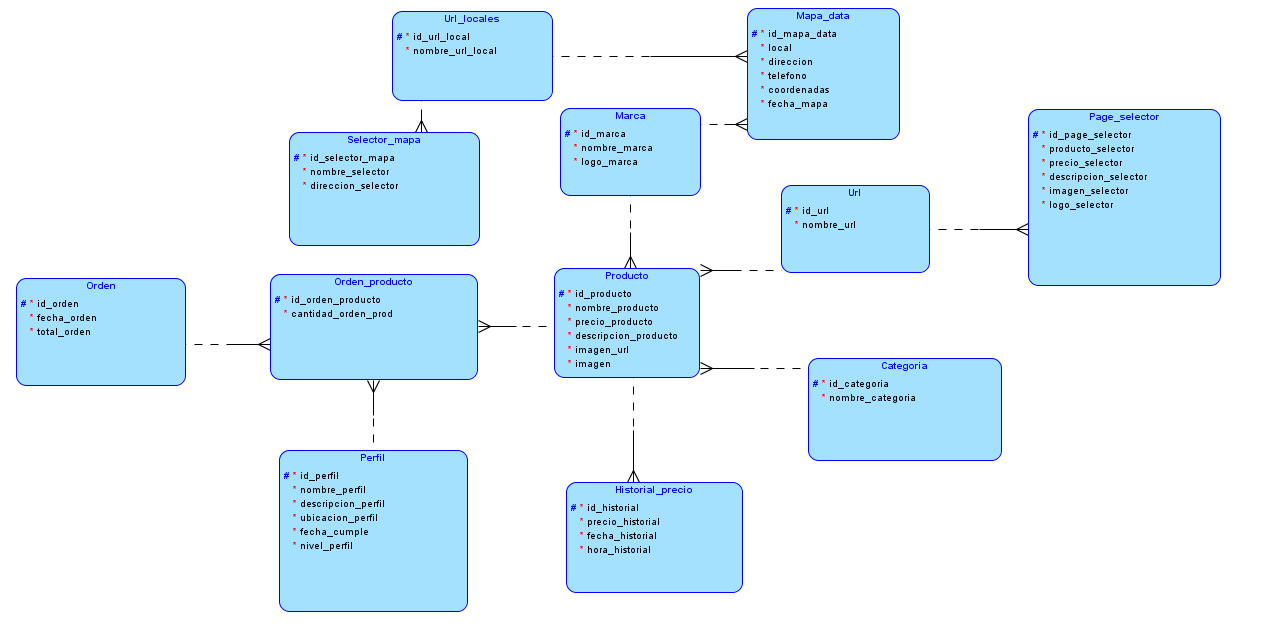
* Asegurar que el sistema cumple con normativas locales relacionadas con la protección de datos y la privacidad de los usuarios.

1. **Arquitectura Lógica**

Este diagrama representará cómo las acciones de entrada y salida de datos se realizan para cumplir las funcionalidades del sistema, como la filtración de precios y el acceso a los menús.

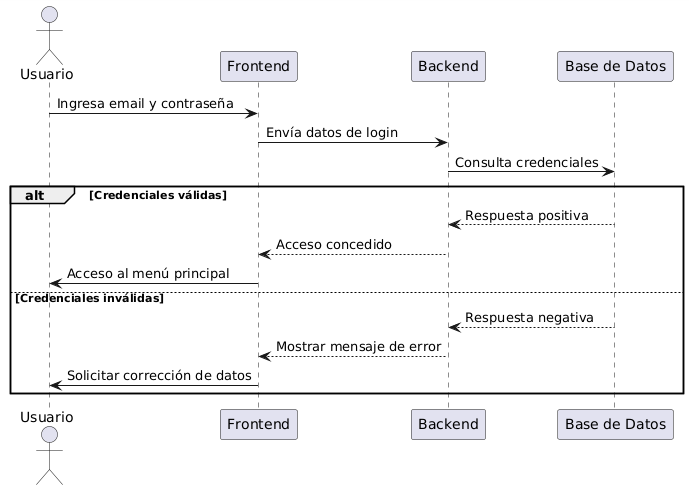
**Parte Estructural**

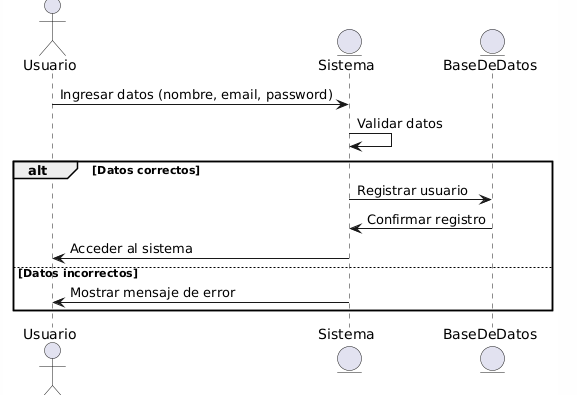
Este es el modelado de la Base de datos



**Parte Dinámica**

Este diagrama muestra el proceso de registro del usuario y la validación de datos. Si el usuario ingresa información incorrecta, el sistema envía un mensaje de error.

****

****

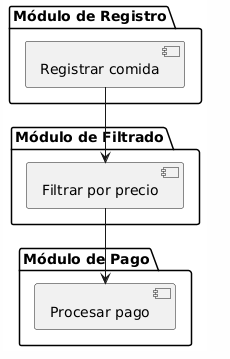
1. **Arquitectura de Procesos**

Este diagrama muestra el flujo de procesos del sistema, como el registro del usuario, la búsqueda de comida y el proceso de pago.



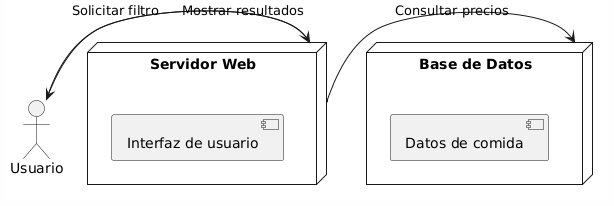
1. **Arquitectura de desarrollo**

Este diagrama muestra los módulos principales, como el módulo de registro de comida rápida, filtrado de precios y pago.

****

1. **Arquitectura física**

Este diagrama muestra la infraestructura del sistema, como los servidores que gestionan la interacción del usuario a través de HTTPS, con conexión a la base de datos.

****

1. **Escenarios**

Esta sección detalla los escenarios funcionales y no funcionales que han obtenido mayor prioridad tras el análisis de requerimientos. Incluye el diagrama de casos de uso, la descripción de los casos prioritarios, y los escenarios en los que atributos de calidad juegan un rol significativo.

**Especificación de Casos de Uso Relevantes**

Los casos de uso más relevantes para el desarrollo de la arquitectura se determinaron con base en los siguientes criterios:

1. Su implementación abarca varios componentes de la vista de despliegue.
2. Representan un riesgo elevado si fallan.
3. Incluyen conceptos clave y relaciones críticas del dominio del sistema.
4. Involucran atributos de calidad esenciales.

A continuación se listan los casos de uso relevantes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Código** | **Nombre** | **Actores** | **Prioridad** |
| CU-001 | Filtro de precios dinámico | Usuario | Muy alta |
| CU-002 | Carrito de compras | Usuario | Alta |
| CU-002 | Integracion de metodos de pago | Usuario, sistema | Muy Alta |
| CU-004 | Gestión de reportes analiticos | Administradores | Media |
| CU-005 | Sistema de autenticación | usuario, sistema | Alta |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Especificación de los Escenarios de Calidad Relevantes**

Tras consultar con los stakeholders, se identificaron y priorizaron los siguientes escenarios de calidad:

* **ID:** QS1

**Nombre:** Rendimiento: Respuesta del filtro de precios  
**Sinopsis:** El usuario debe poder filtrar y obtener resultados de comidas rápidas en función de su rango de precios en tiempo real.

**Entorno:** Uso normal del sistema.  
**Cambio en el entorno:** Alta concurrencia (>100 usuarios simultáneos).  
**Comportamiento esperado:** El sistema debe responder en menos de 3 segundos para todas las consultas realizadas.  
**Medida:** Tiempo de respuesta promedio por consulta.  
**Prioridad Arquitectónica:** Alta.  
**Aplicación:** Global.

* **ID:** QS2  
  **Nombre:** Disponibilidad: Carrito de compras activo  
  **Sinopsis:** Los usuarios deben poder agregar, modificar y eliminar productos en su carrito sin interrupciones.  
  **Entorno:** Proceso de selección y compra en línea.  
  **Cambio en el entorno:** Fallo temporal en la base de datos.  
  **Comportamiento esperado:** El sistema debe preservar los datos del carrito en caché hasta que la base de datos esté disponible nuevamente.  
  **Medida:** No más de 5 minutos de tiempo de recuperación.  
  **Prioridad Arquitectónica:** Muy Alta.  
  **Aplicación:** Global.
* **ID:** QS3  
  **Nombre:** Portabilidad: Compatibilidad multiplataforma  
  **Sinopsis:** El sistema debe ser accesible desde diferentes dispositivos y navegadores modernos.  
  **Entorno:** Uso desde dispositivos con diferentes resoluciones y sistemas operativos.  
  **Cambio en el entorno:** Uso en dispositivos móviles de baja gama.  
  **Comportamiento esperado:** Interfaz funcional y adaptativa en todos los dispositivos probados.  
  **Medida:** Pruebas de adaptabilidad con una tasa de éxito del 95%.  
  **Prioridad Arquitectónica:** Alta.  
  **Aplicación:** Local.

1. **Tamaño y desempeño**

**Tiempo de respuesta:**

* El sistema no debe tardar más de **3 segundos** en devolver resultados al usuario tras una consulta a la base de datos.

**Tiempo de respuesta en transacciones:**

* Para generar reportes analíticos, el sistema no debe exceder los **5 minutos**, incluso en escenarios de alta carga.

**Capacidad de almacenamiento:**

* El almacenamiento mínimo disponible para los usuarios será de **500 MB**.
* La base de datos MySQL contará con al menos **1 GB** de capacidad inicial para datos críticos como precios, filtros y reportes.

**Infraestructura:**

* El sistema está diseñado para operar en un entorno de servidor activo-activo, garantizando alta disponibilidad y tolerancia a fallos.

**Patrones y modularización:**

* Se adoptará una arquitectura modular basada en separación de responsabilidades.
* Se utilizará el patrón **provider** para delegar servicios como pagos externos, asegurando extensibilidad y facilidad de mantenimiento.

**Estrategia de procesos:**

* Los servicios asíncronos se gestionarán mediante un sistema basado en temporizadores que verifican cambios en colas de mensajería o archivos. Esto simplifica la integración y reduce riesgos operativos.