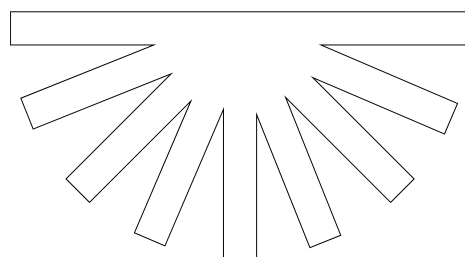


TAKE NO KAMU

# MANUAL TÉCNICO



PRESENTADO POR:

---

Adrián Allard

Carlos Gutiérrez

Emanuel Herrera

Arantza Sanchez

17/06/2025

## Contenidose

1. Introducción.....	5
2. Objetivo del sistema .....	5
3. Requerimientos e instalación .....	5
3.1 Requerimientos en línea .....	5
3.1.1 Dispositivos compatibles .....	5
3.1.2 Navegadores compatibles.....	6
3.1.3 Resolución de pantalla recomendada .....	6
3.1.4 Conectividad.....	6
3.2 Requerimientos en Local .....	6
3.2.1 Dispositivos compatibles .....	6
3.2.2 Requerimientos del Sistema.....	6
3.2.3 Instalación de Librerías .....	7
3.3 Base de datos .....	7
3.4 Instalación.....	7
4. Descripción general del sistema.....	7
4.1 Propósito del sistema.....	7
4.2 Justificación del sistema .....	8
4.3 Alcance del sistema – Interfaz de usuario.....	10
4.3.1 Funcionalidades incluidas (lo que cubre) .....	10
4.3.2 Funcionalidades fuera de alcance (lo que no cubre).....	11
4.4 Descripción Interfaz adminstradores.....	12
4.5 Alcance del sistema – Interfaz de administración .....	13
4.5.1 Funcionalidades incluidas (lo que cubre) .....	13
4.5.2 Funcionalidades fuera de alcance (lo que no cubre).....	14
5. Arquitectura del sistema .....	15
5.1 Introducción al Estilo Arquitectónico .....	15
5.1.1 Principios de Diseño.....	15
5.1.2 Patrones Adoptados .....	16
5.1.3 Justificación de la Arquitectura Elegida .....	17

5.1.4 Visión de la Arquitectura a Alto Nivel .....	17
5.2 Capa de Presentación (Frontend).....	18
5.2.1 Tecnologías Utilizadas.....	18
5.2.2 Diseño Responsivo .....	18
5.2.3 Accesibilidad.....	18
5.3 Componentes Principales y su Interacción .....	19
5.3.1 Cliente (Navegador) .....	19
5.3.2 Servidor Web (Flask).....	19
5.3.3 Base de Datos (MySQL) .....	21
5.3.4 Servicio de Correo (SMTP) .....	21
5.4 Capa de Persistencia (Base de Datos) .....	22
5.4.1 Esquema de Datos.....	22
5.4.2 Acceso a Datos en el Código .....	24
5.4.3 Integridad Referencial y Restricciones .....	27
5.4.4 Backups y Recuperación.....	28
5.4.5 Índices y Optimización .....	28
5.4.6 Diagrama Entidad-Relación.....	29
5.5 Servicios Externos .....	29
Responsabilidad general.....	29
Envío de correos electrónicos (SMTP).....	29
Tecnologías utilizadas: .....	30
Variables almacenadas: .....	30
5.6 Seguridad y Control de Acceso.....	31
5.6.1 Sesiones Seguras .....	31
5.6.5 Expiración de Códigos .....	33
5.6.6 Registro de eventos críticos .....	34
5.7 Justificación de Tecnologías .....	35
5.8 Escalabilidad y Mantenibilidad.....	36
Escalabilidad .....	36
Mantenibilidad .....	36
5.9 Conclusión de la Arquitectura .....	37

6.	Código fuente .....	38
7.	Modelos UML .....	77
	Diagrama de casos de uso .....	77
	Diagrama de clase .....	78
	Diagrama de secuencia de procesos clave .....	79
	Diagrama de secuencia proceso de encuesta .....	80
	Diagrama de actividades .....	81
	Diagrama de Estados – encuesta .....	83
	Diagrama de Estados - usuario .....	84
8.	Control de versiones .....	84
	8.1 Registro de actualizaciones y cambios BD: (Base de Datos) .....	84
	8.2 Registro de actualizaciones y cambios: Interfaz de usuario .....	88
	8.3 Registro de actualizaciones y cambios: Interfaz de administrador.....	92
	8.4 Registro de actualizaciones y cambios: Diseño .....	94
9.	Flujo .....	108
	<b>Interfaz de usuario</b> .....	108
	Interfaz de administrador .....	112
10.	Funciones adicionales .....	113
	8.1. “Recuérdame” (Login).....	113
	8.2. “¿Olvidaste tu contraseña?” (Login).....	113
	8.3. Reenvío de código (Verificación de cuenta).....	113
	8.4. Generación de cupón.....	114
11.	Diccionario de datos .....	114
12.	Pruebas y Testeos .....	138
	Base de datos.....	138
	Vulnerabilidades en SQL.....	138
	Pruebas en la base de datos .....	140
	Prueba 1: Prueba de carga masiva en base de datos.....	140
	Pruebas en la lógica de la aplicación.....	140
	Prueba 1: Mejora de encriptación de datos.....	140
	Prueba 2: Cambio de lógica .....	141

Prueba 3: Consultas parametrizadas .....	142
Prueba 4: Cifrado Fernet.....	142
Prueba 5: Expiración de códigos de verificación .....	143
Prueba 6: Límite de intentos .....	144
Prueba 7: Uso de variables de entorno .....	144
Prueba 8: Manejo personalizado de errores HTTP .....	146
Prueba 9: Validación de entrada del usuario .....	146
Prueba 10: Control de sesiones en rutas protegidas .....	146
Prueba 11: Pruebas de inyección SQL controladas .....	147
Prueba 12: Registro de eventos sensibles (logging) .....	147
13. Referencias .....	148

# 1. Introducción

El presente manual técnico tiene como objetivo guiar paso a paso a los usuarios del sistema Take No Kamu, una plataforma digital diseñada para clientes de un restaurante que podrán responder una encuesta de satisfacción y obtener recompensas por ello.

Este documento proporciona instrucciones claras para el uso correcto del sistema, abarcando desde el ingreso a la plataforma hasta el recibimiento del cupón de recompensa. Está dirigido a los usuarios que contestaran la encuesta de satisfacción del restaurante.

## 2. Objetivo del sistema

Desarrollar una plataforma web robusta y segura que permita a los restaurantes recopilar encuestas de satisfacción de clientes, gestionar la información recolectada mediante un panel administrativo con visualización gráfica por sucursal y país, y ofrecer automáticamente cupones de descuento a los usuarios como incentivo, todo mediante tecnologías modernas como Flask, MySQL y librerías de análisis de datos, garantizando una experiencia eficiente tanto para el usuario final como para los administradores del sistema.

## 3. Requerimientos e instalación

### 3.1 Requerimientos en línea

Para garantizar el correcto funcionamiento de la plataforma, se recomienda que el usuario cuente con las siguientes condiciones mínimas:

#### 3.1.1 Dispositivos compatibles

- Computadoras de escritorio o portátiles con sistema operativo Windows, macOS o Linux.
- Dispositivos móviles con sistema operativo Android (versión 9 o superior) o iOS (versión 13 o superior) [Esto es exclusivamente para el uso en línea].

### 3.1.2 Navegadores compatibles

- Google Chrome
- Microsoft Edge
- Safari

**IMPORTANTE:** No se garantiza el correcto funcionamiento en navegadores no actualizados o navegadores integrados en aplicaciones externas.

### 3.1.3 Resolución de pantalla recomendada

- Resolución mínima: **360 x 640 px**
- Resolución óptima: **1366 x 768 px** o superior

### 3.1.4 Conectividad

- Conexión a Internet obligatoria.
- Se recomienda una conexión estable (mínimo 1 Mbps) para el envío y recepción de datos, especialmente durante el registro, verificación por correo electrónico y uso de cupones.

## 3.2 Requerimientos en Local

### 3.2.1 Dispositivos compatibles

- Computadoras de escritorio o portátiles con sistema operativo Windows, macOS o Linux.

### 3.2.2 Requerimientos del Sistema

- Python 3.7 o superior instalado.
- Acceso a internet (para enviar correos desde Gmail).
- MySQL Server instalado y corriendo.
- Editor de código (como VS Code, PyCharm, etc.).

### 3.2.3 Instalación de Librerías

- Se proporcionará un documento “requirements.txt” en el cual vendrán todas las librerías a ocupar, el cual se ejecutará con el comando “pip install -r requirements.txt”

## 3.3 Base de datos

Debes tener creada una base de datos llamada encuestas\_db con al menos las siguientes tablas:

- usuarios
- respuestas
- cupones

## 3.4 Instalación

Para instalar el sistema, diríjase al siguiente enlace de GitHub, donde encontrará las dependencias necesarias para hacerlo.

<https://github.com/takenokamu/Encuesta-Satisfaccion->

Este enlace lo llevara a la descarga del repositorio para poder realizar la ejecución del sistema en su equipo.

# 4. Descripción general del sistema

A continuación, se presenta una descripción exhaustiva del sistema Take no Kamu, distribuida en siete apartados clave que abarcan su propósito, justificación, alcance, entorno de operación, público objetivo, principales módulos/componentes y casos de uso más relevantes. Esta sección busca ofrecer una visión completa que sirva de fundamento tanto para el equipo de desarrollo como para cualquier lector técnico que requiera comprender en profundidad la estructura y funcionamiento general de la aplicación.

## 4.1 Propósito del sistema

El propósito principal de Take no Kamu es ofrecer a sus restaurantes y sus clientes una plataforma digital sencilla, segura y eficiente para la recopilación de encuestas de satisfacción. A través de esta herramienta, se persiguen los siguientes objetivos específicos:



1. **Recolección de retroalimentación en tiempo real:**

Permitir que los clientes expresen su grado de satisfacción respecto a diferentes aspectos del servicio (calidad de la comida, tiempo de atención, amabilidad del personal, entre otros) de manera inmediata, desde su dispositivo móvil o computadora. Esta retroalimentación provista en tiempo real facilita la toma de decisiones por parte de la gerencia del restaurante para realizar mejoras continuas.

2. **Automatización del envío de incentivos (cupones):**

Tras completar la encuesta, el sistema genera automáticamente un cupón de descuento (aleatorio entre 35 % y 40 %) y lo envía al correo electrónico registrado por el cliente. De esta forma, se motiva la participación en la encuesta y se fortalece la fidelización del cliente, al mismo tiempo que se digitaliza la gestión de recompensas.

3. **Verificación de autenticidad del usuario:**

El sistema incorpora un proceso de registro y verificación de la cuenta mediante el envío de un código de confirmación al correo del usuario. Este mecanismo asegura que las respuestas a las encuestas provengan de clientes reales y disminuye el riesgo de encuestas fraudulentas o múltiples participaciones no autorizadas.

4. **Gestión centralizada de datos de satisfacción:**

Toda la información recabada a través de las encuestas se almacena de forma segura en la base de datos relacional, permitiendo posteriormente realizar análisis estadísticos o generar reportes (como parte de futuros módulos administrativos). A nivel de esta primera fase, el énfasis está en consolidar resultados precisos para cada sucursal y país seleccionado por el cliente.

Take no Kamu pretende ser un puente eficiente entre el restaurante y su público, canalizando opiniones genuinas, recompensando la participación y, al mismo tiempo, integrando controles de seguridad que garanticen la integridad de los datos recabados.

## 4.2 Justificación del sistema

La decisión de desarrollar Take no Kamu surge de varias necesidades y oportunidades identificadas en el entorno de la industria restaurantera actual:

1. **Necesidad de retroalimentación directa y digitalizada:**

Tradicionalmente, las encuestas de satisfacción en restaurantes se

realizaban en papel o mediante formularios físicos al finalizar la comida. Esto implicaba costos en impresión, logística de recolección manual, y un alto margen de error en la transcripción y consolidación de datos. Asimismo, muchos clientes preferían no comprometerse a llenar un formulario impreso tras su consumo, lo que disminuía la tasa de respuesta. Con el auge de los dispositivos móviles y la conectividad permanente, se hace crucial contar con una plataforma digital que reduzca costos, agilice el proceso y aumente la tasa de respuestas.

**2. Competitividad y fidelización de clientes:**

En mercados saturados, la experiencia del cliente es un factor diferencial. Obtener retroalimentación inmediata y actuar sobre ella genera una percepción de compromiso por parte del restaurante hacia sus clientes. Además, ofrecer un cupón de descuento (entre 35 % y 40 %) fortalece la lealtad del cliente y aumenta la probabilidad de una nueva visita. Por esta razón, integrar la generación automática de estos incentivos es un elemento motivador poderoso.

**3. Control de calidad y ajuste de procesos internos:**

Al recopilar información estructurada y cuantificable, la gerencia del restaurante puede identificar patrones: sucursales con menor puntaje en atención, posibles problemas en tiempos de espera o deficiencias en la calidad de platillos. Esta información es invaluable para ajustar procesos internos (capacitación de meseros, revisión de tiempos de cocina, estándares de presentación de alimentos) y mejorar el servicio general.

**4. Valorización de datos y análisis a futuro:**

Aunque la versión inicial de Take no Kamu se enfoca principalmente en la recolección y notificación del cupón, el hecho de contar con una base de datos ordenada y completa abre la puerta a análisis posteriores (reportes mensuales de satisfacción, dashboards gráficos, segmentación de clientes según geolocalización). Esto justifica la inversión en un sistema bien estructurado desde el punto de vista de arquitectura y diseño de datos.

**5. Validación y autenticidad de encuestas:**

Gracias al proceso de registro y verificación de correo, se garantiza que cada respuesta provenga de un usuario único y real. Esto minimiza la posibilidad de respuestas manipuladas o spam, elevando la confiabilidad de los datos recabados. La validez de la información es crucial para la toma de decisiones y para que los cupones no sean generados sin mérito alguno.

En conjunto, Take no Kamu responde a la necesidad del sector de contar con una herramienta digital que no solo facilite la obtención de retroalimentación, sino que además incorpore incentivos medibles y registre datos confiables para respaldar las decisiones estratégicas del restaurante.

## 4.3 Alcance del sistema – Interfaz de usuario

En esta sección se define con claridad qué funcionalidades están incluidas en el sistema Take no Kamu y cuáles quedan explícitamente fuera de su alcance en la versión actual. Esto ayuda a delimitar expectativas, orientar esfuerzos de desarrollo y facilitar el entendimiento del alcance real del proyecto.

### 4.3.1 Funcionalidades incluidas (lo que cubre)

#### 1. Registro de usuarios:

- Captura de datos personales: nombre, apellido, correo, teléfono y contraseña.
- Validación mínima de formato de correo y número telefónico.
- Medidor de fuerza de la contraseña para garantizar claves robustas.

#### 2. Verificación de cuenta por correo electrónico:

- Generación de un código alfanumérico único para cada registro.
- Envío del código al correo registrado.
- Validación del código ingresado y activación de la cuenta.

#### 3. Inicio de sesión autenticado:

- Ingreso de correo y contraseña verificados.
- Opción “Recuérdame” para mantener sesión activa.
- Opción de recuperación de contraseña mediante correo (envío de enlace o código de restablecimiento).

#### 4. Encuesta de satisfacción interactiva:

- Selección de país y sucursal para contextualizar la encuesta.
- Cuatro tipos de preguntas:
  - Escala numérica (barra deslizante 0–10).

- Opción binaria Sí/No (botones de selección).
  - Pregunta abierta (campo de texto limitado a 200 caracteres).
  - Puntuación con pandas (barra deslizante 0–5).
  - Validación de respuestas obligatorias.
- 5. Generación automática de cupones:**
- Cálculo de un porcentaje de descuento aleatorio entre 35 % y 40 %.
  - Creación de un código de cupón único.
  - Restricción de emisión de un nuevo cupón a partir de 30 días desde la última generación para un mismo usuario.
- 6. Envío de notificaciones por correo electrónico:**
- Correo de bienvenida\*\* (potencial, si se desea en versiones futuras).
  - Correo de verificación de cuenta con el código correspondiente.
  - Correo de envío de cupón al finalizar la encuesta, incluyendo imagen y detalles de validez.
- 7. Mensajes de retroalimentación en la interfaz:**
- Indicadores visuales de éxito o error (por ejemplo, “¡Cuenta verificada exitosamente!”, “Contraseña muy débil”, “Cupón enviado correctamente”).
  - Enlaces de ayuda y guías (por ejemplo, “¿Olvidaste tu contraseña?”, “¿No recibiste el código? Reenviar código”).
- 8. Seguimiento y registro de respuestas:**
- Almacenamiento en base de datos de cada respuesta enviada en la encuesta, vinculada al usuario y a la sucursal seleccionada.
  - Registro de la fecha y hora de cada participación.

#### 4.3.2 Funcionalidades fuera de alcance (lo que no cubre)

##### 1. Administración interna de sucursales:

- No existe en esta versión un panel de administración para que el equipo del restaurante agregue o modifique países y sucursales. Dichos datos deben estar precargados en la base de datos por el administrador

## **2. Generación de reportes avanzados o dashboards gráficos:**

- Aunque la base de datos almacena la información de forma estructurada, no se proporciona en esta versión una interfaz de reportes o gráficos para la gerencia. Estos se pueden ver en otra interfaz ((administrador).

## **3. Integración con sistemas de punto de venta (POS):**

- No se contemplan integraciones automáticas con software de facturación o sistemas de cobro del restaurante.

## **4. Personalización de la encuesta por parte del usuario:**

1. La encuesta es fija en su estructura (tipos de preguntas y número de ítems). No se incluye un módulo que permita modificar preguntas, agregar nuevas o eliminar existentes desde la interfaz.

## **5. Registro de respuestas anónimas:**

1. Todas las encuestas están vinculadas a un usuario registrado y verificado; no existe la opción de realizar encuestas de manera anónima.

## **6. Gestión integrada de cupones en terminales de punto de venta:**

1. El sistema envía el cupón por correo, pero no incluye un módulo específico para validar el cupón en un terminal de punto de venta (sería responsabilidad del restaurante implementar el escaneo o validación física).

## **7. Localización avanzada e idiomas múltiples:**

1. Por el momento la aplicación está disponible únicamente en español. No contempla soporte para otros idiomas.

De esta forma, el alcance queda claramente delimitado a las funcionalidades principales de captura de encuesta, verificación de usuario y emisión de cupones, dejando para futuras iteraciones temas de administración interna, reportes avanzados e integraciones adicionales.

## **4.4 Descripción Interfaz adminstradores**

La interfaz de administración del sistema *Take no Kamu* está diseñada para ofrecer a los administradores una plataforma segura y centralizada desde la cual

puedan gestionar el acceso administrativo y visualizar los resultados recolectados a través de las encuestas de satisfacción. Su objetivo principal es facilitar el análisis de datos, brindar visibilidad sobre el desempeño de las distintas sucursales y permitir el registro de nuevos administradores sin necesidad de intervención técnica adicional

## 4.5 Alcance del sistema – Interfaz de administración

En esta sección se especifican claramente las funcionalidades disponibles para los administradores dentro del sistema *Take no Kamu*, así como aquellas que no forman parte del alcance actual en esta versión. Esta delimitación permite enfocar los esfuerzos técnicos en los requerimientos implementados, facilitando el mantenimiento y la evolución futura del sistema.

### 4.5.1 Funcionalidades incluidas (lo que cubre)

#### 1. Panel de administración accesible por ruta específica:

- Acceso a través de un panel exclusivo distinto al de usuarios comunes.

#### 2. Registro de nuevos administradores:

- Acceso a la sección "Registrar administrador" desde el menú principal.
- Captura de los siguientes datos: nombre, apellidos, correo electrónico, teléfono y contraseña.
- Validación de campos básicos (formato de correo y longitud mínima de contraseña).
- Botón de acción "Registrar administrador".
- Confirmación visual con mensaje "Administrador registrado exitosamente".
- Redirección automática a la pantalla de inicio tras registro.

#### 3. Listado de administradores registrados:

- Acceso a la sección "Lista de administradores".
- Visualización de los datos previamente registrados.

#### 4. Inicio de sesión para administradores:

- Ingreso con correo y contraseña en la misma interfaz de inicio de sesión de usuarios.
- Detección automática de tipo de usuario (administrador) para redirigir a la interfaz correspondiente.

#### **5. Visualización de reportes administrativos:**

- Acceso a panel con métricas relacionadas con las encuestas recolectadas.
- Filtros disponibles: país, sucursal, fecha de inicio y fecha de fin.
- Indicadores clave mostrados:
  - Total de respuestas recibidas.
  - Calificación promedio del restaurante (estrellas).
  - Porcentaje de satisfacción del servicio.
  - Tasa de retorno (porcentaje).
- Visualización de tres gráficas:
  - Calificación promedio por sucursal.
  - Distribución de calificaciones.
  - Intención de retorno de los clientes.

#### **6. Revisión de respuestas individuales:**

- Acceso a un área con la lista detallada de respuestas recibidas.
- Datos visibles: nombre del cliente, fecha de participación y respuestas completas de la encuesta.

### **4.5.2 Funcionalidades fuera de alcance (lo que no cubre)**

#### **1. Gestión de administradores con roles o permisos diferenciados:**

- No se contempla la creación de jerarquías administrativas ni la asignación de permisos específicos por tipo de administrador.

#### **2. Edición o eliminación de administradores desde la interfaz:**

- Actualmente no se cuenta con funciones para modificar o eliminar registros de administradores.

### 3. Creación o modificación de encuestas:

- Los administradores no pueden editar el contenido de las encuestas ni agregar nuevas preguntas desde su panel.

### 4. Descarga automatizada de reportes:

- No se incluye exportación de datos en formatos como PDF o Excel desde la interfaz administrativa.

### 5. Gestión directa de usuarios o cupones:

- La administración se limita a la visualización de reportes de encuestas. No incluye gestión directa de usuarios registrados ni de los cupones generados.

## 5. Arquitectura del sistema

### 5.1 Introducción al Estilo Arquitectónico

La arquitectura de Take no Kamu se fundamenta en el patrón multicapa (n-tier) dentro de un modelo Cliente-Servidor, con el objetivo de aislar responsabilidades, maximizar la mantenibilidad y facilitar la escalabilidad. A continuación, se detallan los principios, patrones y decisiones que han definido esta arquitectura.

#### 5.1.1 Principios de Diseño

Cada capa del sistema asume un único conjunto de responsabilidades:

- **Capa de Presentación:** interacción con el usuario, validación ligera en cliente y renderizado de vistas.
- **Capa de Lógica de Negocio:** procesamiento de reglas de negocio (autenticación, validación de encuestas, generación de cupones, envío de correos).
- **Capa de Persistencia:** almacenamiento y recuperación de datos en la base de datos relacional.
- **Capa de Servicios Externos:** integración con SMTP para correo y gestión de archivos estáticos generados.

#### 1. Acoplamiento Bajo y Alta Cohesión



- Cada componente (por ejemplo, el módulo de encuestas o el de reportes) reúne funciones afines, manteniendo el código organizado y fácil de testear.
- Las dependencias entre módulos se reducen a interfaces bien definidas: rutas HTTP, llamadas a funciones de acceso a datos o invocaciones a servicios SMTP.

## 2. Reutilización y Extensibilidad

- Los servicios de envío de correo, hashing de contraseñas o generación de gráficos son funciones desacopladas que pueden reutilizarse en otros proyectos o ampliarse (por ejemplo, cambio a un proveedor de correo profesional).

### 5.1.2 Patrones Adoptados

#### 1. Modelo-Vista-Controlador (MVC) Simplificado

- **Modelo:** las clases y funciones de acceso a datos (encapsuladas en consultas SQL sobre las tablas usuarios, respuestas y cupones).
- **Vista:** plantillas Jinja2 (.html en /templates) que reciben datos del controlador y generan la interfaz final.
- **Controlador:** las rutas de Flask (@app.route) actúan como mediadores entre modelo y vista, validando inputs, manipulando datos y retornando plantillas renderizadas.

#### 2. Arquitectura por Capas

- **Capa 1 – Presentación (Frontend):** validaciones básicas en JavaScript, despliegue de formularios y mensajes flash.
- **Capa 2 – Lógica de Negocio (Backend):** módulos de autenticación, encuesta, generación de cupones y reporte.
- **Capa 3 – Persistencia (Data Access Layer):** funciones que ejecutan sentencias SQL y mapean resultados a estructuras de datos de Python.
- **Capa 4 – Integración de Servicios:** adaptadores SMTP, almacenamiento de archivos estáticos, lectura de variables de entorno.

#### 3. Decorador de Seguridad

- Implementa el patrón Intercepting Filter, interponiendo lógica de autenticación antes de ejecutar la función de cada ruta protegida.

#### 4. Plantillas de Correo y Gráficos como Servicios Internos

- Encapsulan reglas de composición de email (plantillas MIME) y generación de imágenes (Matplotlib) en funciones auxiliares, facilitando su mantenimiento independiente.

### 5.1.3 Justificación de la Arquitectura Elegida

- **Simplicidad y Rapidez de Desarrollo:** Flask permite un arranque rápido sin la complejidad de frameworks más pesados, ideal para prototipos y sistemas de tamaño medio.
- **Escalabilidad Modular:** La separación en módulos (auth, encuesta, admin) facilita migrar a **Flask Blueprints** o incluso a microservicios en futuras iteraciones.
- **Facilidad de Pruebas:** Cada capa y componente puede ser testeado aisladamente—por ejemplo, pruebas unitarias de funciones de hashing, tests de integración en rutas REST o pruebas de carga sobre MySQL.
- **Despliegue Flexible:** Al apoyarse en componentes estándar (WSGI, MySQL, SMTP), el sistema puede ejecutarse tanto en servidores bare-metal, contenedores Docker o plataformas PaaS como Heroku o Render.

### 5.1.4 Visión de la Arquitectura a Alto Nivel

En su conjunto, Take no Kamu se encuadra en un ecosistema en el que:

- El **cliente** (navegador) interactúa exclusivamente con el servidor Flask mediante peticiones HTTP/HTTPS.
- El **servidor** expone rutas RESTful y páginas renderizadas, aplica la lógica de negocio y se comunica con la base de datos MySQL.
- El **sistema de correo** es un servicio externo al que Flask envía mensajes a través de SMTP.
- La **persistencia** reside en una base de datos relacional organizada según el script SQL proporcionado, asegurando integridad y eficiencia en las consultas.

- Los **archivos estáticos dinámicos** (gráficos de reportes) se generan en tiempo real, se almacenan en disco y se sirven directamente al cliente.

## 5.2 Capa de Presentación (Frontend)

### Responsabilidad:

Esta capa es la interfaz directa con el usuario. Su función principal es **mostrar contenido, capturar entradas y presentar resultados** de manera clara y responsiva. Asegura una experiencia de usuario amigable y permite validar datos antes de enviarlos al backend.

### 5.2.1 Tecnologías Utilizadas

- **HTML5**
- **CSS3:** estilos globales y específicos para formularios, botones, layout responsivo y mensajes flash.
- **JavaScript**
- **Jinja2:** permiten reusar componentes (cabeceras, pies de página) y pasar variables del backend (por ejemplo, listas de países, mensajes de flash, nombre de usuario) para renderizado dinámico.

### 5.2.2 Diseño Responsivo

- **Mobile-first:** CSS utiliza media queries para ajustar:
  - Anchura de formularios al ancho del dispositivo.
  - Tamaño de botones y textos.
  - Ocultación de columnas en tablas de reportes\_admin.html cuando la pantalla es pequeña.
- **Pruebas manuales:** validado en:
  - Chrome y Safari (iOS).
  - Chrome y Firefox (Android).

### 5.2.3 Accesibilidad

- **Etiquetas <label>** correctamente asociadas a campos <input>.
- **Contraste suficiente** entre texto y fondo en botones y alertas.
- **Teclado navegable:** tabindex en botones, focus states bien definidos.

## 5.3 Componentes Principales y su Interacción

### 5.3.1 Cliente (Navegador)

- **Responsabilidad:**

- Presentar formularios y vistas HTML al usuario.
- Capturar sus interacciones (clics, envíos de formulario, cambios de selección).
- Mostrar resultados dinámicos (mensajes flash, gráficos) sin recargar toda la página cuando sea posible.

- **Interacciones clave:**

1. **Solicitar páginas**

- GET /login, GET /registro, GET /encuesta, GET /admin/reportes\_sucursal, etc.

2. **Enviar formularios**

- POST /login con { email, password }
- POST /registro con { nombre, apellido, correo, telefono, contrasena }
- POST /encuesta con los campos de la encuesta
- POST /recuperar\_contrasena, /verificar\_codigo\_recuperacion, /nueva\_contrasena

3. **Recepción de respuestas**

- HTML generado por Jinja2.
- Imágenes de gráficos en /static/images/.
- Mensajes flash (clases CSS .flash-success, .flash-danger).

### 5.3.2 Servidor Web (Flask)

- **app.py**

- Crea la instancia de la aplicación Flask.
- Carga la configuración (config.py).
- Registra Blueprints desde routes/.
- Inicializa la extensión MySQL y otras librerías.
- **config.py**
  - Lee variables de entorno (.env) usando python-dotenv.
  - Define MYSQL\_HOST, MYSQL\_USER, MYSQL\_PASSWORD, MYSQL\_DB.
  - Define EMAIL\_SENDER, EMAIL\_PASSWORD.
  - Establece SECRET\_KEY para sesiones seguras.
- **Blueprints (routes/)**
  - auth.py:
    - Rutas de autenticación (/login, /logout, /registro, /verificar\_codigo, /recuperar\_contrasena, /verificar\_codigo\_recuperacion, /nueva\_contrasena).
    - Utiliza encriptador.py para hash y generación de códigos.
  - encuesta.py:
    - Rutas /encuesta (GET y POST).
    - Función verificar\_actividad\_reciente() para limitar a 30 días.
    - Llama a models/respuesta.py y models/cupon.py para persistir datos y generar cupones.
  - admin.py:
    - Ruta /admin/reportes\_sucursal.
    - Consulta respuestas y usuarios via models/respuesta.py y models/usuario.py.
    - Usa Pandas para DataFrame y Matplotlib para gráficos, guarda PNGs en static/images/.
  - errors.py:

- Manejadores `@app.errorhandler(404)` y `@app.errorhandler(500)` que renderizan vistas de error.
- **encriptador.py / encriptador\_datos.py**
  - Funciones auxiliares para hashing de contraseñas (`generate_password_hash`, `check_password_hash`).
  - Generación de códigos de verificación y recuperación.
  - Pueden incluir validaciones regex y sanitización de datos.

### 5.3.3 Base de Datos (MySQL)

- **Esquema (Base de datos Final.sql)**
  - usuarios (PK `id_usuario`, nombre, apellido, correo, telefono, contraseña, verificado, `codigo_verificacion`, `fecha_codigo`).
  - respuestas (PK `id_respuesta`, FK `id_usuario`, campos de encuesta, `fecha_respuesta`).
  - cupones (PK `id_cupon`, FK `id_usuario`, `codigo`, porcentaje, `fecha_emision`, `fecha_vencimiento`, utilizado).
- **Models (models/)**
  - `usuario.py`: funciones `get_user_by_email()`, `create_user()`, `verify_user()`, `update_password()`.
  - `respuesta.py`: funciones `save_response()`, `get_last_response_date()`, `fetch_responses(filters)`.
  - `cupon.py`: funciones `create_coupon()`, `check_coupon_validity()`.

Cada función abre un cursor MySQL, ejecuta la sentencia SQL correspondiente y cierra el cursor. El commit se hace en la capa de ruta tras el procesado.

### 5.3.4 Servicio de Correo (SMTP)

- **Configuración en config.py**
  - `EMAIL_SENDER = os.getenv('EMAIL_SENDER')`
  - `EMAIL_PASSWORD = os.getenv('EMAIL_PASSWORD')`
- **Envío de correos**

- En routes/auth.py y en el módulo de cupones:
  - enviar\_codigo\_verificacion(destinatario, codigo)
  - enviar\_codigo\_recuperacion(destinatario, codigo)
  - enviar\_cupon(destinatario, codigo, porcentaje)
- Uso de smtplib.SMTP\_SSL('smtp.gmail.com', 465) y objetos EmailMessage / MIMEMultipart.
- **Gestión de errores:**
  - Captura excepciones al enviar correo y registra en registro\_seguridad.log.

## 5.4 Capa de Persistencia (Base de Datos)

La Capa de Persistencia es la columna vertebral donde se guardan y recuperan todos los datos críticos de Take no Kamu: usuarios, respuestas de encuestas y cupones. Asegura la consistencia, integridad y seguridad de la información, al tiempo que ofrece un rendimiento adecuado para consultas de lectura y escritura frecuentes.

### 5.4.1 Esquema de Datos

El script SQL “Base de datos Final.sql” define el esquema completo. A continuación se detallan cada una de las tablas, sus campos, tipos y relaciones:

#### 1.- Base de datos

```
CREATE DATABASE `encuestas_db` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;
```

#### 2.- Tabla usuarios

- **Propósito:** almacenar las credenciales y estado de cada usuario.
- **Estructura:**

```
CREATE TABLE `usuarios` (
  `id_usuario` int NOT NULL AUTO_INCREMENT,
  `privilegios` int NOT NULL DEFAULT '0',
  `nombre` varchar(50) NOT NULL,
  `apellido` varchar(50) NOT NULL,
  `correo` varchar(100) NOT NULL,
  `telefono` varchar(15) NOT NULL,
  `contraseña` varchar(255) NOT NULL,
  `fecha_registro` datetime DEFAULT CURRENT_TIMESTAMP,
  `verificado` tinyint(1) DEFAULT '0',
  `codigo_verificacion` varchar(6) DEFAULT NULL,
  `fecha_codigo` datetime DEFAULT NULL,
  PRIMARY KEY (`id_usuario`),
  UNIQUE KEY `correo` (`correo`),
  UNIQUE KEY `telefono_UNIQUE` (`telefono`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

### Índices & Restricciones:

- UNIQUE(correo), UNIQUE(telefono)
- Clave primaria en id\_usuario.

### 3.- Tabla respuestas

- **Propósito:** almacenar cada conjunto de respuestas de la encuesta, asociado a un usuario.
- **Estructura:**

```
CREATE TABLE `respuestas` (
  `id_respuesta` int NOT NULL AUTO_INCREMENT,
  `pais` varchar(50) DEFAULT NULL,
  `sucursal` varchar(50) DEFAULT NULL,
  `calidad_comida` int DEFAULT NULL,
  `tiempo_espera` enum('si','no') DEFAULT NULL,
  `atencion_personal` int DEFAULT NULL,
  `agrado_sucursal` enum('si','no') DEFAULT NULL,
  `volveria_visitar` enum('si','no') DEFAULT NULL,
  `area_mejora` text,
  `calificacion_general` int DEFAULT NULL,
  `fecha_respuesta` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `id_usuario` int DEFAULT NULL,
  PRIMARY KEY (`id_respuesta`),
  KEY `fk_usuario_respuesta` (`id_usuario`),
```

```
CONSTRAINT `fk_usuario_respuesta` FOREIGN KEY (`id_usuario`) REFERENCES `usuarios`
(`id_usuario`) ON DELETE SET NULL
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900
_ai_ci;
```



- **Integridad Referencial:**

- Si un usuario se elimina, sus respuestas.id\_usuario se pone a NULL, preservando el historial.

#### 4. Tabla cupones

- **Propósito:** guardar los cupones generados y su estado de uso.
- **Estructura:**

```
CREATE TABLE `cupones` (  
  `id_cupon` int NOT NULL AUTO_INCREMENT,  
  `id_usuario` int NOT NULL,  
  `codigo` varchar(6) NOT NULL,  
  `porcentaje` int NOT NULL,  
  `fecha_emision` datetime DEFAULT CURRENT_TIMESTAMP,  
  `fecha_vencimiento` datetime NOT NULL,  
  `utilizado` tinyint(1) DEFAULT '0',  
  PRIMARY KEY (`id_cupon`),  
  UNIQUE KEY `codigo_UNIQUE` (`codigo`),  
  KEY `fk_usuario_cupon` (`id_usuario`),  
  CONSTRAINT `fk_usuario_cupon` FOREIGN KEY (`id_usuario`) REFERENCES `usuarios` (`id_usuario`) ON DELETE CASCADE  
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

#### Comportamiento de Borrado:

- Si se elimina el usuario, sus cupones se eliminan automáticamente (ON DELETE CASCADE).

### 5.4.2 Acceso a Datos en el Código

En lugar de un ORM, cada **ruta** de Flask que necesita acceder a la base de datos utiliza **consultas SQL** directas:

- **Apertura de cursor:**

```
cur = mysql.connection.cursor()
```

- **Ejemplos de consultas:**

- **Registro de usuario**

```
cur.execute("SELECT * FROM usuarios WHERE correo = %s", (correo,))
```

```
...
```

```
cur.execute("""
```

```
INSERT INTO usuarios (privilegios, nombre, apellido, correo, telefono,  
contrasena, codigo_verificacion)
```

```
VALUES (%s, %s, %s, %s, %s, %s, %s)
```

```
""", (0, nombre, apellido, correo, telefono, hash_pass, codigo))
```

- **Verificación de código:**

```
cur.execute("SELECT codigo_verificacion FROM usuarios WHERE correo = %s",  
(correo,))
```

```
...
```

```
cur.execute("""  
UPDATE usuarios  
SET codigo_verificacion = NULL, verificado = TRUE  
WHERE correo = %s  
""", (correo,))
```

- **Guardado de respuestas:**

```
cur.execute("""  
INSERT INTO respuestas (  
id_usuario, pais, sucursal, calidad_comida, tiempo_espera,  
atencion_personal, agrado_sucursal, volveria_visitar,  
area_mejora, calificacion_general, fecha_respuesta  
) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, NOW())  
""", (id_usuario, pais, sucursal, calidad_comida, tiempo_espera,  
atencion_personal, agrado_sucursal, volveria_visitar,  
area_mejora, calificacion_general))
```

- **Generación de cupón:**

La función `generar_codigo_cupon()` se encarga de crear un código alfanumérico aleatorio de 6 caracteres. Este código sirve como identificador único del cupón de descuento que se le otorgará al usuario tras contestar la encuesta.

```
def generar_codigo_cupon():  
    caracteres = string.ascii_uppercase + string.digits  
    return "".join(random.choice(caracteres) for _ in range(6))
```

- **Preparación y envío del cupón**

La función `enviar_cupon(destinatario, codigo, porcentaje)` envía un correo personalizado con un mensaje de agradecimiento, el código del cupón y una imagen correspondiente al porcentaje de descuento.

```
mensaje = MIMEMultipart()  
mensaje['Subject'] = f'Tu cupón de {porcentaje}% de descuento - Gracias por tu encuesta'  
mensaje['From'] = EMAIL_SENDER  
mensaje['To'] = destinatario
```

- **Contenido textual del correo**

Se genera el cuerpo del mensaje de texto que irá en el correo, incluyendo el código del cupón y su vigencia.

```
texto = f"""  
Hola,
```

¡Gracias por responder nuestra encuesta de satisfacción!

Como agradecimiento, te enviamos un cupón de {porcentaje}% de descuento para tu próxima visita.

Tu código de cupón es: {codigo}

Este cupón es válido por 30 días a partir de hoy.

¡Esperamos verte pronto!

Saludos,  
El equipo de Encuestas  
'''

- **Adjuntar Imagen del Cupón**

Se verifica si existe una imagen en la ruta correspondiente y se adjunta al correo como una imagen embebida.

```
ruta_imagen = os.path.join(STATIC_FOLDER, 'images', f'cupon_{porcentaje}.png')
if os.path.exists(ruta_imagen):
    with open(ruta_imagen, 'rb') as img:
        imagen = MIMEImage(img.read())
        imagen.add_header('Content-ID', '<cupon>')
        imagen.add_header('Content-Disposition', 'inline',
filename=f'cupon_{porcentaje}.png')
        mensaje.attach(imagen)
```

- **Envío del Correo al Usuario**

El mensaje es enviado al destinatario usando SMTP con cifrado SSL, autenticándose con la cuenta del sistema.

```
with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
    smtp.login(EMAIL_SENDER, EMAIL_PASSWORD)
    smtp.send_message(mensaje)
```

- **Manejo de Errores**

Si ocurre un error durante el proceso de envío del correo, se captura la excepción y se imprime un mensaje en la consola.

```
except Exception as e:
    print(f"Error al enviar email con cupón: {e}")
    return False
```

### 5.4.3 Integridad Referencial y Restricciones

- **Llaves primarias:** garantizan unicidad de registros.

- **Índices únicos:** en usuarios.correo, usuarios.telefono, cupones.codigo para búsquedas rápidas y evitar duplicados.
- **Llaves foráneas:**
  - respuestas.id\_usuario → usuarios.id\_usuario (ON DELETE SET NULL para conservar historial aun si el usuario se elimina).
  - cupones.id\_usuario → usuarios.id\_usuario (ON DELETE CASCADE para remover cupones de usuarios borrados).
- **Tipos de datos apropiados:**
  - VARCHAR para texto corto, TEXT para comentarios libres.
  - ENUM para valores de sí/no.

#### 5.4.4 Backups y Recuperación

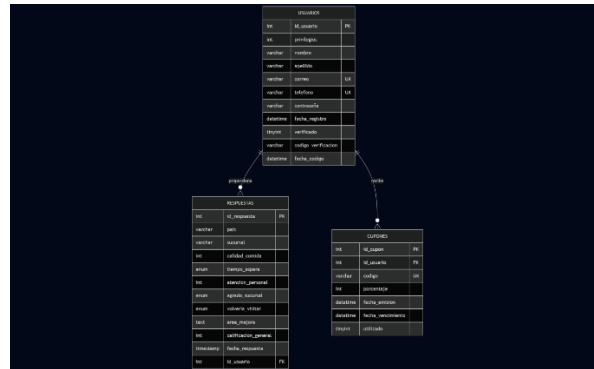
- **Backups periódicos** de encuestas\_db a través de mysqldump.
- **Plan de restauración documentado:**
  1. Detener la aplicación.
  2. mysql -u root -p encuestas\_db < backup.sql.
  3. Validar integridad con consultas de prueba.
  4. Levantar la aplicación.
- **Monitoreo de salud de la DB mediante herramientas como MySQL Workbench o dashboards de servicio gestionado.**

#### 5.4.5 Índices y Optimización

- **Índices automáticos:**
  - Clave primaria en cada tabla.
  - Índices únicos en usuarios(correo), usuarios(telefono), cupones(codigo).
- **Índices adicionales recomendados:**
  - Índice en respuestas(fecha\_respuesta) para acelerar filtrado por rango de fechas en reportes.

- Índice compuesto en respuestas(pais, sucursal) si la consulta de sucursales es frecuente.

#### 5.4.6 Diagrama Entidad-Relación



Este ERD brinda una visión global de la estructura de datos y las relaciones que sostienen la lógica de negocio.

## 5.5 Servicios Externos

## Responsabilidad general

Los servicios externos abarcan funcionalidades que no forman parte del núcleo del sistema, pero que son indispensables para su correcto funcionamiento. Incluyen servicios de correo electrónico, librerías para análisis y visualización de datos, encriptación, y manejo seguro de configuraciones. A continuación se describen los principales:

## Envío de correos electrónicos (SMTP)

El sistema envía correos para:

- Verificación de cuentas
- Recuperación de contraseñas
- Entrega de cupones de descuento

Esto se realiza mediante un servidor SMTP, utilizando el protocolo SMTP sobre SSL:

**smtplib.SMTP\_SSL('smtp.gmail.com', 465)**

## Tecnologías utilizadas:

- `smtpplib`: Establece la conexión segura al servidor de Gmail.
- `email.message.EmailMessage`, `MIMEMultipart`, `MIMEText`, `MIMEImage`: Se utilizan para construir correos en formato HTML, con soporte para imágenes inline (como cupones).
- El diseño del correo es personalizado y atractivo visualmente para el usuario.

## Seguridad:

Las credenciales necesarias para autenticar el envío de correos (`EMAIL_SENDER` y `EMAIL_PASSWORD`) se almacenan de forma segura en variables de entorno (ver siguiente punto).

### Variables de entorno (.env)

Para evitar exponer información sensible (como contraseñas, claves secretas o configuraciones de despliegue), se utiliza un archivo `.env` cargado con:

```
from dotenv import load_dotenv  
  
load_dotenv()
```

## Variables almacenadas:

- `SECRET_KEY`: Clave criptográfica usada por Flask y cryptography.
- Credenciales de MySQL (`DB_USER`, `DB_PASSWORD`, `DB_NAME`, `DB_HOST`)
- Credenciales SMTP (`EMAIL_SENDER`, `EMAIL_PASSWORD`)
- Variables de entorno del sistema: `PORT`, `FLASK_ENV`, etc.

Este mecanismo garantiza una configuración portable, segura y desacoplada del código.

## Generación y almacenamiento de gráficos

El sistema genera gráficas estadísticas que muestran información de uso, como respuestas por sucursal o país.

#### **Librerías utilizadas:**

- Pandas y NumPy: Para procesamiento de datos.
- Matplotlib: Para visualización (gráficas de barras, pastel, etc.)

#### **Almacenamiento:**

- Los gráficos se guardan programáticamente en la carpeta: `static/images/`, con nombres como `chart_sucursales.png`.
- Se usa `os.makedirs()` para crear automáticamente la carpeta si no existe.
- Estos gráficos están disponibles para visualización directa desde el frontend en los reportes de administración.

## **5.6 Seguridad y Control de Acceso**

La seguridad es un pilar fundamental de Take no Kamu. A continuación, se detallan los mecanismos implementados para garantizar la confidencialidad de los datos, la integridad de las operaciones y la protección frente a ataques comunes.

### **5.6.1 Sesiones Seguras**

- **SECRET\_KEY:**
  - Definida en el código y cargada por el mismo

```
app.secret_key = os.getenv('SECRET_KEY')
```

Se usa para firmar las cookies de sesión y evitar su manipulación.

- **Cookies HTTP-Only:**
- Flask marca automáticamente las cookies de sesión como `HttpOnly`, impidiendo su lectura vía JavaScript y reduciendo el riesgo de ataques XSS.



Las sesiones de usuario se protegen mediante una clave secreta (SECRET\_KEY) cargada desde el archivo. env, y configuradas con cookies HTTP-Only, lo que impide el acceso a las cookies mediante scripts del lado del cliente.

### 5.6.2 Control de acceso a rutas

Se implementa un decorador personalizado login\_required que restringe el acceso a rutas que requieren autenticación. Este decorador verifica si la clave 'usuario\_id' está presente en la sesión; en caso contrario, redirige al usuario a la página de login mostrando un mensaje de advertencia.

```
def login_required(f):  
    """Decorador para proteger rutas que requieren autenticación"""  
    @wraps(f)  
    def decorated_function(*args, **kwargs):  
        if 'usuario_id' not in session:  
            flash("Debes iniciar sesión para acceder a esta página", "danger")  
            return redirect(url_for('login'))  
        return f(*args, **kwargs)  
    return decorated_function
```

Además, las rutas administrativas bajo /admin/\* cuentan con verificaciones explícitas de privilegios (if usuario['privilegios'] != 'admin') para asegurar que solo usuarios autorizados puedan acceder a las funciones administrativas.

### 5.6.3 Prevención de ataques de fuerza bruta

El sistema utiliza una estrategia básica de mitigación de fuerza bruta mediante un contador de intentos fallidos de inicio de sesión (session['intentos\_fallidos']). Al alcanzar tres intentos fallidos, el usuario es redirigido automáticamente al flujo de recuperación de contraseña.

### 5.6.4 Gestión segura de contraseñas

Las contraseñas de los usuarios son almacenadas utilizando hashes generados por `werkzeug.security.generate_password_hash`. Para verificar las credenciales, se emplea `check_password_hash`, garantizando que las contraseñas nunca se comparen ni se almacenen en texto plano.

donde se anotan eventos sensibles, como inicios de sesión fallidos, intentos de acceso no autorizados y generación de códigos de recuperación. Esto permite auditar posibles incidentes de seguridad y facilita la detección de patrones anómalos.

### 5.6.5 Expiración de Códigos

- **Verificación de cuenta:**

- El código de verificación se almacena en `usuarios.codigo_verificacion`; la columna `fecha_codigo` no se usa en la verificación inicial, por lo que no expira por defecto.
- Se puede extender el control comparando `fecha_codigo` con `NOW()` si fuese necesario.

- **Recuperación de contraseña:**

- Al generar un código de recuperación se guarda también la marca de tiempo:

```
UPDATE usuarios SET codigo_verificacion = %s, fecha_codigo = NOW() WHERE  
                correo = %s
```

- Al verificar:

```
tiempo_codigo = resultado[1] # fecha_codigo
```

```
if (datetime.now() - tiempo_codigo).total_seconds() <= 1800:
```

```
    # válido
```

```
else:
```

```
    error = "El código ha expirado. Solicita uno nuevo."
```

**Validez:** 30 minutos (1800 segundos).

#### **Cupones de descuento:**

- La columna cupones.fecha\_vencimiento se calcula como NOW() + INTERVAL 30 DAY.

fecha\_vencimiento = (datetime.now() + timedelta(days=30)).strftime('%Y-%m-%d %H:%M:%S')

#### **5.6.6 Registro de eventos críticos**

El sistema mantiene un archivo de registro de seguridad (registro\_seguridad.log) donde se anotan eventos sensibles, como inicios de sesión fallidos, intentos de acceso no autorizados y generación de códigos de recuperación. Esto permite auditar posibles incidentes de seguridad y facilita la detección de patrones anómalos.

- **Configuración inicial:**

```
import logging
```

```
logging.basicConfig(
```

```
    filename='registro_seguridad.log',
```

```
    level=logging.INFO,
```

```
    format='%(asctime)s - %(levelname)s - %(message)s'
```

```
)
```

- **Eventos registrados:**

- Envío de código de verificación:

```
logging.info(f"Código de verificación enviado al correo: {destinatario}")
```

- **Inicio de sesión exitoso:**

```
logging.info(f"Inicio de sesión exitoso para usuario: {correo}")
```

- **Intentos fallidos de login:**

`logging.warning(f'Intento fallido de inicio de sesión para: {correo}')`

- **Fallos al enviar correos:**

`logging.error(f'Error al enviar email de recuperación: {e}')`

Estos registros permiten auditar el comportamiento del sistema, detectar patrones de abuso y depurar problemas en producción.

## 5.7 Justificación de Tecnologías

- **Flask:** Se utilizó Flask por ser un micro-framework ligero y modular que facilita la creación de prototipos funcionales en poco tiempo. Su estructura flexible permite una arquitectura clara y sencilla, lo que resulta adecuado para aplicaciones web pequeñas o medianas.
- **MySQL/InnoDB:** Se seleccionó MySQL como sistema de gestión de base de datos relacional por su robustez, escalabilidad y compatibilidad multiplataforma. El motor de almacenamiento InnoDB proporciona soporte para transacciones, claves foráneas y control de concurrencia, lo cual es esencial para garantizar la integridad de los datos.
- **Pandas & Matplotlib:** Estas bibliotecas fueron empleadas para realizar análisis de datos directamente en Python y generar reportes gráficos automáticos (por sucursal o país) sin depender de herramientas externas. Pandas facilita la manipulación de datos tabulares, mientras que Matplotlib permite la creación de visualizaciones estáticas que se integran fácilmente en el sistema mediante archivos de imagen.
- **Jinja2:** El motor de plantillas Jinja2, incluido por defecto en Flask, permite separar la lógica de negocio de la presentación, generando HTML de forma segura y dinámica. También proporciona protección contra inyecciones HTML, lo cual contribuye a mejorar la seguridad del sistema.
- **dotenv:** La librería python-dotenv permite cargar automáticamente variables de entorno desde un archivo `.env`, facilitando la configuración del

sistema sin exponer datos sensibles como contraseñas, claves secretas o credenciales SMTP directamente en el código fuente.

- **smtplib:** Se optó por smtplib como solución directa para el envío de correos electrónicos, utilizando SMTP\_SSL con autenticación a través de Gmail. Esta elección permite tener un mayor control del proceso de envío, incluyendo mensajes con contenido HTML e imágenes embebidas (por ejemplo, los cupones). Además, ofrece la posibilidad de escalar fácilmente a servicios de correo especializados en el futuro si el sistema crece.

## 5.8 Escalabilidad y Mantenibilidad

El sistema está diseñado con una arquitectura modular que favorece tanto su escalabilidad como su mantenibilidad. A continuación, se describen los elementos clave que respaldan estas cualidades:

### Escalabilidad

- **Separación de responsabilidades:** El código fuente está organizado en archivos específicos para rutas, configuración (config.py), funcionalidades de encriptación y generación de reportes, esta división permite añadir nuevos módulos (por ejemplo, nuevas rutas o modelos) sin afectar directamente el núcleo del sistema.
- **Persistencia y análisis desacoplados:** La base de datos MySQL almacena la información de usuarios, respuestas y cupones, mientras que el análisis y la generación de reportes se realiza con Pandas y Matplotlib, lo que permite escalar el backend de análisis sin modificar el esquema de datos.
- **Recursos estáticos y plantillas:** El uso de carpetas static/ y templates/ puede permitir que la interfaz sea modificable e independiente del backend, lo que facilita integrar frameworks modernos (como React o Vue) si se desea en el futuro.
- **Variables de entorno y configuración externa:** El uso de .env permite mover el sistema a otros entornos (producción, desarrollo, pruebas) sin modificar el código fuente, facilitando su despliegue en servidores remotos o contenedores.

### Mantenibilidad

- **Uso de decoradores reutilizables:** Funcionalidades como el control de acceso (login\_required) están encapsuladas en decoradores reutilizables, lo

que mejora la legibilidad y permite aplicar seguridad a nuevas rutas fácilmente.

- **Estructura clara y comentada:** El archivo principal `app.py` y otros componentes clave están comentados de forma que permiten comprender rápidamente su funcionalidad. Esto facilita el trabajo en equipo y la incorporación de nuevos desarrolladores.
- **Historial de versiones y pruebas planificadas:** Aunque la carpeta `tests/` está vacía, su inclusión anticipa la intención de incorporar pruebas unitarias en el futuro, lo cual es una buena práctica que apunta a una arquitectura mantenible.
- **Uso de convenciones y bibliotecas estándar:** Se emplean bibliotecas ampliamente adoptadas (como `werkzeug.security`, `smtpplib`, `pandas`, `matplotlib`), lo cual facilita el mantenimiento por parte de cualquier desarrollador familiarizado con Python.

## 5.9 Conclusión de la Arquitectura

La arquitectura de Take no Kamu combina solidez, simplicidad y escalabilidad, siguiendo principios de diseño bien establecidos en el desarrollo de software. Cada componente cumple una responsabilidad específica y se comunica mediante interfaces claras: HTTP para la interacción entre el cliente y el servidor, y SQL para la persistencia de datos en la base MySQL.

El uso de un patrón por capas facilita la separación de preocupaciones: la lógica de negocio se mantiene aislada del frontend (plantillas HTML con Jinja2) y del almacenamiento de datos, igualmente se incorporan mecanismos de seguridad estándar como el uso de `SECRET_KEY`, hashing de contraseñas, cookies HTTP-only, y validaciones en rutas protegidas, asegurando tanto la confidencialidad como la integridad de la información.

Desde el punto de vista de mantenibilidad, la estructura modular del sistema, el uso de variables de entorno y el registro de eventos críticos en archivos log permiten detectar, depurar y adaptar el sistema con rapidez ante cambios futuros, la generación automatizada de reportes y cupones, el almacenamiento organizado de imágenes, y la configuración por entorno también contribuyen a una administración eficiente.

Esta base arquitectónica es suficientemente robusta como para permitir nuevas iteraciones sin necesidad de una reestructuración mayor. Entre las extensiones previstas se incluyen:

- Un panel administrativo con gestión de usuarios, sucursales y estadísticas.
- Dashboards de datos en tiempo real con actualizaciones automáticas.
- Escalamiento hacia microservicios para aislar funcionalidades como el envío de correos o la generación de cupones.

Take no Kamu parte de una arquitectura clara y extensible que no solo responde a las necesidades actuales del sistema, sino que también sienta las bases para su crecimiento sostenido y adaptable.

## 6. Código fuente

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
from flask_mysqldb import MySQL
import random
import smtplib
from datetime import datetime, timedelta
from email.message import EmailMessage
import os
from functools import wraps
import hashlib
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg') # Para usar matplotlib en modo no interactivo
import numpy as np
from datetime import datetime, timedelta
from flask import send_from_directory
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
```

```

from email.mime.image import MIMEImage
import string
from dotenv import load_dotenv
load_dotenv() # cargar variables desde .env
from werkzeug.security import generate_password_hash, check_password_hash
import re
import logging

logging.basicConfig(filename='registro_seguridad.log', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

# Definir la ruta de la carpeta para los gráficos
STATIC_FOLDER = os.path.join(os.path.dirname(os.path.abspath(__file__)),
                              'static')
IMAGES_FOLDER = os.path.join(STATIC_FOLDER, 'images')
os.makedirs(IMAGES_FOLDER, exist_ok=True)

app = Flask(__name__)
app.secret_key = os.environ.get('SECRET_KEY', '5mcJAvC298$7')

# Configuración de MySQL
app.config['MYSQL_HOST'] = os.environ.get('MYSQL_HOST', 'localhost')
app.config['MYSQL_USER'] = os.environ.get('MYSQL_USER', 'root')
app.config['MYSQL_PASSWORD'] = os.environ.get('MYSQL_PASSWORD',
'Ericko11$')
app.config['MYSQL_DB'] = os.environ.get('MYSQL_DB', 'encuestas_db')

# Configuración de email

```



```
EMAIL_SENDER = os.environ.get('EMAIL_SENDER', 'takenokamu@gmail.com')
```

```
EMAIL_PASSWORD = os.environ.get('EMAIL_PASSWORD', 'kcdvjijtgcsucvmb')
```

```
mysql = MySQL(app)
```

```
# ----- Funciones auxiliares -----
```

```
# Agregar esta función para generar códigos de cupón aleatorios alfanuméricos
```

```
def generar_codigo_cupon():
```

```
    """Genera un código alfanumérico aleatorio de 6 caracteres para cupones"""
```

```
    caracteres = string.ascii_uppercase + string.digits
```

```
    return "".join(random.choice(caracteres) for _ in range(6))
```

```
# Agregar función para enviar cupón por correo
```

```
def enviar_cupon(destinatario, codigo, porcentaje):
```

```
    """Envía un correo con el cupón de descuento correspondiente"""
```

```
    try:
```

```
        # Crear un mensaje multipart
```

```
        mensaje = MIMEMultipart()
```

```
        mensaje['Subject'] = f'Tu cupón de {porcentaje}% de descuento - Gracias por  
tu encuesta'
```

```
        mensaje['From'] = EMAIL_SENDER
```

```
        mensaje['To'] = destinatario
```

```
    # Texto del correo
```

```
    texto = f'''
```

```
    Hola,
```

¡Gracias por responder nuestra encuesta de satisfacción!

Como agradecimiento, te enviamos un cupón de {porcentaje}% de descuento para tu próxima visita.

Tu código de cupón es: {codigo}

Este cupón es válido por 30 días a partir de hoy.

¡Esperamos verte pronto!

Saludos,

El equipo de Encuestas

'''

```
# Agregar parte de texto al mensaje
```

```
parte_texto = MIMEText(texto)
```

```
mensaje.attach(parte_texto)
```

```
# Agregar imagen del cupón
```

```
ruta_imagen = os.path.join(STATIC_FOLDER, 'images',  
f'cupon_{porcentaje}.png')
```

```
if os.path.exists(ruta_imagen):
```

```
    with open(ruta_imagen, 'rb') as img:
```

```
        imagen = MIMEImage(img.read())
```

```
        imagen.add_header('Content-ID', '<cupon>')
```

```

        imagen.add_header('Content-Disposition', 'inline',
filename=f'cupon_{porcentaje}.png')
        mensaje.attach(imagen)

# Enviar correo
with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
    smtp.login(EMAIL_SENDER, EMAIL_PASSWORD)
    smtp.send_message(mensaje)
    return True
except Exception as e:
    print(f"Error al enviar email con cupón: {e}")
    return False

```

```

def hash_password(password):
    return generate_password_hash(password)

```

```

def generar_codigo_verificacion():
    """Genera un código de verificación de 6 dígitos"""
    return str(random.randint(100000, 999999))

```

```

def enviar_codigo_verificacion(destinatario, codigo):
    """Envía un correo con el código de verificación"""
    try:
        mensaje = EmailMessage()
        mensaje['Subject'] = 'Tu código de verificación - Encuesta de Satisfacción'
        mensaje['From'] = EMAIL_SENDER

```

```
mensaje['To'] = destinatario
```

```
mensaje.set_content(f"
```

```
Hola,
```

```
Gracias por registrarte en nuestro sistema de encuestas de satisfacción.
```

```
Tu código de verificación es: {codigo}
```

```
Si no solicitaste este código, puedes ignorar este correo.
```

```
Saludos,
```

```
El equipo de Encuestas
```

```
")
```

```
with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
```

```
    smtp.login(EMAIL_SENDER, EMAIL_PASSWORD)
```

```
    smtp.send_message(mensaje)
```

```
logging.info(f"Código de verificación enviado al correo: {destinatario}")
```

```
return True
```

```
except Exception as e:
```

```
    logging.error(f"Error al enviar email de verificación: {e}")
```

```
return False
```

```
def login_required(f):
```

```
    """Decorador para proteger rutas que requieren autenticación"""
```

```
    @wraps(f)
```

```
    def decorated_function(*args, **kwargs):
```

```

if 'usuario_id' not in session:

    flash("Debes iniciar sesión para acceder a esta página", "danger")

    return redirect(url_for('login'))

    return f(*args, **kwargs)

return decorated_function

```

# Modifica la función verificar\_actividad\_reciente para incluir verificación de cupon

```

def verificar_actividad_reciente(id_usuario):

    """Verifica si el usuario ha respondido la encuesta en los últimos 30 días"""

    cur = mysql.connection.cursor()

    cur.execute("""

        SELECT fecha_respuesta FROM respuestas

        WHERE id_usuario = %s

        ORDER BY fecha_respuesta DESC

        LIMIT 1

    """, (id_usuario,))

    ultima_respuesta = cur.fetchone()

    if ultima_respuesta:

        fecha_ultima = ultima_respuesta[0]

        hace_un_mes = datetime.now() - timedelta(days=30)

        if fecha_ultima > hace_un_mes:

            return True

    return False

```

```

def enviar_codigo_recuperacion(destinatario, codigo):

```

"""Envía un correo con el código de recuperación de contraseña"""

try:

```
mensaje = EmailMessage()
```

```
mensaje['Subject'] = 'Recuperación de Contraseña - Encuesta de Satisfacción'
```

```
mensaje['From'] = EMAIL_SENDER
```

```
mensaje['To'] = destinatario
```

```
mensaje.set_content(f"""
```

```
Hola,
```

Has solicitado recuperar tu contraseña en nuestro sistema de encuestas de satisfacción.

Tu código de verificación es: {codigo}

Este código expirará en 30 minutos.

Si no solicitaste este código, puedes ignorar este correo.

Saludos,

El equipo de Encuestas

```
""")
```

```
with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
```

```
    smtp.login(EMAIL_SENDER, EMAIL_PASSWORD)
```

```
    smtp.send_message(mensaje)
```

```
logging.info(f"Código de recuperación enviado al correo: {destinatario}")
```

```
return True
```

```
except Exception as e:
```

```
    logging.error(f"Error al enviar email de recuperación: {e}")
```

```
    return False
```

```
# --- Funciones auxiliares para gráficos ---
```

```
def generar_grafico_barras(datos, etiquetas, titulo, etiqueta_x, etiqueta_y,  
    nombre_archivo, promedio=None):
```

```
    """
```

```
    Genera un gráfico de barras
```

```
    Args:
```

```
        datos: Lista de valores numéricos para las barras
```

```
        etiquetas: Lista de etiquetas para las barras
```

```
        titulo: Título del gráfico
```

```
        etiqueta_x: Etiqueta del eje X
```

```
        etiqueta_y: Etiqueta del eje Y
```

```
        nombre_archivo: Nombre del archivo a guardar (sin extensión)
```

```
        promedio: Valor del promedio para dibujar una línea horizontal (opcional)
```

```
    """
```

```
    plt.figure(figsize=(12, 6))
```

```
    plt.bar(etiquetas, datos, color='#335435', alpha=0.8)
```

```
    if promedio is not None:
```

```
        plt.axhline(y=promedio, color='r', linestyle='--',
```

```
                    label=f'Promedio: {promedio}')
```

```
    plt.legend()
```

```

plt.ylim(0, max(datos) * 1.1 if datos else 5.5)
plt.xticks(rotation=45, ha='right')
plt.ylabel(etiqueta_y)
plt.xlabel(etiqueta_x)
plt.title(titulo)
plt.tight_layout()
plt.savefig(os.path.join(IMAGES_FOLDER, f'{nombre_archivo}.png'))
plt.close()

```

```

def generar_grafico_pastel(datos, etiquetas, titulo, nombre_archivo,
colores=None):

```

```

    """

```

Genera un gráfico de pastel

Args:

datos: Lista de valores numéricos para las porciones

etiquetas: Lista de etiquetas para las porciones

titulo: Título del gráfico

nombre\_archivo: Nombre del archivo a guardar (sin extensión)

colores: Lista de colores para las porciones (opcional)

```

    """

```

```

    if not colores:

```

```

        colores = ['#4CAF50', '#F44336', '#2196F3', '#FFC107', '#9C27B0', '#FF5722']

```

```

plt.figure(figsize=(8, 8))

```

```

plt.pie(datos, labels=etiquetas, colors=colores, autopct='%1.1f%%',

```



```

        startangle=90, shadow=True)

plt.axis('equal')

plt.title(titulo)

plt.tight_layout()

plt.savefig(os.path.join(IMAGES_FOLDER, f'{nombre_archivo}.png'))

plt.close()

def generar_grafico_histograma(datos, bins, etiquetas_bins, titulo, etiqueta_x,
etiqueta_y, nombre_archivo):
    """
    Genera un histograma

    Args:
        datos: Serie de pandas con los datos
        bins: Lista con los límites de los bins
        etiquetas_bins: Lista con las etiquetas de los ejes X
        titulo: Título del gráfico
        etiqueta_x: Etiqueta del eje X
        etiqueta_y: Etiqueta del eje Y
        nombre_archivo: Nombre del archivo a guardar (sin extensión)
    """

    plt.figure(figsize=(10, 6))
    plt.hist(datos, bins=bins, rwidth=0.8, color='#335435', alpha=0.8)
    plt.xticks(etiquetas_bins)
    plt.xlabel(etiqueta_x)
    plt.ylabel(etiqueta_y)
    plt.title(titulo)

```

```
plt.tight_layout()
plt.savefig(os.path.join(IMAGES_FOLDER, f'{nombre_archivo}.png'))
plt.close()
```

# ---- Rutas ----

```
@app.route('/')
def index():
    """Ruta principal que redirecciona al login"""
    return redirect(url_for('login'))
```

"""

### 3. MODIFICAR LA RUTA DE LOGIN

-----

Reemplaza la parte de verificación de contraseña en la ruta /login:

"""

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    """Ruta para el inicio de sesión"""
    error = None
    usuario = None

    if request.method == 'POST':
        correo = request.form.get('email', "").strip()
        contrasena = request.form.get('password', "")

        if not correo or not contrasena:
```

```

        error = "Por favor, completa todos los campos."
        return render_template('login.html', error=error)

# Validar formato de correo electrónico
if not re.match(r"^[^@]+@^[^@]+\.[^@]+$", correo):
    error = "Correo electrónico inválido"
    return render_template('login.html', error=error)

cur = mysql.connection.cursor()
cur.execute("""
    SELECT id_usuario, nombre, verificado, privilegios, contraseña,
codigo_verificacion
    FROM usuarios
    WHERE correo = %s
""", (correo,))
usuario = cur.fetchone()
cur.close()

# Verificamos si el usuario existe
if usuario:
    # Primero verificamos si el usuario no está verificado pero tiene un código
    if usuario[2] == 0 and usuario[5]: # verificado=0 y tiene código de
verificación
        session['correo_verificacion'] = correo
        flash("Tu cuenta aún no ha sido verificada. Te hemos redirigido a la
página de verificación.", "warning")
        return redirect(url_for('verificar_codigo'))

```

```

# Ahora verificamos la contraseña con check_password_hash
if check_password_hash(usuario[4], contrasena):
    if usuario[2] == 1: # verificado
        # Reiniciar contador de intentos fallidos si existe
        if 'intentos_fallidos' in session:
            session.pop('intentos_fallidos')
        if 'email_intentos' in session:
            session.pop('email_intentos')

        session['usuario_id'] = usuario[0]
        session['nombre'] = usuario[1]
        session['privilegios'] = usuario[3]

        logging.info(f'Inicio de sesión exitoso para usuario: {correo}')

    # Redireccionar según privilegios
    if usuario[3] == 1: # Específicamente si es administrador (privilegio =
1)
        return redirect(url_for('admin_reportes_sucursal'))
    else: # Usuario normal
        return redirect(url_for('encuesta'))
else:
    # Usuario no verificado, lo redirigimos a la verificación
    session['correo_verificacion'] = correo

    flash("Tu cuenta aún no ha sido verificada. Por favor completa la
verificación.", "warning")
    return redirect(url_for('verificar_codigo'))
else:

```

```

# Contraseña incorrecta

logging.warning(f"Intento fallido de inicio de sesión para el correo:
{correo}")

# Incrementar contador de intentos fallidos

if 'intentos_fallidos' not in session or 'email_intentos' not in session or
session['email_intentos'] != correo:

    session['intentos_fallidos'] = 1

    session['email_intentos'] = correo

else:

    session['intentos_fallidos'] += 1

# Verificar si se han superado los intentos máximos
if session.get('intentos_fallidos', 0) >= 3:

    session['correo_recuperacion'] = correo

    # Reiniciar contador

    session.pop('intentos_fallidos', None)

    session.pop('email_intentos', None)

    logging.warning(f"Múltiples intentos fallidos de inicio de sesión para:
{correo}. Redirigiendo a recuperación.")

    return redirect(url_for('recuperar_contrasena'))

intentos_restantes = 3 - session.get('intentos_fallidos', 0)

error = f"Contraseña incorrecta. Te quedan {intentos_restantes}
intentos."

else:

    error = "No existe una cuenta con ese correo electrónico."

```

```
return render_template('login.html', error=error)
```

```
@app.route('/registro', methods=['GET', 'POST'])
```

```
def registro():
```

```
    """Ruta para el registro de usuarios"""
```

```
    error = None
```

```
    if request.method == 'POST':
```

```
        nombre = request.form.get('nombre', "").strip()
```

```
        apellido = request.form.get('apellido', "").strip()
```

```
        correo = request.form.get('correo', "").strip()
```

```
        telefono = request.form.get('telefono', "").strip()
```

```
        contrasena = request.form.get('contrasena', "")
```

```
    # Validaciones básicas
```

```
    if not all([nombre, apellido, correo, telefono, contrasena]):
```

```
        error = "Todos los campos son obligatorios"
```

```
        return render_template('registro.html', error=error)
```

```
    # Validar formato de correo electrónico
```

```
    if not re.match(r"^[^@]+@[^@]+\.[^@]+", correo):
```

```
        error = "Correo electrónico inválido"
```

```
        return render_template('registro.html', error=error)
```

```
    cur = mysql.connection.cursor()
```

```
    cur.execute("SELECT * FROM usuarios WHERE correo = %s", (correo,))
```

```

if cur.fetchone():
    error = "El correo ya está registrado"
else:
    codigo = generar_codigo_verificacion()
    # Almacenar contraseña hasheada con el nuevo método
    contrasena_hash = hash_password(contrasena)

    cur.execute("""
        INSERT INTO usuarios (privilegios, nombre, apellido, correo, telefono,
        contraseña, codigo_verificacion)
        VALUES (%s, %s, %s, %s, %s, %s, %s)
        """, (0, nombre, apellido, correo, telefono, contrasena_hash, codigo))
    mysql.connection.commit()
    cur.close()

    if enviar_codigo_verificacion(correo, codigo):
        logging.info(f"Nuevo usuario registrado: {correo}")
        session['correo_verificacion'] = correo
        return redirect(url_for('verificar_codigo'))
    else:
        error = "Error al enviar el código de verificación. Intenta de nuevo."

cur.close()

return render_template('registro.html', error=error)

```

```

@app.route('/verificar_codigo', methods=['GET', 'POST'])
def verificar_codigo():
    """Ruta para verificar el código enviado por correo"""
    mensaje = None
    verificado = False

    if 'correo_verificacion' not in session:
        return redirect(url_for('registro'))

    correo = session.get('correo_verificacion')

    # Verificar si necesitamos reenviar un código (cuando llegamos desde login)
    reenviar = request.args.get('reenviar', 'false') == 'true'
    if request.method == 'GET' and reenviar:
        codigo = generar_codigo_verificacion()
        cur = mysql.connection.cursor()
        cur.execute("""
            UPDATE usuarios
            SET codigo_verificacion = %s
            WHERE correo = %s
            """, (codigo, correo))
        mysql.connection.commit()
        cur.close()

    if enviar_codigo_verificacion(correo, codigo):
        mensaje = "Se ha enviado un nuevo código de verificación a tu correo."
    else:

```



```

    mensaje = "Error al enviar el código de verificación. Intenta nuevamente."

if request.method == 'POST':
    codigo_usuario = request.form.get('codigo', "").strip()

    if not codigo_usuario:
        mensaje = "Por favor, ingresa el código de verificación"
    else:
        cur = mysql.connection.cursor()
        cur.execute("SELECT codigo_verificacion FROM usuarios WHERE correo
= %s", (correo,))
        resultado = cur.fetchone()

        if resultado and resultado[0] == codigo_usuario:
            cur.execute("""
            UPDATE usuarios
            SET codigo_verificacion = NULL, verificado = TRUE
            WHERE correo = %s
            """, (correo,))
            mysql.connection.commit()
            mensaje = "¡Cuenta verificada exitosamente!"
            verificado = True
            session.pop('correo_verificacion', None)
            flash("Tu cuenta ha sido verificada. Ahora puedes iniciar sesión.",
"success")
            logging.info(f"Cuenta verificada exitosamente para: {correo}")
            return redirect(url_for('login'))
        else:

```

```
mensaje = "Código incorrecto. Intenta nuevamente."
```

```
logging.warning(f"Intento fallido de verificación para el correo: {correo}")
```

```
cur.close()
```

```
# Agregamos un botón en la plantilla para reenviar el código
```

```
return render_template('verificar_codigo.html', mensaje=mensaje,  
verificado=verificado, correo=correo)
```

```
@app.route('/logout')
```

```
def logout():
```

```
    """Cierra la sesión del usuario"""
```

```
    session.clear()
```

```
    flash("Has cerrado sesión correctamente", "success")
```

```
    return redirect(url_for('login'))
```

```
# Modifica la ruta /encuesta para que se genere y envíe el cupón
```

```
@app.route('/encuesta', methods=['GET', 'POST'])
```

```
@login_required
```

```
def encuesta():
```

```
    """Ruta para responder la encuesta"""
```

```
    id_usuario = session['usuario_id']
```

```
# Verificar si el usuario ya respondió en los últimos 30 días
```

```
if verificar_actividad_reciente(id_usuario):
```

```
flash("Ya has respondido la encuesta este mes. ¡Gracias!", "warning")  
return redirect(url_for('agradecimiento'))
```

```
# Procesar envío del formulario
```

```
if request.method == 'POST':
```

```
    try:
```

```
        # Obtener datos del formulario con validación
```

```
        pais = request.form.get('pais', "").strip()
```

```
        sucursal = request.form.get('sucursal', "").strip()
```

```
        calidad_comida = int(request.form.get('calidad_comida', 0))
```

```
        tiempo_espera = request.form.get('tiempo_espera', "").strip()
```

```
        atencion_personal = int(request.form.get('atencion_personal', 0))
```

```
        agrado_sucursal = request.form.get('agrado_sucursal', "").strip()
```

```
        volveria_visitar = request.form.get('volveria_visitar', "").strip()
```

```
        area_mejora = request.form.get('area_mejora', "").strip()
```

```
        calificacion_general = int(request.form.get('calificacion_general', 0))
```

```
# Validación básica
```

```
if not all([pais, sucursal, tiempo_espera, agrado_sucursal, volveria_visitar]):
```

```
    flash("Por favor, complete todos los campos requeridos", "danger")
```

```
    return redirect(url_for('encuesta'))
```

```
# Guardar respuesta en la base de datos
```

```
cur = mysql.connection.cursor()
```

```
cur.execute("""
```

```
    INSERT INTO respuestas (
```

```
        id_usuario, pais, sucursal, calidad_comida, tiempo_espera,
```

```

        atencion_personal, agrado_sucursal, volveria_visitar,
        area_mejora, calificacion_general, fecha_respuesta
    ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, NOW())
""" , (
    id_usuario, pais, sucursal, calidad_comida, tiempo_espera,
    atencion_personal, agrado_sucursal, volveria_visitar,
    area_mejora, calificacion_general
))
mysql.connection.commit()

# Generar cupón aleatorio (30%, 35% o 40%)
porcentajes = [30, 35, 40]
porcentaje_elegido = random.choice(porcentajes)

# Generar código de cupón único
while True:
    codigo_cupon = generar_codigo_cupon()
    # Verificar que el código no exista ya
    cur.execute("SELECT id_cupon FROM cupones WHERE codigo = %s",
(codigo_cupon,))
    if not cur.fetchone():
        break

# Calcular fecha de vencimiento (30 días desde hoy)
fecha_vencimiento = (datetime.now() + timedelta(days=30)).strftime('%Y-
%m-%d %H:%M:%S')

# Guardar cupón en la base de datos

```

```

cur.execute("""
    INSERT INTO cupones (
        id_usuario, codigo, porcentaje, fecha_vencimiento
    ) VALUES (%s, %s, %s, %s)
""", (
    id_usuario, codigo_cupon, porcentaje_elegido, fecha_vencimiento
))

mysql.connection.commit()

# Obtener email del usuario

cur.execute("SELECT correo FROM usuarios WHERE id_usuario = %s",
(id_usuario,))

correo_usuario = cur.fetchone()[0]

cur.close()

# Enviar cupón por correo

enviar_cupon(correo_usuario, codigo_cupon, porcentaje_elegido)

flash("¡Gracias por tu respuesta! Te hemos enviado un cupón de
descuento.", "success")

return redirect(url_for('agradecimiento'))

except Exception as e:

    print(f"Error al guardar la respuesta: {e}")

    flash("Ocurrió un error. Intenta nuevamente.", "danger")

    return redirect(url_for('encuesta'))

```

```
return render_template('encuesta.html', nombre=session.get('nombre', ""))
```

```
@app.route('/agradecimiento')
```

```
@login_required
```

```
def agradecimiento():
```

```
    """Página de agradecimiento después de completar la encuesta"""
```

```
    return render_template('agradecimiento.html')
```

```
@app.route('/admin/reportes_sucursal')
```

```
@login_required
```

```
def admin_reportes_sucursal():
```

```
    """Página para visualizar reportes por sucursal"""
```

```
    # Verificar si el usuario es administrador
```

```
    if session.get('privilegios', 0) != 1: # Específicamente verificar si es  
administrador (privilegio = 1)
```

```
        flash("No tienes permiso para acceder a esta página", "danger")
```

```
        return redirect(url_for('encuesta'))
```

```
    # Obtener parámetros de filtro
```

```
    pais_seleccionado = request.args.get('pais', "")
```

```
    sucursal_seleccionada = request.args.get('sucursal', "")
```

```
    fecha_inicio = request.args.get('fecha_inicio', "")
```

```
    fecha_fin = request.args.get('fecha_fin', "")
```

```
    # Si no hay fechas seleccionadas, usar último mes
```

```
    if not fecha_inicio:
```

```

    fecha_inicio = (datetime.now() - timedelta(days=30)).strftime('%Y-%m-%d')
if not fecha_fin:
    fecha_fin = datetime.now().strftime('%Y-%m-%d')

# Consultar datos
cur = mysql.connection.cursor()

# Lista de países disponibles
cur.execute("SELECT DISTINCT pais FROM respuestas ORDER BY pais")
paises = [row[0] for row in cur.fetchall()]

# Lista de sucursales (filtradas por país si está seleccionado)
if pais_seleccionado:
    cur.execute("SELECT DISTINCT sucursal FROM respuestas WHERE pais = %s ORDER BY sucursal", (pais_seleccionado,))
else:
    cur.execute("SELECT DISTINCT sucursal FROM respuestas ORDER BY sucursal")
sucursales = [row[0] for row in cur.fetchall()]

# Construir consulta base con filtros para resumen
query_base = """
    SELECT r.*, u.nombre, u.apellido
    FROM respuestas r
    JOIN usuarios u ON r.id_usuario = u.id_usuario
    WHERE 1=1
"""

params = []

```

```
if pais_seleccionado:
```

```
    query_base += " AND r.pais = %s"
```

```
    params.append(pais_seleccionado)
```

```
if sucursal_seleccionada:
```

```
    query_base += " AND r.sucursal = %s"
```

```
    params.append(sucursal_seleccionada)
```

```
if fecha_inicio:
```

```
    query_base += " AND r.fecha_respuesta >= %s"
```

```
    params.append(fecha_inicio)
```

```
if fecha_fin:
```

```
    query_base += " AND r.fecha_respuesta <= %s"
```

```
    params.append(fecha_fin + " 23:59:59")
```

```
# Obtener datos con filtros
```

```
cur.execute(query_base + " ORDER BY r.fecha_respuesta DESC",  
tuple(params))
```

```
datos = cur.fetchall()
```

```
# Cerrar cursor
```

```
cur.close()
```

```
# Procesar datos con pandas
```

```
if datos:
```



```

# Nombres de columnas según tu estructura de tablas
columnas = ['id_respuesta', 'pais', 'sucursal', 'calidad_comida',
            'tiempo_espera', 'atencion_personal', 'agrado_sucursal',
            'volveria_visitar', 'area_mejora', 'calificacion_general',
            'fecha_respuesta', 'id_usuario', 'nombre', 'apellido']

# Crear DataFrame
df = pd.DataFrame(datos, columns=columnas)

# Convertir explícitamente las columnas numéricas
df['calificacion_general'] = pd.to_numeric(df['calificacion_general'],
errors='coerce')

df['calidad_comida'] = pd.to_numeric(df['calidad_comida'], errors='coerce')

df['atencion_personal'] = pd.to_numeric(df['atencion_personal'],
errors='coerce')

# Calcular métricas asegurando que los valores son numéricos
total_respuestas = len(df)

# Calcular promedios solo con valores válidos
promedio_calificacion = round(df['calificacion_general'].mean(), 1) if not
df['calificacion_general'].isna().all() else 0

satisfaccion_servicio = round(df['atencion_personal'].mean() * 10, 1) if not
df['atencion_personal'].isna().all() else 0

# Para volveria_visitar, contar las respuestas afirmativas
tasa_retorno = round((df['volveria_visitar'] == 'si').sum() /
df['volveria_visitar'].count() * 100, 1) if df['volveria_visitar'].count() > 0 else 0

```

```

# Procesar datos por sucursal
datos_sucursales = []

# Agrupar por sucursal
sucursales_grupo = df.groupby('sucursal')

for sucursal, grupo in sucursales_grupo:

    # Calcular indicadores para cada sucursal
    total_sucursal = len(grupo)

    # Calidad comida (promedio)
    calidad_promedio = round(grupo['calidad_comida'].mean(), 1) if not
grupo['calidad_comida'].isna().all() else 0

    # Atención personal (promedio)
    atencion_promedio = round(grupo['atencion_personal'].mean(), 1) if not
grupo['atencion_personal'].isna().all() else 0

    # Tiempo de espera (porcentaje de respuestas 'si')
    tiempo_adequado = round((grupo['tiempo_espera'] == 'si').sum() /
grupo['tiempo_espera'].count() * 100, 1) if grupo['tiempo_espera'].count() > 0 else 0

    # Calificación general (promedio)
    calificacion_promedio = round(grupo['calificacion_general'].mean(), 1) if not
grupo['calificacion_general'].isna().all() else 0

    # Intención de retorno (porcentaje de respuestas 'si')
    intencion_retorno = round((grupo['volveria_visitar'] == 'si').sum() /
grupo['volveria_visitar'].count() * 100, 1) if grupo['volveria_visitar'].count() > 0 else 0

```

```

datos_sucursales.append({
    'sucursal': sucursal,
    'total_respuestas': total_sucursal,
    'calidad_comida': calidad_promedio,
    'atencion_personal': atencion_promedio,
    'tiempo_espera': tiempo_adequado,
    'calificacion_general': calificacion_promedio,
    'intencion_retorno': intencion_retorno
})

```

# Ordenar por calificación general (descendente)

```

datos_sucursales = sorted(datos_sucursales, key=lambda x:
x['calificacion_general'], reverse=True)

```

# 1. Generar gráfico de calificaciones por sucursal

if total\_respuestas > 0:

```

plt.figure(figsize=(12, 6))

```

# Limitamos a las 10 primeras sucursales si hay muchas

```

sucursales_plot = datos_sucursales[:10] if len(datos_sucursales) > 10 else
datos_sucursales

```

# Crear listas para el gráfico

```

nombres_sucursales = [item['sucursal'] for item in sucursales_plot]

```

```

calificaciones = [item['calificacion_general'] for item in sucursales_plot]

```

```

plt.bar(nombres_sucursales, calificaciones, color='#335435', alpha=0.8)

```

```

plt.axhline(y=promedio_calificacion, color='r', linestyle='--',
            label=f'Promedio: {promedio_calificacion}')
plt.ylim(0, 5.5)
plt.xticks(rotation=45, ha='right')
plt.ylabel('Calificación Promedio')
plt.title('Calificación Promedio por Sucursal')
plt.legend()
plt.tight_layout()
plt.savefig(os.path.join(IMAGES_FOLDER, 'chart_sucursales.png'))
plt.close()

```

# 2. Generar gráfico de distribución de calificaciones

```
valid_ratings = df['calificacion_general'].dropna()
```

```
if len(valid_ratings) > 0:
```

```
    plt.figure(figsize=(10, 6))
```

```
    # Crear histograma de calificaciones con valores válidos
```

```
    plt.hist(valid_ratings, bins=[0.5, 1.5, 2.5, 3.5, 4.5, 5.5],
```

```
            rwidth=0.8, color='#335435', alpha=0.8)
```

```
    plt.xticks([1, 2, 3, 4, 5])
```

```
    plt.xlabel('Calificación')
```

```
    plt.ylabel('Número de Respuestas')
```

```
    plt.title('Distribución de Calificaciones Generales')
```

```
    plt.tight_layout()
```

```
    plt.savefig(os.path.join(IMAGES_FOLDER, 'chart_distribucion.png'))
```

```
    plt.close()
```

```

# 3. Generar gráfico de pastel para intención de retorno

plt.figure(figsize=(8, 8))

# Contar valores de volveria_visitar
volveria_counts = df['volveria_visitar'].value_counts()

# Asegurar que 'si' y 'no' existan en el conteo
si_count = volveria_counts.get('si', 0)
no_count = volveria_counts.get('no', 0)

counts = [si_count, no_count]

labels = ['Sí', 'No']

colors = ['#4CAF50', '#F44336']

plt.pie(counts, labels=labels, colors=colors, autopct='%1.1f%%',
startangle=90, shadow=True)

plt.axis('equal') # Para que el gráfico sea circular
plt.title('Intención de Retorno')
plt.tight_layout()
plt.savefig(os.path.join(IMAGES_FOLDER, 'chart_retorno.png'))
plt.close()

# Preparar datos para la tabla de respuestas individuales
# Formatear la columna de fecha para mostrarla mejor
df['fecha_formateada'] = df['fecha_respuesta'].apply(lambda x:
x.strftime('%d/%m/%Y %H:%M') if pd.notnull(x) else "")

# Crear lista de respuestas para mostrar en la tabla
respuestas_tabla = []
for _, row in df.iterrows():
    respuestas_tabla.append({

```

```

'nombre': f'{row['nombre']} {row['apellido']}',
'fecha': row['fecha_formateada'],
'calidad_comida': row['calidad_comida'],
'tiempo_espera': 'Sí' if row['tiempo_espera'] == 'si' else 'No',
'atencion_personal': row['atencion_personal'],
'agrado_sucursal': 'Sí' if row['agrado_sucursal'] == 'si' else 'No',
'volveria_visitar': 'Sí' if row['volveria_visitar'] == 'si' else 'No',
'area_mejora': row['area_mejora'],
'calificacion_general': row['calificacion_general']
})

```

else:

```

# Sin datos
total_respuestas = 0
promedio_calificacion = 0
satisfaccion_servicio = 0
tasa_retorno = 0
datos_sucursales = []
respuestas_tabla = []

```

```

return render_template('reportes_admin.html',
    paises=paises,
    sucursales=sucursales,
    pais_seleccionado=pais_seleccionado,
    sucursal_seleccionada=sucursal_seleccionada,
    fecha_inicio=fecha_inicio,
    fecha_fin=fecha_fin,

```

```

        total_respuestas=total_respuestas,
        promedio_calificacion=promedio_calificacion,
        satisfaccion_servicio=satisfaccion_servicio,
        tasa_retorno=tasa_retorno,
        datos_sucursales=datos_sucursales,
        respuestas=respuestas_tabla) # Enviar los datos de respuestas
individuales

```

```

@app.route('/recuperar_contrasena', methods=['GET', 'POST'])

```

```

def recuperar_contrasena():

```

```

    """Ruta para solicitar recuperación de contraseña"""

```

```

    error = None

```

```

    mensaje = None

```

```

    # Verificar si el usuario fue redirigido tras intentos fallidos

```

```

    correo_redirect = session.get('correo_recuperacion', "")

```

```

    if request.method == 'POST':

```

```

        correo = request.form.get('email', "").strip()

```

```

        if not correo:

```

```

            error = "Por favor, ingresa tu correo electrónico"

```

```

        elif not re.match(r"^[^@]+@^[^@]+\.[^@]+$", correo):

```

```

            error = "Correo electrónico inválido"

```

```

        else:

```

```

            cur = mysql.connection.cursor()

```

```

            cur.execute("SELECT id_usuario FROM usuarios WHERE correo = %s",
(correo,))

```

```

usuario = cur.fetchone()

if usuario:

    # Generar código de verificación
    codigo = generar_codigo_verificacion()

    # Guardar código en la base de datos
    cur.execute("""
UPDATE usuarios
SET codigo_verificacion = %s, fecha_codigo = NOW()
WHERE correo = %s
""", (codigo, correo))
    mysql.connection.commit()

    # Enviar correo con código
    if enviar_codigo_recuperacion(correo, codigo):
        session['correo_recuperacion'] = correo
        mensaje = "Se ha enviado un código de verificación a tu correo"
        logging.info(f"Solicitud de recuperación de contraseña para: {correo}")
        return redirect(url_for('verificar_codigo_recuperacion'))
    else:
        error = "Error al enviar el código. Intenta nuevamente."
else:
    error = "No existe una cuenta con ese correo electrónico"
    logging.warning(f"Intento de recuperación para correo inexistente:
{correo}")

```



```

cur.close()

# Si el usuario fue redirigido desde login por intentos fallidos y aún no se ha
enviado el formulario

elif correo_redirect and request.method == 'GET':

    cur = mysql.connection.cursor()

    cur.execute("SELECT id_usuario FROM usuarios WHERE correo = %s",
(correo_redirect,))

    usuario = cur.fetchone()

if usuario:

    # Generar código de verificación

    codigo = generar_codigo_verificacion()

    # Guardar código en la base de datos

    cur.execute("""
UPDATE usuarios
SET codigo_verificacion = %s, fecha_codigo = NOW()
WHERE correo = %s
""", (codigo, correo_redirect))

    mysql.connection.commit()

    # Enviar correo con código

    if enviar_codigo_recuperacion(correo_redirect, codigo):

        mensaje = f"Se ha enviado un código de verificación a {correo_redirect}"

        logging.info(f"Solicitud de recuperación de contraseña (redireccionada)
para: {correo_redirect}")

        return redirect(url_for('verificar_codigo_recuperacion'))

```

```

        else:
            error = "Error al enviar el código. Intenta nuevamente."
    else:
        error = "No existe una cuenta con ese correo electrónico"

    cur.close()

    return render_template('recuperar_contrasena.html', error=error,
mensaje=mensaje, correo_prefill=correo_redirect)

@app.route('/verificar_codigo_recuperacion', methods=['GET', 'POST'])
def verificar_codigo_recuperacion():
    """Ruta para verificar el código de recuperación"""
    error = None

    if 'correo_recuperacion' not in session:
        return redirect(url_for('recuperar_contrasena'))

    if request.method == 'POST':
        codigo = request.form.get('codigo', "").strip()
        correo = session.get('correo_recuperacion')

        if not codigo:
            error = "Por favor, ingresa el código de verificación"
        else:
            cur = mysql.connection.cursor()
            cur.execute("""

```

```

SELECT codigo_verificacion, fecha_codigo
FROM usuarios
WHERE correo = %s
"""', (correo,))
resultado = cur.fetchone()

if resultado and resultado[0] == codigo:

    # Verificar que el código no tenga más de 30 minutos
    tiempo_actual = datetime.now()
    tiempo_codigo = resultado[1]

    if tiempo_codigo and (tiempo_actual - tiempo_codigo).total_seconds() <=
1800:

        # Marcar el código como verificado
        session['codigo_verificado'] = True
        logging.info(f"Código de recuperación verificado para: {correo}")
        return redirect(url_for('nueva_contrasena'))
    else:
        error = "El código ha expirado. Solicita uno nuevo."
        logging.warning(f"Intento de usar código expirado para: {correo}")
    else:
        error = "Código incorrecto. Intenta nuevamente."
        logging.warning(f"Intento fallido de verificación para recuperación de
contraseña: {correo}")

cur.close()

return render_template('verificar_codigo_recuperacion.html', error=error)

```

```

@app.route('/nueva_contrasena', methods=['GET', 'POST'])
def nueva_contrasena():
    """Ruta para establecer nueva contraseña"""
    error = None

    if 'correo_recuperacion' not in session or 'codigo_verificado' not in session:
        return redirect(url_for('recuperar_contrasena'))

    if request.method == 'POST':
        correo = session.get('correo_recuperacion')
        nueva_contrasena = request.form.get('nueva_contrasena', "")
        confirmar_contrasena = request.form.get('confirmar_contrasena', "")

        if not nueva_contrasena or not confirmar_contrasena:
            error = "Por favor, completa todos los campos"
        elif nueva_contrasena != confirmar_contrasena:
            error = "Las contraseñas no coinciden"
        else:
            # Actualizar contraseña en la base de datos
            contraseña_hash = hash_password(nueva_contrasena)

            cur = mysql.connection.cursor()
            cur.execute("""
                UPDATE usuarios
                SET contraseña = %s, codigo_verificacion = NULL, fecha_codigo =
                NULL
            """)

```

```

        WHERE correo = %s
        """ , (contrasena_hash, correo))
    mysql.connection.commit()
    cur.close()

    # Limpiar variables de sesión
    session.pop('correo_recuperacion', None)
    session.pop('codigo_verificado', None)

    logging.info(f"Contraseña actualizada exitosamente para: {correo}")
    flash("Tu contraseña ha sido actualizada exitosamente", "success")
    return redirect(url_for('login'))

return render_template('nueva_contrasena.html', error=error)

@app.errorhandler(404)
def pagina_no_encontrada(error):
    """Manejador para errores 404"""
    return render_template('404.html'), 404

@app.errorhandler(500)
def error_servidor(error):
    """Manejador para errores 500"""
    return render_template('500.html'), 500

if __name__ == "__main__":
    # Usa las variables de entorno PORT si está disponible (para Render)

```

```
port = int(os.environ.get('PORT', 5000))

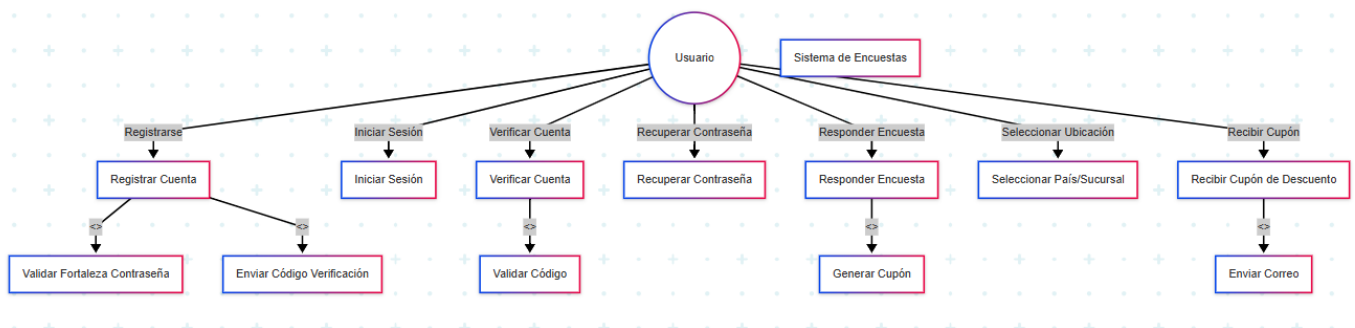
# Modo debug solo en desarrollo
debug = os.environ.get('FLASK_ENV') == 'development'

app.run(host='0.0.0.0', port=port, debug=debug)
```

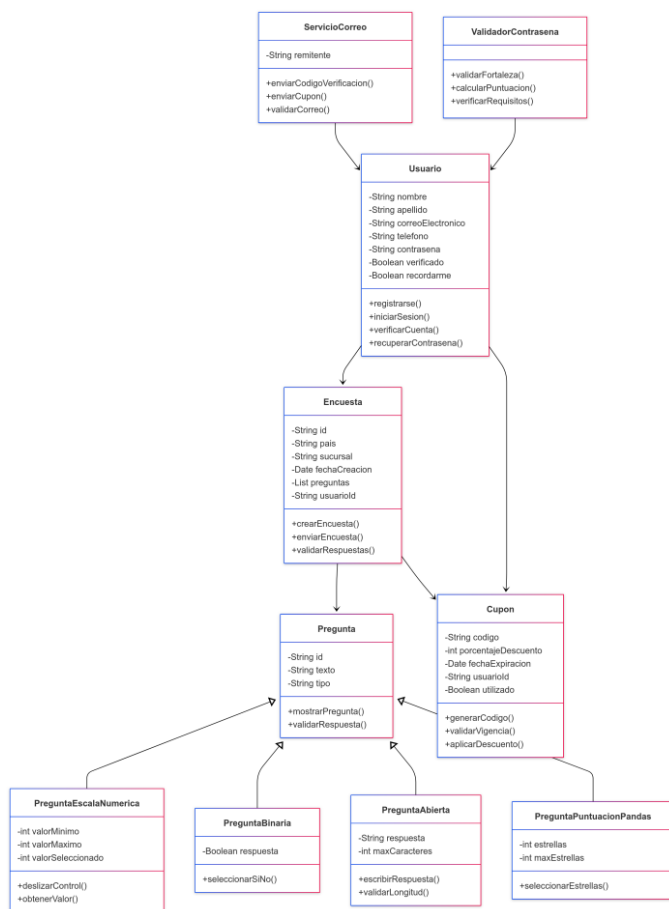
## 7. Modelos UML

A continuación, se presentan descripciones concisas de los diagramas UML desarrollados para Take no Kamu:

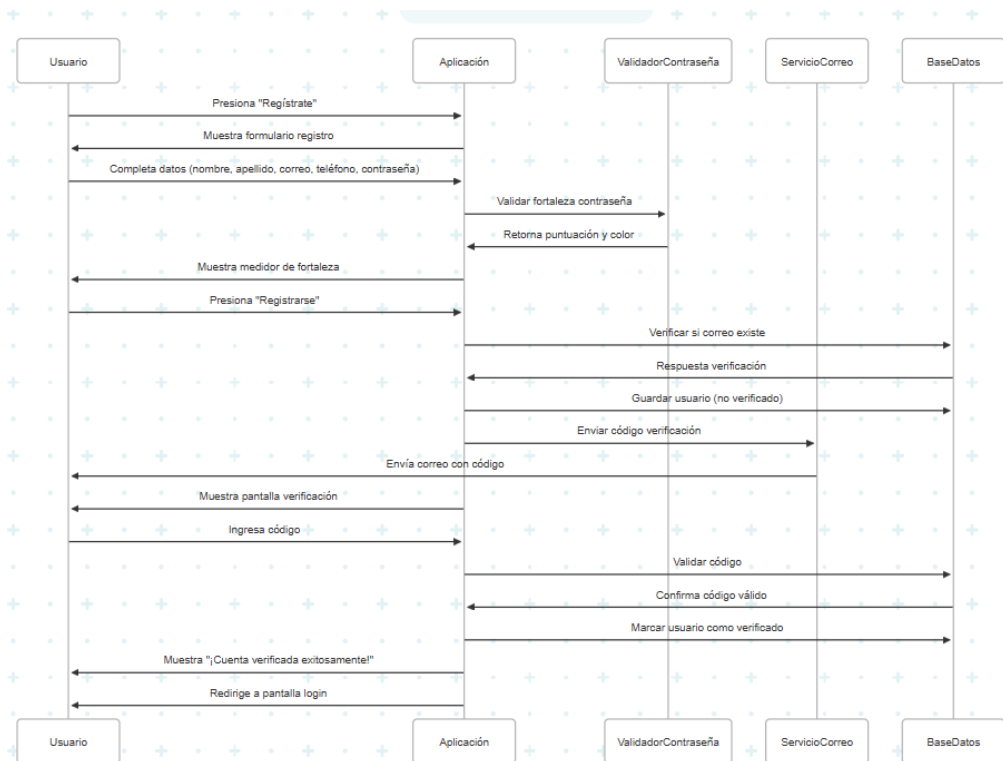
### Diagrama de casos de uso



## Diagrama de clase

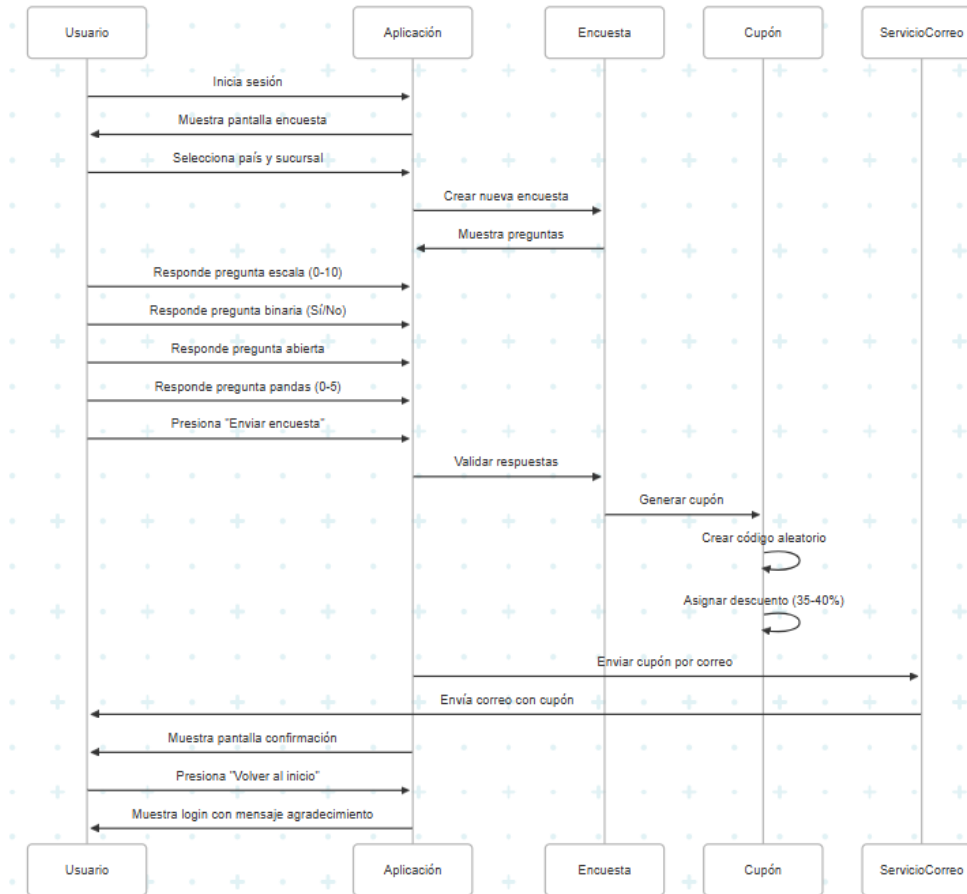


## Diagrama de secuencia de procesos clave





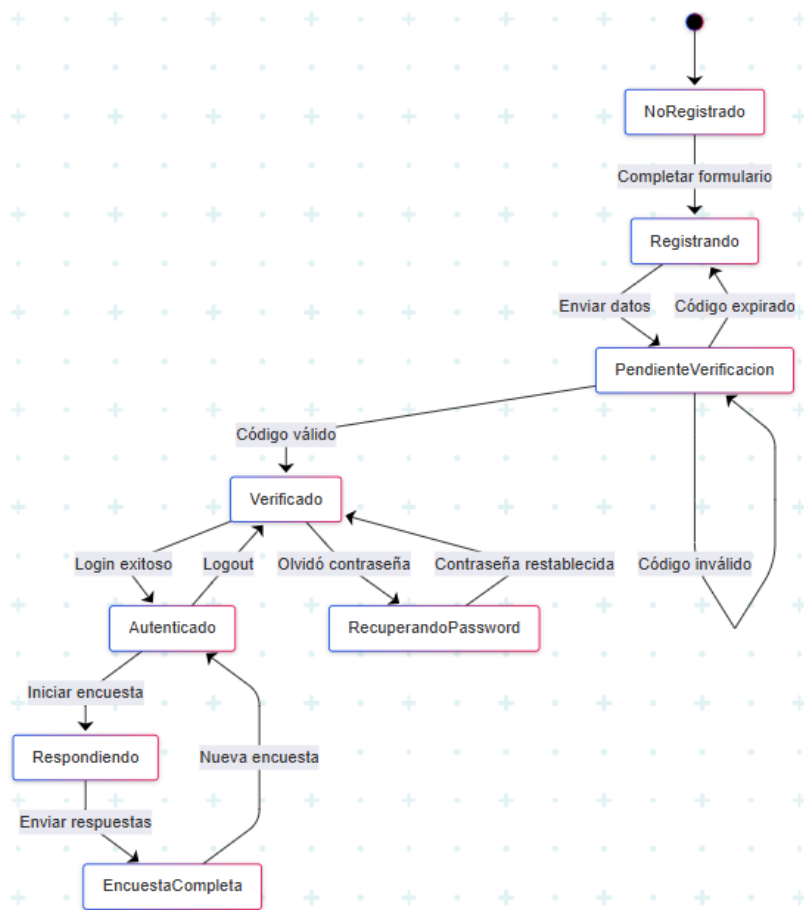
## Diagrama de secuencia proceso de encuesta



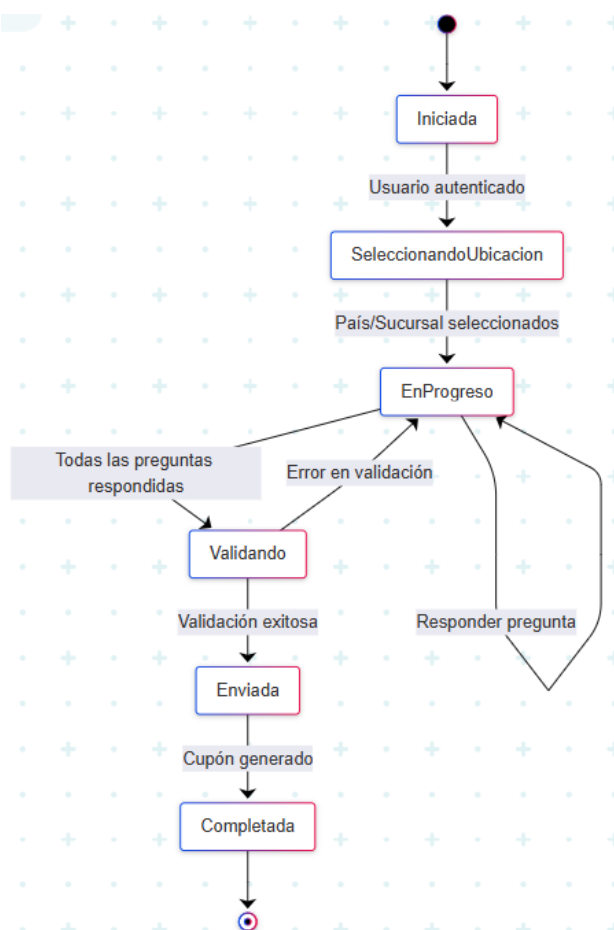
## Diagrama de actividades



## Diagrama de Estados – encuesta



## Diagrama de Estados - usuario



## 8. Control de versiones

### 8.1 Registro de actualizaciones y cambios BD: (Base de Datos)

#### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

CAMPO	DESCRIPCIÓN
NOMBRE DEL PROYECTO	Take No Kamu
FECHA DEL REGISTRO	16/03/2025
VERSIÓN DE LA APLICACIÓN	1.0
RESPONSABLE DEL CAMBIO	Adrián Emmanuel Allard Hernández
EQUIPO INVOLUCRADO	Base de datos

### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
TIPO DE CAMBIO	Creación de la base de datos
DESCRIPCIÓN DETALLADA	Creación de tres entidades y la relación entre las mismas
RAZÓN DEL CAMBIO	Creación
IMPACTO DE CAMBIO	Significativo
MÓDULOS AFECTADOS	Desarrollo y base de datos

### APROBACIÓN DE SEGUIMIENTO

CAMPO	DESCRIPCIÓN
ESTADO DEL CAMBIO	Terminado
APROBADO POR	Adrián Emmanuel Allard Hernández Carlos Gael Gutiérrez Flores
OBSERVACIONES	Propuesta inicial de la base de datos

### IMPLEMENTACIÓN Y VALIDACIÓN

CAMPO	DESCRIPCIÓN
FECHA IMPLEMENTACIÓN	16/03/2025
DESARROLLADOR RESPONSABLE	Adrián Emmanuel Allard Hernández
MÉTODO DE IMPLEMENTACIÓN	MySQL
PRUEBAS REALIZADAS	Inserción de datos
RESULTADO DE PRUEBAS	Positivos

### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

CAMPO	DESCRIPCIÓN
NOMBRE DEL PROYECTO	Take no Kamu
FECHA DEL REGISTRO	20/03/2025
VERSIÓN DE LA APLICACIÓN	2.0
RESPONSABLE DEL CAMBIO	Adrián Emmanuel Allard Hernández
EQUIPO INVOLUCRADO	Base de datos

### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
TIPO DE CAMBIO	Corrección de la base de datos
DESCRIPCIÓN DETALLADA	Se borró la entidad de preguntas, ya que la implementación de estas se encuentra en la página web de la encuesta y se modificaron las otras dos entidades para complementar los datos cambiados.

<b>RAZÓN DEL CAMBIO</b>	La versión anterior presentaba redundancias.
<b>IMPACTO DE CAMBIO</b>	Significativo
<b>MÓDULOS AFECTADOS</b>	Desarrollo y base de datos

#### APROBACIÓN DE SEGUIMIENTO

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>ESTADO DEL CAMBIO</b>	Terminado
<b>APROBADO POR</b>	Adrián Emmanuel Allard Hernández Carlos Gael Gutiérrez Flores
<b>OBSERVACIONES</b>	Corrección basada en observaciones de revisión técnica inicial

#### IMPLEMENTACIÓN Y VALIDACIÓN

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>FECHA IMPLEMENTACIÓN</b>	20/03/2025
<b>DESARROLLADOR RESPONSABLE</b>	Adrián Emmanuel Allard Hernández
<b>MÉTODO DE IMPLEMENTACIÓN</b>	MySQL
<b>PRUEBAS REALIZADAS</b>	Inserción de datos
<b>RESULTADO DE PRUEBAS</b>	Positivos

#### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>NOMBRE DEL PROYECTO</b>	Take no Kamu
<b>FECHA DEL REGISTRO</b>	30/03/2025
<b>VERSIÓN DE LA APLICACIÓN</b>	3.0
<b>RESPONSABLE DEL CAMBIO</b>	Adrián Emmanuel Allard Hernández
<b>EQUIPO INVOLUCRADO</b>	Base de datos

#### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>TIPO DE CAMBIO</b>	Modificación de la base de datos
<b>DESCRIPCIÓN DETALLADA</b>	En la entidad usuarios se le agrego un atributo llamado código-verificación se almacena el código de verificación de la cuenta, hasta que este usuario lo ocupe.

<b>RAZÓN DEL CAMBIO</b>	Modificaciones necesarias para el correcto funcionamiento de la base de datos.
<b>IMPACTO DE CAMBIO</b>	Moderado
<b>MÓDULOS AFECTADOS</b>	Base de datos

#### APROBACIÓN DE SEGUIMIENTO

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>ESTADO DEL CAMBIO</b>	Terminado
<b>APROBADO POR</b>	Adrián Emmanuel Allard Hernández Carlos Gael Gutiérrez Flores
<b>OBSERVACIONES</b>	Modificación de la base de datos

#### IMPLEMENTACIÓN Y VALIDACIÓN

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>FECHA IMPLEMENTACIÓN</b>	30/03/2025
<b>DESARROLLADOR RESPONSABLE</b>	Adrián Emmanuel Allard Hernández
<b>MÉTODO DE IMPLEMENTACIÓN</b>	MySQL
<b>PRUEBAS REALIZADAS</b>	Inserción de datos
<b>RESULTADO DE PRUEBAS</b>	Positivos

#### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>NOMBRE DEL PROYECTO</b>	Take no Kamu
<b>FECHA DEL REGISTRO</b>	5/04/2025
<b>VERSIÓN DE LA APLICACIÓN</b>	4.0
<b>RESPONSABLE DEL CAMBIO</b>	Adrián Emmanuel Allard Hernández
<b>EQUIPO INVOLUCRADO</b>	Base de datos

#### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>TIPO DE CAMBIO</b>	Modificación de la base de datos
<b>DESCRIPCIÓN DETALLADA</b>	Se agregó la entidad “cupones”, en la cual registrara el código, el porcentaje, la fecha de emisión y la fecha de vencimiento, además que cada cupón se ligara con el usuario al que se le haya mandado dicho cupón



<b>RAZÓN DEL CAMBIO</b>	Necesidad de entrega y relación de cupones
<b>IMPACTO DE CAMBIO</b>	Moderado
<b>MÓDULOS AFECTADOS</b>	Base de datos

#### APROBACIÓN DE SEGUIMIENTO

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>ESTADO DEL CAMBIO</b>	Terminado
<b>APROBADO POR</b>	Adrián Emmanuel Allard Hernández Carlos Gael Gutiérrez Flores
<b>OBSERVACIONES</b>	Modificación de la base de datos

#### IMPLEMENTACIÓN Y VALIDACIÓN

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>FECHA IMPLEMENTACIÓN</b>	5/04/2025
<b>DESARROLLADOR RESPONSABLE</b>	Adrián Emmanuel Allard Hernández
<b>MÉTODO DE IMPLEMENTACIÓN</b>	MySQL
<b>PRUEBAS REALIZADAS</b>	Inserción de datos
<b>RESULTADO DE PRUEBAS</b>	Positivos

## 8.2 Registro de actualizaciones y cambios: Interfaz de usuario

#### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>NOMBRE DEL PROYECTO</b>	Take no Kamu
<b>FECHA DEL REGISTRO</b>	20/03/2025
<b>VERSIÓN DE LA APLICACIÓN</b>	1.0
<b>RESPONSABLE DEL CAMBIO</b>	Carlos Gael Gutiérrez Flores
<b>EQUIPO INVOLUCRADO</b>	Interfaz

#### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>TIPO DE CAMBIO</b>	Creación de la interfaz para clientes
<b>DESCRIPCIÓN DETALLADA</b>	Se desarrolló la interfaz inicial del sistema dirigida a los usuarios, en donde podrán realizar su registro, iniciar sesión y responder una encuesta de satisfacción donde podrán recibir cupones de descuento al completarla. Esta interfaz permite una navegación clara y accesible para el usuario final, facilitando la recolección

	de información desde el primer contacto con el sistema.
<b>RAZÓN DEL CAMBIO</b>	Creación
<b>IMPACTO DE CAMBIO</b>	Significativo
<b>MÓDULOS AFECTADOS</b>	Interfaz

#### APROBACIÓN DE SEGUIMIENTO

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>ESTADO DEL CAMBIO</b>	Terminado
<b>APROBADO POR</b>	Carlos Gael Gutiérrez Flores
<b>OBSERVACIONES</b>	Primera versión funcional y navegable de la interfaz cliente lista para pruebas.

#### IMPLEMENTACIÓN Y VALIDACIÓN

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>FECHA IMPLEMENTACIÓN</b>	20/03/2025
<b>DESARROLLADOR RESPONSABLE</b>	Carlos Gael Gutiérrez Flores
<b>MÉTODO DE IMPLEMENTACIÓN</b>	Python
<b>PRUEBAS REALIZADAS</b>	Visualización, navegación entre pantallas
<b>RESULTADO DE PRUEBAS</b>	Positivos

#### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>NOMBRE DEL PROYECTO</b>	Take no Kamu
<b>FECHA DEL REGISTRO</b>	22/03/2025
<b>VERSIÓN DE LA APLICACIÓN</b>	2.0
<b>RESPONSABLE DEL CAMBIO</b>	Carlos Gael Gutiérrez Flores
<b>EQUIPO INVOLUCRADO</b>	Interfaz

#### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>TIPO DE CAMBIO</b>	Modificación de interfaz
<b>DESCRIPCIÓN DETALLADA</b>	Se ajustaron las lógicas de verificación de cuenta.
<b>RAZÓN DEL CAMBIO</b>	Simplificar el proceso de comprobación de datos de usuario.
<b>IMPACTO DE CAMBIO</b>	Bajo
<b>MÓDULOS AFECTADOS</b>	Interfaz

#### APROBACIÓN DE SEGUIMIENTO

CAMPO	DESCRIPCIÓN
ESTADO DEL CAMBIO	Terminado
APROBADO POR	Carlos Gael Gutiérrez Flores
OBSERVACIONES	Verificación probada con varios usuarios, sin incidencias.

#### IMPLEMENTACIÓN Y VALIDACIÓN

CAMPO	DESCRIPCIÓN
FECHA IMPLEMENTACIÓN	22/03/2025
DESARROLLADOR RESPONSABLE	Carlos Gael Gutiérrez Flores
MÉTODO DE IMPLEMENTACIÓN	Python
PRUEBAS REALIZADAS	Pruebas manuales de ingreso de datos válidos e inválidos
RESULTADO DE PRUEBAS	Positivos

#### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

CAMPO	DESCRIPCIÓN
NOMBRE DEL PROYECTO	Take no Kamu
FECHA DEL REGISTRO	25/03/2025
VERSIÓN DE LA APLICACIÓN	3.0
RESPONSABLE DEL CAMBIO	Carlos Gael Gutiérrez Flores
EQUIPO INVOLUCRADO	Interfaz

#### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
TIPO DE CAMBIO	Modificación de interfaz
DESCRIPCIÓN DETALLADA	Tras la verificación por código enviado al correo, ahora la interfaz redirige automáticamente a la pantalla de registro o login, en lugar de mostrar un mensaje estático.
RAZÓN DEL CAMBIO	Mejorar la experiencia de usuario y reducir pasos innecesarios después de la verificación.
IMPACTO DE CAMBIO	Moderado
MÓDULOS AFECTADOS	Interfaz

#### APROBACIÓN DE SEGUIMIENTO

CAMPO	DESCRIPCIÓN
ESTADO DEL CAMBIO	Terminado
APROBADO POR	Carlos Gael Gutiérrez Flores

**OBSERVACIONES**

Flujo de verificación validado en escenarios de uso real.

**IMPLEMENTACIÓN Y VALIDACIÓN**

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>FECHA IMPLEMENTACIÓN</b>	25/03/2025
<b>DESARROLLADOR RESPONSABLE</b>	Carlos Gael Gutiérrez Flores
<b>MÉTODO DE IMPLEMENTACIÓN</b>	Python
<b>PRUEBAS REALIZADAS</b>	Simulaciones de registro y verificación completas
<b>RESULTADO DE PRUEBAS</b>	Positivos

**REGISTRO DE ACTUALIZACIONES Y CAMBIOS**

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>NOMBRE DEL PROYECTO</b>	Take no Kamu
<b>FECHA DEL REGISTRO</b>	28/03/2025
<b>VERSIÓN DE LA APLICACIÓN</b>	4.0
<b>RESPONSABLE DEL CAMBIO</b>	Carlos Gael Gutiérrez Flores
<b>EQUIPO INVOLUCRADO</b>	Interfaz

**DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN**

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>TIPO DE CAMBIO</b>	Modificación de interfaz
<b>DESCRIPCIÓN DETALLADA</b>	Se reemplazó el tradicional sistema de puntuación con estrellas por una interfaz gráfica de “estrellas panditas” al finalizar la encuesta, usando íconos personalizados para mejorar la experiencia lúdica del usuario.
<b>RAZÓN DEL CAMBIO</b>	Aumentar la interacción y el atractivo visual en la sección de satisfacción del servicio.
<b>IMPACTO DE CAMBIO</b>	Bajo
<b>MÓDULOS AFECTADOS</b>	Interfaz

**APROBACIÓN DE SEGUIMIENTO**

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>ESTADO DEL CAMBIO</b>	Terminado
<b>APROBADO POR</b>	Carlos Gael Gutiérrez Flores
<b>OBSERVACIONES</b>	Iconos personalizados integrados sin afectar tiempos de carga.

## IMPLEMENTACIÓN Y VALIDACIÓN

CAMPO	DESCRIPCIÓN
FECHA IMPLEMENTACIÓN	28/03/2025
DESARROLLADOR RESPONSABLE	Carlos Gael Gutiérrez Flores
MÉTODO DE IMPLEMENTACIÓN	Python
PRUEBAS REALIZADAS	Interacción con los nuevos íconos.
RESULTADO DE PRUEBAS	Positivos

## 8.3 Registro de actualizaciones y cambios: Interfaz de administrador

### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

CAMPO	DESCRIPCIÓN
NOMBRE DEL PROYECTO	Take no Kamu
FECHA DEL REGISTRO	15/04/2025
VERSIÓN DE LA APLICACIÓN	1.0
RESPONSABLE DEL CAMBIO	Carlos Gael Gutiérrez Flores
EQUIPO INVOLUCRADO	Interfaz

### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
TIPO DE CAMBIO	Creación de interfaz para administración
DESCRIPCIÓN DETALLADA	Se desarrolló la interfaz inicial para administradores del restaurante, con registro y login de usuarios, y panel de control donde pueden visualizar estadísticas generales: gráficas de desempeño, métricas individuales por restaurante y resultados de encuestas.
RAZÓN DEL CAMBIO	Proveer a los administradores herramientas visuales y de registro para monitorizar el rendimiento y la satisfacción del cliente.
IMPACTO DE CAMBIO	Significativo
MÓDULOS AFECTADOS	Interfaz

### APROBACIÓN DE SEGUIMIENTO

CAMPO	DESCRIPCIÓN
ESTADO DEL CAMBIO	Terminado
APROBADO POR	Carlos Gael Gutiérrez Flores

**OBSERVACIONES**

Primera versión de la interfaz de administración validada y lista para pruebas.

**IMPLEMENTACIÓN Y VALIDACIÓN**

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>FECHA IMPLEMENTACIÓN</b>	15/04/2025
<b>DESARROLLADOR RESPONSABLE</b>	Carlos Gael Gutiérrez Flores
<b>MÉTODO DE IMPLEMENTACIÓN</b>	Python
<b>PRUEBAS REALIZADAS</b>	Inicio de sesión, registro de administrador, carga de datos y renderizado de gráficas
<b>RESULTADO DE PRUEBAS</b>	Positivos

**REGISTRO DE ACTUALIZACIONES Y CAMBIOS**

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>NOMBRE DEL PROYECTO</b>	Take no Kamu
<b>FECHA DEL REGISTRO</b>	25/04/2025
<b>VERSIÓN DE LA APLICACIÓN</b>	2.0
<b>RESPONSABLE DEL CAMBIO</b>	Carlos Gael Gutiérrez Flores
<b>EQUIPO INVOLUCRADO</b>	Interfaz

**DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN**

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>TIPO DE CAMBIO</b>	Corrección de interfaz de administración
<b>DESCRIPCIÓN DETALLADA</b>	Se corrigió la sección de reportes de avances, ahora las gráficas cargan correctamente y muestran los datos correctos, eliminando el error que impedía la visualización de algunas series de datos.
<b>RAZÓN DEL CAMBIO</b>	Asegurar que los administradores reciban información fiable y completa en sus reportes.
<b>IMPACTO DE CAMBIO</b>	Moderado
<b>MÓDULOS AFECTADOS</b>	Interfaz

**APROBACIÓN DE SEGUIMIENTO**

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>ESTADO DEL CAMBIO</b>	Terminado
<b>APROBADO POR</b>	Carlos Gael Gutiérrez Flores

## OBSERVACIONES

Reportes de avances validados con datos reales y sin errores en las gráficas.

### IMPLEMENTACIÓN Y VALIDACIÓN

CAMPO	DESCRIPCIÓN
FECHA IMPLEMENTACIÓN	25/04/2025
DESARROLLADOR RESPONSABLE	Carlos Gael Gutiérrez Flores
MÉTODO DE IMPLEMENTACIÓN	Python
PRUEBAS REALIZADAS	Generación de reportes, renderizado de gráficas con diferentes volúmenes de datos.
RESULTADO DE PRUEBAS	Positivos

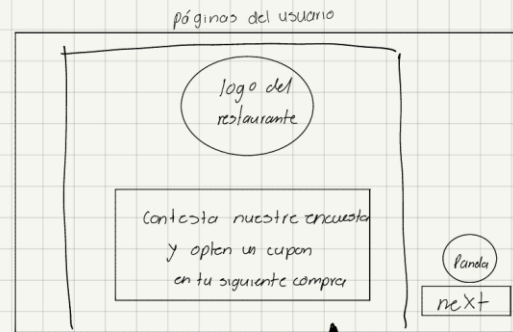
## 8.4 Registro de actualizaciones y cambios: Diseño

### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

CAMPO	DESCRIPCIÓN
NOMBRE DEL PROYECTO	Take no Kamu
FECHA DEL REGISTRO	19/02/2025
VERSIÓN DE LA APLICACIÓN	1.0
RESPONSABLE DEL CAMBIO	Arantza Sánchez Ramírez
EQUIPO INVOLUCRADO	Diseño e Interfaz

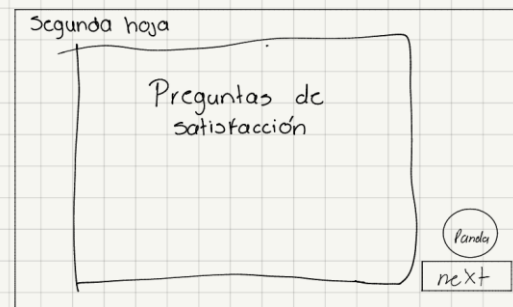
### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
TIPO DE CAMBIO	Propuesta de diseño inicial (primer boceto creado)
DESCRIPCIÓN DETALLADA	Se realizó la propuesta visual de la interfaz del usuario (pantalla principal y secundaria), inspirada en el menú de un restaurante.



Tamaño de ventanas  
800 x 568 píxeles


esa hoja puede  
tener el diseño como  
el de la carta de un  
restaurante



o finalizar

depende cuántas  
preguntas hagamos



	
<b>RAZÓN DEL CAMBIO</b>	Contar con una base visual sólida para comenzar la implementación del diseño.
<b>IMPACTO DE CAMBIO</b>	Significativo
<b>MÓDULOS AFECTADOS</b>	Interfaz y experiencia de usuario

#### APROBACIÓN DE SEGUIMIENTO

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>ESTADO DEL CAMBIO</b>	Terminado
<b>APROBADO POR</b>	Arantza Sánchez Ramírez
<b>OBSERVACIONES</b>	Se autorizó el uso del diseño de restaurante y elementos gráficos propuestos.

#### IMPLEMENTACIÓN Y VALIDACIÓN

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>FECHA IMPLEMENTACIÓN</b>	19/02/2025
<b>DESARROLLADOR RESPONSABLE</b>	Arantza Sánchez Ramírez
<b>MÉTODO DE IMPLEMENTACIÓN</b>	Bocetos a mano digitalizados, discusión y validación en equipo
<b>PRUEBAS REALIZADAS</b>	Revisión de concepto con integrantes y usuarios de prueba
<b>RESULTADO DE PRUEBAS</b>	Positivos: los elementos visuales fueron bien recibidos y considerados viables para su desarrollo

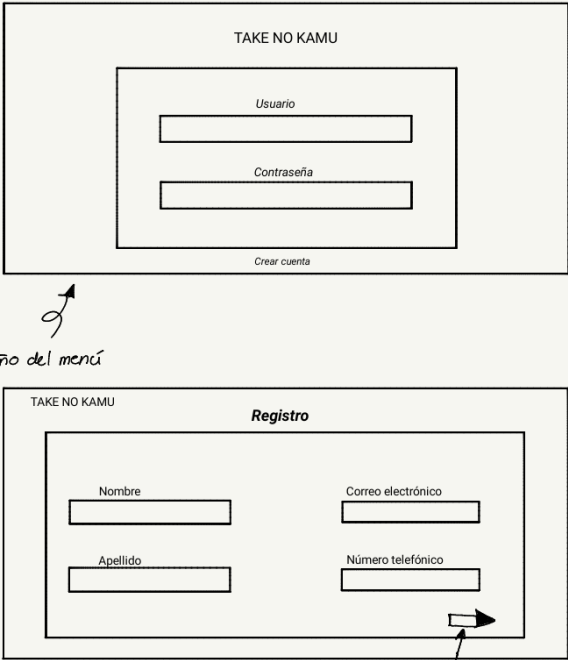
#### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>NOMBRE DEL PROYECTO</b>	Take no Kamu
<b>FECHA DEL REGISTRO</b>	25/02/2025
<b>VERSIÓN DE LA APLICACIÓN</b>	1.1

**RESPONSABLE DEL CAMBIO**  
**EQUIPO INVOLUCRADO**

Arantza Sánchez Ramírez  
Diseño e Interfaz

### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
TIPO DE CAMBIO	Propuesta boceto de interfaz de usuario.
DESCRIPCIÓN DETALLADA	<p>Se diseñó el boceto de la pantalla de registro e ingreso dirigido a usuarios del sistema. Incluye campos para nombre, apellidos, correo electrónico, número telefónico, contraseña y verificación mediante código enviado al correo. Tras el registro, los usuarios acceden a la encuesta y, una vez completada, reciben un cupón como recompensa.</p>  <p><i>diseño del menú</i></p> <p><i>página de registro</i></p> <p><i>siguiente</i></p> <p>• Será redirigido a la página de verificación</p>

TAKE NO KAMU

**Verificación**

Ingrese el código mandado a su correo electrónico

**NEXT**

←  
página de verificación

TAKE NO KAMU

**Registro exitoso**

←  
página de confirmación

- Todas las ventanas tendrán el diseño del restaurante al igual que los botones

TAKE NO KAMU

**Restablecimiento de contraseña**

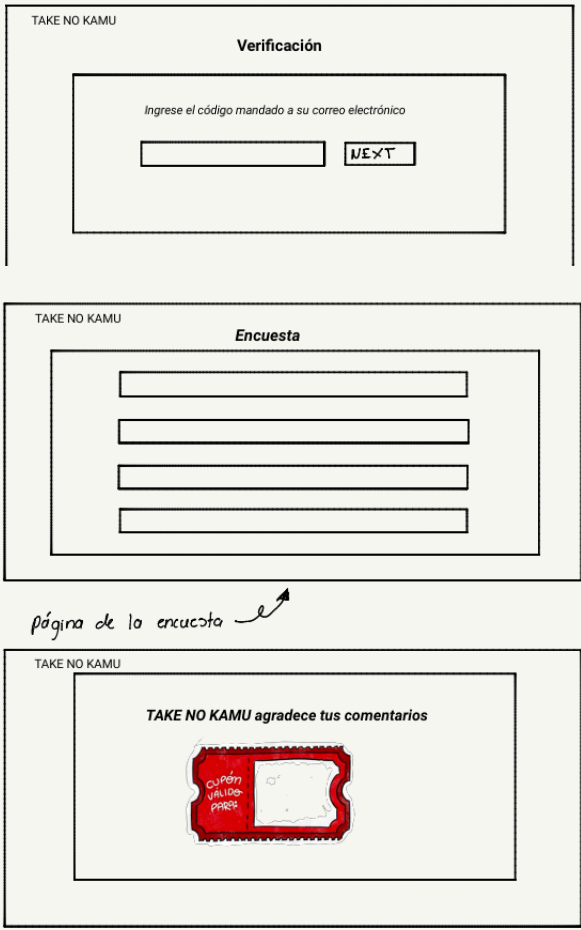
Ingresar tu correo electrónico

**NEXT**

←  
página de recuperación de contraseña

TAKE NO KAMU

**Código enviado al correo electrónico**

	
<b>RAZÓN DEL CAMBIO</b>	Adaptar la interfaz a las necesidades del flujo de registro y acceso de usuarios finales.
<b>IMPACTO DE CAMBIO</b>	Significativo
<b>MÓDULOS AFECTADOS</b>	Interfaz

### APROBACIÓN DE SEGUIMIENTO

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>ESTADO DEL CAMBIO</b>	Terminado
<b>APROBADO POR</b>	Arantza Sánchez Ramírez
<b>OBSERVACIONES</b>	El flujo del usuario fue aprobado como funcional y visualmente claro para desarrollo.

### IMPLEMENTACIÓN Y VALIDACIÓN

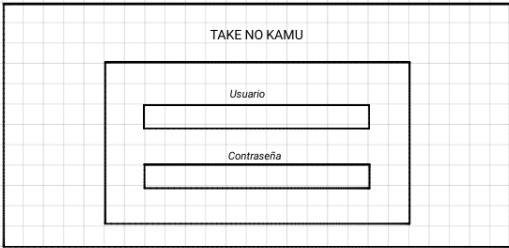
<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>FECHA IMPLEMENTACIÓN</b>	25/02/2025
<b>DESARROLLADOR RESPONSABLE</b>	Arantza Sánchez Ramírez

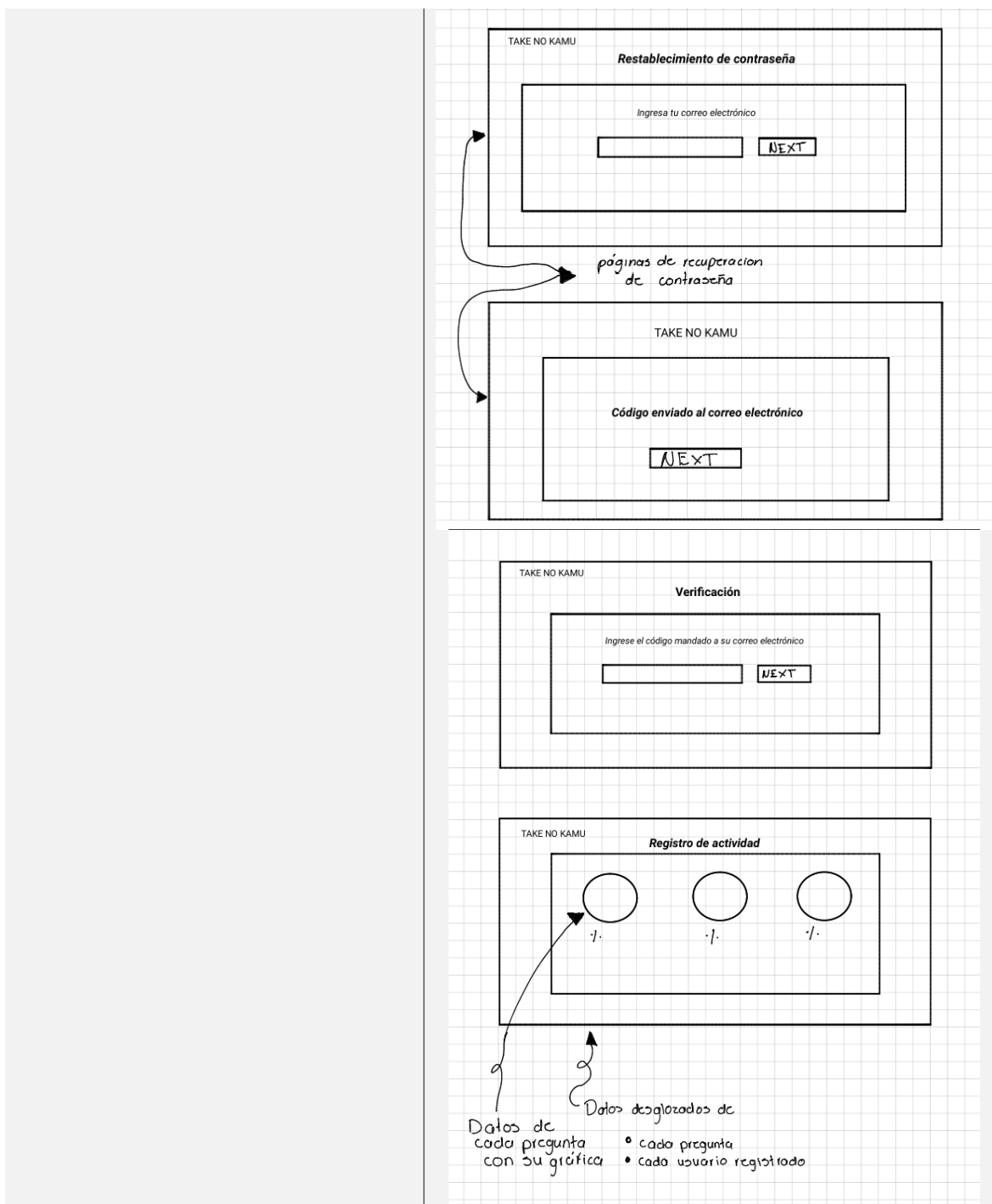
<b>MÉTODO DE IMPLEMENTACIÓN</b>	Boceto digital en software de diseño, diagramación por pantallas
<b>PRUEBAS REALIZADAS</b>	Validación visual del flujo completo (registro > verificación > encuesta > cupón)
<b>RESULTADO DE PRUEBAS</b>	Positivos: el proceso fue intuitivo y viable para su desarrollo en el sistema

### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

CAMPO	DESCRIPCIÓN
<b>NOMBRE DEL PROYECTO</b>	Take no Kamu
<b>FECHA DEL REGISTRO</b>	26/02/2025
<b>VERSIÓN DE LA APLICACIÓN</b>	1.2
<b>RESPONSABLE DEL CAMBIO</b>	Arantza Sánchez Ramírez
<b>EQUIPO INVOLUCRADO</b>	Diseño e interfaz

### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
<b>TIPO DE CAMBIO</b>	Propuesta de boceto de interfaz de cliente (administradores)
<b>DESCRIPCIÓN DETALLADA</b>	<p>Se diseñó el boceto de la pantalla de ingreso para administradores, con campos de usuario y contraseña, validación de cuenta por correo, y función de recuperación de contraseña. Al ingresar, el administrador visualiza un panel con gráficas por cada pregunta de la encuesta, desglose de resultados por usuario, estadísticas detalladas, y registros completos de usuarios.</p>  <p style="text-align: center;">↑ página del cliente</p>



**RAZÓN DEL CAMBIO**

Brindar a los administradores una herramienta visual clara para el monitoreo de encuestas, participación de usuarios y toma de decisiones.

**IMPACTO DE CAMBIO**  
**MÓDULOS AFECTADOS**

Significativo  
Interfaz – Panel de administración

### APROBACIÓN DE SEGUIMIENTO

CAMPO	DESCRIPCIÓN
ESTADO DEL CAMBIO	Terminado
APROBADO POR	Arantza Sánchez Ramírez
OBSERVACIONES	El diseño fue validado como funcional para ser implementado en la sección de administración del sistema.

### IMPLEMENTACIÓN Y VALIDACIÓN

CAMPO	DESCRIPCIÓN
FECHA IMPLEMENTACIÓN	26/02/2025
DESARROLLADOR RESPONSABLE	Arantza Sánchez Ramírez
MÉTODO DE IMPLEMENTACIÓN	Bocetos digitales por secciones (login, recuperación, etc.)
PRUEBAS REALIZADAS	Validación de flujo completo de acceso y navegación del panel administrativo
RESULTADO DE PRUEBAS	Positivos: el diseño fue comprendido por todos los integrantes y aprobado para desarrollo

### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

CAMPO	DESCRIPCIÓN
NOMBRE DEL PROYECTO	Take no Kamu
FECHA DEL REGISTRO	20/02/2025
VERSIÓN DE LA APLICACIÓN	2.0
RESPONSABLE DEL CAMBIO	Arantza Sánchez Ramírez
EQUIPO INVOLUCRADO	Diseño e Interfaz

### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
TIPO DE CAMBIO	Propuesta de diseño de íconos
DESCRIPCIÓN DETALLADA	Se propuso el uso de íconos de “panditas” como elementos visuales distintivos para representar niveles de satisfacción en la encuesta.

	<div> <div> <p>Opciones de diseño</p> </div> <div> <p>Opciones de íconos</p> </div> </div>
<b>RAZÓN DEL CAMBIO</b>	Aumentar la conexión emocional con el usuario mediante elementos visuales únicos y memorables.
<b>IMPACTO DE CAMBIO</b>	Moderado
<b>MÓDULOS AFECTADOS</b>	Interfaz de usuario – Encuesta final

### APROBACIÓN DE SEGUIMIENTO

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>ESTADO DEL CAMBIO</b>	Terminado
<b>APROBADO POR</b>	Arantza Sánchez Ramírez
<b>OBSERVACIONES</b>	Se eligieron íconos de panditas por su estética amigable y diferenciadora.

### IMPLEMENTACIÓN Y VALIDACIÓN


<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>FECHA IMPLEMENTACIÓN</b>	20/02/2025
<b>DESARROLLADOR RESPONSABLE</b>	Arantza Sánchez Ramírez
<b>MÉTODO DE IMPLEMENTACIÓN</b>	Bocetos dibujados a mano
<b>PRUEBAS REALIZADAS</b>	Reacciones y preferencia visual en grupo de prueba interno



## REGISTRO DE ACTUALIZACIONES Y CAMBIOS

CAMPO	DESCRIPCIÓN
NOMBRE DEL PROYECTO	Take no Kamu
FECHA DEL REGISTRO	24/03/2025
VERSIÓN DE LA APLICACIÓN	3.0
RESPONSABLE DEL CAMBIO	Arantza Sánchez Ramírez
EQUIPO INVOLUCRADO	Diseño e Interfaz

## DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
TIPO DE CAMBIO	Propuesta de diseño del logo
DESCRIPCIÓN DETALLADA	Se propuso el logo para el sistema Take no Kamu, buscando reflejar un estilo moderno, amigable y orientado a la temática asiática del restaurante.
	
RAZÓN DEL CAMBIO	Dotar al sistema de una identidad visual clara, coherente y adaptable a diferentes plataformas.
IMPACTO DE CAMBIO	Moderado
MÓDULOS AFECTADOS	Diseño gráfico y elementos de marca del sistema

## APROBACIÓN DE SEGUIMIENTO

CAMPO	DESCRIPCIÓN
ESTADO DEL CAMBIO	Terminado
APROBADO POR	Arantza Sánchez Ramírez
OBSERVACIONES	Se validaron las propuestas visuales y tipográficas para futuras implementaciones del logo.

### IMPLEMENTACIÓN Y VALIDACIÓN

CAMPO	DESCRIPCIÓN
FECHA IMPLEMENTACIÓN	24/03/2025
DESARROLLADOR RESPONSABLE	Arantza Sánchez Ramírez
MÉTODO DE IMPLEMENTACIÓN	Bocetos manuales y versiones digitales en herramientas de diseño vectorial
PRUEBAS REALIZADAS	Evaluación por el equipo para uso en distintos contextos visuales
RESULTADO DE PRUEBAS	Positivos

### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

CAMPO	DESCRIPCIÓN
NOMBRE DEL PROYECTO	Take no Kamu
FECHA DEL REGISTRO	01/04/2025
VERSIÓN DE LA APLICACIÓN	4.0
RESPONSABLE DEL CAMBIO	Arantza Sánchez Ramírez Emanuel Herrera Briseño
EQUIPO INVOLUCRADO	Diseño e Interfaz

### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
TIPO DE CAMBIO	Propuesta de diseño de cupones
DESCRIPCIÓN DETALLADA	Se presentaron tres diseños de cupones que serán entregados al usuario tras completar la encuesta. Cada diseño corresponde a un porcentaje de descuento distinto: 30%, 35% y 40%. Los cupones incluyen colores, íconos e ilustraciones diferentes, conservando la estética del sistema y destacando visualmente el valor del beneficio.

	
<b>RAZÓN DEL CAMBIO</b>	Ofrecer recompensas atractivas que fomenten la participación de los clientes en las encuestas, mientras se mantiene la identidad gráfica del sistema.
<b>IMPACTO DE CAMBIO</b>	Moderado
<b>MÓDULOS AFECTADOS</b>	Interfaz – Recompensas visuales

#### APROBACIÓN DE SEGUIMIENTO

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>ESTADO DEL CAMBIO</b>	Terminado
<b>APROBADO POR</b>	Arantza Sánchez Ramírez Emanuel Herrera Briseño
<b>OBSERVACIONES</b>	Las propuestas fueron validadas con el equipo. Se decidirá el uso final tras discusiones en equipo.

#### IMPLEMENTACIÓN Y VALIDACIÓN

<b>CAMPO</b>	<b>DESCRIPCIÓN</b>
<b>FECHA IMPLEMENTACIÓN</b>	01/04/2025
<b>DESARROLLADOR RESPONSABLE</b>	Arantza Sánchez Ramírez Emanuel Herrera Briseño
<b>MÉTODO DE IMPLEMENTACIÓN</b>	Diseños gráficos digitales adaptables a impresión y uso digital
<b>PRUEBAS REALIZADAS</b>	Presentación de las tres propuestas al equipo de diseño y validación interna de colores y proporciones


## RESULTADO DE PRUEBAS

Positivos: cada cupón fue reconocido fácilmente y se recibió de forma positiva por parte del equipo

### REGISTRO DE ACTUALIZACIONES Y CAMBIOS

CAMPO	DESCRIPCIÓN
NOMBRE DEL PROYECTO	Take no Kamu
FECHA DEL REGISTRO	05/04/2025
VERSIÓN DE LA APLICACIÓN	4.1
RESPONSABLE DEL CAMBIO	Arantza Sánchez Ramírez Emanuel Herrera Briseño
EQUIPO INVOLUCRADO	Diseño e Interfaz

### DESCRIPCIÓN DEL CAMBIO O ACTUALIZACIÓN

CAMPO	DESCRIPCIÓN
TIPO DE CAMBIO	Implementación del diseño final de cupones
DESCRIPCIÓN DETALLADA	<p>Se seleccionó el diseño definitivo para los cupones, el cual presenta a un panda sosteniendo los cupones con la mano. Este diseño se aplicó a los tres tipos de descuento (30%, 35% y 40%) utilizando colores diferentes para distinguirlos. El diseño conserva la línea estética del sistema y refuerza el vínculo visual con el personaje central (el panda).</p> 
RAZÓN DEL CAMBIO	Uniformar visualmente los cupones y generar una identidad reconocible, funcional y amigable para el usuario.
IMPACTO DE CAMBIO	Moderado
MÓDULOS AFECTADOS	Interfaz – Recompensas gráficas del sistema

### APROBACIÓN DE SEGUIMIENTO

CAMPO	DESCRIPCIÓN
ESTADO DEL CAMBIO	Terminado
APROBADO POR	Arantza Sánchez Ramírez Emanuel Herrera Briseño
OBSERVACIONES	Se eligió el diseño por su claridad visual y conexión con la identidad del sistema. Aplicado exitosamente a los tres niveles de descuento.

### IMPLEMENTACIÓN Y VALIDACIÓN

CAMPO	DESCRIPCIÓN
FECHA IMPLEMENTACIÓN	05/04/2025
DESARROLLADOR RESPONSABLE	Arantza Sánchez Ramírez Emanuel Herrera Briseño
MÉTODO DE IMPLEMENTACIÓN	Diseño digital final adaptable a medios impresos y digitales
PRUEBAS REALIZADAS	Revisión de legibilidad, escalabilidad del diseño y validación con muestras digitales
RESULTADO DE PRUEBAS	Positivos

## 9. Flujo

### Interfaz de usuario

Al abrir la aplicación, el usuario es recibido con la pantalla de inicio de sesión, donde aparecerán dos campos claramente etiquetados: uno para ingresar su correo electrónico y otro para la contraseña. Debajo de estos campos, en letras pequeñas y discretas, encontrará el enlace “¿No tienes cuenta? Regístrate”. Al pulsar este enlace, el sistema lo llevará a la pantalla de registro.

En la pantalla de registro, se despliegan cinco casillas obligatorias:

1. Nombre
2. Apellido

3. Correo electrónico
4. Número de teléfono
5. Contraseña

Debajo del campo de contraseña se verá un pequeño medidor de fuerza (una barra que ira creciendo, dependiendo que tan segura o poco segura sea la contraseña, siendo roja y corta si es insegura y verde y larga si es segura) que te ayudará a saber cuán segura es tu clave mientras la escribes. Cuantos más requisitos cumpla, más se llenará el indicador y más fuerte será tu contraseña. Para lograrlo, intenta:

- Escribir al menos 8 caracteres (mejor si pasas de 10).
- Usar alguna letra mayúscula (A–Z).
- Incluir al menos un número (0–9).
- Añadir algún símbolo especial (por ejemplo: !, @, #, \$).

Así, la contraseña será más difícil de adivinar y el medidor lo reflejará llenando más barras o puntos de color

Cada criterio cumple un punto de seguridad:

- +1 si tiene más de 6 caracteres
- +1 adicional si supera los 10 caracteres
- +1 por incluir mayúsculas
- +1 por incluir números
- +1 por incluir símbolos

Bajo estas casillas, un botón grande y llamativo, de color verde, dice “Registrarse”. Justo debajo, como en eco de la pantalla anterior, aparece “¿Ya tienes una cuenta? Inicia sesión”, devolviendo al usuario a la pantalla de login si ya posee credenciales.

Al pulsar “Registrarse” y completar los cinco campos, el usuario es dirigido a la pantalla de verificación de cuenta. En la parte superior leerá el título “Verificación de

cuenta” y un texto instructivo: “Ingresa el código que enviamos a tu correo electrónico”. Inmediatamente debajo, un campo de texto para introducir el código de verificación. Bajo este campo, en texto más pequeño, se advierte que “El código expirará en 15 minutos.” A continuación, un botón verde con la leyenda “Verificar código”. En la esquina inferior de la pantalla, en letras pequeñas y verdes, consta la opción “¿No recibiste el código? Reenviar código”, que permite generar y enviar otro código automáticamente desde la dirección [takenokamu@gmail.com](mailto:takenokamu@gmail.com) (recomendada revisar la carpeta de spam).

Una vez introducido el código válido y pulsado “Verificar código”, aparece un mensaje verde en la parte inferior: “¡Cuenta verificada exitosamente!”. Después de unos instantes, la aplicación redirige al usuario de nuevo a la pantalla de inicio de sesión, lista para ingresar con las credenciales recién creadas.

De regreso en el login, los dos mismos campos (correo y contraseña) vuelven a la vista, ahora con la opción adicional “Recuérdame”, que al marcarla permitirá al usuario mantenerse conectado. Al lado, aparece el enlace “¿Olvidaste tu contraseña?” para recuperar el acceso si lo necesita. Bajo estas opciones, el botón grande y verde “Entrar” da paso al contenido principal de la aplicación.

Al autenticarse correctamente, el usuario accede a la pantalla de la encuesta de satisfacción. Aquí, en la parte superior, dos menús desplegables permiten seleccionar primero el país —por ejemplo, México— y luego la sucursal que desee evaluar. A continuación, se presentan cuatro tipos de preguntas:

1. Escala numérica (0–10):  
Una barra horizontal con un control deslizante circular de color verde que puede arrastrarse de izquierda a derecha; el extremo izquierdo equivale a 0 (muy insatisfecho) y el extremo derecho a 10 (muy satisfecho). Por ejemplo: “¿Qué tal estuvo tu comida?”
2. Opción binaria Sí/No:  
Dos círculos blancos bajo las palabras “Sí” y “No”. Al tocar uno, aparece una

palomita verde dentro del círculo seleccionado. Ejemplo: “¿El tiempo de espera fue adecuado?”

3. Respuesta abierta:

Un cuadro de texto con borde verde y texto gris en su interior que dice: “Escribe tus sugerencias aquí (máximo 200 caracteres)”. El usuario puede escribir libremente comentarios como “¿En qué podemos mejorar?”

4. Puntuación con pandas (0–5):

Similar a la escala numérica, pero con cinco iconos de pandas alineados y un deslizador que se mueve entre 0 y 5, representando las estrellas de calificación de la experiencia global: “¿Cuántas estrellas darías a tu experiencia?”

Una vez respondidas todas las preguntas, al pie de la pantalla se muestra un botón verde, amplio, con el texto “Enviar encuesta” acompañado de un icono de avión de papel. Al pulsarlo, se procesa la información y se avanza a la pantalla de confirmación:

- Un encabezado que dice “¡Gracias por contestar la encuesta!”
- Debajo, un gran círculo verde con una palomita blanca.
- Un mensaje informativo: “Hemos enviado un correo a tu dirección con un cupón especial para tu próxima visita a nuestro restaurante.”
- Otro mensaje: “Esperamos verte pronto.”
- Un recuadro gris claro con el texto:

“Tu cupón te espera, revisa tu bandeja de entrada y disfruta de un descuento especial en tu próxima visita.”

- Finalmente, un botón verde “Volver al inicio” que regresa a la pantalla de login. En esa pantalla, ahora verá un pequeño recuadro verde en la parte superior que dice “¡Gracias por tu respuesta!”, recordándole que ya participó en la encuesta.



El correo de cupón se envía desde takenokamu@gmail.com con asunto:

“Tu cupón de XX % de descuento – Gracias por tu encuesta”

El porcentaje, aleatorizado entre 35 % y 40 %, y el cupón (por ejemplo, “ZWFGTQ”) aparecen en el cuerpo del mensaje junto a una imagen del cupón. Se especifica que el cupón es válido por 30 días y que no se podrá solicitar otro hasta transcurrido ese periodo

## Interfaz de administrador

Para registrar un administrador, primero se accede a la interfaz denominada como “Panel de administración”, donde habrá tres opciones: “Inicio”, “Registrar administrador” y “Lista de administradores”. Después, daremos clic en el botón del medio, el cual es el de “Registrar administrador”. Ahí se pondrán los datos que se piden, como el nombre, apellidos, correo, teléfono y contraseña para dicho administrador. Al final, habrá un botón verde debajo que dirá “Registrar administrador”, el cual deberemos presionar para registrarlo.

Después de eso, se redirigirá a la pantalla de Inicio, en donde nos saldrá el mensaje “Administrador registrado exitosamente”. Luego, en “Lista de administradores”, se verán los datos que hayamos registrado.

Para entrar a la nueva interfaz de administrador, simplemente se inicia sesión como se haría normalmente, en la misma plataforma del usuario, pidiendo correo y contraseña. Al ingresar, veremos la interfaz de administradores, la cual consiste en los reportes de sucursal.

Podremos cambiar los filtros: país, sucursal, fecha de inicio y fecha de fin. Se verá el total de respuestas, calificaciones promedio del restaurante (estrellas), satisfacción de servicio en porcentaje, tasa de retorno en porcentaje, al igual que tres gráficas:

- Calificación promedio por sucursal
- Distribución de calificaciones

- Intención de retorno

## 10. Funciones adicionales

### 8.1. “Recuérdame” (Login)

- **Ubicación:** En la **pantalla de inicio de sesión**, justo debajo de los campos de correo y contraseña, al lado izquierdo.
- **Descripción:** Es una casilla de verificación que, al marcarla, mantiene tu sesión activa incluso después de cerrar la aplicación o el navegador, de modo que no tengas que ingresar tus credenciales cada vez que la abras.

### 8.2. “¿Olvidaste tu contraseña?” (Login)

- **Ubicación:** En la **misma pantalla de inicio de sesión**, se muestra como un enlace de texto pequeño y subrayado, ubicado justo a la derecha de “Recuérdame”.
- **Descripción:** Al hacer clic, se inicia un flujo de recuperación: el sistema te pedirá tu correo, enviará un enlace de restablecimiento y, tras verificarlo, podrás elegir una nueva contraseña.

### 8.3. Reenvío de código (Verificación de cuenta)

- **Ubicación:** En la pantalla de verificación, aparece como un texto pequeño y de color verde, alineado al centro-inferior de la pantalla, debajo del botón “Verificar código”.
- **Descripción:** Si no recibiste el código en tu correo (revisa spam), pulsa “¿No recibiste el código? Reenviar código” para que la aplicación genere y envíe uno nuevo automáticamente desde takenokamu@gmail.com.

## 8.4. Generación de cupón

- **Ubicación del proceso:** Tras enviar la encuesta, se crea un cupón automáticamente. En la pantalla de confirmación, leerás un mensaje que anuncia el envío del cupón por correo.
- **Mecánica interna:**
  - Se genera un porcentaje aleatorio entre 35 % y 40 %.
  - Se crea un código alfanumérico (por ejemplo, “ZWFGTQ”) y se asocia al usuario en la base de datos.
- **Validez y restricción:**
  - El cupón vence a los 30 días de su envío (se muestra la fecha de expiración en el correo).
  - Hasta que no pasen esos 30 días, el sistema no permitirá emitir un cupón nuevo para la misma cuenta, evitando duplicados prematuros.

## 11. Diccionario de datos

### Módulos Importados

Nombre del Módulo	Descripción	Uso en el Proyecto
<b>flask</b>	Framework web ligero para Python.	Se utiliza para crear rutas, renderizar templates, gestionar sesiones, redireccionamientos y mostrar mensajes (flash).
<b>flask_sqlalchemy</b>	Extensión de Flask para conectarse a	Proporciona una interfaz sencilla para ejecutar queries sobre MySQL.

	una base de datos MySQL.	
<b>random</b>	Biblioteca estándar de Python para generar números aleatorios.	Se puede utilizar para generar códigos o tokens aleatorios (por ejemplo, para verificación por correo).
<b>smtplib</b>	Protocolo de envío de correos (SMTP) de Python.	Usado para enviar correos electrónicos, como la verificación de cuenta o recuperación de contraseña.
<b>email.message.EmailMessage</b>	Permite construir mensajes de correo de forma estructurada.	Facilita la creación y formato de correos electrónicos antes de enviarlos.
<b>datetime, timedelta</b>	Módulos para manejo de fechas y horas.	Controla tiempos de expiración (por ejemplo, para tokens de recuperación o sesiones).
<b>os</b>	Módulo para interactuar con el sistema operativo.	Manejo de rutas, acceso a archivos u operaciones del sistema.
<b>functools.wraps</b>	Herramienta para crear decoradores conservando metadatos de la función original.	Se puede usar para proteger rutas con autenticación.
<b>hashlib</b>	Biblioteca para funciones hash seguras.	Se usa para encriptar contraseñas o generar tokens hash.
<b>pandas</b>	Librería para análisis y manipulación de datos.	Se emplea para construir reportes administrativos y análisis de encuestas.

<b>matplotlib.pyplot</b>	Librería para visualización de gráficos.	Generación de gráficos para reportes administrativos.
<b>matplotlib.use('Agg')</b>	Permite a matplotlib trabajar sin una GUI.	Necesario cuando se renderizan gráficos en servidores sin entorno gráfico.
<b>numpy</b>	Biblioteca para operaciones numéricas eficientes.	Apoya el análisis estadístico de resultados de encuestas.
<b>json</b>	Biblioteca estándar para trabajar con datos JSON.	Permite serializar y deserializar datos (e.g., respuestas de usuarios).
<b>io</b>	Módulo para operaciones de entrada/salida con flujos de datos.	Usado para generar archivos en memoria (como imágenes de gráficos).
<b>flask.send_from_directory</b>	Función de Flask para enviar archivos desde un directorio.	Utilizada para descargas de archivos generados (como reportes en PDF o CSV).

### Configuración General de la Aplicación

Nombre	Tipo	Descripción
<b>STATIC_FOLDER</b>	str	Ruta absoluta al directorio static donde se almacenan archivos estáticos como imágenes, CSS o JS. Se construye dinámicamente con os.path.
<b>IMAGES_FOLDER</b>	str	Subdirectorío dentro de static/ destinado a almacenar imágenes

		generadas, como los gráficos de reportes. Se crea si no existe.
<b>app</b>	Flask	Instancia principal de la aplicación Flask. Permite definir rutas, configuraciones y controladores.
<b>app.secret_key</b>	str	Clave secreta utilizada para firmar cookies de sesión. Es sensible y se obtiene desde variables de entorno para mayor seguridad.
<b>os.makedirs(IMAGES_FOLDER, exist_ok=True)</b>	función	Crea el directorio images/ dentro de static si no existe, evitando errores al guardar imágenes.

### Configuración de la Base de Datos (MySQL)

Clave de Configuración	Valor por Defecto	Descripción
<b>MYSQL_HOST</b>	'localhost'	Dirección del host de la base de datos MySQL.
<b>MYSQL_USER</b>	'root'	Usuario de conexión a la base de datos.
<b>MYSQL_PASSWORD</b>	'Ericko11\$'	Contraseña del usuario (debe ocultarse en producción).
<b>MYSQL_DB</b>	'encuestas_db'	Nombre de la base de datos donde se almacenan las encuestas, usuarios, respuestas, etc.

### Configuración para Envío de Correos Electrónicos

Nombre	Tipo	Descripción
<b>EMAIL_SENDER</b>	str	Dirección de correo desde la cual se envían los mensajes (como verificación o recuperación).

<b>EMAIL_PASSWORD</b>	str	Contraseña o clave de aplicación del correo (en este caso, Gmail). Usada por smtplib para autenticarse.
-----------------------	-----	---

### hash\_password(password)

Elemento	Valor
<b>Nombre</b>	hash_password
<b>Parámetros</b>	password (str) – Contraseña en texto plano que se desea proteger.
<b>Retorno</b>	str – Cadena hexadecimal resultante del hash.
<b>Descripción</b>	Utiliza el algoritmo SHA-256 para hashear contraseñas de forma segura. Esto permite almacenar contraseñas en la base de datos sin guardarlas en texto plano.
<b>Uso Común</b>	Registro y validación de contraseñas de usuarios.
<b>Dependencia</b>	hashlib

### generar\_codigo\_verificacion()

Elemento	Valor
<b>Nombre</b>	generar_codigo_verificacion
<b>Parámetros</b>	Ninguno
<b>Retorno</b>	str – Código aleatorio de 6 dígitos.
<b>Descripción</b>	Genera un número aleatorio de 6 cifras en formato de texto. Se utiliza para confirmar la identidad del usuario durante el proceso de verificación.
<b>Uso Común</b>	Envío de códigos de verificación por correo electrónico.
<b>Dependencia</b>	random

### enviar\_codigo\_verificacion(destinatario, codigo)

Elemento	Valor
<b>Nombre</b>	enviar_codigo_verificacion

<b>Parámetros</b>	destinatario (str) – Dirección de correo del usuario codigo (str) – Código de verificación generado.
<b>Retorno</b>	bool – True si el correo se envía exitosamente, False si ocurre un error.
<b>Descripción</b>	Esta función compone y envía un correo electrónico con un código de verificación al usuario usando smtplib y el protocolo SMTP seguro (SSL).
<b>Uso Común</b>	Confirmación de identidad en el registro o recuperación de contraseña.
<b>Dependencias</b>	smtplib, email.message.EmailMessage, EMAIL_SENDER, EMAIL_PASSWORD
<b>Notas</b>	Captura excepciones para evitar fallos graves si el correo no se puede enviar.

### login\_required(f)

Elemento	Valor
<b>Nombre</b>	login_required
<b>Tipo</b>	Decorador
<b>Parámetros</b>	f – Función que se desea proteger.
<b>Retorno</b>	Función decorada.
<b>Descripción</b>	Decorador que protege rutas para que solo usuarios autenticados puedan acceder. Si no hay un usuario_id en la sesión, redirige a la página de login y muestra un mensaje.
<b>Uso Común</b>	Aplicar a rutas que requieren sesión iniciada, como responder encuestas o ver reportes.
<b>Dependencias</b>	session, flash, redirect, url_for, functools.wraps

### verificar\_actividad\_reciente(id\_usuario)



Elemento	Valor
<b>Nombre</b>	verificar_actividad_reciente
<b>Parámetros</b>	id_usuario (int) – ID del usuario a verificar.
<b>Retorno</b>	bool – True si el usuario ya respondió en los últimos 30 días, False si no o si nunca respondió.
<b>Descripción</b>	Consulta la base de datos para verificar si el usuario ha enviado una respuesta a la encuesta en los últimos 30 días.
<b>Uso Común</b>	Evitar que un mismo usuario participe repetidamente en un corto plazo.
<b>Dependencias</b>	mysql, datetime, timedelta
<b>Consulta SQL</b>	

### enviar\_codigo\_recuperacion(destinatario, codigo)

Elemento	Valor
<b>Nombre</b>	enviar_codigo_recuperacion
<b>Parámetros</b>	destinatario (str) – Correo electrónico del usuario.codigo (str) – Código generado para la recuperación.
<b>Retorno</b>	bool – True si el mensaje se envía exitosamente, False si ocurre un error.
<b>Descripción</b>	Función muy similar a enviar_codigo_verificacion. Prepara y envía un correo electrónico con el código de recuperación de contraseña.
<b>Uso Común</b>	En procesos de "¿Olvidaste tu contraseña?"
<b>Dependencias</b>	smtpplib, EMAIL_SENDER, EMAIL_PASSWORD, EmailMessage

### enviar\_codigo\_recuperacion(destinatario, codigo)

Elemento	Valor
<b>Nombre</b>	enviar_codigo_recuperacion

<b>Parámetros</b>	destinatario (str) – Dirección de correo del usuario.codigo (str) – Código de verificación para recuperación.
<b>Retorno</b>	bool – True si el mensaje se envía correctamente, False si ocurre un error.
<b>Descripción</b>	Envía un correo electrónico con un código de verificación temporal para restablecer la contraseña. El correo informa que el código expirará en 30 minutos.
<b>Uso Común</b>	Proceso de recuperación de cuenta cuando el usuario olvida su contraseña.
<b>Dependencias</b>	smtplib, email.message.EmailMessage, EMAIL_SENDER, EMAIL_PASSWORD
<b>Comportamiento</b>	Captura errores con un try-except y reporta cualquier falla en la consola.
<b>Contenido del correo</b>	

**@app.route('/') → index()**

Elemento	Valor
<b>Ruta</b>	/
<b>Nombre de Función</b>	index()
<b>Métodos Permitidos</b>	GET (por defecto)
<b>Descripción</b>	Es la ruta principal de la aplicación. Redirige automáticamente a la página de inicio de sesión (/login).
<b>Uso Común</b>	Facilita que la raíz del sitio web lleve a la página adecuada para comenzar sesión.
<b>Dependencias</b>	redirect, url_for

**@app.route('/registro', methods=['GET', 'POST']) → registro()**

Elemento	Valor
<b>Ruta</b>	/registro

<b>Nombre de Función</b>	registro()
<b>Métodos Permitidos</b>	GET, POST
<b>Descripción</b>	Permite a un nuevo usuario registrarse en el sistema. Maneja tanto la visualización del formulario como el procesamiento del envío.

### | Campos Esperados del Formulario |

Campo	Tipo	Descripción
<b>nombre</b>	str	Nombre del usuario.
<b>apellido</b>	str	Apellido del usuario.
<b>correo</b>	str	Correo electrónico único del usuario.
<b>telefono</b>	str	Número telefónico del usuario.
<b>contrasena</b>	str	Contraseña en texto plano que se encriptará.

### | Dependencias |

- mysql
- hash\_password()
- generar\_codigo\_verificacion()
- enviar\_codigo\_verificacion()
- session
- render\_template, request, redirect, url\_for

**@app.route('/verificar\_codigo', methods=['GET', 'POST']) →  
verificar\_codigo()**

Elemento	Valor
<b>Ruta</b>	/verificar_codigo
<b>Nombre de Función</b>	verificar_codigo()
<b>Métodos Permitidos</b>	GET, POST

<b>Descripción</b>	Permite al usuario ingresar el código enviado por correo para confirmar su cuenta. Si el código es correcto, se marca como verificada y se borra el código de la base de datos.
<b>Requisitos Previos</b>	Debe existir la variable correo_verificacion en la sesión (session). Si no está, se redirige al registro.
<b>Flujo de Datos</b>	

### | Campos Esperados del Formulario |

Campo	Tipo	Descripción
<b>codigo</b>	str	Código de verificación de 6 dígitos que fue enviado al correo del usuario.

### | Variables en la sesión (session) |

Clave	Descripción
<b>correo_verificacion</b>	Correo electrónico del usuario que se está verificando.

**@app.route('/login', methods=['GET', 'POST']) → login()**

Elemento	Valor
<b>Ruta</b>	/login
<b>Nombre de Función</b>	login()
<b>Métodos Permitidos</b>	GET, POST
<b>Descripción</b>	Permite a los usuarios iniciar sesión validando su correo, contraseña y estado de verificación. Maneja intentos fallidos y redirecciona según privilegios.

### Campos Esperados del Formulario

Campo	Tipo	Descripción
<b>email</b>	str	Correo electrónico del usuario.

<b>password</b>	str	Contraseña del usuario (en texto plano, se hashea antes de comparar).
-----------------	-----	---

### Variables de Sesión (session)

Clave	Descripción
<b>usuario_id</b>	ID del usuario autenticado.
<b>nombre</b>	Nombre del usuario.
<b>privilegios</b>	Nivel de privilegio (0 = usuario, 1 = administrador).
<b>intentos_fallidos</b>	Contador de intentos de inicio fallidos por correo.
<b>email_intentos</b>	Correo actual en intento de login.
<b>correo_recuperacion</b>	Correo que se usará si se exceden los intentos permitidos.

### Redirecciones Según Condición

Condición	Redirige a
<b>Usuario administrador (privilegios == 1)</b>	admin_reportes_sucursal
<b>Usuario verificado y estándar (privilegios == 0)</b>	encuesta
<b>Usuario no verificado</b>	Se muestra error
<b>3 intentos fallidos</b>	recuperar_contrasena

### @app.route('/logout') → logout()

Elemento	Valor
<b>Ruta</b>	/logout
<b>Nombre de Función</b>	logout()
<b>Método</b>	GET
<b>Descripción</b>	Cierra la sesión del usuario eliminando todos los datos de session y redirige al login.
<b>Acciones</b>	

### @app.route('/encuesta', methods=['GET', 'POST']) → encuesta()

Elemento	Valor
<b>Ruta</b>	/encuesta
<b>Nombre de Función</b>	encuesta()
<b>Métodos Permitidos</b>	GET, POST
<b>Decorador</b>	@login_required
<b>Descripción</b>	Muestra y procesa el formulario de encuesta para usuarios autenticados. Limita una respuesta por usuario cada 30 días.

### Campos Esperados del Formulario (POST)

Campo	Tipo	Requerido	Descripción
<b>pais</b>	str	✓	País del usuario.
<b>sucursal</b>	str	✓	Sucursal visitada.
<b>calidad_comida</b>	int	✗	Calificación 1–5 de la comida.
<b>tiempo_espera</b>	str	✓	Descripción del tiempo de espera.
<b>atencion_personal</b>	int	✗	Calificación 1–5 de atención.
<b>agrado_sucursal</b>	str	✓	Opinión sobre la sucursal.
<b>volveria_visitar</b>	str	✓	Si volvería o no.
<b>area_mejora</b>	str	✗	Comentario opcional.
<b>calificacion_general</b>	int	✗	Calificación general.

### @app.route('/agradecimiento') → agradecimiento()

Elemento	Valor
<b>Ruta</b>	/agradecimiento
<b>Nombre de Función</b>	agradecimiento()
<b>Método</b>	GET
<b>Decorador</b>	@login_required

<b>Descripción</b>	Muestra una página de agradecimiento después de completar la encuesta.
<b>Plantilla HTML</b>	agradecimiento.html

**@app.route('/admin/reportes\_sucursal') → admin\_reportes\_sucursal()**

Elemento	Valor
<b>Ruta</b>	/admin/reportes_sucursal
<b>Nombre de Función</b>	admin_reportes_sucursal()
<b>Método</b>	GET
<b>Decorador</b>	@login_required
<b>Restricción</b>	Solo accesible por usuarios con privilegios = 1 (administradores).

### Funcionalidad General

Elemento	Valor
<b>Objetivo</b>	Mostrar reportes de encuestas por sucursal con filtros por país, sucursal y fechas.
<b>Filtro por Fechas</b>	Por defecto muestra los últimos 30 días si no se seleccionan fechas.
<b>Origen de Datos</b>	Consulta a la tabla respuestas con JOIN a usuarios.

### Parámetros de Filtro (request.args)

Parámetro	Tipo	Descripción
<b>pais</b>	str	País filtrado. Opcional.
<b>sucursal</b>	str	Sucursal filtrada. Opcional.

<b>fecha_inicio</b>	str (YYYY-MM-DD)	Fecha mínima. Si no se indica, se usa hace 30 días.
<b>fecha_fin</b>	str (YYYY-MM-DD)	Fecha máxima. Si no se indica, se usa fecha actual.

### Métricas Calculadas (con Pandas)

Métrica	Descripción
<b>total_respuestas</b>	Total de respuestas dentro del rango.
<b>promedio_calificacion</b>	Promedio de calificacion_general.
<b>satisfaccion_servicio</b>	Promedio de atencion_personal (escala 1–5) multiplicado por 20 (porcentaje).
<b>tasa_retorno</b>	% de respuestas con volveria_visitar = "si".

### Indicadores por Sucursal

Indicador	Descripción
<b>total_respuestas</b>	Cantidad de respuestas de esa sucursal.
<b>calidad_comida</b>	Promedio de calidad de comida.
<b>atencion_personal</b>	Promedio de atención del personal.
<b>tiempo_espera</b>	% de respuestas con tiempo_espera = "si".
<b>calificacion_general</b>	Promedio de calificación general.
<b>intencion_retorno</b>	% de respuestas afirmativas en volveria_visitar.

**@app.route('/recuperar\_contrasena') → recuperar\_contrasena()**

Elemento	Valor
<b>Ruta</b>	/recuperar_contrasena
<b>Nombre de Función</b>	recuperar_contrasena()



<b>Métodos HTTP</b>	GET, POST
<b>Descripción</b>	Permite a usuarios solicitar recuperación de contraseña mediante correo electrónico. Envía un código de verificación para resetear contraseña.
<b>Plantilla HTML</b>	recuperar_contraseña.html

### Flujo Funcional

Paso	Descripción
1.	En GET, muestra el formulario para ingresar correo. Si el usuario fue redirigido desde login tras intentos fallidos, prellena el correo y genera código automáticamente.
2.	En POST, recibe el correo ingresado, valida existencia en BD.
3.	Si existe usuario, genera un código de verificación (generar_codigo_verificacion()).
4.	Actualiza en BD el campo codigo_verificacion y fecha_codigo para ese usuario.
5.	Envía correo con código mediante enviar_codigo_recuperacion().
6.	Si correo enviado correctamente, guarda correo_recuperacion en sesión y redirige a ruta verificar_codigo_recuperacion.
7.	En caso de error, muestra mensajes adecuados (error o mensaje).

### Variables de Sesión

Variable	Descripción
correo_recuperacion	Guarda el correo para el cual se está haciendo la recuperación, para uso en procesos posteriores.

### Errores/Mensajes

Condición	Mensaje mostrado
No se ingresa correo	"Por favor, ingresa tu correo electrónico"
Correo no existe en BD	"No existe una cuenta con ese correo electrónico"

<b>Error al enviar el código</b>	"Error al enviar el código. Intenta nuevamente."
<b>Código enviado con éxito</b>	"Se ha enviado un código de verificación a tu correo" (o correo prellenado)

### Dependencias Externas

Función externa	Descripción
<b>generar_codigo_verificacion()</b>	Genera un código aleatorio para verificación.
<b>enviar_codigo_recuperacion(correo, codigo)</b>	Envía correo electrónico con el código al usuario.

**@app.route('/verificar\_codigo\_recuperacion') →  
verificar\_codigo\_recuperacion()**

Elemento	Valor
<b>Ruta</b>	/verificar_codigo_recuperacion
<b>Nombre de Función</b>	verificar_codigo_recuperacion()
<b>Métodos HTTP</b>	GET, POST
<b>Descripción</b>	Permite al usuario ingresar el código recibido por correo para validar la recuperación de contraseña. Verifica validez y expiración del código.
<b>Plantilla HTML</b>	verificar_codigo_recuperacion.html

### Flujo Funcional

Paso	Descripción
1.	Si no existe variable correo_recuperacion en sesión, redirige a la ruta de recuperación inicial.
2.	En GET, muestra el formulario para ingresar código.
3.	En POST, valida que se ingrese un código.
4.	Consulta en BD el código almacenado y su fecha de creación para el correo en sesión.
5.	Verifica que el código sea correcto y que no tenga más de 30 minutos de antigüedad.
6.	Si es válido, establece codigo_verificado en sesión y redirige a ruta nueva_contrasena.
7.	Si es inválido o expirado, muestra error correspondiente.

**@app.route('/nueva\_contrasena') → nueva\_contrasena()**

Elemento	Valor
<b>Ruta</b>	/nueva_contrasena
<b>Nombre de Función</b>	nueva_contrasena()
<b>Métodos HTTP</b>	GET, POST
<b>Descripción</b>	Permite al usuario establecer una nueva contraseña tras validar el código de recuperación.
<b>Plantilla HTML</b>	nueva_contrasena.html

### Flujo Funcional

Paso	Descripción
1.	Verifica que en sesión existan correo_recuperacion y codigo_verificado. Si no, redirige a recuperación inicial.
2.	En GET, muestra formulario para ingresar y confirmar nueva contraseña.
3.	En POST, valida que los campos estén completos y las contraseñas coincidan.
4.	Hashea la nueva contraseña con hash_password().

5.	Actualiza la contraseña en la BD y elimina el código de verificación (campo y fecha).
6.	Limpia variables de sesión relacionadas.
7.	Muestra mensaje de éxito y redirige a login.

### Variables de Sesión

Variable	Descripción
<b>correo_recuperacion</b>	Correo electrónico para recuperación, persistente entre rutas.
<b>codigo_verificado</b>	Indica que el código fue validado correctamente.

### Errores/Mensajes

Condición	Mensaje mostrado
<b>No ingresar código (verificación)</b>	"Por favor, ingresa el código de verificación"
<b>Código incorrecto o expirado</b>	"Código incorrecto. Intenta nuevamente." / "El código ha expirado. Solicita uno nuevo."
<b>Campos vacíos (nueva contraseña)</b>	"Por favor, completa todos los campos"
<b>Contraseñas no coinciden</b>	"Las contraseñas no coinciden"
<b>Contraseña actualizada con éxito</b>	Flash: "Tu contraseña ha sido actualizada exitosamente"

### Dependencias Externas

Función externa	Descripción
<b>hash_password(nueva_contrasena)</b>	Hashea la nueva contraseña para almacenarla de forma segura.

**@app.errorhandler(404) → pagina\_no\_encontrada(error)**

Elemento	Valor
<b>Ruta</b>	No aplica (manejador de error 404)
<b>Nombre de Función</b>	pagina_no_encontrada(error)
<b>Métodos HTTP</b>	No aplica (se activa automáticamente ante error 404)
<b>Descripción</b>	Renderiza una página personalizada cuando se solicita una URL no encontrada.
<b>Plantilla HTML</b>	404.html
<b>Código HTTP</b>	404

### **@app.errorhandler(500) → error\_servidor(error)**

Elemento	Valor
<b>Ruta</b>	No aplica (manejador de error 500)
<b>Nombre de Función</b>	error_servidor(error)
<b>Métodos HTTP</b>	No aplica (se activa automáticamente ante error 500)
<b>Descripción</b>	Renderiza una página personalizada cuando ocurre un error interno del servidor.
<b>Plantilla HTML</b>	500.html
<b>Código HTTP</b>	500

### **Arranque de la Aplicación**

Elemento	Valor
<b>Bloque Principal</b>	if __name__ == '__main__':
<b>Comando de ejecución</b>	app.run(debug=os.environ.get('FLASK_DEBUG', 'False') == 'True')

<b>Descripción</b>	Ejecuta la app en modo debug si la variable de entorno FLASK_DEBUG está establecida en 'True'. De lo contrario, ejecuta sin debug.
--------------------	--

## STATIC\_FOLDER

Elemento	Valor
<b>Variable</b>	STATIC_FOLDER
<b>Descripción</b>	Ruta absoluta a la carpeta static donde se almacenan archivos estáticos (CSS, JS, imágenes, etc.).
<b>Valor</b>	os.path.join(os.path.dirname(os.path.abspath(__file__)), 'static')
<b>Notas</b>	Se calcula automáticamente a partir de la ubicación del archivo actual.

## IMAGES\_FOLDER

Elemento	Valor
<b>Variable</b>	IMAGES_FOLDER
<b>Descripción</b>	Ruta a la subcarpeta static/images donde se guardarán los gráficos generados.
<b>Valor</b>	os.path.join(STATIC_FOLDER, 'images')
<b>Notas</b>	La carpeta se crea si no existe mediante os.makedirs(..., exist_ok=True).

## app.secret\_key

Elemento	Valor
<b>Variable</b>	app.secret_key
<b>Descripción</b>	Clave secreta usada por Flask para manejar sesiones y proteger cookies.
<b>Valor por defecto</b>	'5mcJAvC298\$7' (solo si no se define la variable de entorno SECRET_KEY)
<b>Recomendación</b>	En producción debe definirse como variable de entorno segura.

### Configuración de Base de Datos (MySQL)

Clave de Configuración	Valor por Defecto	Descripción
<b>MYSQL_HOST</b>	'localhost'	Dirección del servidor de la base de datos.
<b>MYSQL_USER</b>	'root'	Nombre de usuario para la conexión a la base de datos.
<b>MYSQL_PASSWORD</b>	'Ericko11\$'	Contraseña del usuario de la base de datos.
<b>MYSQL_DB</b>	'encuestas_db'	Nombre de la base de datos a la que se conectará la app.

### Configuración de Correo Electrónico

Variable	Valor por Defecto	Descripción
<b>EMAIL_SENDER</b>	'takenokamu@gmail.com'	Dirección de correo desde la cual se enviarán los mensajes automáticos.
<b>EMAIL_PASSWORD</b>	'kcdvjijtgcsucvmb'	Contraseña o token de aplicación del correo remitente.

### Seguridad y Verificación

Función	Descripción
<b>hash_password(password)</b>	Devuelve una versión hasheada de una contraseña utilizando <code>werkzeug.security.generate_password_hash</code> . Utilizado para almacenar contraseñas de forma segura.

<b>generar_codigo_verificacion()</b>	Genera y devuelve un código de 6 dígitos aleatorio (tipo OTP), como string. Utilizado para verificar identidad vía email.
<b>enviar_codigo_verificacion(destinatario, codigo)</b>	Envía un correo con un código de verificación al usuario. Usa conexión segura SMTP_SSL y los datos de autenticación del remitente. Devuelve True si se envía correctamente, False si hay error.

### Cupones de Descuento

Función	Descripción
<b>generar_codigo_cupon()</b>	Genera un código alfanumérico aleatorio de 6 caracteres (mayúsculas y dígitos). Se usa como identificador único de cupón.
<b>enviar_cupon(destinatario, codigo, porcentaje)</b>	Envía un correo con el código de cupón generado, indicando un porcentaje de descuento. Si existe una imagen de cupón (cupon_X.png) en /static/images/, se adjunta al correo. Usa MIMEMultipart y MIMEImage. Retorna True o False según éxito del envío.

### Control de Acceso

Decorador	Descripción
<b>@login_required(f)</b>	Decorador que redirige a login si no hay una sesión activa con clave 'usuario_id'. Se usa para proteger rutas restringidas a usuarios autenticados.

### verificar\_actividad\_reciente(id\_usuario)

Elemento	Descripción
<b>id_usuario</b>	Identificador único del usuario. Se espera que sea un entero.



<b>mysql.connection.cursor()</b>	Objeto para ejecutar consultas a la base de datos MySQL.
<b>respuestas</b>	Tabla de la base de datos donde se almacenan las respuestas de las encuestas.
<b>fecha_respuesta</b>	Fecha y hora en que el usuario respondió la encuesta.
<b>cur.execute(...)</b>	Consulta SQL que busca la fecha de respuesta más reciente del usuario.
<b>datetime.now()</b>	Obtiene la fecha y hora actual del sistema.
<b>timedelta(days=30)</b>	Objeto que representa una diferencia de 30 días, usado para validar antigüedad.
<b>Condición</b>	Si la respuesta fue hace menos de 30 días, se considera actividad reciente.
<b>Retorno</b>	True si hubo actividad reciente, False en caso contrario.

### **enviar\_codigo\_recuperacion(destinatario, codigo)**

<b>Elemento</b>	<b>Descripción</b>
<b>destinatario</b>	Dirección de correo electrónico del usuario que solicitó recuperar la contraseña.
<b>codigo</b>	Código numérico de verificación (6 dígitos).
<b>EmailMessage()</b>	Clase de email.message usada para crear el contenido del correo.
<b>EMAIL_SENDER</b>	Correo electrónico configurado como remitente del sistema.
<b>EMAIL_PASSWORD</b>	Contraseña o token del correo emisor.
<b>mensaje.set_content(...)</b>	Cuerpo del mensaje enviado en texto plano.
<b>smtpplib.SMTP_SSL(...)</b>	Protocolo seguro para enviar el correo mediante el servidor SMTP de Gmail.
<b>smtp.login(...)</b>	Inicio de sesión en el servidor SMTP con las credenciales configuradas.
<b>smtp.send_message(...)</b>	Envía el mensaje de recuperación al destinatario.
<b>Retorno</b>	True si el envío fue exitoso, False si ocurrió un error.
<b>Logging</b>	Usa logging.info o logging.error para registrar el resultado del envío.

### generar\_grafico\_barras(...)

Elemento	Descripción
<b>datos</b>	Lista de valores numéricos que representan las alturas de cada barra.
<b>etiquetas</b>	Lista de etiquetas (categorías) para el eje X, una por cada barra.
<b>titulo</b>	Texto que aparecerá como título del gráfico.
<b>etiqueta_x</b>	Etiqueta del eje horizontal (X).
<b>etiqueta_y</b>	Etiqueta del eje vertical (Y).
<b>nombre_archivo</b>	Nombre base del archivo donde se guardará el gráfico (sin extensión .png).
<b>promedio</b>	Valor numérico opcional para dibujar una línea horizontal de referencia.
<b>plt.bar(...)</b>	Función de matplotlib que dibuja las barras.
<b>plt.axhline(...)</b>	Dibuja una línea horizontal (por ejemplo, un promedio).
<b>IMAGES_FOLDER</b>	Ruta de la carpeta donde se guarda el gráfico generado.
<b>Archivo generado</b>	Imagen .png del gráfico de barras, almacenada en static/images/.

### generar\_grafico\_pastel(...)

Elemento	Descripción
<b>datos</b>	Lista de valores numéricos, cada uno correspondiente a una porción del pastel.
<b>etiquetas</b>	Lista de etiquetas para identificar cada porción.
<b>titulo</b>	Título que aparecerá en el gráfico.
<b>nombre_archivo</b>	Nombre del archivo .png donde se guardará la imagen.
<b>colores</b>	Lista opcional de colores personalizados para cada porción.
<b>plt.pie(...)</b>	Dibuja el gráfico de pastel con etiquetas y porcentajes.
<b>plt.axis('equal')</b>	Asegura que el pastel tenga forma circular.
<b>Archivo generado</b>	Imagen .png del gráfico de pastel, guardada en static/images/.

### generar\_grafico\_histograma(...)

Elemento	Descripción
<b>datos</b>	Serie de pandas con los datos numéricos a analizar.
<b>bins</b>	Lista de límites que separan los rangos (intervalos) del histograma.
<b>etiquetas_bins</b>	Etiquetas a colocar en el eje X según los bins.
<b>titulo</b>	Título del gráfico.
<b>etiqueta_x</b>	Etiqueta del eje X (eje de las clases o intervalos).
<b>etiqueta_y</b>	Etiqueta del eje Y (frecuencia de cada clase).
<b>nombre_archivo</b>	Nombre del archivo .png a guardar.
<b>plt.hist(...)</b>	Dibuja el histograma con los valores agrupados por intervalos.
<b>Archivo generado</b>	Imagen .png del histograma, guardada en static/images/.

## 12. Pruebas y Testeos

### Base de datos

#### Vulnerabilidades en SQL

##### *Vulnerabilidad 1: Contraseñas almacenadas en texto plano*

Aspecto	Detalle
<b>Vulnerabilidad</b>	Guardar contraseñas sin cifrar pone en riesgo total la seguridad si la base de datos se ve comprometida.
<b>Solución aplicada</b>	El campo contraseña se mantiene como VARCHAR(255), pero las contraseñas se almacenan como <b>hashes bcrypt</b> generados con <code>werkzeug.security.generate_password_hash</code> .

##### Vulnerabilidad 2: Enumeración de usuarios mediante errores de duplicidad

Aspecto	Detalle
<b>Vulnerabilidad</b>	Si al registrar un usuario se muestra un mensaje distinto según si el correo ya existe, puede usarse para detectar correos válidos y obtener información de usuarios.
<b>Solución aplicada</b>	Se mantienen las restricciones UNIQUE en correo y telefono. El manejo de errores es <b>genérico</b> en el código Python (por ejemplo, flash("Error en el registro")), sin exponer la causa exacta.

### *Vulnerabilidad 3: Relaciones de clave foránea sin control de eliminación*

Aspecto	Detalle
<b>Vulnerabilidad</b>	Si no se define un comportamiento de eliminación (ON DELETE), pueden quedar datos “huérfanos” cuando se borra un usuario.
<b>Solución aplicada</b>	- En la tabla <b>respuestas</b> se usa ON DELETE SET NULL para conservar los registros de encuesta aunque el usuario sea eliminado.- En la tabla <b>cupones</b> se usa ON DELETE CASCADE para eliminar automáticamente los cupones asociados a un usuario borrado.

### *Vulnerabilidad 4: Códigos de verificación reutilizables o sin control*

Aspecto	Detalle
<b>Vulnerabilidad</b>	Si los códigos de verificación no expiran o pueden reutilizarse, pueden ser explotados (replay attacks o fuerza bruta).
<b>Solución aplicada</b>	- Cada código de verificación <b>expira a los 10 minutos</b> (campo fecha_codigo).- Se limitan los intentos de verificación en la sesión.- Un código correctamente verificado <b>no puede reutilizarse</b> (se elimina o se marca como usado).

## Pruebas en la base de datos

### Prueba 1: Prueba de carga masiva en base de datos

N°	1
Título	Prueba de carga masiva en base de datos
Propósito	Probar que la base de datos pudiera almacenar una entrada de datos masiva en comparación con pruebas anteriores.
¿Qué se hizo?	Realizamos una sesión con un grupo de 24 alumnos usando la aplicación simultáneamente para resolver encuestas, midiendo cómo reaccionaba la base de datos ante múltiples usuarios concurrentes.
Resultados Observados	Positivo: la base de datos registró correctamente todas las respuestas sin presentar errores o pérdidas de datos.
Conclusión	La base de datos soporta cargas masivas de usuarios concurrentes y mantiene la integridad de la información, entregando cupones a todos los participantes sin fallo.

## Pruebas en la lógica de la aplicación

### Prueba 1: Mejora de encriptación de datos

N°	1
Título	Mejora de encriptación de datos
Propósito	Mejorar el nivel de seguridad de encriptación de datos de los usuarios (se encriptaron las contraseñas)

¿Qué se hizo?	Inicialmente se sustituyó el hasheo de contraseñas por una encriptación usando la biblioteca Werkzeug; luego, como segunda capa, se restituyó el hasheo y se añadió la encriptación de Werkzeug.
Resultados Observados	Positivo: el código reaccionó correctamente a las modificaciones y las contraseñas quedaron doblemente encriptadas.
Conclusión	Es una buena manera de proteger los datos de los usuarios, ya que una vez encriptados no pueden descryptarse, evitando que un ciberdelincuente obtenga esas credenciales.

## Prueba 2: Cambio de lógica

N°	2
Título	Cambio de lógica
Propósito	Cambiar la lógica del código para que un atacante no pueda saber si un correo ya está registrado.
¿Qué se hizo?	Se modificaron los mensajes devueltos por el sistema al registrar correo, de modo que nunca se indique explícitamente “este correo ya existe”.
Resultados Observados	Positivo: el cambio fue ligero y no afectó la funcionalidad; la vulnerabilidad quedó cubierta de manera inmediata.
Conclusión	Aunque el cambio es pequeño a nivel de código, es muy necesario desde el

	punto de vista de la lógica, pues previene la divulgación de información sobre usuarios existentes.
--	---

### Prueba 3: Consultas parametrizadas

N°	3
Título	Consultas parametrizadas
Propósito	Prevenir ataques de inyección SQL.
¿Qué se hizo?	Se reemplazaron las consultas SQL directas por consultas parametrizadas usando cursor.execute, evitando concatenación de strings.
Resultados Observados	Positivo: la base de datos quedó protegida exitosamente y no se logró inyectar código malicioso.
Conclusión	El uso de consultas parametrizadas es una defensa eficaz contra inyección SQL y fortalece significativamente la seguridad de la aplicación.

### Prueba 4: Cifrado Fernet

N°	4
Título	Cifrado Fernet
Propósito	Cifrar correos electrónicos y números telefónicos de usuarios antes de almacenarlos.
¿Qué se hizo?	Se utilizó la biblioteca cryptography (Fernet) para cifrar los datos sensibles

	antes de guardarlos en la base de datos.
Resultados Observados	Positivo: los datos quedaron cifrados correctamente y no son accesibles sin la clave.
Conclusión	El cifrado con Fernet garantiza la confidencialidad de la información sensible de los usuarios, dificultando su exposición en caso de brechas.

### Prueba 5: Expiración de códigos de verificación

N°	5
Título	Expiración de códigos de verificación
Propósito	Limitar la validez de los códigos de verificación y recuperación, evitando que permanezcan activos indefinidamente.
¿Qué se hizo?	Se implementó un temporizador de 30 minutos para cada código, usando <code>datetime.now()</code> y validaciones al verificar el código.
Resultados Observados	Positivo: los códigos expiran tras 30 minutos y rechazan intentos de uso posteriores.
Conclusión	La expiración de códigos mejora la seguridad al reducir la ventana de oportunidad para ataques por reutilización de códigos.



### Prueba 6: Límite de intentos

N°	6
Título	Límite de intentos
Propósito	Limitar intentos de login previniendo ataques de fuerza bruta.
¿Qué se hizo?	Se implementó un sistema de control por IP para limitar envíos simultáneos de formularios provenientes de la misma IP.
Resultados Observados	Positivo: el sistema bloquea peticiones excesivas, mitigando efectivamente los ataques de fuerza bruta.
Conclusión	Es un cambio esencial para proteger la base de datos, impidiendo que atacantes automaticen cientos de solicitudes por minuto desde la misma IP.

### Prueba 7: Uso de variables de entorno

N°	7
Título	Uso de variables de entorno
Propósito	Almacenar datos sensibles para no exponerlos en el código.
¿Qué se hizo?	Se definieron variables de entorno para credenciales sensibles (EMAIL_PASSWORD, SECRET_KEY, EMAIL_PHONE_KEY) en lugar de hardcodearlas en el código.
Resultados Observados	Positivo: las credenciales quedan fuera del repositorio, reduciendo el riesgo de exposición en caso de filtraciones.

Conclusión	El uso de variables de entorno refuerza la seguridad de la aplicación al evitar la exposición de datos sensibles en el código fuente y facilita la gestión de credenciales en diferentes entornos.
------------	--

### Prueba 8: Manejo personalizado de errores HTTP

Nº	8
Título	Manejo personalizado de errores HTTP
Propósito	Evitar filtraciones de información del servidor al usuario.
¿Qué se hizo?	Se implementaron rutas <code>@app.errorhandler(404)</code> y <code>500</code> con plantillas <code>404.html</code> y <code>500.html</code> .
Resultados Observados	Las rutas de error muestran mensajes amigables sin revelar trazas internas.
Conclusión	El sistema evita exponer información crítica en caso de errores inesperados.

### Prueba 9: Validación de entrada del usuario

Nº	9
Título	Validación de entrada del usuario
Propósito	Prevenir inyecciones o errores por datos mal formateados.
¿Qué se hizo?	Se aplicaron expresiones regulares en campos como correo electrónico y se verificó que todos los campos estén completos antes de ejecutar lógica.
Resultados Observados	Datos inválidos como correos sin <code>@</code> fueron correctamente rechazados sin errores de backend.
Conclusión	La validación protege contra inyecciones y errores de lógica en formularios.

### Prueba 10: Control de sesiones en rutas protegidas

Nº	10
Título	Control de sesiones en rutas protegidas
Propósito	Asegurar que solo usuarios autenticados accedan a rutas sensibles.
¿Qué se hizo?	Se utilizó un decorador <code>@login_required</code> en rutas clave como <code>/encuesta</code> y <code>/reportes_admin</code> .

Resultados Observados	Al intentar acceder sin sesión, el sistema redirige al login con un mensaje de advertencia.
Conclusión	Las rutas protegidas funcionan correctamente, impidiendo accesos no autorizados.

### Prueba 11: Pruebas de inyección SQL controladas

Nº	11
Título	Pruebas de inyección SQL controladas
Propósito	Verificar resistencia contra ataques SQL Injection.
¿Qué se hizo?	Se probaron entradas como ' OR 1=1 --, '; DROP TABLE usuarios; -- en formularios de login y recuperación.
Resultados Observados	El sistema rechazó todas las entradas maliciosas; no hubo errores ni acceso no autorizado.
Conclusión	El uso de consultas parametrizadas protege eficazmente contra inyecciones SQL.

### Prueba 12: Registro de eventos sensibles (logging)

Nº	12
Título	Registro de eventos sensibles (logging)
Propósito	Auditar intentos fallidos o sospechosos.
¿Qué se hizo?	Se implementó logging para guardar eventos como errores de verificación, envío de códigos y accesos inválidos en un archivo local.
Resultados Observados	Los eventos fueron registrados en registro_seguridad.log correctamente con fecha, correo e IP.
Conclusión	El registro proporciona trazabilidad ante incidentes de seguridad o mal uso del sistema.

## 13. Referencias

- OWASP Foundation. (2021). OWASP Top Ten Web Application Security Risks. Open Web
- Application Security Project. <https://owasp.org/Top10/>
- Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python. O'Reilly Media.
- Python Software Foundation. (2023). cryptography — Strong encryption for Python.<https://cryptography.io/en/latest/>
- Palit, R., & Parveen, R. (2021). Securing Web Applications: Authentication, Authorization, and Encryption. In Proceedings of the International Conference on Computer and
- Communication Technologies (pp. 345–356). Springer. [https://doi.org/10.1007/978-981-158289-9\\_32](https://doi.org/10.1007/978-981-158289-9_32)
- Vlasenko, I. (2020). Secure Password Storage in Python: Hashing with bcrypt, scrypt and