Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Software Avanzado
Primer Semestre 2024
Catedrático:
Ing. Marco Tulio Aldana Prillwitz
Tutor académico:
Ariana Pérez

Proyecto Fase 1

DOCUMENTACIÓN

Alexandro Provenzale Pérez
201904012

Table of Contents

Especificación detallada de los requisitos funcionales y no funcionales del sistema.

La especificación detallada de los requisitos funcionales y no funcionales del sistema de tickets incluye:

Requisitos Funcionales:

1. **Creación de Tickets:** Los clientes pueden crear tickets a través de un portal web con un formulario intuitivo que incluye campos como nombre, correo electrónico, número de teléfono, descripción del problema, tipo de problema y prioridad.
2. **Seguimiento de Tickets:** Los clientes y el equipo de soporte pueden dar seguimiento a los tickets creados, visualizar su estado actual y recibir notificaciones sobre actualizaciones.
3. **Resolución de Tickets:** El equipo de soporte puede resolver los tickets asignados, registrar la solución proporcionada y cerrar el ticket una vez completado.
4. **Gestión de Usuarios:** Los administradores pueden crear y gestionar usuarios, configurar el sistema y generar informes.
5. **Generación de Informes y Análisis:** El sistema genera informes sobre el volumen de tickets, tiempo de respuesta y satisfacción del cliente, permitiendo análisis para la toma de decisiones estratégicas.
6. **Base de Conocimiento**: Incluye una base de conocimiento con artículos y tutoriales para ayudar a los clientes a resolver problemas comunes.
7. **Encuestas de Satisfacción:** Envío de encuestas de satisfacción a los clientes para evaluar la calidad del servicio recibido.
8. **Integración con otras herramientas:** Posibilidad de integrarse con otras herramientas, como un CRM para la gestión de proveedores y tipos de servicios.

Requisitos No Funcionales:

1. **Seguridad:** El sistema debe ser seguro para proteger la información confidencial de los clientes.
2. **Escalabilidad:** Debe ser escalable para soportar un alto volumen de tickets.
3. **Disponibilidad:** El sistema debe estar disponible 24/7 para garantizar un servicio continuo a los usuarios.

Descripción de cada funcionalidad.

Creación de Tickets

- Permite a los clientes crear tickets a través de un portal web.
- Incluye un formulario intuitivo con campos para nombre, correo electrónico, número de teléfono, descripción del problema, tipo de problema y prioridad.
- Posibilidad de adjuntar archivos como capturas de pantalla o registros de errores.

Seguimiento de Tickets

- Permite a los clientes y al equipo de soporte seguir el estado de los tickets en tiempo real a través del portal web.
- Proporciona información detallada sobre cada ticket, como número de ticket, estado, prioridad, fechas de creación y actualización, agente asignado, historial de comunicaciones y solución al problema.

Resolución de Tickets

- Permite al equipo de soporte realizar acciones como ver detalles del ticket, asignar a un agente, escalar a un nivel superior, agregar comentarios, cambiar estado y proporcionar solución.

Gestión de Usuarios

- Permite la creación de diferentes tipos de usuarios como clientes, agentes y administradores.
- Cada tipo de usuario tiene permisos específicos para acceder a las funcionalidades del sistema.

Informes y Análisis

- Genera informes y análisis sobre el volumen de tickets, tiempo de respuesta y satisfacción del cliente.
- Los informes pueden ser personalizados para mostrar información relevante a cada usuario.
- Ayuda en la toma de decisiones estratégicas para la gestión del servicio de atención al cliente.

Base de Conocimiento

- Incluye una base de conocimiento con artículos y tutoriales para ayudar a los clientes a resolver problemas comunes.

Encuestas de Satisfacción

- Permite enviar encuestas de satisfacción a los clientes para evaluar la calidad del servicio recibido.

Integración con Otras Herramientas

- Posibilidad de integrarse con otras herramientas, como un CRM para la gestión de proveedores y tipos de servicios.

Detalles sobre los tipos de usuarios y sus permisos.

Clientes

Permisos:

- Crear tickets para solicitar asistencia o soporte técnico.
- Seguir el estado de los tickets creados.

Funcionalidades:

- Acceso al formulario de creación de tickets.
- Visualización del estado actual de los tickets.

Agentes

Permisos:

- Ver, asignar, escalar, resolver y responder a tickets.

Funcionalidades:

- Acceso a la lista de tickets pendientes.
- Asignación de tickets a agentes disponibles.
- Registro de soluciones y actualizaciones en los tickets.

Administradores

Permisos:

- Crear y gestionar usuarios.
- Configurar el sistema.
- Generar informes.

Funcionalidades:

- Acceso a la gestión de usuarios.
- Configuración de parámetros del sistema.
- Generación de informes sobre el rendimiento del sistema.

Contratos de microservicios

Encargado	Microservicio	# de contrato	EndPoints	Descripción
Alexandro Provenzale Pérez	Usuarios	1	Crear usuario	Creación de usuario interno
		2	Obtener agentes	Retorna los agentes que se pueden asignar a un ticket
	Ticket	1	Crear ticket	Crea un ticket con los datos de un cliente
		2	Obtener ticket	Obtiene todos los tickets creados y activos
		3	Editar ticket	Edita el estado de un ticket y algunas características
		4	Ver ticket	Accede a un ticket en específico

ID: 001	**NOMBRE:**	
PRIORIDAD:	**HISTORIA DE USUARIO:**	
ESTIMADO:		
MÓDULO: general		

CRITERIOS DE ACEPTACIÓN:

- Mensaje con código 200 para asegurarse de que se encuentra en línea

RUTA: /
MÉTODO: POST
FORMATO DE ENTRADA: JSON
HEADER:

ATRIBUTO	TIPO	DESCRIPCIÓN
Content_type	header	Application/json

BODY:

ATRIBUTO	TIPO	DESCRIPCIÓN

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	TIPO	DESCRIPCIÓN
mensaje	cadena	Mensaje que se mostrará

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	DESCRIPCIÓN
400	Error al ingresar datos
500	Error el usuario ingresado ya existe
600	Contraseña no se cumple los requisitos

PARÁMETROS DE ENTRADA	
PARÁMETROS DE SALIDA EXITOSA	`HEADER: { status: 200 }` `BODY: {'mensaje': 'Servidor en linea'}`
PARÁMETROS DE SALIDA FALLIDA	`HEADER: { status: 400 }` `BODY: {'error': mensaje'}`

ID: 002	**NOMBRE:**	
PRIORIDAD:	**HISTORIA DE USUARIO:**	
ESTIMADO:		
MÓDULO: user		

CRITERIOS DE ACEPTACIÓN:

- Que la inserción de datos se haya realizado correctamente dentro de la base de datos.

RUTA: /crearusuario
MÉTODO: POST
FORMATO DE ENTRADA: JSON
HEADER:

ATRIBUTO	TIPO	DESCRIPCIÓN
Content_type	header	Application/json

BODY:

ATRIBUTO	TIPO	DESCRIPCIÓN
nombre	cadena	Nombre y apellido

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	TIPO	DESCRIPCIÓN
mensaje	cadena	Mensaje que se mostrará

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	DESCRIPCIÓN
400	Error al ingresar datos
500	Error el usuario ingresado ya existe
600	Contraseña no se cumple los requisitos

PARÁMETROS DE ENTRADA	
PARÁMETROS DE SALIDA EXITOSA	`HEADER: { status: 200 }` `BODY: {'mensaje': 'Usuario creado exitosamente'}`
PARÁMETROS DE SALIDA FALLIDA	`HEADER: { status: 400 }` `BODY: {'error': mensaje}`

ID: 003	**NOMBRE:**	
PRIORIDAD:	**HISTORIA DE USUARIO:** Creación de ticket	
ESTIMADO:		
MÓDULO: ticket		

CRITERIOS DE ACEPTACIÓN:

- Los datos deben haberse actualizado correctamente en la base de datos

RUTA: /api/tickets
MÉTODO: POST
FORMATO DE ENTRADA: JSON
HEADER:

ATRIBUTO	TIPO	DESCRIPCIÓN
Content_type	header	Application/json

BODY:

ATRIBUTO	TIPO	DESCRIPCIÓN
nombre	cadena	Nombre y apellido

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	TIPO	DESCRIPCIÓN
mensaje	cadena	Mensaje que se mostrará

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	DESCRIPCIÓN
400	Error al ingresar datos
500	Error el usuario ingresado ya existe
600	Contraseña no se cumple los requisitos

PARÁMETROS DE ENTRADA	
PARÁMETROS DE SALIDA EXITOSA	`HEADER: { status: 200 }` `BODY: {'mensaje': 'Usuario actualizado exitosamente'}`
PARÁMETROS DE SALIDA FALLIDA	`HEADER: { status: 400 }` `BODY: {'error': mensaje}`

ID: 004	**NOMBRE:**	
PRIORIDAD:	**HISTORIA DE USUARIO:** Eliminación de ticket	
ESTIMADO:		
MÓDULO: ticket		

CRITERIOS DE ACEPTACIÓN:

- Se debe haber eliminado correctamente al usuario de la base de datos.

RUTA: /eliminarticekt
MÉTODO: DELETE
FORMATO DE ENTRADA: JSON
HEADER:

ATRIBUTO	TIPO	DESCRIPCIÓN
Content_type	header	Application/json

BODY:

ATRIBUTO	TIPO	DESCRIPCIÓN
nombre	cadena	Nombre y apellido

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	TIPO	DESCRIPCIÓN
mensaje	cadena	Mensaje que se mostrará

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	DESCRIPCIÓN
400	Error al ingresar datos
500	Error el usuario ingresado ya existe
600	Contraseña no se cumple los requisitos

PARÁMETROS DE ENTRADA	
PARÁMETROS DE SALIDA EXITOSA	`HEADER: { status: 200 }` `BODY: {'mensaje': 'Usuario eliminado exitosamente'}`
PARÁMETROS DE SALIDA FALLIDA	`HEADER: { status: 400 }` `BODY: {'error': mensaje}`

ID: 005	**NOMBRE:**	
PRIORIDAD:	**HISTORIA DE USUARIO:**	
ESTIMADO:	Se modifica el estado de un ticket existente	
MÓDULO: ticket		

CRITERIOS DE ACEPTACIÓN:

- Obtener todos los datos de la table de usuario.

RUTA: /putticket
MÉTODO: PUT
FORMATO DE ENTRADA: JSON
HEADER:

ATRIBUTO	TIPO	DESCRIPCIÓN
Content_type	header	Application/json

BODY:

ATRIBUTO	TIPO	DESCRIPCIÓN
nombre	cadena	Nombre y apellido

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	TIPO	DESCRIPCIÓN
mensaje	cadena	Mensaje que se mostrará

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	DESCRIPCIÓN
400	Error al ingresar datos
500	Error el usuario ingresado ya existe
600	Contraseña no se cumple los requisitos

PARÁMETROS DE ENTRADA	

PARÁMETROS DE SALIDA EXITOSA	`HEADER: { status: 200 }`
	`BODY: {datos: [idUser, Nombre, Correo, Telefono, TipoUsuario`
	`] }`

PARÁMETROS DE SALIDA FALLIDA	`HEADER: { status: 400 }` `BODY: {'error': mensaje'}`

ID: 006	**NOMBRE:**	
PRIORIDAD:	**HISTORIA DE USUARIO:**	
ESTIMADO:	Se asigna el usuario a un ticket	
MÓDULO: usuario		

CRITERIOS DE ACEPTACIÓN:

- Obtener todos los usuarios de la base de datos.

RUTA: /asignaruser
MÉTODO: POST
FORMATO DE ENTRADA: JSON
HEADER:

ATRIBUTO	TIPO	DESCRIPCIÓN
Content_type	header	Application/json

BODY:

ATRIBUTO	TIPO	DESCRIPCIÓN
nombre	cadena	Nombre y apellido

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	TIPO	DESCRIPCIÓN
mensaje	cadena	Mensaje que se mostrará

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	DESCRIPCIÓN
400	Error al ingresar datos
500	Error el usuario ingresado ya existe
600	Contraseña no se cumple los requisitos

PARÁMETROS DE ENTRADA	

PARÁMETROS DE SALIDA EXITOSA	`HEADER: { status: 200 }`
	`BODY: {datos: [idUser, Nombre, Correo, Telefono, TipoUsuario`
	`] }`

PARÁMETROS DE SALIDA FALLIDA	`HEADER: { status: 400 }` `BODY: {'error': mensaje'}`

ID: 007	**NOMBRE:**	
PRIORIDAD:	**HISTORIA DE USUARIO:**	
ESTIMADO:		
MÓDULO: user		

CRITERIOS DE ACEPTACIÓN:

- Haberse insertado todos los datos en la tabla.

RUTA: / tipouserinsert
MÉTODO: POST
FORMATO DE ENTRADA: JSON
HEADER:

ATRIBUTO	TIPO	DESCRIPCIÓN
Content_type	header	Application/json

BODY:

ATRIBUTO	TIPO	DESCRIPCIÓN
nombre	cadena	Nombre y apellido

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	TIPO	DESCRIPCIÓN
mensaje	cadena	Mensaje que se mostrará

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	DESCRIPCIÓN
400	Error al ingresar datos
500	Error el usuario ingresado ya existe
600	Contraseña no se cumple los requisitos

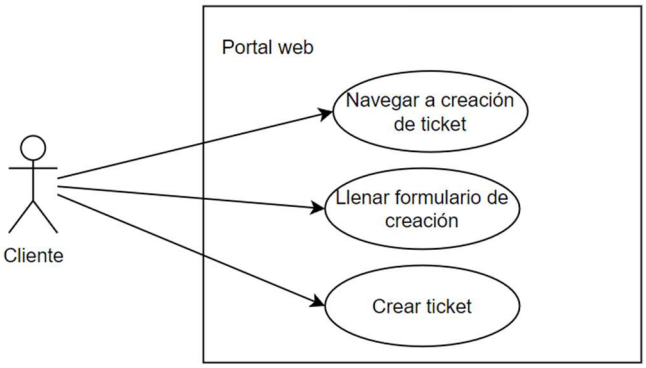
PARÁMETROS DE ENTRADA	

PARÁMETROS DE SALIDA EXITOSA	`HEADER: { status: 200 }`
	`BODY: {mensaje: "Tipo de usuario insertado exitosamente "`
	`}`

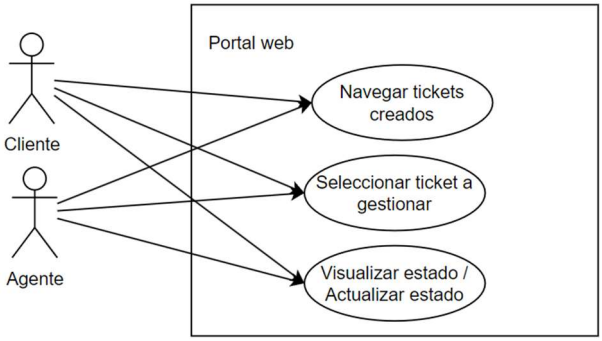
PARÁMETROS DE SALIDA FALLIDA	`HEADER: { status: 400 }` `BODY: {'error': mensaje'}`

Casos de uso

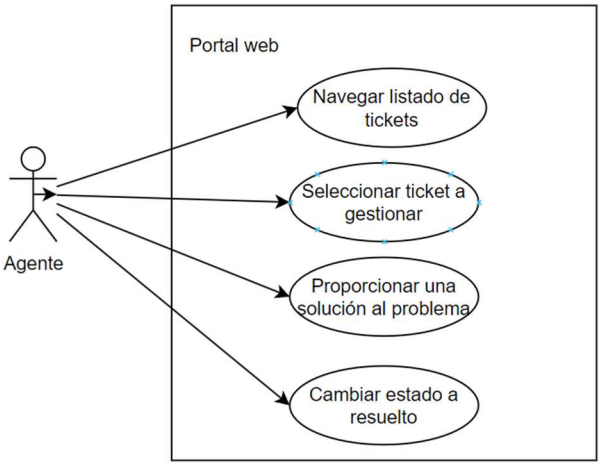
Crear Ticket:



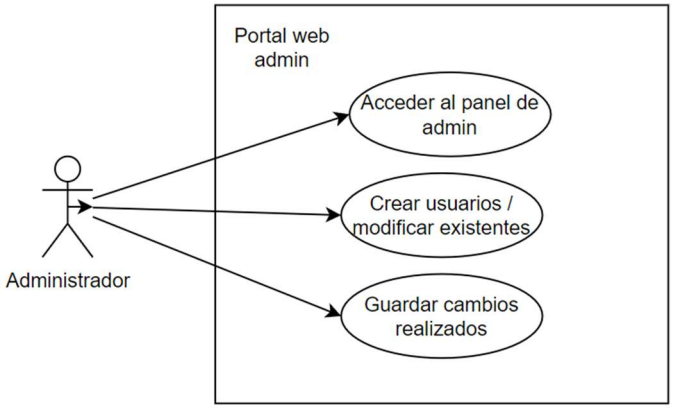
Seguimiento de ticket



Resolver ticket



Gestionar usuarios



Diseño de base de datos

Base de datos NoSQL hecha en MongoDB:

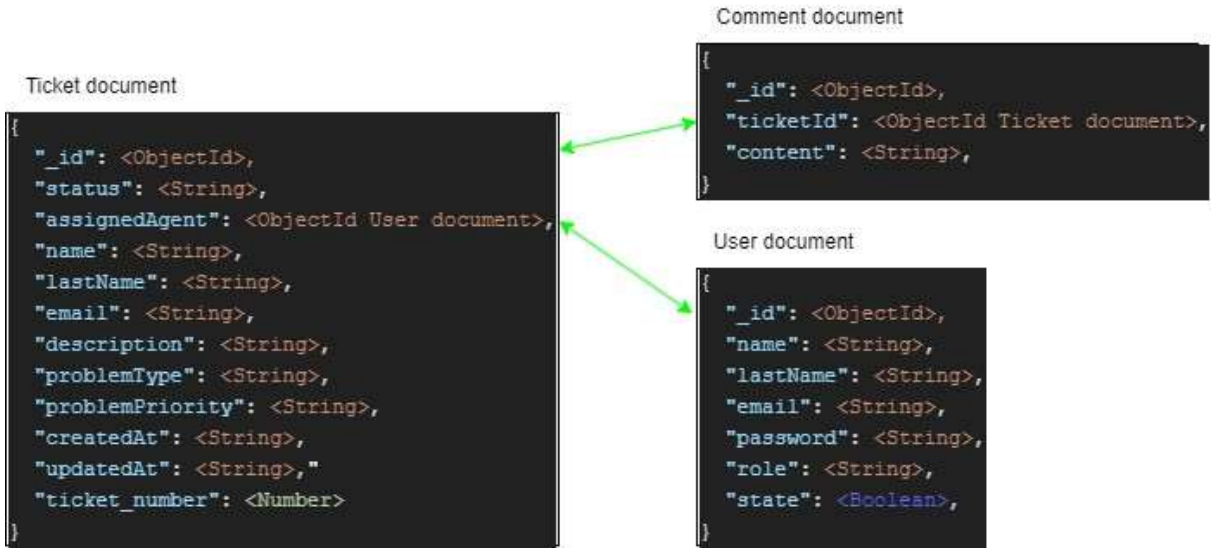


Diagrama de arquitectura