

Solicitudes de visitas guiadas mediante agentes de software

Guided visit requests using software agents

Información del reporte:

Licencia Creative Commons



El contenido de los textos es responsabilidad de los autores y no refleja forzosamente el punto de vista de los dictaminadores, o de los miembros del Comité Editorial, o la postura del editor y la editorial de la publicación.

Para citar este reporte técnico:

Villarreal Brito, G. (2026). Solicitudes de visitas guiadas mediante agentes de software. *Cuadernos Técnicos Universitarios de la DGTIC*, 4(3), páginas (83 - 105). <https://doi.org/10.22201/dgtic.30618096e.2026.4.3.186>

Gustavo Villarreal Brito

Instituto de Ciencias del Mar y Limnología
Universidad Nacional Autónoma de México

gwillarreal@cmarl.unam.mx

ORCID: 0009-0006-9119-1122

Resumen

Las visitas guiadas de la Unidad Académica de Sistemas Arrecifales coadyuvan a las actividades de divulgación científica y vinculación institucional. Antes de la mejora, la recepción de solicitudes dependía principalmente de correos, mensajes y revisiones manuales de disponibilidad, lo que generaba tiempos de respuesta variables, riesgo de omisiones y dificultad para conocer el estado de cada trámite. El objetivo fue diseñar, implementar y validar una arquitectura basada en agentes de *software* para apoyar la recepción, validación, notificación y seguimiento de solicitudes. Estos agentes operan como componentes de automatización basada en reglas, sin utilizar modelos de inteligencia artificial ni sustituir la decisión humana del coordinador.

La metodología incluyó la revisión del proceso manual, la identificación de reglas operativas, el diseño modular y la programación de componentes especializados para consultar agenda, validar formularios, registrar solicitudes, enviar notificaciones, apoyar el agendamiento y emitir confirmaciones.

Como resultado, las solicitudes quedaron centralizadas y asociadas a un estado de seguimiento. En el proceso manual, los tiempos de primera respuesta oscilaron entre 2 h 4 min y 28 días 2 h 50 min, con un promedio de 8.9 días; en el caso automatizado, la confirmación ocurrió en 42 minutos. Se concluyó que la arquitectura propuesta fue adecuada para un servicio con entradas estructuradas,

reglas claras y necesidad de trazabilidad. Como limitación, la decisión final de agenda siguió dependiendo de condiciones logísticas y académicas.

Palabras clave: Agenda digital, automatización administrativa, captura estructurada de solicitudes, gestión de solicitudes, seguimiento operativo.

Abstract

Guided visits at the Unidad Académica de Sistemas Arrecifales are part of its scientific outreach and institutional engagement activities. Before the improvement, request intake relied mainly on emails, messages, and manual availability checks, which generated variable response times, risk of omissions, and difficulty in tracking the status of each request. The objective was to design, implement, and validate a software agent-based architecture to support the intake, validation, notification, and follow-up of guided visit requests. These agents operate as rule-based automation components, without using artificial intelligence models or replacing the coordinator's human decision-making.

The methodology included reviewing the manual process, identifying operational rules, designing a modular architecture, and programming specialized components to query the calendar, validate forms, register requests, send notifications, support scheduling, and issue confirmations. As a result, requests were centralized and associated with a follow-up status. In the manual process, first-response times ranged from 2 h 4 minutes to 28 days 2 h 50 min, with an average of 8.9 days; in the documented automated case, confirmation occurred within 42 minutes. It was concluded that the architecture was suitable for a service with structured inputs, clear rules, and traceability needs. As a limitation, the final scheduling decision continued to depend on logistical and academic conditions.

Keywords: Digital calendar, administrative automation, structured request intake, request management, operational follow-up.

1. INTRODUCCIÓN

Las instituciones de educación superior realizan actividades de vinculación que requieren atención al público, coordinación administrativa y apoyo de plataformas tecnológicas. En estos servicios, la recepción de solicitudes por distintos medios, la revisión manual de disponibilidad y el seguimiento mediante mensajes aislados pueden generar tiempos de respuesta variables, duplicidad de información y dependencia de personas específicas. La literatura sobre automatización de procesos ha señalado que las tareas repetitivas, basadas en reglas y con entradas estructuradas, son candidatas adecuadas para ser apoyadas mediante componentes de *software* especializados (Van der Aalst *et al.*, 2018; Syed *et al.*, 2020).

En la Unidad Académica de Sistemas Arrecifales (UASA) del Instituto de Ciencias del Mar y Limnología (ICML) de la Universidad Nacional Autónoma de México (UNAM), las visitas guiadas formaron parte de las actividades de divulgación científica y vinculación con escuelas, instituciones y público general. Desde 2001, la comisión responsable atendió a más de 7,000 personas mediante recorridos, pláticas y actividades relacionadas con los ecosistemas arrecifales y el trabajo académico desarrollado en la Unidad.

Antes de la mejora, el servicio dependía principalmente de la supervisión manual de cada paso: revisar solicitudes, verificar restricciones operativas, validar datos de contacto, consultar disponibilidad de

espacios, coordinar fechas, comunicar confirmaciones a las instituciones interesadas y registrar la información en los sistemas de apoyo. Este esquema permitía atender las solicitudes, pero aumentaba el riesgo de omisiones en periodos de alta demanda y dificultaba conocer el estado actualizado de cada trámite. Esta situación mostró la necesidad de contar con un mecanismo más ordenado para recibir, validar, notificar y dar seguimiento a las solicitudes.

La solución se basó en agentes de *software*, los cuales operan como componentes programados dentro de una automatización basada en reglas; cabe aclarar que estos agentes no emplean modelos de inteligencia artificial ni toman decisiones autónomas sobre la agenda. Su función consiste en recibir eventos, validar datos, consultar disponibilidad, enviar notificaciones y actualizar estados conforme a reglas previamente definidas.

Estudios sobre bots y agentes conversacionales han mostrado que estos componentes pueden apoyar tareas de coordinación, respuesta y seguimiento cuando se integran de forma controlada a procesos institucionales (Lebeuf *et al.*, 2018; Diederich *et al.*, 2022). Asimismo, en servicios de atención, se ha destacado la importancia de conservar la claridad en la interacción, la trazabilidad y el control humano sobre las decisiones relevantes (Androutsopoulou *et al.*, 2019).

En este contexto, el objetivo fue diseñar, implementar y validar una arquitectura basada en agentes de *software* para automatizar la recepción, validación, notificación y seguimiento de solicitudes de visitas guiadas en la UASA; se mantuvo únicamente la decisión final del coordinador sobre la agenda institucional.

2. DESARROLLO TÉCNICO

2.1 METODOLOGÍA

El trabajo se desarrolló a partir de la revisión del proceso de solicitudes de visitas guiadas, desde el primer contacto de la institución interesada hasta la confirmación final. Se identificaron actividades repetitivas, como recibir mensajes, revisar fechas disponibles, validar datos, consultar al coordinador, registrar la visita y responder al solicitante; asimismo, se identificaron las decisiones que debían mantenerse bajo control humano, especialmente la confirmación definitiva de la fecha. La secuencia general del flujo se resume en el Anexo A.

Con base en el análisis, se definieron las reglas operativas que debía respetar el sistema. Entre ellas, se incluyeron la anticipación mínima para solicitar una visita, los días permitidos, los meses con restricciones, el cupo máximo mensual y los periodos no laborables. Estas reglas ya formaban parte de la operación cotidiana del servicio, pero no estaban integradas en un solo mecanismo automatizado. Por ello, se documentaron y se usaron como base para programar la consulta de disponibilidad y la validación inicial de las solicitudes, como se muestra en la matriz del Anexo C.

Después, se diseñó una arquitectura modular en la que cada agente cumple una función específica: el agente de disponibilidad consulta la agenda; el agente de captura valida el formulario; el agente notificador avisa al coordinador; el agente de agendamiento asistido actualiza el estado de la visita; y los agentes complementarios atienden consultas por correo y generan reportes. La coordinación entre

componentes se realizó mediante una base de datos, de acuerdo con el esquema presentado en el apartado 5 del Anexo B.

La implementación se realizó con herramientas compatibles con la infraestructura disponible en la Unidad. El formulario y la consulta de disponibilidad se desarrollaron en PHP por su integración con el servidor web institucional. Los agentes de notificación, agendamiento y resumen se programaron en Node.js, porque requerían ejecutar tareas programadas, comunicarse con servicios externos y atender eventos de mensajería. Por otro lado, el agente de correo se desarrolló en Python debido a la disponibilidad de bibliotecas estables para consultar buzones y enviar respuestas mediante protocolos estándar. Los fragmentos principales de estos componentes se incluyeron en el Anexo B para documentar la implementación.

Durante el desarrollo, se realizaron pruebas funcionales por componente. Se verificó que el formulario rechazara registros incompletos, que la consulta excluyera fechas ocupadas o bloqueadas, que el agente notificador detectara solicitudes pendientes y que el flujo de confirmación actualizara el estado de la visita. En estas pruebas, se identificó que la validación de fechas era el punto más sensible, porque debía combinar reglas institucionales, disponibilidad real de agenda y confirmación humana.

La validación operativa se centró en comprobar que el sistema respetara las reglas del servicio y que mantuviera la intervención humana en la decisión final. No se buscó sustituir al coordinador, sino reducir las tareas repetitivas previas y posteriores a su decisión. Con ello, la metodología permitió pasar de un proceso distribuido entre correos, mensajes y revisiones manuales, a un flujo documentado mediante agentes especializados y estados persistentes en una base de datos.

2.2 IMPLEMENTACIÓN

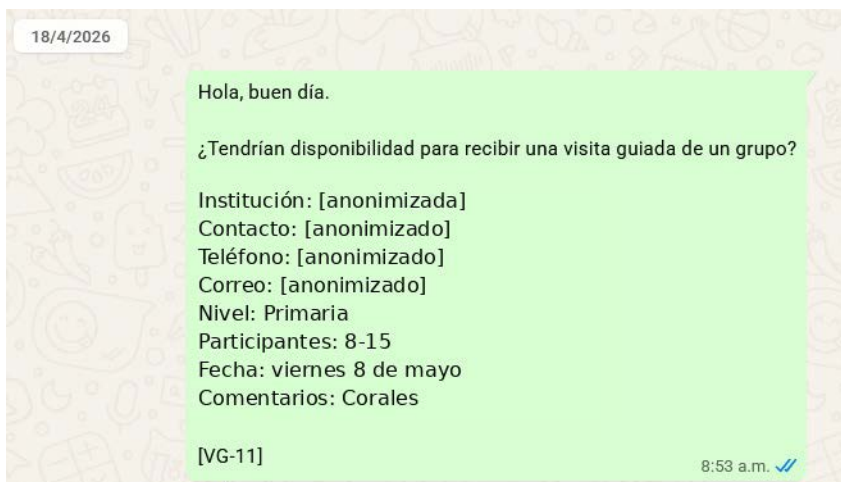
El formulario captura los datos mínimos de la solicitud: institución, responsable, contacto, participantes, nivel académico, fecha solicitada y comentarios. La validación incluyó método de envío, formato de correo y teléfono, campos cerrados, longitudes máximas, CAPTCHA y limitación de tasa. La persistencia de estos datos se vinculó con el esquema de la tabla principal descrito en el apartado 5 del Anexo B.

El agente de disponibilidad consulta la agenda antes del envío de la solicitud y genera una lista de viernes disponibles dentro de un horizonte de doce semanas. Para ello, se aplica reglas institucionales de anticipación mínima, restricción de meses, cupo mensual y bloqueo de días no laborables. La consulta se ejemplifica en el apartado 1 del Anexo B y corresponde a las reglas del Anexo C.

El manejo de seguridad se incorporó desde el formulario: encabezados HTTPS de protección, sanitización de campos de texto y consultas parametrizadas para reducir la exposición a inyección de código. El agente notificador revisa periódicamente las solicitudes pendientes y genera un aviso al coordinador por mensajería instantánea. En la Figura 1 se muestra el mensaje que fue redactado automáticamente con los datos de la solicitud.

Figura 1

Mensaje de solicitud automatizado

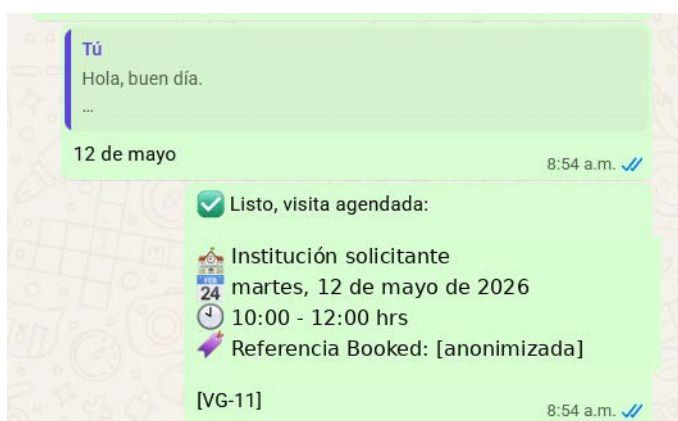


Nota. Mensaje al coordinador para su validación con los datos mínimos de la solicitud.

El diseño conservó la intervención humana en la selección definitiva de fecha, ya que esa decisión dependía de condiciones logísticas y de la disponibilidad de los recursos. El responsable debe responder con una afirmación o con una fecha específica. En la Figura 2 se observa la interacción de confirmación con el agente, en la que se seleccionó una fecha distinta a la propuesta por éste.

Figura 2

Confirmación de visita validada en la agenda



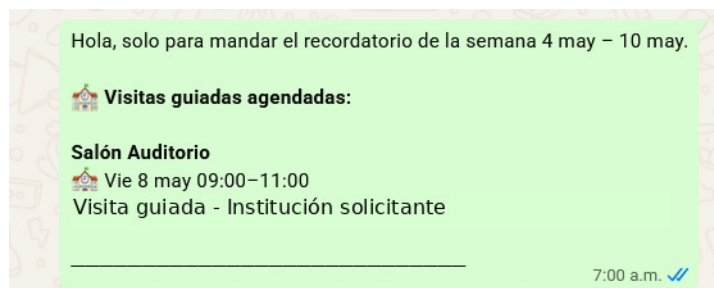
Nota. La figura muestra la respuesta humana del coordinador y la confirmación generada por el agente después de registrar la visita en la agenda institucional y la notificación a los interesados.

La lógica de notificación por WhatsApp se documentó en el apartado 2 del Anexo B; la recepción de respuestas y el agendamiento asistido, en el apartado 7 del Anexo B; el agente de correo, en el apartado 4 del Anexo B; el resumen semanal enviado por WhatsApp, en el apartado 3 del Anexo B; y el bloqueo de

días inhábiles, en el apartado 6 del Anexo B. De forma complementaria, la Figura 3 muestra un ejemplo del resumen semanal enviado al coordinador.

Figura 3

Resumen semanal de visitas



La solución se implementó como un conjunto de agentes especializados y no como un único flujo centralizado en una herramienta de automatización visual. Cada componente respondió a un evento concreto, aplicó reglas del servicio y actualizó el estado de la solicitud sin depender directamente de los demás módulos. Así, el sistema está bajo acoplamiento, facilita el mantenimiento individual de cada agente y conserva evidencia técnica mediante código fuente revisable. La Tabla 1 resume esta distribución funcional.

Tabla 1

Componentes funcionales de la solución de agentes

Componente	Tecnología	Disparador	Función principal
Captura web	PHP	Solicitud HTTPS	Valida y registra solicitudes de visita
Disponibilidad	PHP	Consulta del formulario	Publica fechas disponibles con reglas de cupo
Notificación	Node.js	Tarea programada	Avisa al coordinador sobre nuevas solicitudes
Agendamiento asistido	Node.js	Respuesta del coordinador	Actualiza estados, agenda y confirmación
Correo y resumen	Python / Node.js	Tareas programadas	Atiende consultas y genera reportes semanales

3. RESULTADOS

La solución centralizó la entrada de solicitudes mediante un formulario controlado y redujo la dispersión de información entre correos, mensajes y consultas manuales. El registro en base de datos permitió

conservar un estado único por solicitud, con lo cual se facilitó el seguimiento y disminuyó la probabilidad de duplicidad. Este resultado se relacionó con el flujo general del Anexo A y con el esquema de base de datos documentado en el apartado 5 del Anexo B.

La pizarra de seguimiento permitió visualizar en una sola interfaz el estado de cada solicitud, la fecha agendada, el avance de notificación y los participantes asignados a la visita. Como se muestra en la Figura 4, esta vista facilita la supervisión operativa del proceso sin consultar directamente la base de datos.

Figura 4

Seguimiento de solicitudes de visitas guiadas

La arquitectura basada en agentes permitió incorporar reglas de operación documentadas en el Anexo C, validaciones de seguridad, control de cupo, manejo de fechas no disponibles y persistencia de estados.

El tiempo de aviso al coordinador quedó acotado por el intervalo de ejecución programada del agente notificador. En el diseño evaluado, la notificación pudo emitirse dentro de una ventana máxima de cuatro minutos desde el registro de la solicitud. La evidencia técnica de esta operación se vinculó con la consulta de solicitudes pendientes y la actualización de estado, descritas en el apartado 2 del Anexo B, así como con el servidor interno de mensajería del apartado 7 del Anexo B.

Para dimensionar el cambio operativo, se comparó una muestra de hilos de correo previos con un caso posterior a la implementación, cuya evidencia anonimizada se presenta en el Anexo D. Como se resume en la Tabla 2, los tiempos de primera respuesta observados en el proceso manual variaron entre 2 h 4 min y 28 días 2 h 50 min, con un promedio de 8.9 días. En el flujo automatizado documentado, la confirmación de fecha se realizó en 42 minutos y el registro en agenda ocurrió en el mismo minuto de la validación humana.

Tabla 2

Comparación de indicadores operativos antes y después de la automatización

Indicador	Proceso manual	Proceso automatizado
Tiempo mínimo observado de respuesta	2 h 4 min	41 min
Tiempo máximo observado de respuesta	28 días 2 h 50 min	42 min en el caso documentado
Promedio observado	8.9 días	42 min en el caso documentado
Registro en agenda	Manual, posterior a la confirmación	En el mismo minuto de la validación humana
Trazabilidad	Hilos de correo dispersos	Folio de solicitud y referencia de agenda

El impacto operativo se observó en tres dimensiones: oportunidad de respuesta, al pasar de revisiones manuales a avisos programados; consistencia del servicio, porque todas las solicitudes siguieron las mismas reglas; y capacidad de supervisión, ya que el estado de cada solicitud quedó disponible para consultas posteriores. Esta reducción fue relevante porque la coordinación de visitas guiadas es una actividad complementaria del personal técnico académico.

4. CONCLUSIONES

El proceso de visitas guiadas presentó condiciones adecuadas para su automatización, debido a que combinaba reglas explícitas, entradas estructuradas, actividades repetitivas y necesidad de seguimiento. La solución implementada cumplió el objetivo de automatizar la recepción, validación, notificación y seguimiento de solicitudes, sin sustituir la decisión final del coordinador sobre la agenda institucional. De esta manera, la automatización redujo tareas operativas y fortaleció la atención ordenada del servicio.

Los resultados mostraron que el desacoplamiento entre componentes facilitó la integración del formulario web, la agenda institucional, el correo electrónico y la mensajería instantánea. Esta organización permitió conservar trazabilidad por solicitud, reducir dependencias entre módulos y mantener la intervención humana en la validación final de la fecha.

La principal aportación del trabajo fue demostrar que un servicio universitario de vinculación podía ser transformado en un flujo documentado y verificable, sin perder la flexibilidad necesaria para la toma de decisiones académicas y logísticas. La implementación permitió aplicar reglas de disponibilidad de forma uniforme, disminuir el riesgo de omisiones y establecer una base técnica replicable para otros servicios administrativos de apoyo académico.

AGRADECIMIENTOS

Se agradece a la comisión de visitas guiadas de la Unidad Académica de Sistemas Arrecifales del Instituto de Ciencias del Mar y Limnología por la colaboración brindada para documentar los procesos y validar los requerimientos funcionales de la solución.

REFERENCIAS

- Androutsopoulou, A., Karacapilidis, N., Loukis, E., y Charalabidis, Y. (2019). Transforming the communication between citizens and government through AI-guided chatbots. *Government Information Quarterly*, 36(2), 358–367. <https://doi.org/10.1016/j.giq.2018.10.001>
- Diederich, S., Brendel, A. B., Morana, S., y Kolbe, L. M. (2022). On the design of and interaction with conversational agents: An organizing and assessing review of human-computer interaction research. *Journal of the Association for Information Systems*, 23(1), 96–138. <https://doi.org/10.17705/1jais.00724>
- Lebeuf, C., Storey, M. A., y Zagalsky, A. (2018). Software bots. *IEEE Software*, 35(1), 18–23. <https://doi.org/10.1109/MS.2017.4541027>
- Syed, R., Suriadi, S., Adams, M., Bandara, W., Leemans, S. J. J., Ouyang, C., ter Hofstede, A. H. M., van de Weerd, I., Wynn, M. T., y Reijers, H. A. (2020). Robotic process automation: Contemporary themes and challenges. *Computers in Industry*, 115, 103162. <https://doi.org/10.1016/j.compind.2019.103162>
- Van der Aalst, W. M. P., Bichler, M., y Heinzl, A. (2018). Robotic process automation. *Business & Information Systems Engineering*, 60(4), 269–272. <https://doi.org/10.1007/s12599-018-0542-4>

ANEXO A. ARQUITECTURA FUNCIONAL RESUMIDA

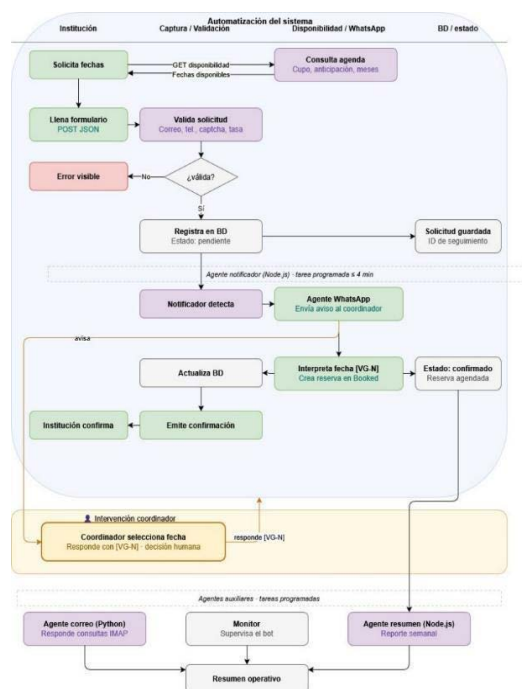
El proceso inicia cuando una institución consulta las fechas disponibles. El agente de disponibilidad revisa la agenda institucional, aplica las reglas del servicio —día permitido, anticipación mínima, meses restringidos y cupo mensual— y devuelve únicamente las fechas válidas. Con la fecha seleccionada, la institución envía el formulario en línea. El agente de captura valida los datos, registra la solicitud en la base de datos con estado “pendiente” y asigna un identificador de seguimiento. Si alguna validación falla, el error se notifica de inmediato.

En un ciclo programado de hasta cuatro minutos, el agente notificador detecta las solicitudes nuevas y envía un aviso automático al coordinador a través del agente WhatsApp, incluyendo los datos de la visita y un *token* de referencia [VG-N]. El coordinador revisa la información y responde con la fecha confirmada. El agente WhatsApp interpreta la respuesta, crea la reserva en el sistema de agenda, actualiza el estado a “confirmado” en la base de datos y envía la confirmación por correo electrónico a la institución solicitante.

De forma complementaria, el agente de correo atiende consultas entrantes respondiendo automáticamente con el enlace al formulario, mientras que el agente de resumen genera un reporte semanal de visitas programadas para el coordinador. Ambos agentes operan en modo lectura sin alterar el flujo principal.

Figura 5

Diagrama conceptual del proceso automatizado



Nota. El diagrama distingue las actividades ejecutadas por agentes de *software*, las interacciones con la base de datos y la intervención del coordinador en la validación final de la fecha.

ANEXO B. FRAGMENTOS REPRESENTATIVOS DE IMPLEMENTACIÓN

Los fragmentos incluidos en este anexo son extractos representativos de la implementación y no corresponden al código fuente completo del sistema. Su propósito es mostrar la lógica técnica principal de cada componente, como validación, consulta de disponibilidad, notificación, actualización de estados y agendamiento asistido. Para proteger la seguridad de la infraestructura institucional, los fragmentos fueron depurados y ajustados con el fin de omitir credenciales, rutas internas, direcciones, identificadores sensibles, parámetros de conexión y cualquier configuración que pudiera comprometer los servicios tecnológicos.

1. Agente de disponibilidad — *disponibilidad.php* (PHP)

Consulta la API REST de Booked Scheduler y filtra los viernes disponibles. El fragmento muestra la lógica central: autenticación, consulta de reservas existentes y cálculo de fechas libres.

Consulta a Booked y cálculo de los viernes disponibles:

```
$auth = booked_post(BOOKED_BASE . '/Authentication/Authenticate',  
    ['username' => BOOKED_USER, 'password' => BOOKED_PASS]);  
  
$token = $auth['sessionToken'];  
$userId = $auth['userId'];  
$authHeaders = [  
    'X-Booked-SessionToken: ' . $token,  
    'X-Booked-UserId: ' . $userId,  
];  
  
// Rango: mínimo 2 semanas desde hoy, 12 semanas adelante  
$inicio = (new DateTime())->modify('+14 days');  
$fin = (clone $inicio)->modify('+12 weeks');  
  
$url = BOOKED_BASE . '/Reservations/?'  
    . '&startDate=' . urlencode($inicio->format('Y-m-d\\  
T00:00:00'))  
    . '&endDate=' . urlencode($fin->format('Y-m-d\\T23:59:59'))  
    . '&resourceId=' . RESOURCE_ID;  
  
$reservas = booked_get($url, $authHeaders)['reservations'] ?? [];  
  
while ($cursor <= $limite && count($disponibles) < 8) {  
    $yyyymmdd = $cursor->format('Y-m-d');  
    $mes_key = $cursor->format('Y-m');  
  
    if (($visitas_por_mes[$mes_key] ?? 0) >= 2) {  
        $cursor->modify('+7 days'); continue;  
    }  
    // Verificar traslape con 10:00-12:00 (slot UTC 15:00-17:00)
```

```
$ocupado = /* comparación de rangos de tiempo */ false;
if (!$ocupado && !esta_bloqueado($yyyymmdd, $DIAS_BLOQUEADOS,
$RANGOS))
    $disponibles[] = ['value'=>$yyyymmdd, 'label'=>"viernes
..."];
    $cursor->modify('+7 days');
}
echo json_encode(['fechas' => $disponibles]);
```

2. Agente notificador — *visitas_notifier.js* (Node.js)

Lee solicitudes no notificadas en la BD, limpia los campos antes de construir el mensaje y lo envía por mensajería instantánea. Una vez confirmado el envío, marca el registro como notificado.

Sanitización de campos y composición del mensaje:

```
function sanitizarCampo(val) {
    return String(val ?? '')
        .replace(/[\x00-\x08\x0B\x0C\x0E-\x1F\x7F]/g, '')
        .replace(/ignore\\s+(previous|all)\\s+instructions?/gi,
        '[eliminado]')
        .replace(/system\\s*/gi, '[eliminado]')
        .trim();
}

function mensajeDefault(v, fechasDisponibles) {
    const fechasStr = fechasDisponibles?.map(f => {
        const dia = DIAS_ES[f.getDay()];
        return `${dia} ${f.getDate()} de ${MESES_ES[f.getMonth()+1]}`;
    }).join(', ') ?? null;

    return `Hola, buen día.

¿Tendrían disponibilidad para recibir una visita guiada?

Institución: ${v.escuela}
Contacto: ${v.responsable} (${v.cargo})
Teléfono: ${v.telefono}
Nivel: ${v.nivel_academico}
Fecha: ${fechasStr ?? v.mes_tentativo}

Gracias.

[VG-${v.id}]`;
```

}

Ciclo principal: consulta, envío y actualización de estado:

```
async function main() {
  const { rows } = await pool.query(
    "SELECT * FROM visitas_guiadas"
    + " WHERE notificado = FALSE ORDER BY creado_en ASC"
  );

  for (const visita of rows) {
    const mensaje = await componerMensaje(visita, fechasDisponi-
bles);

    const res = await enviarWhatsApp(mensaje);
    if (res.ok) {
      await pool.query(
        "UPDATE visitas_guiadas SET notificado = TRUE WHERE
id = $1",
        [visita.id]
      );
    }
    await new Promise(r => setTimeout(r, 3000)); // pausa entre
envíos
  }
  await pool.end();
}
```

3. Agente de resumen — *agenda_semanal.js* (Node.js)

Consulta los tres recursos del auditorio en Booked Scheduler, filtra las reservas cuyo título contiene la palabra "visita" y envía el resumen semanal. Si no hay visitas registradas, no genera mensaje.

```
// Consultar los 3 recursos en paralelo
const consultas = Object.keys(RECURSOS).map(rid =>
  httpsGet(`${BOOKED_BASE}/Reservations/?` +
    `startDateTime=${inicio}&endDateTime=${fin}&resourceI-
d=${rid}`,
    headers
  ).then(data => ({ rid, reservas: data.reservations || [] })))
);
const resultados = await Promise.all(consultas);

const visitasPorRecurso = {};
for (const { rid, reservas } of resultados) {
  visitasPorRecurso[rid] = reservas.filter(
```

```

        r => /visita/i.test(r.title || '')
    );
}

if (totalVisitas === 0) { log(`Sin visitas esta semana.`); return; }

let msg = `Hola, recordatorio semana ${semanaStr}.\n\n`;
for (const [rid, visitas] of Object.entries(visitasPorRecurso)) {
    if (!visitas.length) continue;
    msg += `*${RECURSOS[rid]}*\n`;
    for (const r of visitas) {
        msg += ` ${fmtFecha(r.startDate)} ${fmtHora(r.startDate)}-${fmtHora(r.endDate)}\n`;
        msg += ` ${r.title||''}.slice(0,60)}\n`;
    }
}
await enviarWhatsApp(msg);

```

4. Agente de correo — *visitas_mail.py* (Python)

Revisa la bandeja institucional por IMAP, detecta correos que pidan informes sobre visitas guiadas y responde indicando el formulario. Los correos automáticos (*noreply*, *mailer-daemon*) son omitidos.

Detección de solicitudes de informes:

```

PALABRAS_INFORMES = [
    r'inform[eo]s?',      r'informaci[oó]n',
    r'solicitar?\\s+visita', r'visita\\s+guiada',
    r'disponibilidad',    r'c[ó]mo\\s+(?:solicitar|agendar)',
    r'requisitos?',       r'nos?\\s+gustar[íiI]a',
]

def pide_informes(asunto: str, cuerpo: str) -> bool:
    texto = (asunto + ' ' + cuerpo).lower()
    return any(re.search(p, texto) for p in PALABRAS_INFORMES)

```

Ciclo principal de revisión:

```

for uid in uids:                                # solo correos no leídos
    msg = email.message_from_bytes(...)
    asunto = sanitizar_header(decode_str(msg.get('Subject', '')))
    _, addr = parseaddr(msg.get('From', ''))

    # Omitir correos automáticos
    if re.search(r'noreply|no-reply|mailer-daemon', addr, re.I):

```

```
mail.store(uid, '+FLAGS', '\\\\Seen'); continue

if pide_informes(asunto, extraer_cuerpo(msg)):
    enviar_respuesta(addr, nombre, asunto, message_id)
    logging.info(f'Respuesta enviada a {addr}')

mail.store(uid, '+FLAGS', '\\\\Seen')    # marcar leído
```

5. Esquema de base de datos — *schema.sql* (PostgreSQL)

La tabla principal del módulo de visitas guiadas. El campo “notificado” es el mecanismo de coordinación entre el agente de captura y el agente notificador; los demás campos registran la trazabilidad completa de cada solicitud.

```
CREATE TABLE IF NOT EXISTS visitas_guiadas (
    id                SERIAL PRIMARY KEY,
    correo            VARCHAR(200) NOT NULL,
    escuela           VARCHAR(300) NOT NULL,
    responsable       VARCHAR(200) NOT NULL,
    telefono          VARCHAR(30)  NOT NULL,
    cargo             VARCHAR(200),
    participantes     VARCHAR(20),
    nivel_academico   VARCHAR(200),
    mes_tentativo     VARCHAR(30),
    fecha_solicitada  DATE,
    fecha_agendada   DATE,
    comentarios       TEXT,
    notificado        BOOLEAN DEFAULT FALSE,
    estado            VARCHAR(30) DEFAULT 'pendiente',
    creado_en         TIMESTAMP DEFAULT NOW()
);
```

6. Script auxiliar — *bloquear_dias.js* (Node.js)

Bloquea días inhábiles y periodos vacacionales en Booked, creando reservas administrativas en el Salón Auditorio. Se ejecuta manualmente antes de cada ciclo escolar para sincronizar la agenda pública con el calendario institucional.

```
// Días individuales y rangos a bloquear
const DIAS = [
    { y:2026, m:1, d:1, titulo:'Día inhábil - Año nuevo' },
    { y:2026, m:5, d:1, titulo:'Día inhábil - Día del trabajo' },
    // ...
];

const RANGOS = [
```

```

    { ini:[2026,3,30], fin:[2026,4,3], titulo:'Asueto académico' },
    { ini:[2026,7,6], fin:[2026,7,24], titulo:'Vacaciones adminis-
trativas' },
  ];

  // Para cada fecha, crear reserva con título 'BLOQUEADO - <motivo>'
  const res = await httpsPost(`${BOOKED_BASE}/Reservations/`, {
    resourceId: RESOURCE_ID,
    startDateTime: `${fecha}T09:30:00`,
    endDateTime: `${fecha}T15:30:00`,
    title: `BLOQUEADO - ${t.titulo}`,
  }, headers);

  if (/conflicto|conflict/i.test(res.errors?.join(';') || '')) {
    log(`Ya tiene reserva, omitiendo: ${fecha}`);
    continue;
  }

```

7. Agente de WhatsApp — *index.js* (Node.js)

Es el componente central del sistema de notificación. Implementa tres responsabilidades en un mismo proceso: (a) un servidor HTTP interno que recibe mensajes de los demás agentes y los envía por WhatsApp; (b) el *listener* de mensajes entrantes que detecta respuestas del coordinador mediante el *token* [VG-N]; y (c) el agente de agendamiento asistido que, al recibir una fecha válida, crea la reserva en Booked, actualiza la base de datos y envía la confirmación por correo al solicitante.

Servidor HTTP interno:

```

// Los demás agentes (notifier, agenda_semanal) llaman a este en-
dpoint
app.post('/enviar', async (req, res) => {
  const { numero, mensaje } = req.body;
  if (!sock) return res.json({ error: 'Bot no conectado' });

  const jid = numero.includes('@')
    ? numero
    : `${numero}@s.whatsapp.net`;
  await sock.sendMessage(jid, { text: mensaje });
  log(`ENVIADO → ${jid}: ${mensaje}`);
  res.json({ ok: true });
});

app.get('/mensajes', (req, res) => {
  const lineas = fs.existsSync(LOG_FILE)
    ? fs.readFileSync(LOG_FILE, 'utf8').trim().split('\n').sli-

```

```
ce(-50)
    : [];
    res.json(lineas);
  });
```

Detección de respuestas del coordinador:

```
sock.ev.on('messages.upsert', async ({ messages, type }) => {
  if (type !== 'notify') return;
  for (const msg of messages) {
    const texto = msg.message?.conversation
      || msg.message?.extendedTextMessage?.text || '';

    const ctxInfo = msg.message?.extendedTextMessage?.contextInfo || null;
    const citado = ctxInfo?.quotedMessage?.conversation
      || ctxInfo?.quotedMessage?.extendedTextMessage?.text || '';

    const textoCompleto = texto + (citado ? ' ' + citado : '');

    if (textoCompleto.includes('[VG-'])) {
      await procesarRespuesta(msg.key.remoteJid, textoCompleto);
    }
  }
});
```

Extracción de fecha en español:

```
function parsearFecha(texto) {
  const m = texto.match(
    /(\d{1,2})\s+de\s+(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre)/i
  );
  if (!m) return null;
  const fecha = new Date(
    new Date().getFullYear(),
    MESES_PARSE[m[2].toLowerCase()] - 1,
    parseInt(m[1])
  );
  if (fecha < new Date()) fecha.setFullYear(fecha.getFullYear() + 1);
  return fecha;
}
```

Agendamiento asistido: reserva en Booked + confirmación por correo:

```

async function procesarRespuesta(jid, texto) {
  const tokenMatch = texto.match(/\[VG-(\d+)\]/i);
  if (!tokenMatch) return;

  const visitaId = parseInt(tokenMatch[1]);
  const fecha = parsearFecha(texto);

  if (!fecha) {
    await sock.sendMessage(jid, {
      text: `No pude identificar la fecha [VG-${visitaId}].`
        + ` Indica: "viernes 12 de junio [VG-${visitaId}]"`
    });
    return;
  }

  const { rows } = await pool.query(
    `SELECT * FROM visitas_guiadas WHERE id = $1`, [visitaId]
  );
  const visita = rows[0];

  const bookedRes = await crearReservaBooked(visita, fecha);
  const ref = bookedRes.referenceNumber || bookedRes.reservatio-
nId;

  await pool.query(
    `UPDATE visitas_guiadas SET fecha_agendada=$1, referencia_
booked=$2 WHERE id=$3`,
    [fecha, ref, visitaId]
  );

  // Confirmar al coordinador por WhatsApp
  const diaStr = fecha.toLocaleDateString('es-MX',
    { weekday: 'long', day: 'numeric', month: 'long', year: 'nume-
ric' });
  await sock.sendMessage(jid, {
    text: `Visita agendada:\n ${visita.escuela}\n ${diaStr}`
      + `\n Referencia: ${ref} [VG-${visitaId}]`
  });

  // Notificar al solicitante por correo
  await enviarEmailContacto(visita, diaStr);
}

```

Conexión a WhatsApp y reconexión automática:

```

async function conectar() {

```

```
const { state, saveCreds } = await useMultiFileAuthState(AUTH_
DIR);
const { version } = await fetchLatestBaileysVersion();

sock = makeWASocket({
  version, auth: state,
  logger: pino({ level: 'silent' }),
  printQRInTerminal: false,
});

sock.ev.on('connection.update', ({ connection, lastDisconnect,
qr }) => {
  if (qr) qrcode.generate(qr, { small: true }); // escanear
al iniciar

  if (connection === 'close') {
    const code = lastDisconnect?.error?.output?.statusCode;
    const noReconectar = [
      DisconnectReason.loggedOut,
      DisconnectReason.connectionReplaced, // otra ins-
tancia activa
    ];
    if (!noReconectar.includes(code))
      setTimeout(conectar, 3000); // reconexion automáti-
ca
  }
});
sock.ev.on('creds.update', saveCreds);
}
conectar();
```

ANEXO C. MATRIZ DE REGLAS OPERATIVAS DEL SISTEMA

Las reglas operativas son las condiciones, restricciones y criterios explícitos que gobiernan el comportamiento del sistema y delimitan qué solicitudes son válidas, cuándo se pueden atender y cómo deben procesarse. Para efectos de la implementación, se clasificaron en reglas del servicio y reglas técnicas.

Reglas del servicio

Estas reglas definieron las condiciones institucionales bajo las cuales podía atenderse una visita guiada:

1. Las visitas se realizan únicamente los viernes.
2. Se requiere una anticipación mínima de dos semanas entre la solicitud y la fecha de visita.
3. No se realizan visitas durante julio ni en el mes de diciembre.

4. El cupo máximo es de dos visitas por mes.
5. El horario ordinario de visita es de 10:00 a 12:00 horas.
6. Los días inhábiles y periodos vacacionales institucionales se bloquean en la agenda antes de cada ciclo escolar.

Reglas técnicas

Estas reglas definieron la forma en que el sistema valida, procesa y protege las solicitudes:

1. Sólo se aceptan solicitudes por método POST con contenido JSON.
2. El tamaño máximo de la petición se limita para prevenir el abuso del formulario.
3. El correo electrónico debe tener formato válido.
4. El teléfono debe cumplir con un patrón de dígitos definido.
5. Los campos cerrados, como nivel académico y cargo, se validan contra listas blancas.
6. Cada campo tiene una longitud máxima permitida.
7. Se requiere verificación CAPTCHA para confirmar que la solicitud proviene de una persona.
8. Se aplica limitación de tasa con un máximo de cinco intentos por hora por dirección de origen.
9. Las operaciones en base de datos se realizan mediante consultas parametrizadas.
10. El agente notificador ejecuta su ciclo en intervalos de hasta cuatro minutos.
11. La respuesta del coordinador debe incluir el *token* [VG-N] y una fecha en lenguaje natural para activar el agendamiento.
12. Los correos automáticos, como *noreply* o *mailer-daemon*, son ignorados por el agente de correo.
13. La caché de disponibilidad tiene una vigencia de treinta minutos para reducir la carga sobre Booked Scheduler.

ANEXO D. EVIDENCIA OPERATIVA DE TIEMPOS DE RESPUESTA

Este anexo presenta evidencia operativa utilizada para estimar los tiempos de respuesta antes y después de la implementación del sistema. Para proteger la privacidad de las instituciones solicitantes, se anonimizaron nombres de escuelas, personas responsables, correos electrónicos, teléfonos y referencias internas. Las fechas y horas se conservaron únicamente con fines de análisis operativo.

1. Evidencia del proceso manual previo

Antes de la implementación, las solicitudes de visitas guiadas se recibían principalmente por correo electrónico. La atención dependía de la revisión manual de mensajes, la consulta de disponibilidad en agenda, la validación de fechas posibles y la comunicación posterior con la institución solicitante. Esta forma de operación generaba tiempos de respuesta variables, especialmente en periodos con alta

demanda o cuando era necesario ajustar fechas por cupo, días inhábiles o disponibilidad del personal responsable.

En la muestra revisada, se identificaron respuestas que ocurrieron desde el mismo día hasta varias semanas después de la solicitud inicial. También se observaron hilos con varios intercambios de correo antes de llegar a una confirmación o ajuste de fecha. Esta evidencia mostró que el proceso manual dependía de la revisión periódica de mensajes y de la consulta caso por caso de la disponibilidad.

Figura 6

Ejemplo anonimizado de solicitud atendida mediante correo electrónico con tiempo de respuesta extendido

Solicitud de visita escolar (Institución A)		
	Solicitante / texto del correo anonimizado — evento 1	vie, 15 nov 2024, 9:01 a.m.
	Solicitante / texto del correo anonimizado — evento 2	mar, 19 nov 2024, 3:11 p.m.
	Solicitante / texto del correo anonimizado — evento 3	jue, 12 dic 2024, 3:59 p.m.
	Solicitante / texto del correo anonimizado — evento 4	vie, 19 dic 2024, 10:51 a.m.
	Solicitante / texto del correo anonimizado — evento 5	lun, 16 dic 2024, 9:22 a.m.
	Solicitante / texto del correo anonimizado — evento 6	jue, 23 ene 2025, 10:58 a.m.
	Solicitante / texto del correo anonimizado — evento 7	lun, 27 ene 2025, 9:50 a.m.

Figura 7

Ejemplo anonimizado de coordinación manual para revisar disponibilidad y ajustar fechas

Hilo de coordinación manual — texto y remitente anonimizados (1)	Inicio del flujo	mie, 4 sept 2024, 12:04 p.m.
Hilo de coordinación manual — texto y remitente anonimizados (2)		vie, 6 sept 2024, 11:04 a.m.
Hilo de coordinación manual — texto y remitente anonimizados (3)		vie, 6 sept 2024, 2:09 p.m.
Hilo de coordinación manual — texto y remitente anonimizados (4)		jue, 12 sept 2024, 12:56 p.m.
Hilo de coordinación manual — texto y remitente anonimizados (5)		vie, 19 sept 2024, 9:43 a.m.
Hilo de coordinación manual — texto y remitente anonimizados (6)		jue, 19 sept 2024, 12:47 p.m.
Fin		jue, 26 sept 2024, 9:23 a.m.

2. Evidencia del proceso posterior a la implementación

Después de la implementación, el sistema permitió recibir solicitudes mediante un formulario controlado, registrar los datos en una base de datos, generar un folio de seguimiento y notificar automáticamente al coordinador. La respuesta humana continuó siendo necesaria para confirmar la fecha definitiva, pero el flujo redujo las tareas manuales previas y posteriores a esa decisión.

En el caso documentado, el agente notificó una nueva solicitud al coordinador a las 5:04 p.m. La primera respuesta humana se recibió a las 5:45 p.m. y la confirmación de fecha ocurrió a las 5:46 p.m. En ese mismo minuto, el sistema registró la visita en la agenda y generó la referencia correspondiente. Esto

permitió observar un tiempo de 42 minutos entre la notificación automática y la confirmación de fecha, así como un registro en agenda realizado en el mismo minuto de la validación humana.

Figura 8

Ejemplo anonimizado de notificación automática, respuesta humana y confirmación del flujo



Figura 9

Ejemplo anonimizado de confirmación humana y registro automatizado en agenda



3. Síntesis de la comparación operativa

La evidencia revisada permitió comparar dos condiciones de operación. En el proceso manual, la primera respuesta dependía de la revisión de correos y podía variar entre algunas horas y varias semanas. En el proceso automatizado, la solicitud quedó registrada desde el inicio, se notificó al coordinador en una ventana controlada y se conservó trazabilidad mediante folio y referencia de agenda.

La comparación no tuvo como propósito sustituir la decisión del coordinador, sino reducir actividades repetitivas alrededor de esa decisión: búsqueda de mensajes, revisión inicial de datos, aviso de solicitudes pendientes, seguimiento de estado y registro posterior. Esta mejora fue relevante porque la coordinación de visitas guiadas no correspondía a un recurso dedicado exclusivamente a dicha actividad, sino a una función complementaria realizada por personal técnico académico junto con sus actividades principales. En este sentido, la automatización favoreció la continuidad del servicio sin que el personal asignado descuidara las funciones técnicas y académicas para las cuales fue contratado o designado.