SERVICIOS DE NOTIFICACIÓN Y DISTRIBUCIÓN DE REPORTES

Carlos Robles C39310
Luis Rivera C36589
Jonathan Moreno C35380
Dilan Flores C32965

```
{
    0 references
    public class EmailTaskDto
    {
        0 references
        public string CorrelationId { get; set; } = string.Empty;
        0 references
        public string ToEmail { get; set; } = string.Empty;
        0 references
        public string Subject { get; set; } = string.Empty;
        0 references
        public string Message { get; set; } = string.Empty;
        0 references
        public int CustomerId { get; set; }

    }
```

```
public class MessagingTaskDto
{
    0 references
    public string CorrelationId { get; set; } = string.Empty;
    0 references
    public string PhoneNumber { get; set; } = string.Empty;
    0 references
    public string Message { get; set; } = string.Empty;
    0 references
    public int CustomerId { get; set; }
}
```

DISEÑO DEL SISTEMA

Lo primero que se realizó fue el diseño de los dos objetos que recibiría el sistema: Email Server y Message Server. Esto se hizo desde el servidor Hangfire, con el objetivo de definir hacia dónde se enviarían los datos correspondientes a cada servidor .

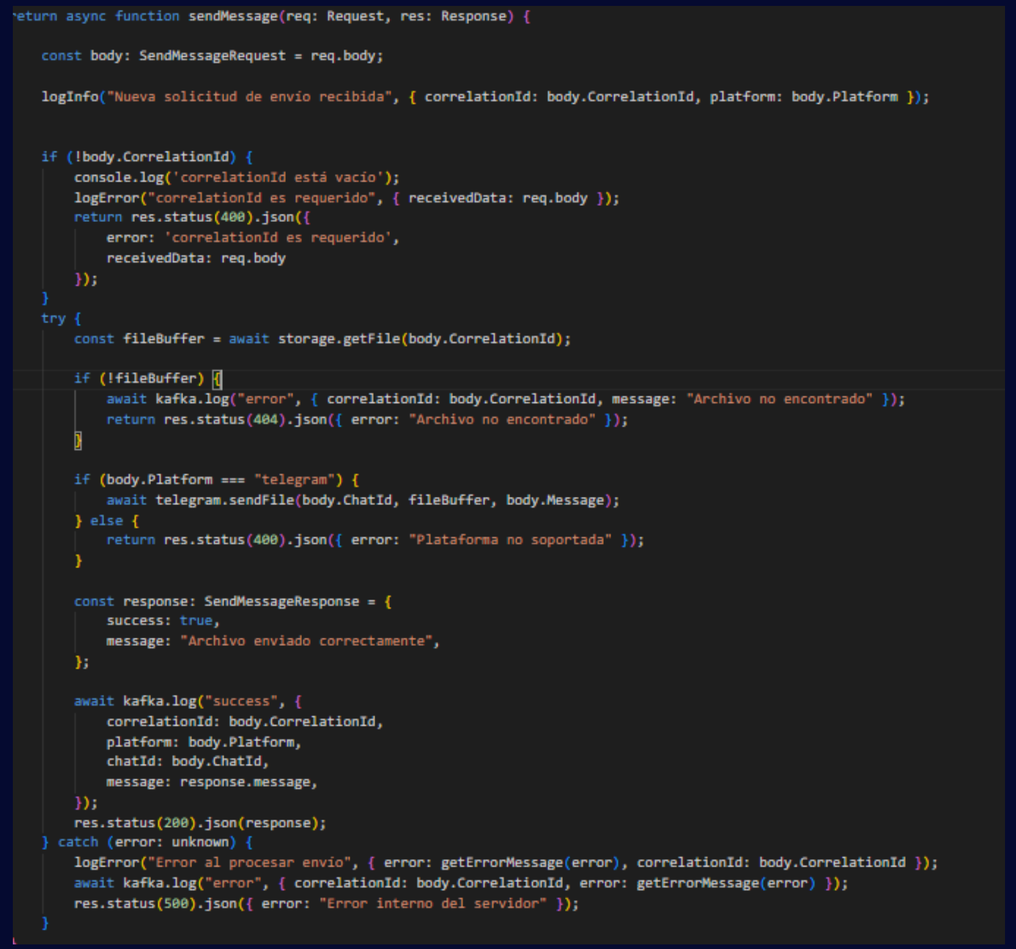
IMPLEMENTACIÓN DEL AGENDAMIENTO DE CORREOS Y MENSAJES

El sistema se encarga de programar de forma automática el envío de correos y mensajes usando el servidor Hangfire. Cuando llega una solicitud, se guardan los datos necesarios y se agenda la tarea para que se ejecute unos minutos después. De esta forma, el servidor puede seguir funcionando sin interrupciones. Además, se registran los detalles de cada tarea para tener control y saber si se completó correctamente o si ocurrió algún error.

```
[HttpPost("schedule-email")]
0 references
public async Task<IActionResult> ScheduleEmailTask([FromBody] EmailTaskDto emailTask)
{
    try
    {
        _logger.LogInformation("Recibida solicitud de programación de email. CorrelationId: {CorrelationId}", emailTask.CorrelationId);

        var jobId = _hangfire.Schedule<IEmailJobService>(
            job => job.SendEmailAsync(
                emailTask.CorrelationId,
                emailTask.ToEmail,
                emailTask.Subject,
                emailTask.Message,
                emailTask.CustomerId
            ),
            TimeSpan.FromMinutes(1)
        );

        _logger.LogInformation("Tarea de email programada exitosamente. JobId: {JobId}, CorrelationId: {CorrelationId}", jobId, emailTask.CorrelationId);

        return Ok(new
        {
            JobId = jobId,
            CorrelationId = emailTask.CorrelationId,
            Status = "Scheduled",
            ScheduledTime = DateTime.UtcNow.AddMinutes(1),
            Message = "Email task scheduled successfully",
            EmailTo = emailTask.ToEmail
        });
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error al programar tarea de email. CorrelationId: {CorrelationId}", emailTask.CorrelationId);
        return StatusCode(500, new { Error = "Error al programar tarea de email", Details = ex.Message });
    }
}

[HttpPost("schedule-messaging")]
0 references
public async Task<IActionResult> ScheduleMessagingTask([FromBody] MessagingTaskDto messagingTask)
{
    try
    {

        _logger.LogInformation("Datos recibidos - CorrelationId: '{CorrelationId}', PhoneNumber: '{PhoneNumber}', Message: '{Message}', CustomerId: {CustomerId}",
            messagingTask.CorrelationId ?? "NULL",
            messagingTask.PhoneNumber ?? "NULL",
            messagingTask.Message ?? "NULL",
            messagingTask.CustomerId);

        _logger.LogInformation("Recibida solicitud de programación de messaging. CorrelationId: {CorrelationId}", messagingTask.CorrelationId);


        var jobId = _hangfire.Schedule<IMessagingJobService>(
            job => job.SendMessageAsync(
                messagingTask.CorrelationId,
                messagingTask.PhoneNumber,
                "telegram",
                messagingTask.Message
            ),
            TimeSpan.FromMinutes(1)
        );

        _logger.LogInformation("Tarea de messaging programada exitosamente. JobId: {JobId}, CorrelationId: {CorrelationId}", jobId, messagingTask.CorrelationId);

        return Ok(new
        {
            JobId = jobId,
            CorrelationId = messagingTask.CorrelationId,
            Status = "Scheduled",
            ScheduledTime = DateTime.UtcNow.AddMinutes(1),
            Message = "Messaging task scheduled successfully",
            PhoneNumber = messagingTask.PhoneNumber
        });
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error al programar tarea de messaging. CorrelationId: {CorrelationId}", messagingTask.CorrelationId);
        return StatusCode(500, new { Error = "Error al programar tarea de messaging", Details = ex.Message });
    }
}
```

• ENVIO DE MESAJES TELEGRAM

Esta función en el servidor message server se encarga de manejar el envío de mensajes y archivos por medio de la plataforma Telegram. Cuando llega una solicitud, el sistema valida los datos, busca el archivo correspondiente y lo envía al destinatario. Además, registra cada acción o error en Kafka para mantener un control del proceso y asegurar que todo funcione correctamente.



```
return async function sendMessage(req: Request, res: Response) {

    const body: SendMessageRequest = req.body;

    logInfo("Nueva solicitud de envío recibida", { correlationId: body.CorrelationId, platform: body.Platform });


    if (!body.CorrelationId) {
        console.log('correlationId está vacío');
        logError("correlationId es requerido", { receivedData: req.body });
        return res.status(400).json({
            error: 'correlationId es requerido',
            receivedData: req.body
        });
    }
    try {
        const fileBuffer = await storage.getFile(body.CorrelationId);

        if (!fileBuffer) {
            await kafka.log("error", { correlationId: body.CorrelationId, message: "Archivo no encontrado" });
            return res.status(404).json({ error: "Archivo no encontrado" });
        }

        if (body.Platform === "telegram") {
            await telegram.sendFile(body.ChatId, fileBuffer, body.Message);
        } else {
            return res.status(400).json({ error: "Plataforma no soportada" });
        }

        const response: SendMessageResponse = {
            success: true,
            message: "Archivo enviado correctamente",
        };

        await kafka.log("success", {
            correlationId: body.CorrelationId,
            platform: body.Platform,
            chatId: body.ChatId,
            message: response.message,
        });
        res.status(200).json(response);
    } catch (error: unknown) {
        logError("Error al procesar envío", { error: getErrorMessage(error), correlationId: body.CorrelationId });
        await kafka.log("error", { correlationId: body.CorrelationId, error: getErrorMessage(error) });
        res.status(500).json({ error: "Error interno del servidor" });
    }
```

ENVIO DE GMAIL

El servidor Email server recibe los datos de la tarea programada del servidor hangfire y ejecuta la funcion send_email .

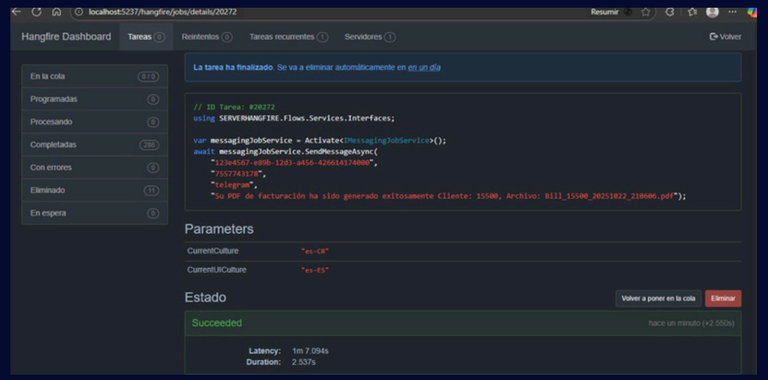
```python
@router.post("/send", status_code=status.HTTP_200_OK)
async def send_email(
    request: EmailSendRequest,
    email_service: EmailService = Depends(get_email_service),
    storage_service: StorageService = Depends(get_storage_service)
):

    pdf_file = await storage_service.get_file(request.correlation_id)

    response = await email_service.send_email(request, pdf_file)

    return response
```

Esta funcion se encarga de enviar correos electrónicos con archivos adjuntos. Primero construye el contenido y el mensaje, luego lo envía mediante SMTP y devuelve un resultado indicando éxito o fracaso.

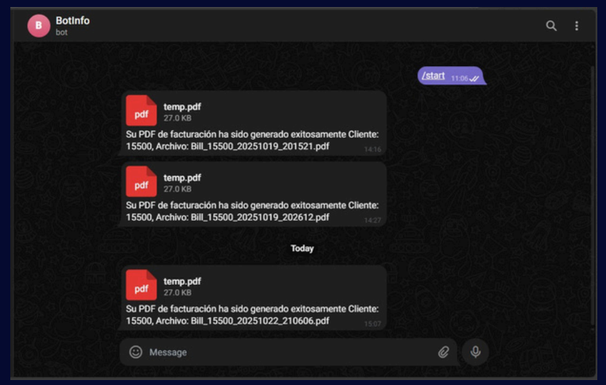
```python
class EmailService(IEmailService):

    async def send_email(self, email_request: EmailSendRequest, pdf_file: bytes) -> dict:

        try:
            body_html = build_email_body(email_request)

            msg = build_email_message(email_request, body_html, pdf_file)

            await send_email_smtp(msg, email_request.recipient_email)

            return {
                "status": "success",
                "message": Messages.EMAIL_SEND_SUCCESS,
                "correlation_id": email_request.correlation_id
            }

        except Exception as e:
            error_message = str(e).lower()

            if "email destinatario rechazado" in error_message or "recipients refused" in error_mes
                raise EmailRecipientRejectedException(Messages.EMAIL_REJECTED_RECIPIENT)
            elif "error de autenticación" in error_message or "authentication" in error_message:
                raise EmailAuthenticationException(Messages.EMAIL_AUTHENTICATION_ERROR)
            else:
                raise EmailSendException(Messages.EMAIL_SEND_FAILED)
```

RESULTADOS

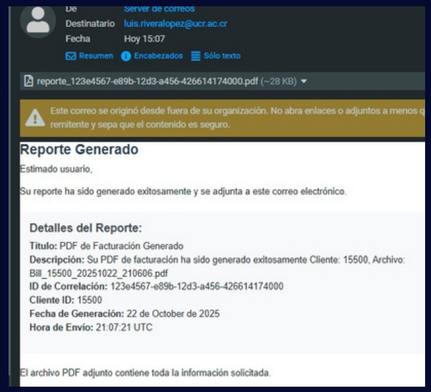
Dashboard de hangfire donde se muestra las tareas programadas a los diferentes servidores



La llegada de el pdf que envia el servidor messaje server al bot de telegram



La llegada de el pdf que envia el servidor Email server al correo destinatario



GRACIAS