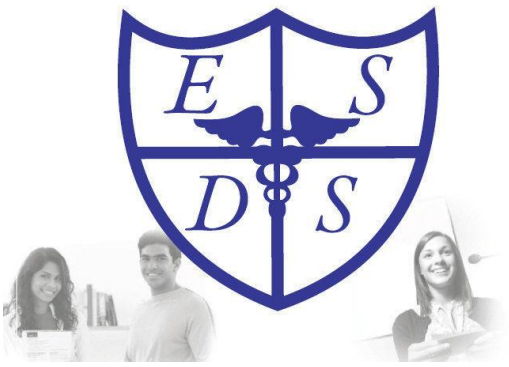
Documentación Tecnica

Crombieversario

Crombie

Escuela Normal Superior y Superior de Comercio N° 46 Domingo Guzmán Silva



“Practicas Profesionalizantes II”

Contenido

Arquitectura del Sistema

Frontend (React + Vite)

Ubicación: crombieversario-app

Interfaz de usuario. Permite visualizar estadísticas, mails enviados, mails con error, editar mensajes, ver empleados y gestionar imágenes.

Componentes principales:

Pages: Vistas principales (Dashboard, Mails Enviados, Mails Error, Editor de Mensaje, Empleados, Calendario).

Componentes: Estadísticas, formularios, tablas, gráficos, hooks de configuración y datos.

Comunicación: Consume la API REST del backend usando fetch/axios y una API key.

Backend (Node.js + Express)

Ubicación: src

Rol:

Provee la API REST, gestiona la lógica de negocio, el envío de mails, el tracking de apertura, la configuración y la gestión de imágenes.

Módulos principales:

server.js:

Expone endpoints REST para mails, estadísticas, imágenes, configuración y tracking.

Valida la API key.

Sirve archivos estáticos (imágenes).

index.js:

Proceso principal y cron: detecta aniversarios, genera mensajes, envía mails, registra logs y errores.

Se comunica con la base de datos y dispara eventos.

eventos.js:

Lógica de eventos personalizados (detección de aniversarios, generación de mensajes).

Inserta el pixel de tracking en los mails.

db.js:

Modelos de datos (Mongoose): SentLog, FailedEmailLog, Config, Aniversario.

Funciones de acceso y manipulación de la base de datos.

Funciones de estadísticas (mensuales, anuales, semanales).

utils/

Utilidades para manejo de archivos, generación y guardado de API key, etc.

Base de Datos (MongoDB)

Ubicación:

Instancia local o remota, accedida vía Mongoose.

Rol:

Almacena logs de mails enviados, mails fallidos, configuración, imágenes,aniversarios y los usuarios que se crean.

Colecciones principales:

sentlogs: Logs de mails enviados (con fecha, abierto, etc).

failedemaillogs: Logs de mails fallidos.

configs: Plantilla de mensaje y rutas de imágenes.

aniversarios: Datos de aniversarios detectados.

Users: Usuarios con email, su respectiva imagen y su rol.

Almacenamiento de Imágenes

Ubicación:

Carpeta local public/uploads en el backend.

Rol:

Almacena imágenes de aniversario que se adjuntan a los mails y se gestionan desde el frontend.

Seguimiento de Apertura de Mails (Tracking Pixel)

Funcionamiento:

El backend expone un endpoint /track/:email/:anniversaryNumber que sirve un pixel transparente.

Cuando el destinatario abre el mail y el cliente carga la imagen, el backend marca el mail como abierto en la base de datos.

Seguridad

API Key:

El backend requiere una API key para acceder a los endpoints protegidos.

El frontend obtiene y usa esta API key en cada petición.

Diagrama de Interconexión (Texto)

Resumen de roles y conexiones

Frontend: Solicita datos y muestra información, permite gestión de mensajes e imágenes.

Backend: Expone API REST, ejecuta lógica de negocio, gestiona mails, imágenes y estadísticas.

Base de datos: Guarda toda la información relevante (logs, config, aniversarios).

Almacenamiento de imágenes: Sirve imágenes para mails y frontend.

Nodemailer: Envía los correos electrónicos.

Tracking Pixel: Marca mails como abiertos cuando el destinatario los visualiza.

Resumen de roles y conexiones

- Frontend: Solicita datos y muestra información, permite gestión de mensajes e imágenes.
- Backend: Expone API REST, ejecuta lógica de negocio, gestiona mails, imágenes y estadísticas.
- Base de datos: Guarda toda la información relevante (logs, config, aniversarios).
- Almacenamiento de imágenes: Sirve imágenes para mails y frontend.
- Nodemailer: Envía los correos electrónicos.
- Tracking Pixel: Marca mails como abiertos cuando el destinatario los visualiza.

Descripción General

Crombieversario es un sistema integral diseñado para la gestión y automatización del envío de correos electrónicos de aniversario laboral a los empleados de Crombie. El sistema permite personalizar mensajes, adjuntar imágenes, realizar seguimiento de la apertura de los correos y visualizar estadísticas detalladas sobre los envíos y aperturas, todo desde una interfaz web moderna y fácil de usar.

Componentes Principales

Frontend (React + Vite)

- Proporciona una interfaz de usuario intuitiva para:
    - Visualizar estadísticas de mails enviados y abiertos (por mes, año y semana).
    - Consultar el historial de mails enviados y mails con error.
    - Editar y previsualizar la plantilla de mensaje de aniversario.
    - Gestionar empleados y aniversarios.
    - Subir y administrar imágenes asociadas a los mensajes.
- Se comunica con el backend mediante una API REST protegida por API Key.

Backend (Node.js + Express)

- Expone endpoints REST para:
    - Obtener y actualizar la configuración y plantilla de mensajes.
    - Consultar y registrar empleados y aniversarios.
    - Gestionar imágenes (subida, listado, borrado).
    - Consultar estadísticas de envíos y aperturas.
    - Registrar y consultar logs de mails enviados y con error.
    - Servir el pixel de tracking para registrar la apertura de correos.
- Incluye un proceso programado (cron) que detecta automáticamente los aniversarios laborales y envía los correos personalizados a los empleados correspondientes.
- Utiliza Nodemailer para el envío de correos electrónicos.

Base de Datos (MongoDB)

- Almacena:
    - Logs de mails enviados y abiertos.
    - Logs de mails fallidos.
    - Configuración y plantilla de mensajes.
    - Datos de empleados y aniversarios.
    - Rutas y metadatos de imágenes.

Almacenamiento de Imágenes

- Las imágenes que se adjuntan a los mensajes se almacenan en el servidor backend y pueden ser gestionadas desde el frontend.

Seguimiento de Apertura de Correos

- Cada correo enviado incluye un pixel de seguimiento único.
- Cuando el destinatario abre el correo y el cliente de correo carga la imagen, el backend registra la apertura en la base de datos.

Flujo de Trabajo

1. Carga de empleados:
   Los empleados y sus fechas de ingreso se cargan en el sistema (vía archivo o API).
2. Detección de aniversarios:
   Un proceso automático revisa diariamente qué empleados cumplen aniversario laboral.

3. Generación y envío de correos:
   El sistema genera un mensaje personalizado (usando la plantilla y las imágenes configuradas) y lo envía al empleado.
4. Registro de logs:
   Cada envío exitoso o fallido se registra en la base de datos.
5. Seguimiento de apertura:
   Si el destinatario abre el correo, el sistema lo detecta mediante el pixel de tracking.
6. Visualización y gestión:
   Los administradores pueden consultar estadísticas, ver el historial de envíos, mails con error, editar la plantilla y gestionar imágenes desde el frontend.

Características Destacadas

- Automatización total del proceso de detección y envío de correos de aniversario.
- Personalización avanzada de mensajes y adjuntos.
- Seguimiento en tiempo real de la apertura de correos.
- Panel de estadísticas con gráficos y reportes por mes, año y semana.
- Gestión centralizada de empleados, imágenes y configuración.
- Seguridad mediante autenticación por API Key.

Usuarios Destinados

- Área de marketing y Recursos Humanos: Para automatizar y monitorear los saludos de aniversario.
- Administradores de sistemas: Para gestionar la configuración y supervisar el funcionamiento.
- Empleados: Como destinatarios de los mensajes personalizados.

Introducción

“Crombieversario” es una aplicación web que automatiza el envío de felicitaciones de aniversario laboral. El sistema consta de tres capas principales: un Frontend construido con React y Vite, un Backend basado en Node.js y Express, y una base de datos MongoDB.

Al acceder a la interfaz, los administradores pueden crear y editar plantillas de correo, incorporar variables como el nombre del empleado y los años de servicio, y asociar imágenes conmemorativas. El flujo de autenticación se maneja mediante JSON Web Tokens (JWT), lo que garantiza que solo los usuarios con rol super_admin o staff puedan modificar configuraciones o consultar registros.

En el Backend, cada endpoint está protegido por middleware de autenticación. El módulo de eventos se encarga de identificar, a través de una tarea programada configurada con node-cron, qué usuarios cumplen aniversario en la fecha actual. Para cada uno, se ensamblan los datos de plantilla, variables de contexto e imagen correspondiente, y se envía el

correo utilizando Nodemailer. Los resultados se registran en las colecciones SentLog y FailedEmailLog, junto con metadatos que permiten posteriormente generar estadísticas de envíos totales, aperturas y errores.

La base de datos MongoDB almacena documentos de usuario (incluyendo email, hash de contraseña, rol y fecha de ingreso), configuraciones de plantilla y registros de envío. Se han definido índices sobre los campos de fecha para optimizar la detección de aniversarios.

1.1. Objetivo

El objetivo principal de "Crombieversario" es proporcionar una solución completamente automatizada y fiable para el envío de correos de felicitación de aniversario laboral. Esta herramienta está diseñada para integrarse de manera transparente con los sistemas de Recursos Humanos existentes, reduciendo al mínimo la intervención manual y garantizando la puntualidad y personalización de cada mensaje.

Para lograrlo, se establecen los siguientes objetivos específicos:

1. Automatización del Proceso de Detección: Implementar un cron job configurado mediante node-cron que ejecute diariamente una rutina de detección de aniversarios. Esta rutina deberá leer las fechas de ingreso almacenadas en MongoDB y filtrar aquellos registros cuya fecha coincida con el día actual, teniendo en cuenta posibles desviaciones de zona horaria. El componente eventos.js será responsable de orquestar este flujo.

2. Gestión Integral de Activos Multimediales: Permitir la carga, validación y almacenamiento de imágenes conmemorativas en formatos estándar (JPEG, PNG). Las imágenes se gestionarán mediante el directorio public/uploads y estarán referenciadas desde los documentos de configuración. Se implementarán validaciones de tamaño y formato al momento de la carga.

3. Envío Robusto y Escalable de Correos: Integrar Nodemailer con un servicio SMTP configurable a través de variables de entorno. El servicio de envío deberá gestionar colas, reintentos en caso de errores transitorios y respetar límites de tasa para evitar bloqueos por parte del proveedor de correo. Se considerará la posibilidad de desacoplar el envío en un microservicio independiente para mejorar la escalabilidad.

4. Registro y Trazabilidad de Eventos: Definir esquemas de log (SentLog y FailedEmailLog) que almacenen metadatos detallados de cada intento de envío: timestamp, destinatario, plantilla utilizada, estado (éxito o tipo de error) y código de respuesta SMTP. Adicionalmente, se desarrollarán endpoints que permitan la consulta y filtrado de estos logs para auditorías internas.

5. Visualización de Métricas en el Frontend: Construir componentes de visualización en React que muestren métricas clave, tales como número total de correos enviados, tasa de apertura, cantidad de errores y tendencias mensuales. Estos componentes deberán ser reutilizables y configurables para distintos periodos de tiempo.

6. Seguridad y Control de Acceso: Implementar autenticación basada en JWT para proteger todos los endpoints críticos del backend. Se definirá un middleware de autorización que verifique roles (super_admin y staff) antes de permitir operaciones sensibles, como modificación de plantillas o eliminación de registros.

1.2. Alcance

El alcance de Crombieversario Automático incluye:

- Integración con la API de PeopleForce para la obtención automática de datos de empleados, tales como nombre, correo y fecha de ingreso.
- Gestión completa del ciclo de vida de las plantillas de correo y activos multimedia necesarios para las felicitaciones.
- Implementación de un servicio de envío de correos automatizado, con reintentos.
- Registro, almacenamiento y consulta de logs de envío y errores.
- Módulo de tareas programadas para la detección diaria de aniversarios.

Excluye:

- Gestión de otros tipos de notificaciones vía email (cumpleaños personales, promociones internas, etc.).
- Integración con servicios de mensajería distintos al correo electrónico.

1.3. Audiencia

La documentación está dirigida a los siguientes perfiles:

- Equipo de Desarrollo Frontend: responsables de la implementación de la interfaz y componentes React.
- Equipo de Desarrollo Backend: encargados de la API en Node.js, lógica de negocio y cron jobs.
- Ingenieros de DevOps: planteamiento de despliegue, configuración de entornos, monitoreo y escalabilidad.
- Equipo de QA: diseño y ejecución de pruebas unitarias, de integración y de aceptación.
- Stakeholders de Recursos Humanos y Marketing: supervisión de requerimientos funcionales y validación de la solución.

2.Definiciones y Especificación de Requerimientos

2.1. Definición general del proyecto

“Crombieversario” es una plataforma modular que se integra con la información de los empleados de Crombie, gracias a la API de “PeopleForce”, para gestionar de manera autónoma el envío de correos de felicitación de aniversario laboral. El usuario principal es el staff de Marketing, que accederán a través de una interfaz web para configurar plantillas, administrar usuarios y revisar estadísticas.

El proyecto contempla tres módulos principales:

- Frontend (crombieversario-app): Aplicación cliente en React + Vite, responsable de la experiencia de usuario.
- Backend (proyecto-practicas): API RESTful en Node.js + Express que atiende peticiones del frontend y controla la lógica de negocio.
- Base de Datos (MongoDB): Persistencia de datos de usuarios, configuraciones y logs.

2.2. Requisitos funcionales

RF01. Gestión de Usuarios: El sistema debe permitir crear, leer, actualizar y eliminar cuentas de usuario, asignando roles (super_admin, staff).

RF02. Autenticación y Autorización: Implementar login mediante JWT; las rutas deben verificar la validez del token y el rol asociado.

RF03. Configuración de Plantillas: Posibilitar la creación, edición y eliminación de la plantilla de correo.

RF04. Carga y Administración de Imágenes: Permitir subir, listar y eliminar imágenes conmemorativas desde la interfaz, validando formatos y tamaños.

RF05. Detección de Aniversarios: Ejecutar diariamente un proceso que identifique usuarios con aniversario de ingreso y obtenga sus datos para el envío.

RF06. Envío de Correos Automático: Generar y enviar correos personalizados mediante Nodemailer, registrando el resultado en logs.

RF07. Registro de Logs: Almacenar cada intento de envío en SentLog o FailedEmailLog, incluyendo timestamp, destinatario, plantilla utilizada y estado.

RF08. Visualización de Estadísticas: Mostrar en el dashboard métricas de envíos totales, tasa de apertura y errores, filtrables por periodo.

RF09. Configuración de Cron: Exponer en el backend parámetros de configuración del cron (hora de ejecución, zona horaria).

RF10. Seguridad de Datos: Proteger la carga y visualización de datos sensibles mediante HTTPS y configuración de CORS.

RF11. Manejo de Errores: El sistema debe capturar y mostrar errores de forma clara en el frontend y registrar detalles en el backend.

3.Procedimientos de Instalación y Prueba

3.1. Entorno y dependencias

Requisitos previos

- Node.js (v18 o superior recomendado)
- npm (gestor de paquetes de Node.js)
- MongoDB (En la realización del proyecto se uso MongoDB Compass)
- Git (opcional, para clonar el repositorio)
- Cuenta de Gmail (para el envío de correos automáticos, se recomienda usar una contraseña de aplicación)

  Requisitos previos

Dependencias principales

Backend (proyecto-practicas):

- express
- mongoose
- dotenv
- bcryptjs
- jsonwebtoken
- multer
- node-cron
- nodemailer
- axios
- dayjs

Estas dependencias están listadas en el archivo package.json.

Frontend (crombieversario-app):

- react
- react-dom
- react-router-dom
- axios
- vite (para desarrollo y build)

Otras dependencias de UI y utilidades según el archivo package.json.

3.2 Instalación.

1. Ingresa a este link: [2Diego2/Crombieversario-Autom-tico]

Opciones:

Clonar Repositorio.

- Ve a la página principal del repositorio y haz clic en el botón "Code" (Código).

Copia la URL del repositorio

- Elige el método de clonación: HTTPS o SSH.
- HTTPS: fácil de usar, sólo necesitas tu usuario/contraseña o token.
- SSH: más seguro, requiere que tengas configuradas claves SSH en tu cuenta

Abre la terminal

- Usa Git Bash (Windows), Terminal (macOS/Linux) o la integrada de tu editor (como VS Code).

Navega hasta la carpeta destino

- cd ruta/a/tu/carpeta

Clona el repositorio

- Ejecuta: git clone https://github.com/usuario/proyecto.git

Verifica que se clono correctamente

Cambia al directorio recién creado

- Cd nombre-del-repositorio

Luego, ejecuta:

- ls

y/o

- git log --oneline

Descargar repositorio completo como ZIP.

Ve a la página principal del repositorio en Github.

Haz clic en el botón verde "Code" que aparece arriba junto a la lista de archivos

En el menú desplegable, selecciona "Download ZIP" o "Descargar ZIP"

Tu navegador comenzará la descarga de un archivo comprimido con todo el contenido del repositorio.

Una vez finalizado, descomprime el archivo en tu equipo para acceder a los archivos más recientes.

3.2. Instalación del frontend

Una vez instalado el repositorio en nuestro dispositivo, se puede realizar la instalación tanto del backend y frontend.

Ingresamos a nuestra carpeta principal

- cd ruta/a/tu/carpeta

Instalamos las dependencias

cd crombieversario-app

npm install

3.3. Instalación del backend

Ingresamos a nuestra carpeta del backend

- *cd ruta/a/tu/carpetaprincipal*
- *cd proyecto-practicas –> npm install*

- *cd src -->npm install*

3.4. Configuración de la base de datos

Debemos ingresar a https://cloud.mongodb.com/

Creamos un cluster, por ejemplo: "Crombieversario" --> Ingresamos a "Browse Collections"

Creamos una base de datos llamada "Crombieversario" y creamos una de las 4 colecciones "sentLogs", que guardara los mails que se enviaron.

Realizamos la creación de las siguientes 3 colecciones:

"Users" --> Guarda los usuarios que se crean, con sus respectivos roles.

"FailedEmailLog" --> Guarda los mails que fallaron.

"Config"--> Guarda las rutas de las imágenes y el mensaje que se envía junto con los mails.

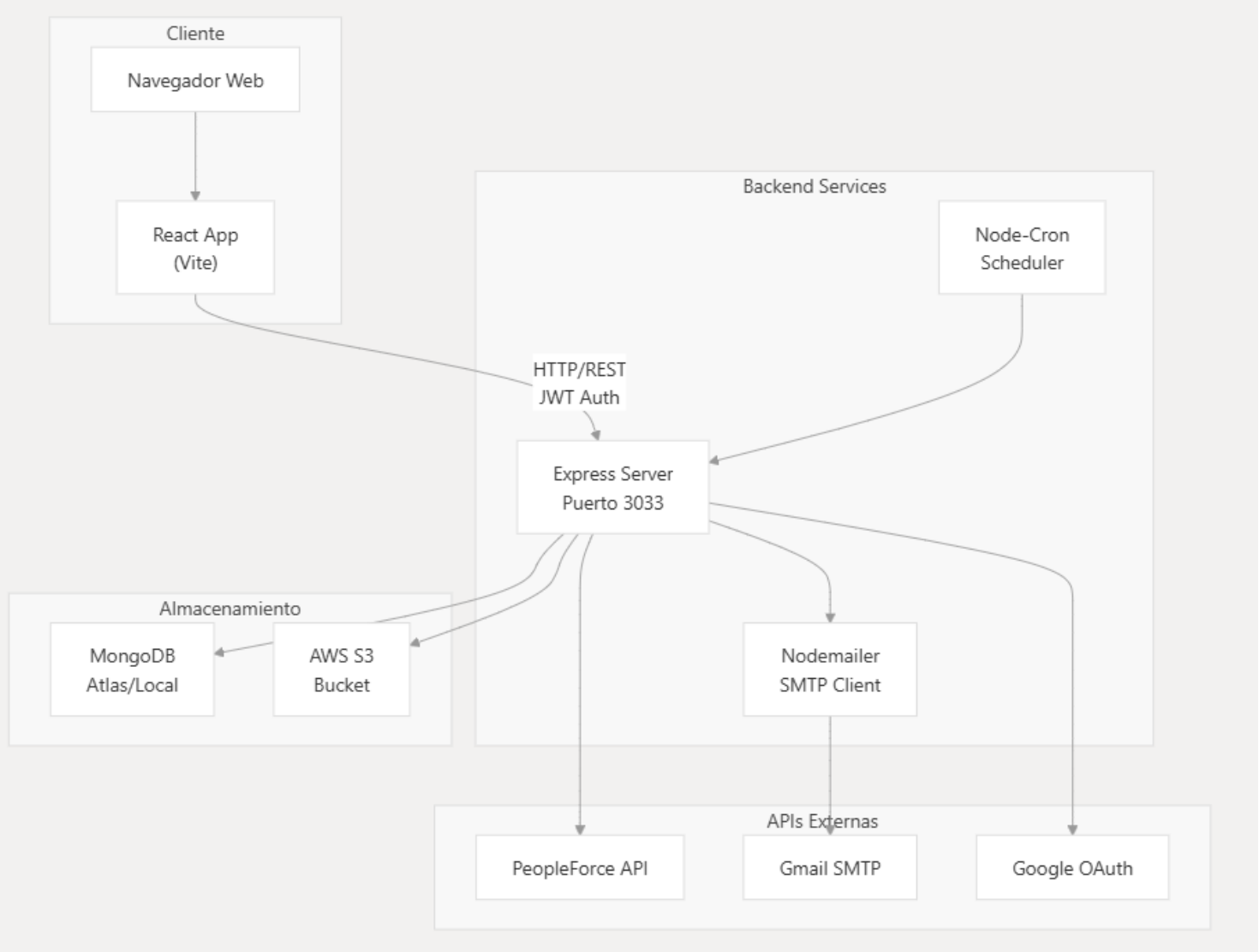
4. Arquitectura del Sistema

4.1. Visión General de la Arquitectura

El sistema Crombieversario Automático es una aplicación full-stack que automatiza el envío de correos de aniversario y cumpleaños para empleados. La arquitectura sigue un patrón cliente-servidor con tres capas principales:

- **Frontend**: Aplicación React con Vite
- **Backend**: API REST con Node.js y Express
- **Base de Datos**: MongoDB para persistencia de datos
- **Almacenamiento**: AWS S3 para imágenes

4.2. Diagrama de Interconexión de Componentes



4.3. Descripción de Módulos

4.3.1. Frontend (React + Vite)

El frontend está construido con React 19.1.0 y Vite 7.0.0

Componentes principales:

- **App.jsx**: Punto de entrada, maneja autenticación y enrutamiento
- **EditorMensaje.jsx**: Editor de plantillas de correo y gestión de imágenes
- **EmpleadosPage.jsx**: Gestión de empleados y roles

Hooks personalizados:

- **useAuth.js**: Gestión de autenticación JWT
- **useConfig.js**: Configuración de mensajes e imágenes
- **useEstadisticasMail.js**: Estadísticas de correos
- **useEventosProximos.js**: Eventos de calendario

Dependencias clave:

- **axios**: Cliente HTTP para llamadas API
- **jwt-decode**: Decodificación de tokens JWT
- **@fullcalendar/react**: Visualización de calendario
- **recharts**: Gráficos de estadísticas

4.3.2. Backend (Node.js + Express)

El backend expone una API REST en el puerto 3033

Estructura:

- **server.js**: Servidor Express y endpoints API
- **index.js**: Scheduler de tareas cron

Endpoints principales:

- Autenticación:**POST /api/users/createPUT /api/users/update-role-password**
- Configuración: **GET /api/configPUT /api/config**
- Imágenes: **POST /api/upload-image/:anniversaryNumber**
- Estadísticas: **GET /api/email-stats/yearlyGET /api/email-stats/monthly**
- Empleados: **GET /trabajadores**

4.3.3. Base de Datos (MongoDB)

MongoDB se utiliza para almacenar datos de empleados, configuración, logs y usuarios

Configuración:

- URI de conexión: MongoDB Atlas (recomendado) o local
- Base de datos:**Crombieversario**

Colecciones principales (inferidas del código):

- **users**: Usuarios del sistema con roles
- **sentLogs**: Registro de correos enviados exitosamente
- **errorLogs**: Registro de errores en envíos
- Configuración de mensajes e imágenes

4.3.4. Almacenamiento de Imágenes

Las imágenes se almacenan en AWS S3

Configuración S3:

- Región: **us-east-2**
- Bucket: **crombieversario**
- Credenciales: Access Key ID y Secret Access Key

4.4. Dependencias Externas

APIs de terceros:

- **PeopleForce API**: Sincronización de datos de empleados
- **Google OAuth**: Autenticación de usuarios
- **Gmail SMTP**: Envío de correos mediante Nodemailer
- **AWS S3**: Almacenamiento de imágenes

5. Descripción de Procesos y Servicios

5.1. Detección de Aniversarios (cron)

El sistema utiliza para ejecutar tareas programadas. node-cron

Flujo de automatización:

1. **Sincronización de datos desde PeopleForce API**
2. **Consulta de aniversarios y cumpleaños del día**
3. **Generación de correos personalizados**
4. **Obtención de imágenes desde AWS S3**
5. **Envío mediante Gmail SMTP**

5.2. Generación y envío de correos

El sistema utiliza **Nodemailer con Gmail SMTP** para el envío automatizado de correos . Las credenciales se configuran mediante variables de entorno GMAIL_USER y GMAIL_APP_PASSWORD

El flujo de generación incluye:

1. Consulta diaria de empleados con cumpleaños o aniversarios desde la colección trabajadores
2. Obtención de la plantilla de mensaje desde la colección config
3. Reemplazo de variables como {{nombre}} y {{años}} con datos del empleado
4. Selección de imagen correspondiente desde AWS S3 según años de aniversario
5. Envío mediante Gmail SMTP
6. Registro del resultado en sentLogs (éxito) o errorLogs (fallo)

5.3. Gestión de logs y estadísticas

El sistema mantiene dos colecciones para auditoría:

sentLogs: Registra envíos exitosos con
campos recipientEmail, recipientName, eventType, anniversaryYear, sentAt, messageContent, imageAttached

errorLogs: Almacena fallos
con recipientEmail, recipientName, eventType, errorMessage, errorStack, attemptedAt, retryCount

Los endpoints de estadísticas disponibles son:

- GET /api/email-stats/yearly: Estadísticas anuales
- GET /api/email-stats/monthly?year=YYYY: Estadísticas mensuales

7 Seguridad

7.1. Autenticación y autorización (API Key)

El sistema implementa **autenticación dual**:

1. **JWT para usuarios**: Tokens generados tras login, almacenados en localStorage y enviados en header Authorization
2. **Google OAuth**: Autenticación mediante GOOGLE_CLIENT_ID y GOOGLE_CLIENT_SECRET
3. **API Key interna**: Variable API_KEY para comunicación entre servicios

El sistema de roles distingue entre SUPER_ADMIN y STAFF, controlando acceso a funcionalidades como creación de usuarios.

7.2. Validación de entrada

Se observan validaciones en el frontend:

- **Contraseñas**: Mínimo 6 caracteres y confirmación de coincidencia
- **Imágenes**: Solo formato PNG permitido
- **Números de aniversario**: Solo enteros positivos

El hook useAuth maneja errores de autenticación (401/403) redirigiendo al login

7.3. Almacenamiento seguro de credenciales

Las credenciales se gestionan mediante **variables de entorno** en archivos .env:

- Contraseñas hasheadas con **bcrypt** en la colección users
- Gmail App Password (no contraseña normal)
- JWT Secret para firma de tokens
- AWS credentials para S3

Importante: Los archivos .env contienen credenciales en texto plano y deben estar en .gitignore.

8 Gestión de Imágenes

8.1. Estructura de carpetas

El sistema utiliza **AWS S3** como almacenamiento de imágenes con el bucket crombieversario en la región us-east-2. La estructura de carpetas sigue una convención específica:

Imágenes de aniversario (raíz del bucket):

- Patrón: {año}.png (ej: 1.png, 5.png, 10.png, 25.png)
- Uso: Imágenes personalizadas según años de servicio del empleado
- Acceso: URLs públicas directas desde S3

Otras categorías de assets:

- profiles/{employeeId}.jpg: Fotos de perfil sincronizadas desde PeopleForce
- logos/{variant}.png: Logos corporativos para branding
- backgrounds/{template}.png: Fondos para plantillas de email

Esta estructura permite una organización clara donde las imágenes de aniversario están en la raíz para acceso rápido, mientras que otros assets están categorizados en subdirectorios.

8.2. Endpoints de subida, listado y borrado

Endpoint de Subida:

POST /api/upload-image/:anniversaryNumber

El frontend implementa esta funcionalidad en el hook useConfig mediante la función uploadImageApi. El proceso:

1. Crea un FormData con el archivo seleccionado
2. Envía petición POST con header Content-Type: multipart/form-data
3. Incluye autenticación JWT en headers mediante getAuthHeader()
4. El parámetro :anniversaryNumber determina el nombre del archivo en S3 (ej: 5.png)

Validaciones previas al upload:

- Archivo seleccionado no vacío
- Número de aniversario válido (entero positivo)
- Formato PNG exclusivamente (rechaza otros formatos)

Endpoint de Listado:

GET /api/config

Este endpoint retorna la configuración completa incluyendo el array imagePaths con todas las URLs de S3. El frontend aplica ordenamiento numérico automático:

```
const sortedImagePaths = imagePaths.sort((a, b) => {
  const yearA = parseInt(a.match(/(\d+)\.png$/)[1]);
  const yearB = parseInt(b.match(/(\d+)\.png$/)[1]);
  return yearA - yearB;
});
```

Esto garantiza que las imágenes se muestren en orden ascendente (1, 5, 10, 25, etc.) independientemente del orden de inserción.

Endpoint de Borrado:

DELETE /api/delete-image

Body: { imageUrl: "https://crombieversario.s3.us-east-2.amazonaws.com/5.png" }

Implementado en deleteImageApi useConfig.js:105-118 . Envía la URL completa de S3 en el body de la petición DELETE, y el backend se encarga de eliminar el objeto del bucket y actualizar la colección config en MongoDB.

8.3. Retorno y uso en el frontend

Las imágenes se retornan como **URLs públicas de S3** en formato:

https://crombieversario.s3.us-east-2.amazonaws.com/{año}.png

El componente EditorMensaje consume estas URLs directamente en tags <img>:

```
config.imagePaths.map((path) => {
  const fileName = path.split("/").pop();
  const anniversaryNum = fileName.split(".")[0];

  return (
    <div key={path} className="img">
      <img src={path} alt={`Aniversario ${anniversaryNum}`} />
      <button onClick={() => handleDeleteImage(path)}>×</button>
      <p>{anniversaryNum} años</p>
    </div>
  );
})
```

Características del renderizado:

- Extrae el número de años del nombre del archivo (5.png → 5)
- Muestra miniaturas de 100x100px con object-fit: cover
- Botón de eliminación posicionado absolutamente en esquina superior derecha

- Etiqueta descriptiva debajo de cada imagen

Flujo completo de actualización:

1. Usuario sube/elimina imagen
2. Backend actualiza S3 y MongoDB
3. Frontend ejecuta refetchConfig() para obtener lista actualizada
4. Estado config se actualiza automáticamente
5. Componente re-renderiza con nuevas imágenes

9 Despliegue

9.1. Requisitos de entorno de producción

Infraestructura base:

1. **Node.js**: Versión LTS recomendada para estabilidad
2. **MongoDB**:
    - Opción recomendada: MongoDB Atlas (cloud)
    - Alternativa: Instancia local con Docker
3. **AWS S3**: Bucket configurado con permisos de lectura/escritura
4. **Gmail SMTP**: Cuenta con App Password habilitado (no contraseña normal)
5. **PeopleForce API**: Acceso con API Key válida

Configuración Docker (recomendada para producción):

El proyecto incluye docker-compose.yml con tres servicios README.md:93-97 :

- mongodb: Base de datos con persistencia de volúmenes
- backend: Node.js/Express en puerto 3033
- frontend: React servido en puerto 80 (Nginx)

Variables de entorno críticas:

```
# Producción
MONGO_URI=mongodb+srv://user:pass@cluster.mongodb.net/Crombieversario
SERVER_BASE_URL=http://backend:3033  # Comunicación interna
FRONTEND_BASE_URL=https://tudominio.com  # Acceso público
API_BASE_URL=https://tudominio.com
```

Seguridad:

- JWT_SECRET: Clave robusta para firma de tokens
- API_KEY: Autenticación interna entre servicios
- Google OAuth credentials para autenticación social

9.2. Procedimiento de despliegue continuo

Estado actual: No hay configuración CI/CD implementada. El proyecto tiene la base para despliegue containerizado pero requiere configuración adicional.

Evidencia de preparación para despliegue:

- Dockerfiles para backend y frontend
- Comentario sugiere uso previo de AWS Elastic Beanstalk
- Variables de entorno separadas para desarrollo/producción

Procedimiento manual recomendado:

1. **Build de imágenes Docker**:

```
docker-compose build
```

2. **Push a registry** (Docker Hub, ECR, etc.):

```
docker tag backend:latest registry/backend:v1.0
docker push registry/backend:v1.0
```

3. **Despliegue en servidor**:

```
docker-compose up -d
```

4. **Verificación de servicios**:
    - Backend: http://servidor:3033/api/config
    - Frontend: http://servidor:80

Propuesta de CI/CD automatizado (no implementado):

- GitHub Actions para build automático en push a main
- Tests unitarios como gate de calidad
- Deploy automático a staging/producción
- Rollback automático en caso de fallo

9.3. Monitoreo y alertas

Estado actual: **No implementado**. El sistema solo registra logs en MongoDB sin alertas proactivas.

Logging existente:

El sistema mantiene dos colecciones de auditoría:

1. **sentLogs**: Envíos exitosos con campos recipientEmail, sentAt, messageContent, imageAttached
2. **errorLogs**: Fallos con errorMessage, errorStack, attemptedAt, retryCount

Endpoints de consulta manual:

- GET /api/email-stats/yearly: Estadísticas anuales
- GET /api/email-stats/monthly?year=YYYY: Estadísticas mensuales

10 Propuesta de mejoras futuras

Prioridad Alta

1. **Sistema de reintentos automáticos**:
    - Usar campo retryCount de errorLogs para lógica de reintento
    - Cron job secundario que procese errores con retryCount < 3
    - Backoff exponencial (1h, 4h, 24h)
    - Notificación a admin tras 3 intentos fallidos
2. **CI/CD completo**:
    - GitHub Actions con stages: test → build → deploy
    - Ambientes separados: development, staging, production
    - Secrets management con GitHub Secrets
    - Rollback automático si health check falla

Prioridad Media

3. **Monitoreo y observabilidad**:
    - **Sentry** para tracking de errores con stack traces
    - **AWS CloudWatch** para métricas de infraestructura
    - **Prometheus + Grafana** para métricas custom (emails/día, tasa de error)
    - Alertas vía Slack/email cuando error rate > 5%
4. **Validación backend robusta**:
    - Librería Joi o Zod para validación de esquemas

- Middleware de validación en todos los endpoints
- Sanitización de inputs para prevenir XSS/injection
- Actualmente solo hay validación frontend EditorMensaje.jsx:92-120

5. **Gestión de secretos**:
    - Migrar de .env a **AWS Secrets Manager** o **HashiCorp Vault**
    - Rotación automática de credenciales
    - Auditoría de acceso a secretos
    - Actualmente credenciales en texto plano .env:1-49

Prioridad Baja

6. **Rate limiting**:
    - Middleware express-rate-limit en endpoints públicos
    - Límite de 100 req/min por IP
    - Protección contra ataques DDoS
7. **Suite de tests**:
    - Jest para tests unitarios de funciones puras
    - Supertest para tests de integración de API
    - Cypress para tests E2E del frontend
    - Coverage mínimo del 70