EmailService

September 4, 2026

EmailService

Kind: Service

Source: `atloria-monorepo/apps/api/src/email/email.service.ts`

EmailService — one send API, three transports, picked at startup:

1. gateway — the Sherkety email gateway (EMAIL_GATEWAY_URL + EMAIL_GATEWAY_KEY).
   Managed sending from noreply@atloria.app; the platform default in prod.
2. smtp — nodemailer via SMTP_* env vars (self-host escape hatch).
3. console — logs the email instead of sending; local-dev fallback.

The gateway is SendGrid-v3-compatible and carries attachments (base64, max 10 per message) — SMTP remains only as the self-host escape hatch.

`EmailService` is the backend email-sending façade that exposes a single `send` API while selecting one of three transports at startup: Sherkety gateway (default in prod), SMTP via nodemailer, or a console logger for local development. It centralizes email delivery concerns (auth/config, transport selection, fallbacks) so the rest of the system can send emails without knowing the underlying provider. When attachments are present, it bypasses the gateway (no attachment support) and falls back to SMTP if configured, otherwise logs a warning and prints the email.

Methods

Method	Signature	Returns	Description
`sendMail`	`sendMail(opts: SendEmailOptions)`	`Promise<boolean>`	Send an email.

When something fails

- `EmailService` handles failure in 2 places: it turns it into a return value in all 2.

Diagram

```
sequenceDiagram
  autonumber
  participant App as NestJS App Code
  participant Email as EmailService
  participant GW as Sherkety Gateway Transport
  participant SMTP as SMTP (nodemailer)
  participant Console as Console Transport

  App->>Email: send(message)
  Note over Email: Transport chosen at startup\n(gateway | smtp | console)

  alt message has attachments
    Note over Email: Gateway contract has no attachments
    alt SMTP configured
      Email->>SMTP: sendWithAttachments(message)
      SMTP-->>Email: result
    else SMTP not configured
      Email->>Console: warn + log(message)
      Console-->>Email: ok (logged)
    end
  else no attachments
    alt Transport = gateway
      Email->>GW: send(message)
      GW-->>Email: result
    else Transport = smtp
      Email->>SMTP: send(message)
      SMTP-->>Email: result
    else Transport = console
      Email->>Console: log(message)
      Console-->>Email: ok (logged)
    end
  end

  Email-->>App: delivery result / completion
```

Usage

```
import { Injectable } from '@nestjs/common';
import { EmailService } from './email.service';

@Injectable()
export class InviteService {
  constructor(private readonly email: EmailService) {}

  async sendInvite(to: string, inviteUrl: string) {
    await this.email.send({
      to,
```

```
      subject: 'You've been invited to Atloria',
      // shape may vary; keep to your app's email DTO/contract
      html: `<p>Join here: <a href="${inviteUrl}">${inviteUrl}</a></p>`,
      text: `Join here: ${inviteUrl}`,
    });
  }

  async sendReport(to: string, pdfBuffer: Buffer) {
    await this.email.send({
      to,
      subject: 'Your monthly report',
      text: 'Attached is your monthly report.',
      attachments: [
        {
          filename: 'report.pdf',
          content: pdfBuffer,
          contentType: 'application/pdf',
        },
      ],
    });
  }
}
```

AI Coding Instructions

- Treat `EmailService` as the only email entrypoint; do not instantiate transports directly in feature modules—inject the service and call `send`.
- Remember: gateway does **not** support attachments; if you add attachments to a message, ensure SMTP is configured in environments where delivery matters, or expect console fallback + warning.
- Keep transport selection/config at startup via environment variables (`EMAIL_GATEWAY_URL/KEY`, `SMTP_*`); avoid per-request switching or ad-hoc overrides.
- When modifying the send contract/DTO, update all transports consistently (gateway, SMTP, console) and preserve the fallback behavior for attachments.

Referenced By

- `ScheduledReportsService` (DEPENDS_ON)
- `AuthService` (DEPENDS_ON)
- `ChangelogSubscriptionsService` (DEPENDS_ON)
- `DocAutomationService` (DEPENDS_ON)
- `EmailModule` (MODULE_PROVIDES)
- `EmailModule` (MODULE_EXPORTS)

- `InvitationService` (DEPENDS_ON)
- `IssuesService` (DEPENDS_ON)
- `OpsAlertsService` (DEPENDS_ON)
- `ProjectDomainsService` (DEPENDS_ON)
- `DocsAccessService` (DEPENDS_ON)
- `ReaderMagicLinkService` (DEPENDS_ON)
- `StaleAlertService` (DEPENDS_ON)
- `TechnicalDocsGenerationService` (DEPENDS_ON)