UNIVERSIDADE ESTADUAL DA PARAÍBA
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS - CAMPUS VII
DISCIPLINA: Métodos Avançados de Programação
CURSO DE CIÊNCIA DA COMPUTAÇÃO
DOCENTE: Prof. Giovanna Trigueiro

PROJETO

Central de Comunicação estudantil por e-mail

EQUIPE:

Matheus de Sousa Medeiros

Felipe Oliveira

Renan Martins de Lima

Pedro Henrique Almeida Guimarães

PATOS – PB

INTRODUÇÃO:

O projeto **CCEE (Central de Comunicação Estudantil por E-mail)** tem como finalidade desenvolver uma aplicação web que consolide a comunicação institucional entre escolas e seus alunos. A aplicação permite que estudantes visualizem, em um ambiente unificado, as mensagens recebidas por e-mail acadêmico, apresentadas de forma clara e organizada.

A proposta surgiu da observação de que muitos estudantes perdem informações importantes por não acompanharem regularmente os canais oficiais de comunicação, como o e-mail institucional. Assim, o sistema busca resolver esse problema centralizando essas mensagens em uma interface moderna, leve e acessível.

A aplicação foi desenvolvida com **React** e **TypeScript**, adotando práticas modernas de front-end, como:

- Organização em componentes reutilizáveis (ex: Background, Chat);
- Separação de responsabilidades entre layout, lógica e exibição;
- Utilização do padrão de projeto **Template Method** para estruturar mensagens com cabeçalho, corpo e rodapé de forma extensível;
- Potencial para extensão com outros padrões, como Visitor, caso se deseje tratar diferentes tipos de mensagens dinamicamente.

O resultado é uma interface que permite ao aluno visualizar as comunicações recebidas, com foco na experiência do usuário e facilidade de manutenção do código. O projeto também reforça conceitos fundamentais de engenharia de software, como versionamento adequado com Git e aplicação dos princípios SOLID no design dos componentes.

DESENVOLVIMENTO:

O projeto CCEE foi desenvolvido utilizando React com TypeScript, com o objetivo de criar uma aplicação web que centralize a exibição de e-mails acadêmicos. O front-end é estruturado em componentes reutilizáveis e prioriza legibilidade, escalabilidade e facilidade de manutenção.

A aplicação é client-side (SPA - Single Page Application), estruturada com as seguintes tecnologias:

- React: biblioteca principal para construção da interface;
- TypeScript: fornece tipagem estática para maior segurança e clareza no código;
- CSS modularizado: para estilização dos componentes;
- Vite ou React Scripts: ferramenta de build rápida e moderna (dependendo do projeto original).

A organização do projeto segue uma arquitetura **modular e orientada a componentes**, separando responsabilidades de maneira clara. A estrutura básica é:

```
src/
├── components/        → componentes reutilizáveis (ex: Background)
├── pages/             → páginas principais (ex: Chat)
├── templates/         → templates abstratos (ex: MessageTemplate)
├── stylesheets/       → arquivos de estilo organizados por contexto
├── App.tsx            → ponto de entrada da aplicação
└── main.tsx           → renderização principal
```

Padrões Utilizados no Projeto (CCEE):

Template Method (Comportamental)

Implementado na renderização das mensagens:

- A classe abstrata MessageTemplate define a estrutura base para exibir uma mensagem (com renderHeader(), renderBody() e renderFooter()).
- Classes concretas, como TextMessage, herdam e implementam os detalhes específicos de cada parte.
- O método render() define o *esqueleto fixo*, enquanto os métodos abstratos permitem personalizações específicas.

Motivação: permite que tipos diferentes de mensagem sejam renderizados mantendo um formato comum e consistente na interface.

Separação de Responsabilidades (SRP - SOLID):

Cada componente possui uma única responsabilidade:

- Background cuida exclusivamente da camada visual de fundo;
- Chat é responsável pela exibição das mensagens;
- TextMessage cuida da estrutura da mensagem em si.

Isso melhora a organização e torna o código mais manutenível e testável.

Uso do Padrão Chain of Responsibility:

No contexto do projeto **CCEE**, o padrão **Chain of Responsibility** pode ser aplicado (ou planejado) para lidar com o **processamento de mensagens antes do envio ou exibição**. Esse padrão permite que uma cadeia de objetos trate uma solicitação de forma flexível, onde cada elemento da cadeia decide se lida com a requisição ou a repassa ao próximo.

Decisões de Implementação:

- Componentização como prioridade: facilitando testes, legibilidade e reaproveitamento.
- Uso de tipagem com TypeScript: aumenta a robustez e detecta erros em tempo de desenvolvimento.
- Separação clara por pastas: organização lógica e coesa para acelerar a leitura e edição do código.
- Simplicidade de layout: prioriza clareza visual e foco no conteúdo (mensagens).
- Base para extensibilidade: o uso do Template Method e da estrutura de componentes permite que novas funcionalidades (como filtro de mensagens, categorias ou notificações) sejam adicionadas facilmente**.**

Como parte da solução proposta para o projeto CCEE, foi considerado o uso de uma API Gemini para realizar o envio automatizado de e-mails institucionais. Essa integração permite que o sistema não apenas exiba mensagens recebidas pelos estudantes, mas também gere e envie novas comunicações diretamente pela aplicação, caso a interface de administrador ou secretaria venha a ser implementada.

Finalidade da API

A API Gemini atua como serviço de envio de e-mails, fornecendo uma interface segura e eficiente para a transmissão de mensagens. Através de requisições HTTP (REST), é possível:

- Enviar e-mails individuais ou em massa para estudantes;
- Personalizar remetente, assunto, corpo da mensagem e anexos;
- Registrar logs de envio, facilitando auditorias e controle.

Como a Integração Funciona (Fluxo Básico)

1. O sistema React coleta os dados do e-mail a ser enviado (remetente, destinatário, conteúdo etc.);
2. Os dados são enviados para um servidor intermediário (backend) via HTTP (por segurança e controle de autenticação);
3. O backend realiza uma chamada POST para a API Gemini com as credenciais e o corpo da mensagem;
4. A API Gemini processa a solicitação e envia o e-mail ao destinatário;
5. O sistema registra e notifica o resultado (sucesso ou erro de envio).

Segurança e Autenticação

Para proteger o envio de e-mails via API, a Gemini utiliza autenticação via chave de API (API Key), que deve ser armazenada em um servidor backend seguro — não no front-end — a fim de evitar exposições indevidas.

Benefícios da Integração

- Automação do envio de mensagens institucionais;
- Escalabilidade: permite o envio em massa sem sobrecarregar a infraestrutura da instituição;
- Rastreabilidade: logs de entrega e falha ajudam na auditoria e prestação de contas;
- Customização: corpo do e-mail pode ser adaptado para diferentes contextos acadêmicos (avisos, notificações, convites, lembretes).

Possibilidades Futuras

- Interface administrativa para criação e envio direto pela equipe pedagógica;
- Histórico de mensagens enviadas com status de leitura;
- Integração com sistemas de matrícula para envio automático em eventos como inscrições, provas ou renovação de matrícula.

CONCLUSÃO:

O desenvolvimento do projeto demonstrou, na prática, como soluções web podem ser aplicadas para melhorar a comunicação entre instituições de ensino e seus alunos. Por meio de uma aplicação construída com React e TypeScript, foi possível criar uma interface simples, intuitiva e eficiente para exibir mensagens acadêmicas de forma centralizada.

A estrutura do sistema foi cuidadosamente planejada para manter clareza e organização, utilizando padrões de projeto como o Template Method, que viabiliza a expansão do sistema com diferentes tipos de mensagens. O uso de componentes reutilizáveis e a separação de responsabilidades reforçam as boas práticas de desenvolvimento front-end e facilitam a manutenção a longo prazo.

Além da exibição de mensagens, a integração com a API Gemini representa um passo importante rumo à automação da comunicação institucional, permitindo o envio dinâmico de e-mails diretamente pela aplicação, com segurança e rastreabilidade.

O projeto também abriu espaço para diversas possibilidades futuras, como filtros personalizados, autenticação de usuários, notificações em tempo real, e o desenvolvimento de um painel administrativo completo.

Em resumo, o CCEE oferece uma base sólida e extensível para a modernização da comunicação estudantil, aliando praticidade de uso com princípios sólidos de engenharia de software.