Centro Universitário da Católica de Santa Catarina em Joinville

Engenharia de Software

Synk

Cristian Prochnow

Joinville

2025

Resumo

A publicação de conteúdos de forma distribuída entre vários canais de comunicação pode ser uma tarefa complexa e propensa a erros, o que torna o gerenciamento desses passos um empecilho para o alcance de resultados eficientes. Neste contexto, o objetivo da aplicação é proporcionar um meio integrado com as redes sociais, oferecendo um ambiente que gerencia o conteúdo por meio de estruturas pré-cadastradas e também publicação de forma automatizada. Isso é possível pelo cadastro de modelos (*templates*) de publicação, e também integração do sistema com APIs de plataformas de comunicação. Desta forma, o projeto é uma solução prática que resolve a ineficiência da operação e minimiza a propensão a erros, contribuindo para o aumento de produtividade de times de *marketing* e afins.

Palavras-chave: automação; redes sociais; gestão de conteúdo.

Abstract

Publishing content in a distributed way across various communication channels can be a complex and error-prone task, which makes managing these steps an obstacle to achieving efficient results. In this context, the aim of the application is to provide a means of integration with social networks, offering an environment that manages content through pre-registered structures and also automated publication. This is made possible by the registration of publication templates, as well as the system's integration with communication platform APIs. In this way, the project is a practical solution that solves operational inefficiency and minimizes the propensity for errors, helping to increase the productivity of marketing teams and the like.

Keywords: automation; social networks; content management.

Sumário

Introdução

No cenário digital atual, a presença corporativa em redes sociais é fundamental para fomentar o engajamento com clientes. A interação direta com os usuários dessas plataformas possibilita a construção de uma imagem e a identificação da companhia. Consequentemente, a conexão ocorre tanto pelo que a empresa oferece, quanto pela sua identidade institucional.

Contudo, a manutenção da presença em tantas plataformas oferecidas, replicando o mesmo conteúdo, é suscetível a erros e também ineficiente se executada manualmente. A publicação do mesmo conteúdo em plataformas distintas oferece inúmeras lacunas de padronização, dada a particularidade de cada uma delas. Tendo isso em vista, o projeto “Synk” foi criado para assegurar que o gerenciamento de conteúdo inter-plataformas ocorra de maneira coerente e eficiente.

O objetivo da plataforma é otimizar o trabalho manual por meio de integrações com diversas plataformas de conteúdo, ao qual o texto pré-cadastrado é modificado pelo usuário e publicado uma única vez, adaptando-se ao contexto de integrações ao qual esteja contido. É importante ressaltar que o Synk não se configura como uma plataforma CRM (Customer Relationship Management ou Gestão de Relacionamento com o Cliente), tendo como intuito primordial minimizar as lacunas de ineficiência inerentes à abordagem manual de publicação de conteúdo.

Descrição do Projeto

Synk é uma plataforma de gerenciamento de conteúdo, que integra funcionalidades de gestão de modelos (*templates*), gestão de publicações e também conexão com plataformas de comunicação como Discord e Telegram. A publicação nas plataformas, pode ser realizada de forma manual ou assíncrona.

O gerenciamento de modelos (*templates*) é feito manualmente, na formatação de texto que for desejada. O conteúdo será integrado automaticamente ao selecionar o registro na tela de publicações, permitindo a edição para aquele caso, mas mantendo a versão original criada anteriormente.

Para gerenciar as integrações, é indispensável o registro de perfis de integração, que atuam como *gateways* para as publicações. Cada perfil pode englobar múltiplas platafromas, servindo como um agrupador de canais de comunicação. Tais integrações armazenam as credenciais de acesso (*tokens* ou credenciais) esperadas pelas APIs de cada plataforma.

Nesse contexto, o gerenciamento de publicações é responsável por utilizar as configurações dos módulos supracitados. Ao selecionar o *template* desejado para publicação, o conteúdo pode ser livremente editado para adequar-se ao contexto do usuário. Em seguida, seleciona-se o perfil de integração, que — como citado anteriormente — agrega um conjunto de redes sociais com suas respectivas credenciais. Ao confirmar, esse conteúdo editado é assincronamente inserido em cada uma das integrações do conjunto selecionado, adaptando-se ao formato esperado por cada rede para a publicação.

Especificação Técnica

Componentes Principais

1. **Módulo de Perfil do Usuário**: responsável pelo cadastro do usuário e autenticação dos usuários.
2. **Módulo de Templates**: permite operações de CRUD (Criar, Ler, Atualizar, Excluir) para os templates de *posts*. Cada template é estritamente vinculado ao usuário que o criou.
3. **Módulo de Integrações**: permite operações de CRUD (Criar, Ler, Atualizar, Excluir) para os perfis de integração cadastrados. Cada perfil de integração é composto por uma ou mais integrações, nas quais são registradas — individualmente — as credenciais. Cada integração é estritamente vinculada ao usuário que o criou.
4. **Módulo de Publicação**: Utiliza um *template* e uma ou mais integrações ativas para realizar a postagem do conteúdo através das APIs correspondentes. Responsável por tratar as respostas (sucesso/erro) de cada API. Cada publicação é estritamente vinculada ao usuário que o criou.

Requisitos de Software

Requisitos Funcionais

Módulo 1: Gestão de Perfil do Usuário

- **RF-001**: O sistema deve permitir que um novo usuário se cadastre fornecendo Nome, E-mail e Senha.
- **RF-002**: O sistema deve permitir o login por meio de e-mail e senha.

Módulo 2: Gestão de Templates

- **RF-003**: O usuário autenticado deve poder criar um novo template, fornecendo obrigatoriamente um Nome para o template e o Conteúdo do post.
- **RF-004**: O sistema deve listar todos os templates criados pelo usuário.
- **RF-005**: O usuário deve poder editar o nome e o conteúdo de um template existente.

Módulo 3: Gestão de Integrações

- **RF-006**: O usuário deve poder adicionar uma nova integração, selecionando uma plataforma disponível (ex: Telegram, Discord) e fornecendo as credenciais de acesso necessárias (via OAuth ou chave de API).
- **RF-007**: O usuário deve poder atribuir um nome/apelido personalizado para cada integração cadastrada (ex: "Servidor Discord do Time Dev", "Grupo Telegram do Comercial").
- **RF-008**: O sistema deve listar todas as integrações configuradas pelo usuário, indicando a plataforma e o nome personalizado.
- **RF-009**: O usuário deve poder remover uma integração existente.

Módulo 4: Fluxo de Publicação

- **RF-010**: O usuário deve poder selecionar um de seus templates existentes para publicação.
- **RF-011**: O sistema deve permitir que o usuário edite livremente o conteúdo do *post* para aquela publicação específica, sem alterar o template original.
- **RF-012**: O usuário deve poder selecionar uma ou mais de suas integrações ativas como destino para a publicação.
- **RF-013**: O sistema deve enviar o conteúdo final para as APIs de cada uma das integrações de destino selecionadas.

Requisitos Não-Funcionais

- **RNF-001**: A aplicação web deve ser totalmente funcional nas duas últimas versões estáveis dos principais navegadores: Google Chrome, Mozilla Firefox, Safari e Microsoft Edge.
- **RNF-002**: A interface deve ser responsiva e se adaptar para garantir uma boa experiência de uso em dispositivos móveis (smartphones), tablets e desktops.
- **RNF-003**: O tratamento de dados pessoais dos usuários deve estar em total conformidade com a Lei Geral de Proteção de Dados (LGPD) do Brasil.
- **RNF-004**: A forma como a Synk interage com as APIs de outras plataformas deve respeitar rigorosamente os Termos de Serviço e as políticas de uso de cada uma delas para evitar o bloqueio da aplicação.
- **RNF-005**: O prazo de envio à API das integrações, de maneira assíncrona, deve ser até 1 hora após sua publicação dentro da plataforma.
- **RNF-006**: O agendamento da publicação das postagens nas plataformas é determinado pela fila de processamento assíncrono dos respectivos registros. Não há previsão para agendamentos em datas e horas específicos de tarefas relacionadas.
- **RNF-007**: O *delay* para carregamento das telas deve ser de, no máximo, 500ms.

Diagramas UML

Nos diagramas de Caso de Uso estão representados 3 cenários principais que estão presentes na aplicação. O primeiro deles, que pode ser encontrado na figura 1, é a representação da criação de um *template* abordando todas as etapas de inserção do conteúdo; o segundo, que pode ser visto na figura 2, representa os passos para criação de um perfil de integração, nas quais as credenciais de acesso às APIs serão vinculadas; o último, que está disponível na figura 3, é o processo de criar uma publicação, ao qual une o resultados dos passos do *template* e de integração.

Figura 1: Diagrama de caso de uso para criação de template

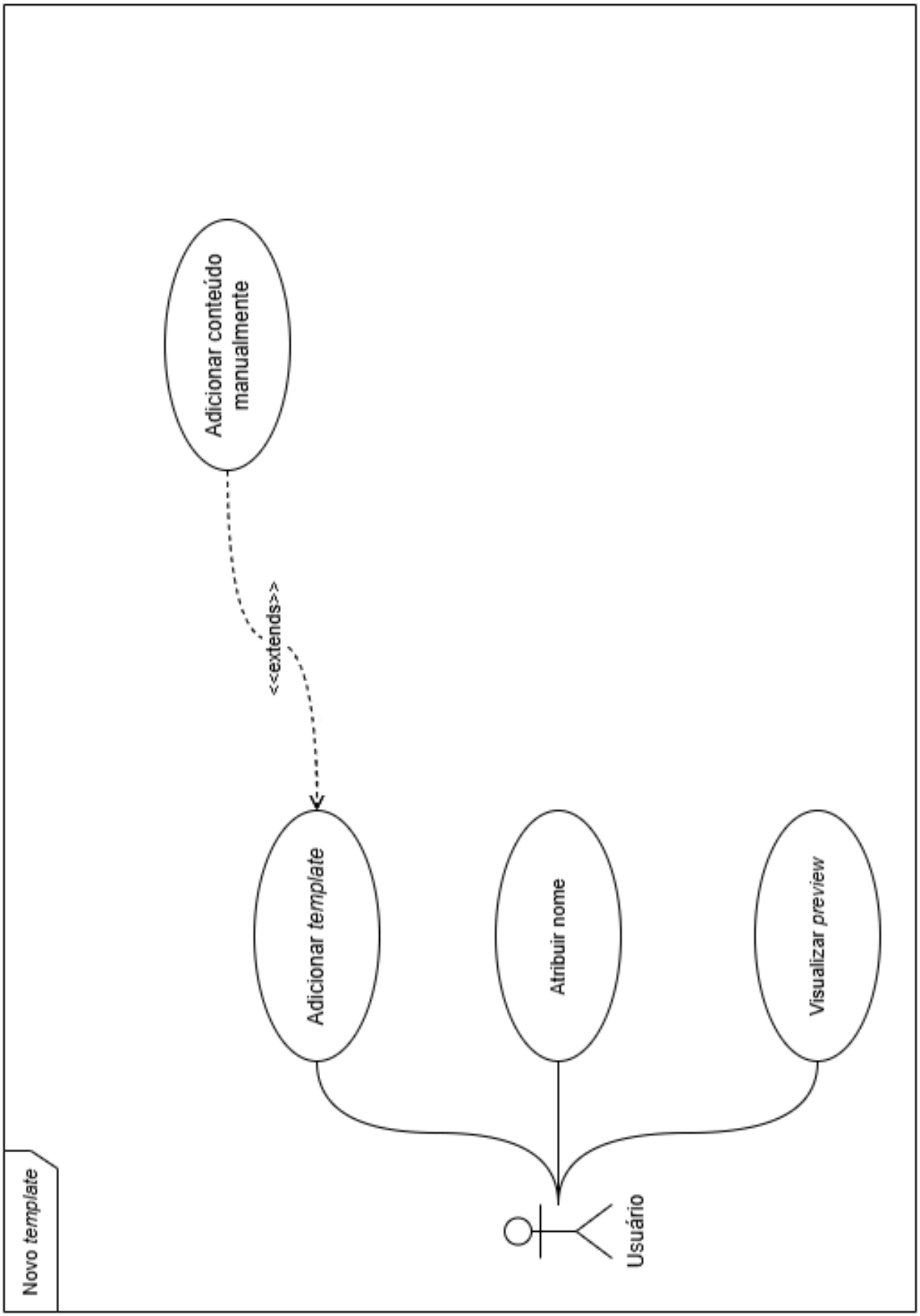


Figura 2: Diagrama de caso de uso para criação de perfil de integração

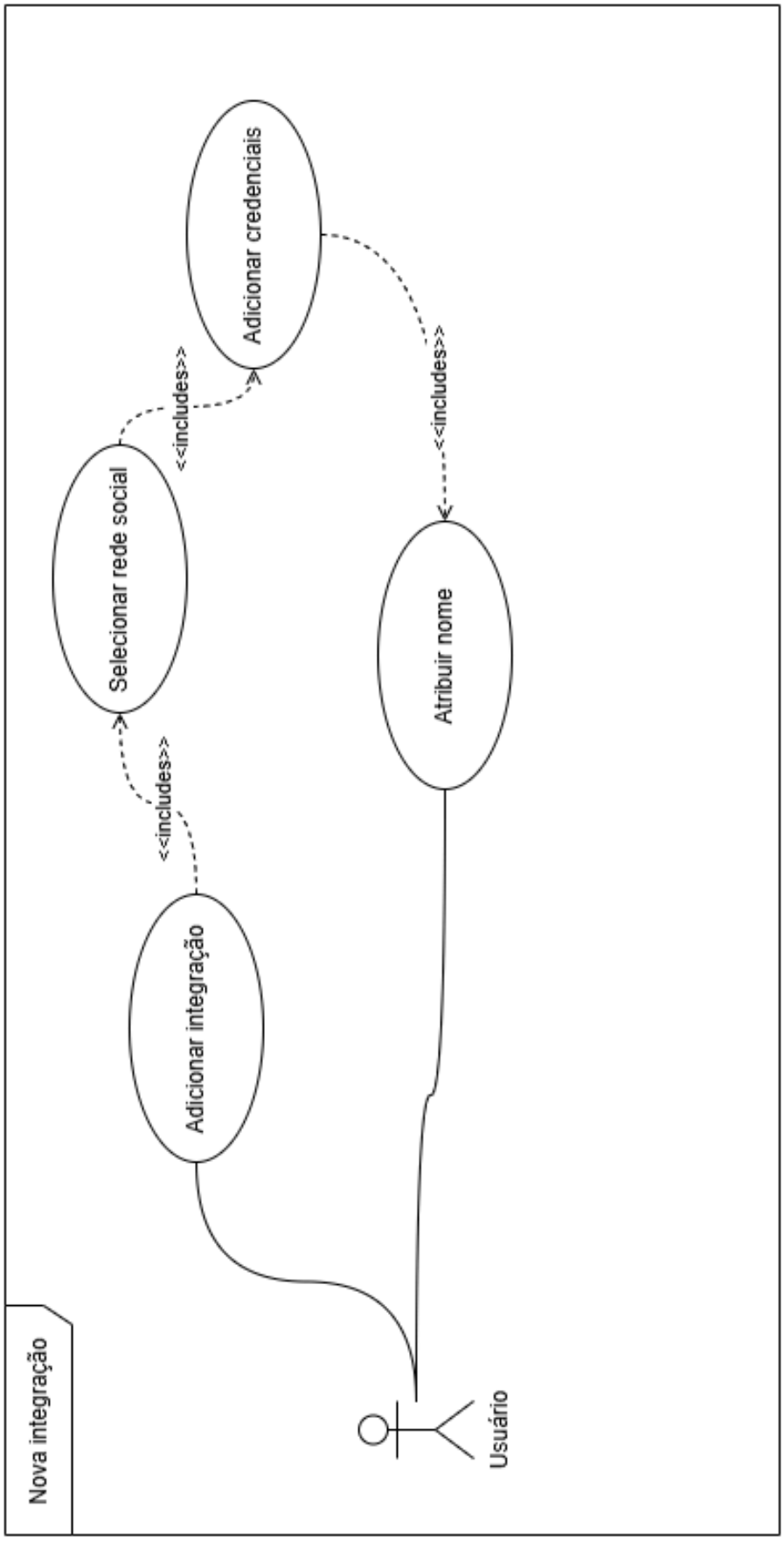
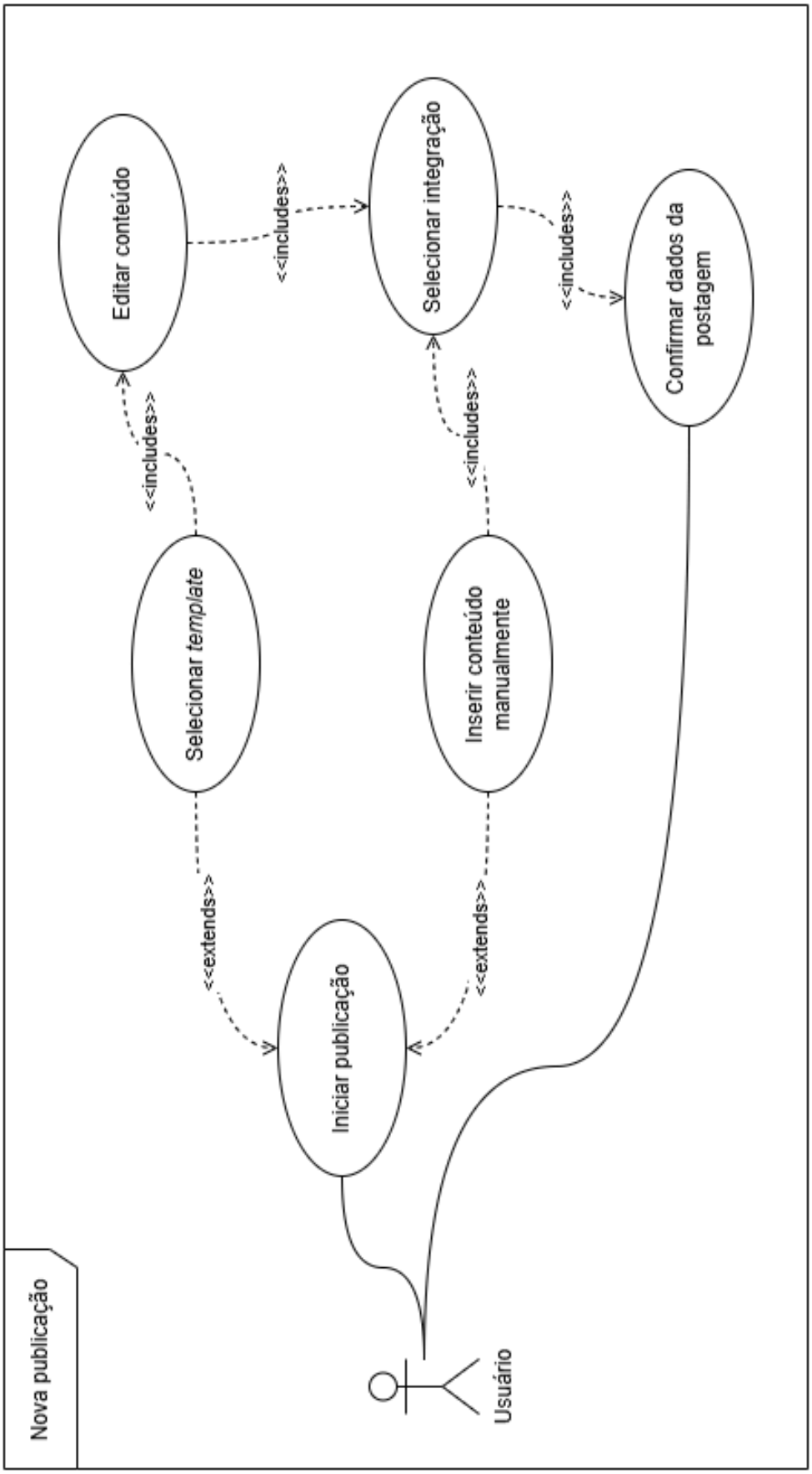


Figura 3: Diagrama de caso de uso de criação de publicação



Considerações de Design

A arquitetura está separada em apresentação e em processamento principal e paralelo. Cada um desses itens é separado em serviços distintos, que consomem o mesmo banco de dados. Importante ressaltar que não se trata de microsserviços, visto que a instância do banco de dados é compartilhado entre todos. Sendo assim, são apenas serviços separados para o acoplamento de responsabilidades.

Arquitetura

- **Frontend** para interação direta com os usuários por meio da web, realizando a comunicação diretamente com o serviço de Backend. Esse serviço usa a instância local do cache do navegador ao qual está sendo usado. É de grande relevância também destacar que a aplicação não possui suporte a interações *offline*, necessitando estritamente de uma conexão à internet para que funciona corretamente.
- **Backend** fica responsável pelo processamento principal da aplicação. Ele fica responsável por receber todas as chamadas do Frontend. A autenticação é feita por meio do serviço de Auth, que fica responsável pela autenticação do usuário. Há também à disponilidade uma instância de cache com foco em performance, além da interação direta com a instância de banco de dados.
- **Auth** é o serviço responsável por registrar o login dos usuários, interagindo tanto com o banco de dados, quanto com o serviço de cache para guardar a sessão.
- **Queuer** é o serviço de publicação que recebe as chamadas do Backend e serve o serviço Publisher.
- **Publisher** recebe as chamadas do Queuer relacionadas à publicação dos posts por meio das integrações devidas, que também interage com o banco de dados para busca de informações.

Modelagem C4

O diagrama na Figura 4 ilustra a separação inicial do sistema, compreendendo o núcleo do Synk e suas interações com meios externos. Primeiramente, há a interação com publicações de acesso público para importação de novos templates. Em segundo lugar, o sistema externo engloba um conjunto de APIs externas das plataformas de publicação de conteúdo (APIs do Discord e Telegram).

O diagrama na Figura 5 detalha o núcleo do Synk, explicitando as interações internas desse sistema. A interação direta do usuário inicia-se na aplicação web, que realiza todas as requisições para visualização e registro junto ao serviço de API Gateway. Este serviço interage diretamente com o Banco de Dados. O serviço principal também se comunica com o Serviço de Autenticação para validar os usuários ativos.

Para detalhar o API Gateway, onde a lógica de negócios é predominantemente armazenada, o diagrama da Figura 6 ilustra as interações. O processo inicia no Router, que recebe as requisições direcionadas ao API Gateway e valida a existência das rotas. Em seguida, consulta o autenticador para verificar se o usuário possui as credenciais necessárias. Após a validação, a comunicação é direcionada para cada um dos controllers da aplicação, que representam os módulos da plataforma: perfil do usuário, templates, integrações e publicações.

Figura 4: Primeiro nível do diagrama C4 (contexto do sistema)

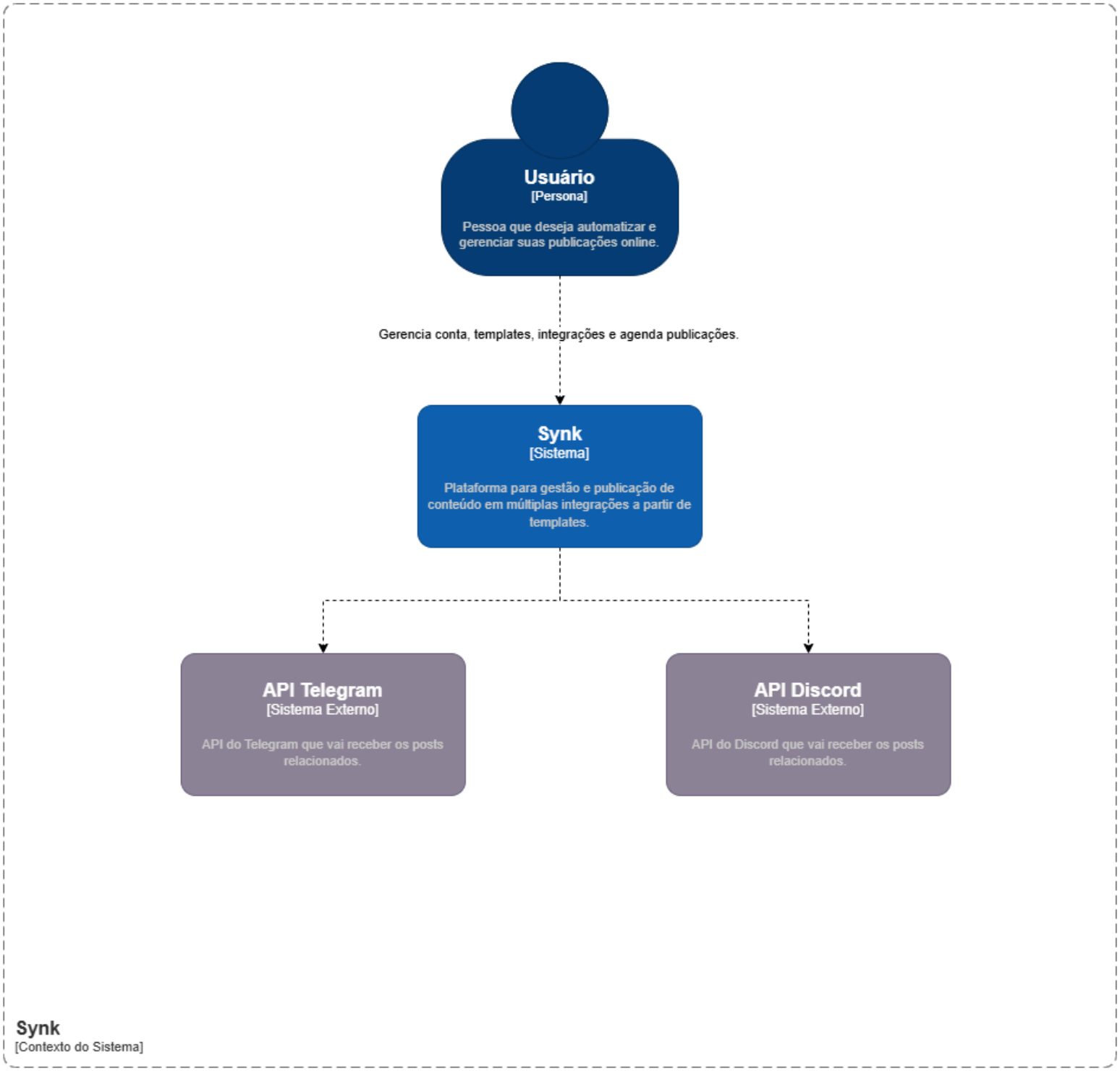


Figura 5: Segundo nível do diagrama C4 (contêineres do sistema)

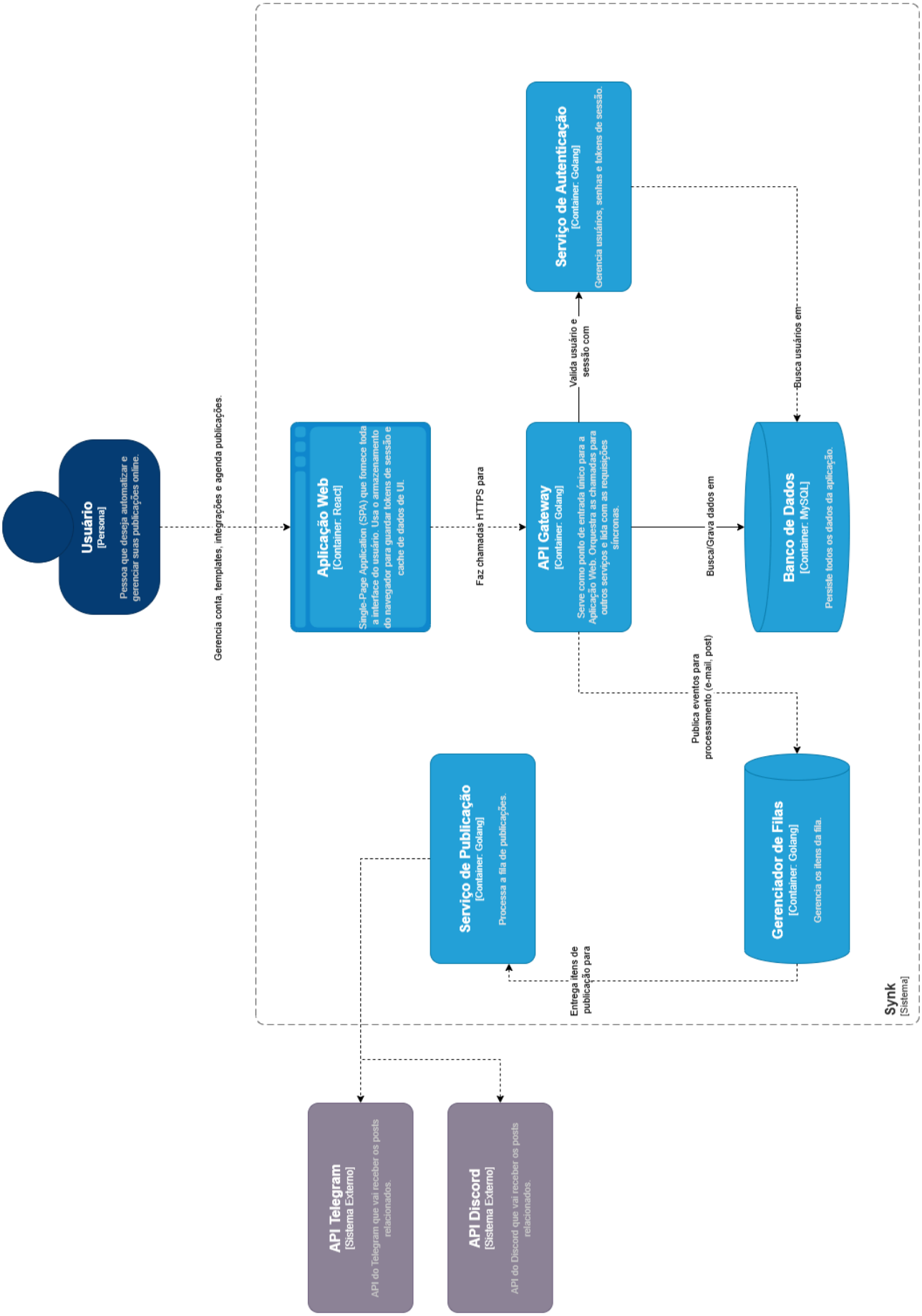
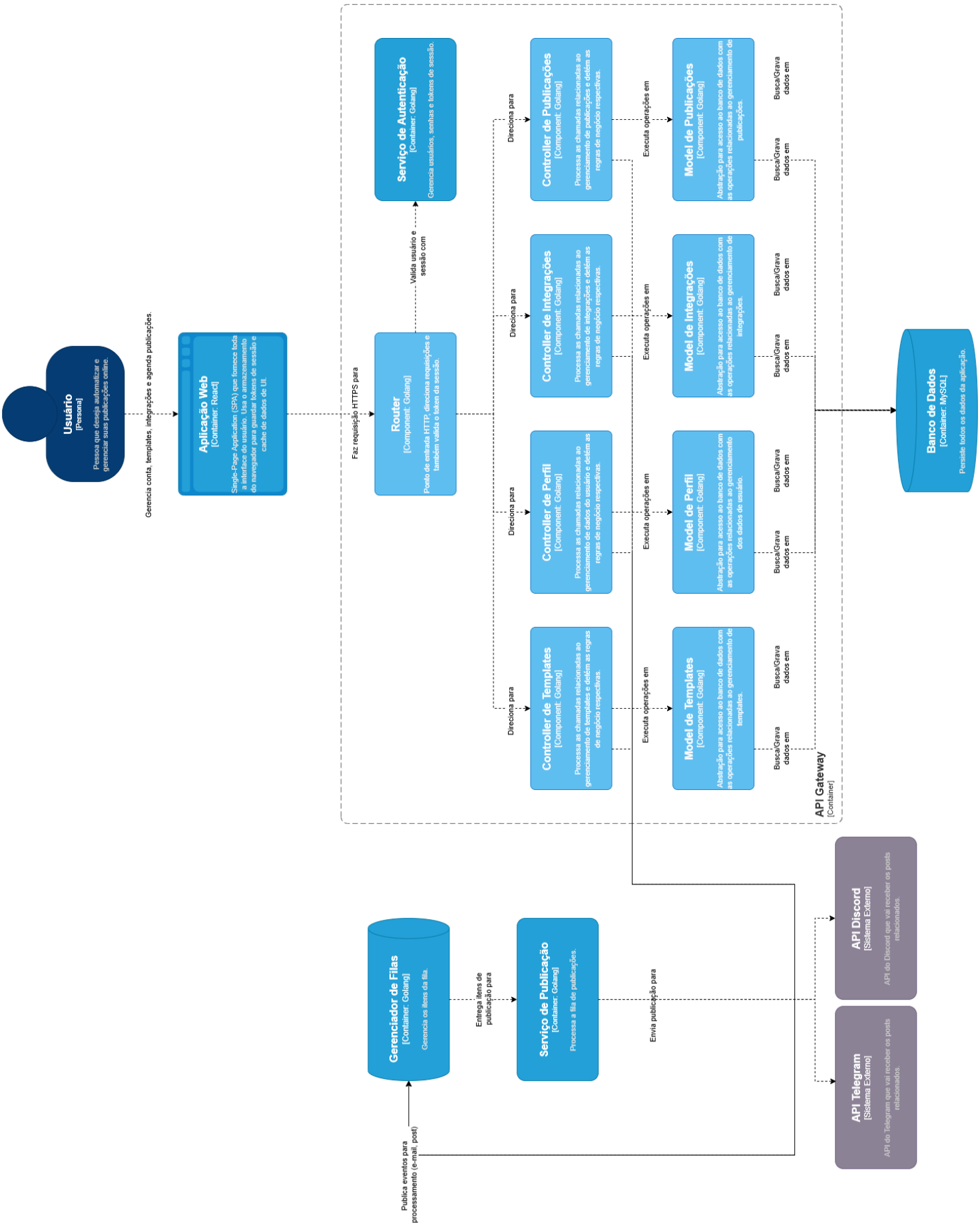


Figura 6: Terceiro nível do diagrama C4 (componentes do sistema)



Stack Tecnológica

A interface do frontend é construída utilizando React. Para a compilação do código TypeScript, que será empregado em conjunto com esta biblioteca, utiliza-se o Vite. Desta forma, busca-se a aplicação mais pura possível do React, sem o uso de frameworks, visando uma ferramenta robusta, leve e de alta performance para a interação com a interface. O principal foco dessa escolha é unir a possibilidade de uma interface responsiva com a performance das chamadas retornadas dos serviços para renderização praticamente instantânea.

Para o armazenamento em cache local de determinados dados, otimizando a usabilidade no navegador, emprega-se o recurso nativo da plataforma, o Local Storage. Isso permite uma interface simples, porém altamente eficiente, para a persistência de dados básicos e/ou extensos.

O banco de dados utilizado é o MySQL, uma ferramenta consolidada com vasta documentação disponível. Adicionalmente, a versão mais recente da ferramenta aprimorou o desempenho e introduziu novos recursos, tornando seu uso ainda mais vantajoso.

No que concerne aos serviços distribuídos da aplicação, tanto síncronos quanto assíncronos, emprega-se a linguagem Golang, sem a utilização de qualquer framework. A premissa é que essa interação seja a mais direta possível, utilizando as ferramentas inerentes à própria linguagem. O objetivo é empregar bibliotecas externas apenas quando estritamente necessário para problemas específicos, nos quais as tecnologias nativas não possam ser adequadamente adaptadas. Isto se deve principalmente à exigência da aplicação ser o mais performática possível. Por ser uma linguagem compilada, apenas o carregamento realmente necessário será realizado.

A infraestrutura de nuvem e o processo de deploy da aplicação estão hospedados em uma máquina virtual (VM) na Google Cloud Platform (GCP). Esta escolha é justificada pela flexibilidade e escalabilidade que o ambiente da GCP oferece.

Todo o código-fonte é versionado e hospedado no GitHub, que atua como repositório central. Os processos de Integração Contínua (CI) e Entrega Contínua (CD) são automatizados por meio do GitHub Actions, assegurando que cada atualização no código seja testada e implantada de forma padronizada e segura diretamente na VM, otimizando o fluxo de desenvolvimento e entrega.

O gerenciamento do projeto, incluindo o planejamento de tarefas, o acompanhamento do progresso e a organização de sprints, é conduzido por meio do GitHub Projects. Esta ferramenta é diretamente integrada ao repositório de código, o que facilita a vinculação de *issues*, *pull requests* e discussões às tarefas do projeto. Consequentemente, obtém-se uma visão clara e centralizada do andamento do desenvolvimento com este conjunto de ferramentas, facilitando o desenvolvimento e a evolução da aplicação.

Plano de Testes

O presente plano tem como objetivo formalizar e sistematizar o processo de verificação e validação da aplicação. A finalidade é assegurar que todos os requisitos funcionais e não-funcionais, especificados neste documento, sejam integralmente atendidos. Este processo visa garantir a qualidade, estabilidade e usabilidade do software, mitigando a ocorrência de defeitos em ambiente de produção e garantindo a conformidade com as diretrizes do projeto.

Escopo

O escopo dos testes abrange a totalidade dos módulos e funcionalidades desenvolvidas, bem como a aderência aos atributos de qualidade definidos.

- **Funcionalidades em Escopo:**
    - Módulo 1: Gestão de Perfil do Usuário.
    - Módulo 2: Gestão de Templates.

    - Módulo 3: Gestão de Integrações.
    - Módulo 4: Fluxo de Publicação.
- **Requisitos Não-Funcionais em Escopo:**
    - Compatibilidade, responsividade, desempenho, segurança e conformidade legal.
- **Fora do Escopo:**
    - Testes de estresse e carga com volume massivo de usuários simultâneos.
    - Testes de penetração conduzidos por agentes externos.
    - Análise de performance das APIs de plataformas de terceiros (o teste se limitará a validar a integração e o respeito aos seus termos de uso).

Tecnologias

A metodologia de testes de software foi segmentada para alinhar-se às melhores práticas de cada ecossistema tecnológico do projeto. No que concerne aos serviços de backend, desenvolvidos em Golang, optou-se pela utilização exclusiva do ferramental de testes nativo da própria linguagem. Essa escolha se fundamenta na integração direta com o código-fonte, na eficiência da execução de testes unitários e de integração sem a necessidade de dependências externas, e na aderência à filosofia de simplicidade e performance do Go. Em contrapartida, para a validação da interface do usuário (frontend), o framework Jest foi selecionado como ferramenta principal. A escolha do Jest justifica-se por sua abrangência no ecossistema JavaScript, oferecendo uma solução completa com corredor de testes (test runner), biblioteca de asserções e funcionalidades avançadas de simulação (mocking) e testes de snapshot, essenciais para garantir a estabilidade e a consistência visual dos componentes de interface.

Critérios de Avaliação

Para que a fase de testes seja considerada concluída e a aplicação aprovada, os seguintes critérios devem ser atendidos:

- **Critérios de Entrada:** 100% das funcionalidades descritas no escopo devem estar com o desenvolvimento concluído e implantadas no ambiente de homologação.
- **Critérios de Saída:**
    - 100% dos casos de teste planejados devem ter sido executados.
    - Inexistência de defeitos classificados como críticos (impeditivos de uso de uma funcionalidade principal) ou graves.
    - Todos os defeitos encontrados devem estar devidamente documentados, e direcionados para correção em versões futuras.

Considerações de Segurança

- **Hashing de Senhas e Dados Sensíveis**: As senhas dos usuários e outros dados sensíveis são armazenadas no banco de dados MySQL empregando algoritmos de hashing robustos e atualizados, como Bcrypt ou Argon2. É imperativo que senhas nunca sejam armazenadas em texto plano, assegurando a proteção das credenciais mesmo em cenários de violação do banco de dados. Isso pode ser extendido também a dados sensíveis, garantindo uma camada de segurança adicional para com o tratamento dos dados do usuário.

- **Gerenciamento de Sessão com JWT**: A autenticação de usuários é orquestrada por meio de JSON Web Tokens (JWT). Após a autenticação, o backend Golang emite um token assinado, transmitido ao frontend. Este token é mandatório para todas as requisições a endpoints protegidos. Os tokens possuem um período de

expiração conciso, minimizando a janela de vulnerabilidade em caso de comprometimento.

- **Tokens de Uso Único**: O link de verificação de e-mail incorpora um token seguro, de uso único e com tempo de expiração restrito, prevenindo ataques de replay ou uso indevido.
- **Criptografia em Trânsito**: Toda a comunicação entre o cliente (navegador), o frontend React e os serviços de backend em Golang é obrigatoriamente efetuada via HTTPS (TLS), garantindo a integridade e confidencialidade dos dados durante a transmissão.
- **Validação de Entradas no Backend**: Nenhuma entrada proveniente do cliente é considerada intrinsecamente confiável. Todos os serviços em Golang implementam validação e sanitização rigorosas de todas as entradas para mitigar ataques como:
    - **SQL Injection**: Exclusivamente através do uso de prepared statements e queries parametrizadas para interagir com o MySQL.
    - **Cross-Site Scripting (XSS)**: Assegurando que qualquer conteúdo gerado pelo usuário (nomes, conteúdo de templates) renderizado seja devidamente "escapado".
    - **Server-Side Request Forgery (SSRF)**: A funcionalidade de importação por URL apresenta um risco de SSRF. O backend valida a URL recebida, garantindo que aponte para endereços públicos e não para recursos da rede interna da GCP.

E, corroborando com o que foi dito, é importante também ressaltar que as considerações de segurança da aplicação foram concebidas como um pilar fundamental da arquitetura, adotando uma abordagem multicamada para proteger tanto os dados do usuário quanto a integridade do sistema. Em estrita conformidade com os princípios da Lei Geral de Proteção de Dados (LGPD), o tratamento de

credenciais de acesso recebe atenção prioritária. Para tanto, as senhas dos usuários são submetidas a algoritmos de hashing criptográfico assimétrico, com a adição de um salt individual para cada registro, medida que garante a confidencialidade das informações e inviabiliza a recuperação da senha original mesmo em caso de acesso indevido à base de dados. Complementarmente à proteção dos dados em repouso, foi implementado um sistema de trilhas de auditoria (audit logs) que registra cronologicamente as ações críticas realizadas pelos usuários, como tentativas de login, alterações de perfil e exclusão de dados. Este mecanismo é essencial não apenas para a rastreabilidade e a eventual análise forense em caso de incidentes, mas também como uma ferramenta proativa para a detecção de atividades anômalas, estabelecendo uma camada de segurança robusta que assegura a integridade da aplicação e a responsabilização (accountability) por suas operações.

Referências

REACT. React. Disponível em: https://react.dev/. Acesso em: 28 nov. 2025.

VITE. Vite. Disponível em: https://vitejs.dev/. Acesso em: 28 nov. 2025.

GOLANG. Go: The Go Programming Language. Disponível em: https://go.dev/. Acesso em: 28 nov. 2025.

MYSQL. MySQL. Disponível em: https://www.mysql.com/. Acesso em: 28 nov. 2025

GOOGLE CLOUD PLATFORM. Google Cloud Platform (GCP). Disponível em: https://cloud.google.com/. Acesso em: 28 nov. 2025.

GITHUB. GitHub. Disponível em: https://github.com/. Acesso em: 28 nov. 2025.

GITHUB. GitHub Actions. Disponível em: https://github.com/features/actions. Acesso em: 28 nov. 2025.

GITHUB. GitHub Projects. Disponível em: https://github.com/features/project-management. Acesso em: 28 nov. 2025.

C4 MODEL. C4 model. Disponível em: https://c4model.com/. Acesso em: 28 nov. 2025.