Documentação Arquitetural

SGP - Sistema de Gerência de Psicoterapia

Ester Holanda Ravette
Henrique Lima Pires
Jhordanna Gonçalves De Oliveira
José Mykael Alves Nogueira
Marcos Vitor Souza Freire

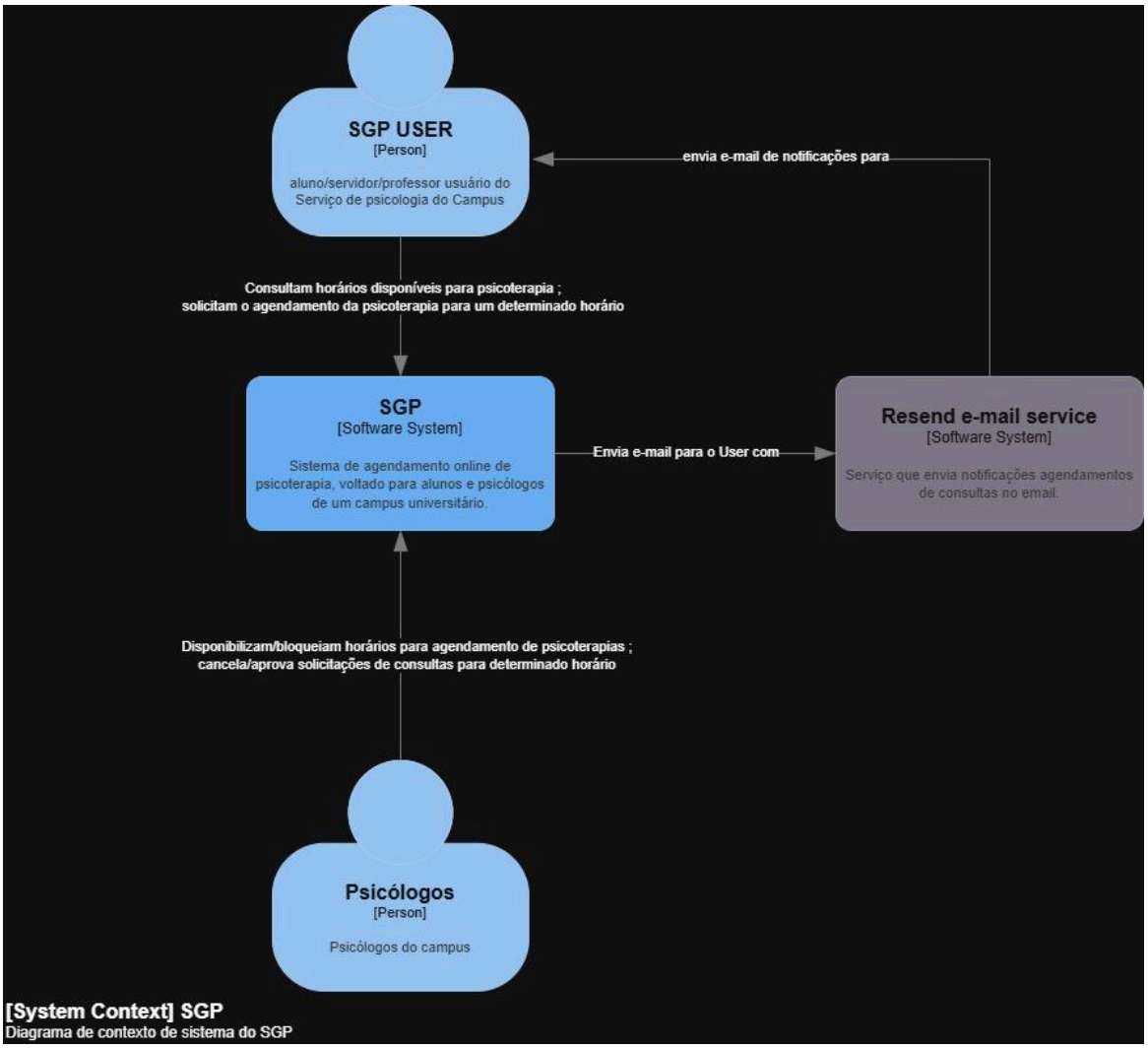
Quixadá - 2025

Sumário

Sistema de Gerência de Psicoterapia

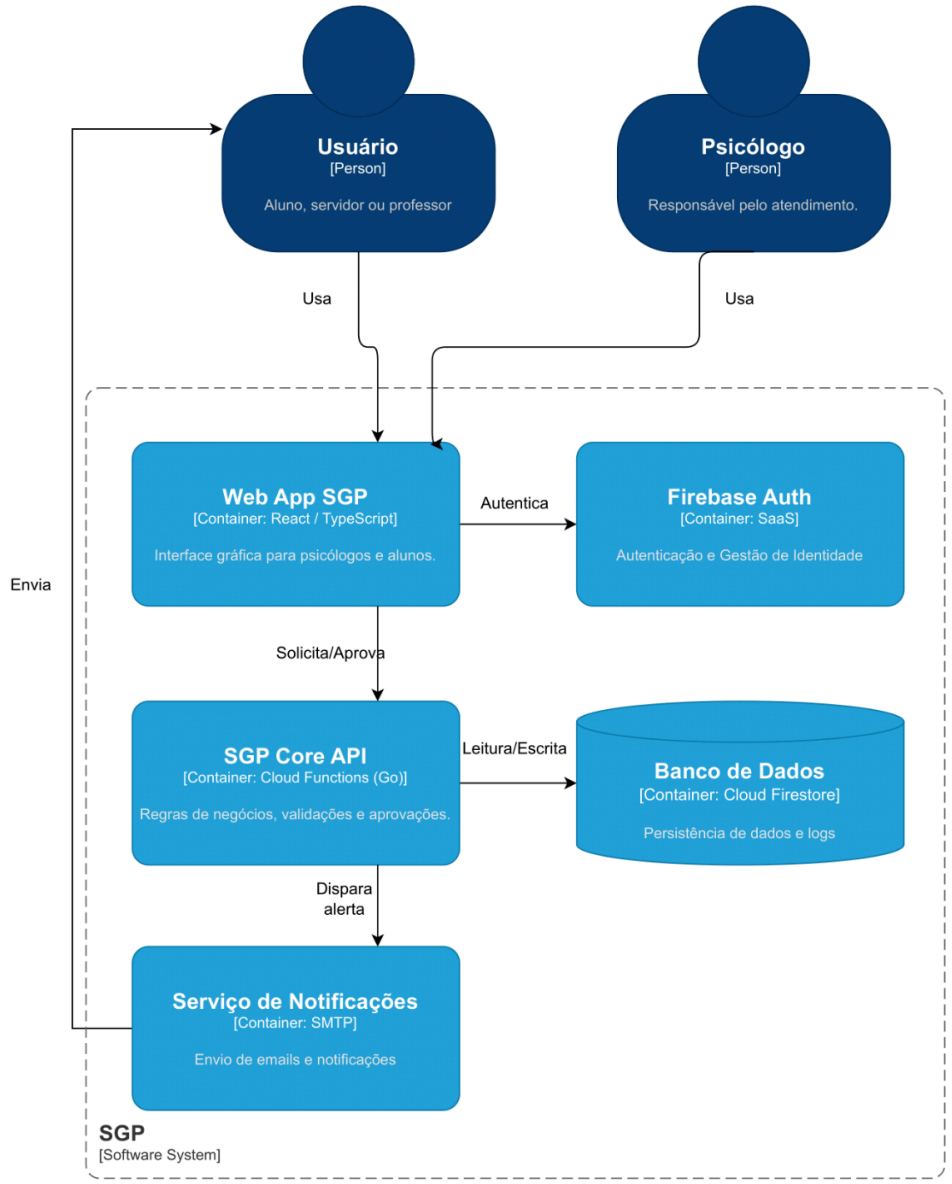
SGP

Visão de Contexto



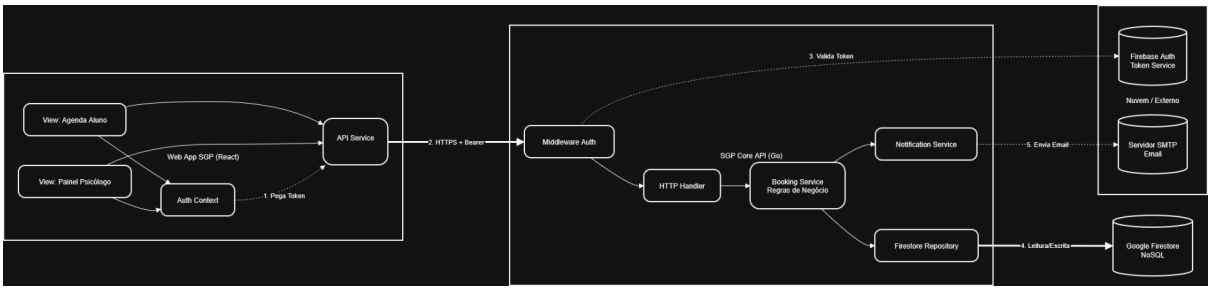
O **SGP** é o sistema principal que intermedia o agendamento de psicoterapia. Existem dois perfis de usuário: o **SGP USER** (Alunos/Servidores) que agenda, e os Psicólogos que gerenciam a agenda. O SGP envia e-mail para o User (aluno ou psicólogo) para confirmar, alertar ou notificar sobre mudanças (cancelamentos, aprovações, etc.)

Visão de Contêiner



Este diagrama define a topologia **Serverless** do sistema. O Web App SGP (React/TS) é a interface que os usuários acessam. Ele se comunica exclusivamente com a SGP Core API (Cloud Functions em Go), que representa a camada FaaS (Functions as a Service) do Backend. A API e o Web App utilizam os serviços BaaS (Firebase Auth e Cloud Firestore) para segurança e dados, demonstrando uma clara separação de responsabilidades entre a interface, a lógica e os serviços gerenciados.

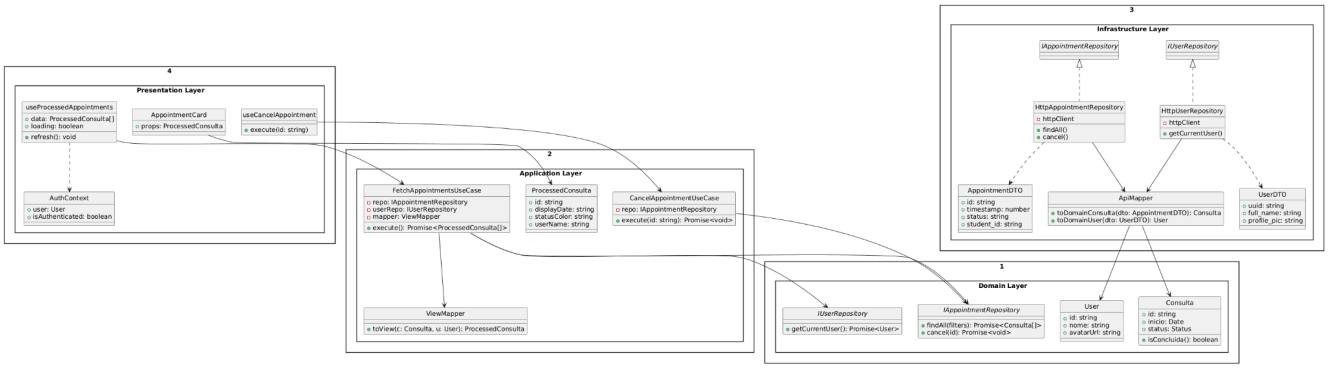
Visão de Componentes



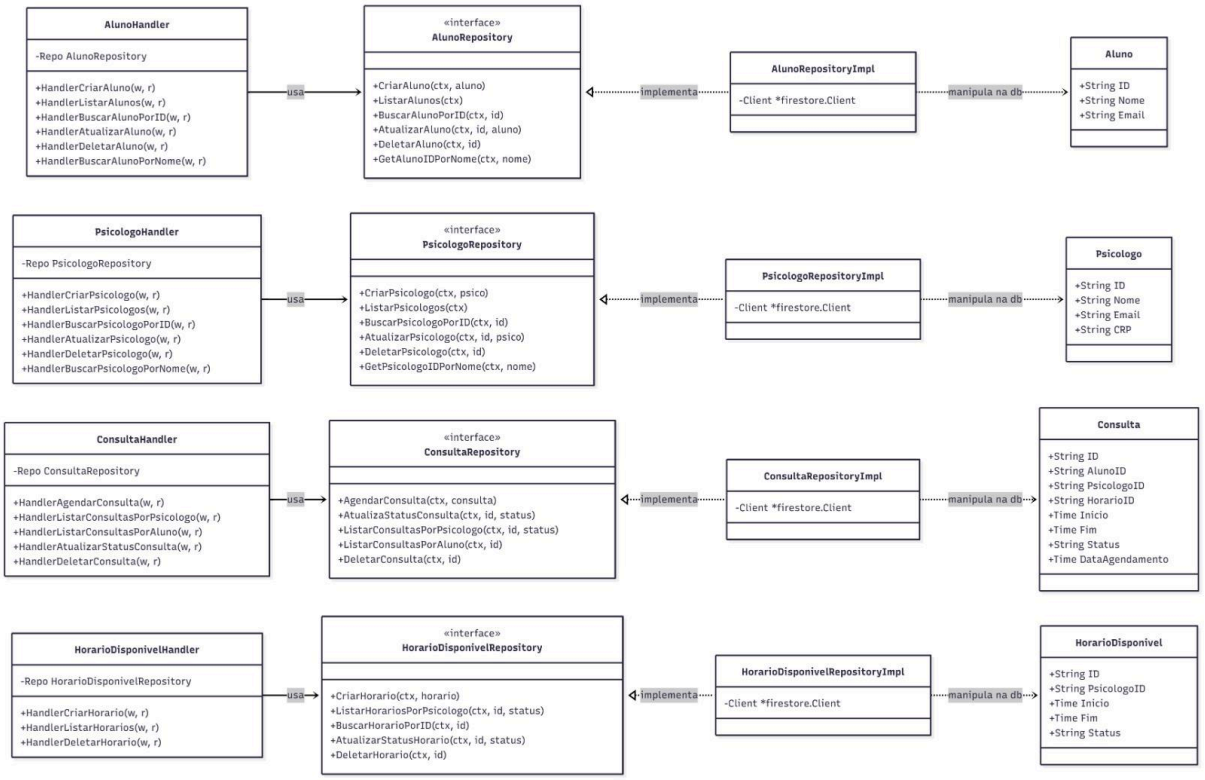
O fluxo de uma requisição é interceptado pelo Middleware Auth para validação do JWT (Segurança Transversal). Se aprovada, o HTTP Handler invoca o Booking Service, o componente central de Regras de Negócio. O Service utiliza o Firestore Repository (que implementa o Padrão Repository) para realizar transações atômicas no banco de dados e aciona o Notification Service para enviar e-mails de forma assíncrona, garantindo alta coesão e baixo acoplamento.

Visão de Código

- **Frontend**



- **Backend**



Mecanismos de Comunicação e Fluxo de Dados

- **Fluxo de Comunicação Síncrona (Agendamento/Aprovação)**

A comunicação entre o Web App SGP e o SGP Core API (Go) é baseada em HTTP e é segura por autenticação Bearer JWT:

1. **Pega Token**: O Auth Context (Frontend) obtém o Token JWT do Firebase.
2. **Requisição Segura**: O API Service (Frontend) envia a requisição HTTP com o Token JWT no cabeçalho.
3. **Validação de Token**: A requisição chega ao Middleware Auth (Backend), que valida o Token com o Firebase Auth Token Service.
4. **Processamento**: Se o token for válido, o HTTP Handler chama o Booking Service para executar a regra de negócio.

- **Fluxo de Comunicação Assíncrona (Notificação)**

As notificações são desacopladas da transação principal:

1. **Gatilho**: O Booking Service (após a transação de agendamento) chama o Notification Service.

2. **Ação**: O Notification Service envia a mensagem ao Servidor SMTP Email de forma não-bloqueante.

- ## Fluxo de Autenticação

Processo responsável por obter e validar as credenciais do usuário:

1. O usuário informa email e senha no Web App;
2. O Web App envia os dados para o Firebase Auth;
3. O Firebase Auth verifica as credenciais e retorna um Token JWT;
4. O Auth Context armazena o Token JWT localmente;
5. A partir desse momento, todas as requisições ao Backend incluem o token no cabeçalho HTTP.

Este fluxo é pré-condição para todos os demais.

- ## Fluxo de Cadastro

O cadastro utiliza diretamente o Firebase Auth e a API:

1. O usuário preenche email institucional, nome e senha no Web App;
2. O Web App envia os dados ao Firebase Auth para criação da conta;
3. O Firebase Auth retorna um identificador único (UID);
4. O Web App envia o UID à SGP Core API;
5. A API registra o usuário no Firestore com seu papel (aluno/servidor/psicólogo).

- ## Fluxo de Consulta ao Banco de Dados (Firestore)

Presente em praticamente todos os casos de uso:

1. O Web App envia requisição autenticada à API;
2. O Middleware Auth valida o Token JWT;
3. O Handler invoca o Service correspondente (ex.: Booking Service, Prontuário Service);
4. O Service chama o Firestore Repository para leitura/escrita;
5. O Firestore retorna dados ao Repository → Service → Handler → Web App.

Esse fluxo é totalmente síncrono.

- ## Fluxo de Cancelamento de Consulta

Uma extensão do fluxo principal de agendamento:

1. O aluno ou psicólogo envia solicitação de cancelamento;
2. O Middleware Auth valida a identidade do usuário;
3. O Booking Service verifica as regras de negócio (prazo, papel do usuário, horário);
4. O Service atualiza o status da consulta no Firestore;
5. O Notification Service é acionado e envia e-mail às partes envolvidas.

- ## Fluxo de Prontuário (Criação/Atualização)

Fluxo essencial para psicólogos:

1. O psicólogo acessa a tela de prontuário e envia o número de matrícula;
2. O Service consulta o Firestore para preencher automaticamente nome e email;
3. O psicólogo preenche os campos necessários;
4. A API valida permissões pelo Middleware Auth + regras de negócio;
5. O Prontuário Service salva ou atualiza o documento no Firestore;
6. O sistema retorna confirmação ao Web App.

- ## Fluxo de Bloqueio e Liberação de Datas

Somente psicólogos podem liberar ou bloquear horários:

1. O psicólogo seleciona data e horário no Web App;
2. O Web App envia requisição autenticada à API;
3. O Booking Service valida regras de negócio (RN004 – horário de funcionamento);
4. O Service grava a ação no Firestore;
5. O Web App atualiza o calendário automaticamente.

- ## Fluxo de Inscrição em Atendimentos Coletivos

1. O aluno visualiza os atendimentos disponíveis (Firestore → Web App);
2. Seleciona um atendimento público ou um convite privado;
3. O Web App envia a requisição autenticada para a API;
4. O Service verifica vagas, permissões e regras;
5. O aluno é inscrito no Firestore;
6. O Notification Service envia o e-mail de confirmação.

- ## Fluxo de Agendamentos Individuais

1. O agendamento individual utiliza a API e o Firestore para registrar a solicitação:
2. O aluno visualiza os horários disponíveis no Web App;
3. Seleciona a data e o horário desejados;
4. O Web App envia a solicitação autenticada para a SGP Core API;
5. O Middleware Auth valida o Token JWT com o Firebase Auth;

6. O Booking Service verifica disponibilidade, conflitos e regras de negócio;
7. O Booking Service registra a solicitação no Firestore via Firestore Repository;
8. O horário é marcado como pendente para evitar novos agendamentos;
9. O Notification Service envia um e-mail ao psicólogo notificando a nova solicitação.

- ### Fluxo de Aprovação de Agendamentos

A aprovação utiliza a API para validar permissões e atualizar o status da consulta:

1. O psicólogo visualiza no Web App as solicitações pendentes;
2. Seleciona a solicitação que deseja aprovar;
3. O Web App envia a aprovação autenticada para a SGP Core API;
4. O Middleware Auth valida o Token JWT e confirma o papel de psicólogo;
5. O Booking Service verifica regras de negócio e permissões;
6. O Booking Service atualiza o status da consulta no Firestore para “confirmada”;
7. O horário é bloqueado definitivamente na agenda;
8. O Notification Service envia um e-mail ao aluno informando a aprovação.

Decisões Arquiteturais

- ### Serverless

- A escolha pelo estilo Serverless é a **decisão de infraestrutura central**. Essa abordagem é implementada através da combinação de FaaS (Functions as a Service, com o Backend em Go) e BaaS (Backend as a Service, com o uso de serviços como Firebase Auth e Firestore para persistência e autenticação). Essa estratégia é crucial para cumprir o RNF03 (Disponibilidade), pois oferece alta disponibilidade e escalabilidade automática em todas as camadas (código, banco de dados e autenticação), sem a necessidade de gerenciar servidores.

  Requisitos atendidos:

- **RNF003 – Disponibilidade:** serviços serverless possuem redundância nativa e SLA elevado.
- **RNF001 – Usuários simultâneos:** escalabilidade automática permite alta taxa de requisições simultâneas.
- **RNF002 – Tempo de Resposta:** Cloud Functions executam sob demanda com baixa latência.

- **RF004–RF007 – Agendamentos:** garante que operações críticas ocorram mesmo em horários de pico.

.

● Clean Architecture

A organização lógica do Backend é estritamente guiada pela Clean Architecture. O objetivo principal é a separação de preocupações, isolando as regras de negócio centrais (Services) de detalhes técnicos como o banco de dados (Repository) ou o protocolo HTTP (Handlers). Essa independência é fundamental para garantir a alta testabilidade e a manutenibilidade do core da aplicação, prevenindo que alterações na infraestrutura afetem a lógica de negócio do agendamento.

- **RN006 – Gerenciamento de usuários e permissões:** regras de acesso ficam centralizadas e independentes da infraestrutura.
- **RN001 – Privacidade dos registros:** facilita aplicar políticas de acesso restrito ao prontuário e consultas.
- **RF011–RF012 – Prontuário e Anotações:** regras de negócio ficam encapsuladas em serviços testáveis.
- **RNF005 – Acessibilidade:** a separação entre frontend e backend evita que ajustes de UI impactem regras de negócio.
- **Manutenibilidade geral** (tacitamente exigida pelo tamanho do sistema).

● Tecnologia Backend: Go (Golang)

O Golang foi selecionado para o desenvolvimento do Backend devido às suas características de performance e concorrência. Go é altamente eficiente para lidar com múltiplas requisições simultâneas de agendamento, utilizando goroutines para gerenciar a concorrência de forma leve e rápida.

Requisitos atendidos:

- **RNF002 – Tempo de Resposta:** Go mantém baixa latência mesmo sob carga.
- **RNF001 – Usuários simultâneos:** goroutines permitem atendimento eficiente de múltiplas requisições.
- **RF004–RF007 – Agendamentos e aprovações:** requerem validar horários, carregar dados e gravar transações rapidamente.

- **RN003 – Agendamento e gerenciamento de consultas:** execução rápida de regras complexas garante fluidez.

● Tecnologia Frontend: React e TypeScript

A interface do usuário é construída com React e TypeScript. O React foi escolhido pela componentização, facilitando a construção de interfaces complexas e reutilizáveis (como o calendário de agendamento e os cards de consulta). O TypeScript é essencial para adicionar tipagem estática ao Frontend, o que reduz erros em tempo de execução, melhora a segurança e a manutenibilidade do código. Essa combinação permite o SGP ter uma interface acessível e compatível com diversos dispositivos.

Essa escolha atende:

- **RNF004 – Suporte Multiplataforma** (desktop, mobile e diferentes navegadores)
- **RNF005 – Acessibilidade** (leitores de tela, contraste, texto alternativo, navegação acessível)

● Firebase Auth

O Firebase Auth é o serviço responsável pela autenticação e autorização dos usuários no SGP. Ele fornece criação de contas, login e emissão de Tokens JWT utilizados em todas as requisições autenticadas. A integração com o Middleware Auth garante que apenas alunos, servidores e psicólogos devidamente autenticados possam acessar o sistema.

Essa decisão atende:

- **RF01 – Cadastro** (criação segura de contas)
- **RF02 – Login** (validação de credenciais)
- **RN006 – Gerenciamento de Usuários e Permissões**
- **RNF003 – Disponibilidade** (infraestrutura global resiliente)
- **RNF005 – Acessibilidade** (fluxos compatíveis com WCAG (Web Content Acessibilitu Guidelines) quando combinados ao frontend)

● Banco de Dados: Google Cloud Firestore

O Cloud Firestore é o banco de dados NoSQL utilizado para armazenar todas as informações essenciais do sistema, como usuários, consultas, prontuários, atendimentos coletivos e registros de bloqueio/liberação de datas. O modelo orientado a documentos oferece alta flexibilidade e desempenho, além de suportar escalabilidade automática sem necessidade de manutenção de servidores.

Essa escolha atende:

- **RNF001 – Usuários simultâneos** (operações concorrentes em grande volume)
- **RNF002 – Tempo de resposta do sistema** (baixa latência e indexação automática)
- **RF004–RF007 – Agendamentos**
- **RF011–RF016 – Prontuários e registros de atendimento**

A integração com o Firestore Repository dentro da Clean Architecture reforça o desacoplamento entre domínio e infraestrutura. Também permite manter consistência e auditoria sem a necessidade de gerenciar servidores.

- ### Serviço de Notificações (SMTP)

O envio de mensagens é realizado por meio de um servidor SMTP. O Notification Service opera de forma assíncrona, garantindo que o usuário receba notificações sem impactar o tempo de resposta das requisições principais. As notificações contemplam confirmações de agendamento, cancelamentos, aprovações, convites e lembretes automáticos.

Essa decisão atende:

- **RF020 – Notificações de Consultas**
- **RF004–RF007 – Fluxos de agendamento**
- **RN003 – Regras de agendamento e comunicação**
- **RNF002 – Tempo de resposta** (como o envio é assíncrono, ele não impacta o tempo de resposta da API, garantindo o cumprimento desse requisito)

Relacionamento entre Arquitetura e Regras/Requisitos

A arquitetura proposta foi definida para atender diretamente às necessidades funcionais e não funcionais do SGP, conforme os requisitos:

Requisitos Funcionais (RF) atendidos pela Arquitetura

- **RF01–RF03**: Autenticação via Firebase Auth + Middleware JWT
- **RF004–RF007**: Agendamentos implementados no Booking Service
- **RF011–RF012**: Prontuário e anotações via Firestore + controle de acesso
- **RF013–RF015**: Cancelamento, bloqueio e liberação de datas

- **RF020**: Notificações/lembranças via SMTP

Requisitos Não Funcionais (RNF) atendidos

- **RNF001**: Escalabilidade automática (Firestore + Cloud Functions)
- **RNF002**: Baixa latência (Go + Firestore)
- **RNF003**: Alta disponibilidade (arquitetura serverless)
- **RNF004**: Responsividade (React)

- **RNF005**: Acessibilidade (padrões WCAG 2.1 AA)

Regras de Negócio (RN) suportadas

- **RN001**: Privacidade — middleware, claims e controle de acesso
- **RN003**: Agendamento — Booking Service e Firestore
- **RN004–RN005**: Validação de datas e prazos
- **RN006**: Papéis e permissões (via JWT + claims)