Relatório - Mind System

Equipe

Aluno	GitHub	Cargo
Gustavo Rodrigues Ribeiro	GustavooRibas	Arquiteto de Software (AS)
Murillo Gordo de Andrade	murilloandrade	Testador (TT)
Davi de Souza Fernandes	sfDavi	Desenvolvedor Backend (DB) e Desenvolvedor Frontend (DF)
Felipe Alves Leão de Araújo	FelipeAlvesLeao	Analista de Requisitos (AR)

1. Arquitetura Lógica (parcial ou a priori)

1.1. Camadas Lógicas

O sistema adota uma **arquitetura em quatro camadas lógicas**, conforme o modelo proposto por Bezerra (2015):

Camada	Responsabilidade	Dependências
Apresentação (Interface)	Responsável pela interação com o usuário, exibição de informações e captura de dados (login, questionários, consultas). Implementa páginas web e telas responsivas.	Depende da Camada de Aplicação para requisitar e enviar dados.
Aplicação (Serviços e Controle)	Orquestra os fluxos do sistema (login, responder questionário, agendamento, geração de laudos). Controla regras de navegação, autenticação e comunicação entre apresentação e domínio.	Depende da Camada de Domínio para aplicar as regras de negócio.
Domínio (Negócio)	Contém as regras de negócio e entidades centrais como **Aluno**, **Psicólogo**, **Administrador**, **Questionário**, **Laudo**, **Consulta** e **PesquisaFeedback**.	Depende da camada de infraestrutura para persistência e serviços técnicos.
Infraestrutura (Serviços Técnicos)	Fornece suporte técnico como persistência, autenticação (via **Autenticacao** e **API_Google**), geração de relatórios (via **GeradorPDF**) e integração externa.	Suporta todas as demais camadas.

Essa separação favorece **baixo acoplamento** e **alta coesão**, permitindo evolução modular do sistema.

Princípio aplicado: camadas superiores dependem das inferiores, mas não o contrário (princípio de dependência unidirecional).

1.2. Subsistemas / Módulos / Estereótipos

Subsistema / Módulo	Serviços providos	Dependências
Módulo de Autenticação (`Usuario`, `Aluno`, `Psicologo`, `Administrador`, `Autenticacao`)	Login, cadastro, login social via Google.	Depende da **Camada de Infraestrutura** para validação e tokens, além da Interface (frontend).
Módulo de Questionário (`Questionario`, `Pergunta`, `Resposta`)	Criação, ativação, cálculo de resultados e controle de perguntas.	Depende de **Pergunta** e **Resposta** (Domínio), aplicação e banco de dados.
Módulo de Consultas (`Consulta`, `API_Google`)	Agendamento, cancelamento e execução de consultas com integração ao Google Meet.	Depende de **API_Google** (Infraestrutura).
Módulo de Laudos (`Laudo`, `GeradorPDF`)	Geração e envio de relatórios psicopedagógicos em PDF.	Depende de **GeradorPDF** e **Aluno/Psicólogo**, ou seja, domínio e infraestrutura.
Módulo de Feedback (`PesquisaFeedback`)	Criação e análise de pesquisas de satisfação.	Depende de **Administrador** e **Pergunta** (Domínio).
Módulo Administrativo	Cadastro e gestão de psicólogos, questionários e pesquisas.	Depende da **Camada de Aplicação**.

Cada subsistema segue o estereótipo `<<subsystem>>` e interage via interfaces bem definidas (por exemplo, aplicação → domínio via controladores).

1.3. Classes de Domínio

Principais classes e responsabilidades, conforme o diagrama:

Classe	Responsabilidade	Dependências
Usuario	Classe base que representa todos os perfis do sistema.	Superclasse de **Aluno**, **Psicologo**, **Administrador**.
Aluno	Realiza questionários, agenda consultas e acessa laudos.	Depende de **Questionario**, **Consulta**, **Laudo**, **PesquisaFeedback**.
Psicologo	Conduz consultas e gera laudos.	Depende de **Laudo** e **Consulta**.
Administrador	Gerencia cadastros e questionários.	Depende de **Questionario**, **PesquisaFeedback**, **Psicologo**.
Questionario	Conjunto de perguntas avaliativas.	Depende de **Pergunta**.
Pergunta	Define os itens de questionários e pesquisas.	Dependência de **Resposta**.

Classe	Responsabilidade	Dependências
Resposta	Armazena as respostas do aluno.	Dependente de **Pergunta** e **Aluno**.
Consulta	Representa sessões entre aluno e psicólogo.	Integração com **API_Google**.
Laudo	Resultado da avaliação vocacional.	Usa **GeradorPDF**.
Autenticacao	Responsável pelo login e tokens.	Depende de **Usuario** e **API_Google**.
API_Google	Integração com Google OAuth e Google Meet.	Fornece serviços à **Consulta** e **Autenticacao**.
GeradorPDF	Gera relatórios e laudos em PDF.	Fornece serviço à **Laudo**.

- Todas as classes de domínio apresentam alta coesão (cada uma tem papel claro) e baixo acoplamento (interagem por interfaces).

---

2. Arquitetura Física (parcial ou a priori)

2.1. Camadas Físicas

O sistema seguirá a **arquitetura cliente-servidor em três camadas** (three-tier):

1. **Camada de Apresentação (Frontend)**
    - Executada no navegador do usuário (cliente web).
    - Responsável pela interface gráfica.
2. **Camada de Aplicação (Backend)**
    - Executada em um servidor web.
    - Processa requisições, aplica regras de negócio e gera respostas.
3. **Camada de Dados (Database Server)**
    - Responsável pelo armazenamento persistente das informações.

2.2. Plataformas e Tecnologias

Componente	Função	Tecnologia sugerida
Cliente (Frontend)	Interface com o usuário	React.js / Next.js / Bootstrap / HTML5
Servidor de Aplicação (Backend)	Lógica de negócio e APIs REST	Node.js com Express ou Java Spring Boot
Banco de Dados	Persistência de entidades	PostgreSQL / MongoDB

Componente	Função	Tecnologia sugerida
Middleware (Autenticação e Integrações)	Login via OAuth, geração de PDF, integração com Google Meet	Google API, JWT, PDFKit / iText
Servidor de Arquivos (opcional)	Armazenamento de laudos e PDFs	AWS S3 / Firebase Storage

2.3. Componentes e Tecnologias

Camada	Componentes Principais	Tecnologias aplicadas
Apresentação (Frontend)	Módulos de Login, Dashboard, Questionários, Agenda, Feedback, ou seja, Interface Web, Formulários e Dashboards	React + Axios + HTML/CSS
Aplicação (Backend)	Controladores REST, Serviços de Autenticação, Questionário, Consulta, Laudo, Feedback	Node.js + Express + JWT
Domínio	Entidades e Regras de Negócio (Usuarios, Questionarios, Perguntas, Respostas, Laudos, Consultas)	ORM (Sequelize / Hibernate)
Infraestrutura	Banco de Dados, APIs externas, Gerador de PDF	Google API, PDFKit, PostgreSQL

3. Padrões de Projeto

Padrão	Descrição	Vantagem
Façade	Cria uma interface de alto nível que simplifica a interação entre módulos complexos (ex.: entre Camada de Aplicação e Infraestrutura).	Reduz o acoplamento e simplifica o uso de serviços externos (como API Google e GeradorPDF).
Factory Method	Centraliza a criação de objetos complexos, como **Questionario** e **Laudo**, delegando às subclasses a definição concreta.	Facilita manutenção e extensão quando há múltiplos tipos de questionários ou laudos.
Pub/Sub	Pode ser aplicado para notificar alunos e psicólogos sobre mudanças em consultas ou novos laudos.	Permite comunicação reativa e desacoplada entre usuários e sistema (RabbitMQ).