Subaru

KIOKU
Documento de Arquitetura de Software

Versão 1.0

Índice Analítico

Documento de Arquitetura de Software

1. Introdução

O objetivo desse documento é passar uma visão geral da arquitetura do sistema do **Kioku** com a finalidade de estabelecer o contexto, o escopo e a estrutura deste documento.

1.1 Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema. O documento irá adotar uma estrutura baseada na visão "4+1" de modelo de arquitetura, conforme (Kruchten, 1994).

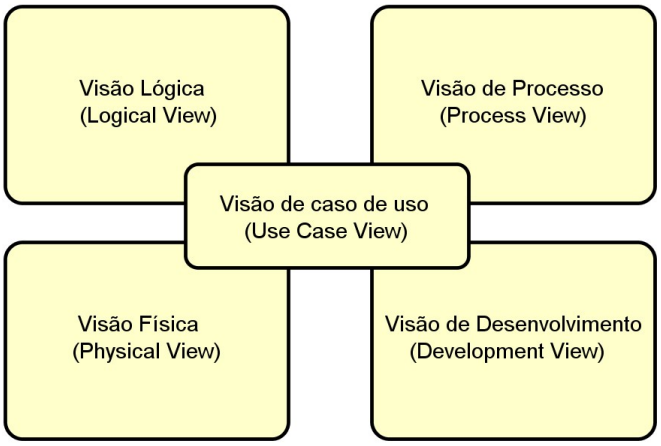


Figura 1 - Visões 4+1 da arquitetura de software (Kruchten, 1994)

1.2 Escopo

Este documento aplica-se exclusivamente ao sistema Kioku - Plataforma de Repetição Espaçada. O Kioku é um *software* de estudo e memorização que utiliza o método de Repetição Espaçada (SRS) através de *flashcards*, visando oferecer uma interface mais amigável e acessível em comparação com ferramentas de mercado.

A arquitetura descrita engloba todas as camadas do sistema, desde o Cliente (Web e Mobile) até o Servidor de Aplicação e a Persistência de Dados. O escopo do DAS não se aprofunda na lógica de *software* de terceiros.

O sistema Kioku deve ser compatível com as plataformas Web e Mobile (Android/IOS), conforme o Requisito Não Funcional RNF18.

1.3 Definições, Acrônimos e Abreviações

Acrônimo/Abreviação	Definição
AOO	Arquitetura Orientada a Objetos
API	*Application Programming Interface* (Interface de Programação de Aplicações)

Acrônimo/Abreviação	Definição
AWS	*Amazon Web Services* (Plataforma de serviços em nuvem para armazenamento e sincronização).
DAS	Documento de Arquitetura de Software
Deck	Um conjunto de *flashcards* agrupados por tema.
Flashcard	Um cartão de estudo com frente e verso, podendo conter texto, imagem ou áudio.
RNF	Requisito Não Funcional
SRS	*Spaced Repetition Scheduler* (Algoritmo de agendamento de revisões).
UML	*Unified Modeling Language* (Linguagem Unificada de Modelagem)

1.4 Referências

Os seguintes documentos são a base para a definição desta Arquitetura de Software:

1. **DRS** - Documento de Requisitos de Software (Kioku) - Versão 1.0 (Data: 03/09/2025). *Fonte primária para Requisitos Funcionais e Não Funcionais.*
2. **DV -** Documento de Visão (Kioku) - Versão 1.0 (Data: 03/09/2025). *Fonte para o objetivo, motivação e visão geral do produto.*
3. **NSA** - Notas sobre Arquitetura de Software. *Referência conceitual sobre estilos e princípios arquiteturais.*
4. **PPA -** U5-6 - Padrões de Projeto e Arquitetura. *Referência conceitual sobre Design Patterns (GoF) e Arquitetura Hexagonal.*

2. Requisitos e Restrições da Arquitetura

Esta seção detalha os requisitos e objetivos de software que têm um impacto direto sobre a arquitetura, bem como as restrições aplicáveis.

Requisito	*Solução*
Linguagem	**TypeScript/Node.js** (Back-end) e **React/React Native** (Front-end) O uso de React Native para atende essa necessidade de compatibilidade com dispositivos móveis. TypeScript oferece tipagem forte, aumentando a manutenibilidade e diminuindo bugs.
Plataforma	**AWS** (Amazon Web Services). O Documento de Visão e Requisitos já menciona a AWS para armazenamento e sincronização

Segurança	**TLS 1.3** e **Hash Seguro (Argon 2/bcrypt).** A arquitetura deve incluir uma API de Autenticação separada e seguir o princípio de proteção de Dados de Usuári**o,** em conformidade com a LGPD
Persistência	Banco de Dados **Relacional**, e **SQL**. Permite modelar com clareza as relações complexas de Decks, Cartas e Progresso do Usuário.
Internacionalização	Nesse primeiro momento não terá necessidade de internacionalização.

3. Visão de Casos de Uso

Esta seção lista os Casos de Uso (CU) que representam a funcionalidade central e significativa do sistema Kioku, ou que enfatizam um ponto complexo e específico da arquitetura.

A funcionalidade do Kioku é centrada em seis categorias: Criação/Manutenção, Revisão, Agendamento (SRS), Registro de Progresso, Sincronização e Criação de Perfil.

ID	Título	Requisito Funcional (RF) / Prioridade	Cobertura Arquitetural / Complexidade
US17	Estudar Deck	RF17 (Must Have)	Fluxo Multi-camada Crítico: Envolve a interface (Camada Cliente), a busca por cartas agendadas (Camada de Persistência) e a lógica de estudo (Camada de Negócio).
US18	Agendar Cartas (SRS)	RF18 (Must Have)	Lógica de Domínio Complexa: É o coração do sistema, onde o algoritmo SRS é executado, impactando a Camada de Negócio diretamente com a Regra de Negócio RN01. A execução da função SRS é complexa e exige um ponto de acesso isolado no core.
US19	Sincronizar Dispositivos	RF19 (Could Have)	Arquitetura Distribuída/Confiabilidade: Ilustra a necessidade de um servidor de nuvem (AWS), forçando a arquitetura a lidar com comparação de versões, upload e download de dados, e potenciais conflitos de concorrência.
US02	Importar Deck de Flashcards	RF02 (Should Have)	Padrão Adaptador: O Cenário 3 exige o reconhecimento e importação de um formato externo. Isso requer um Objeto Adaptador na Camada de Serviço para converter a interface externa em um modelo interno, ilustrando o padrão Adapter.
US01	Criar Deck	RF01 (Must Have)	Integridade Transacional/Unicidade: Enfatiza a regra de unicidade (Cenário 1) e a obrigatoriedade de campos

ID	Título	Requisito Funcional (RF) / Prioridade	Cobertura Arquitetural / Complexidade
			(Cenário 2). Garante que a transação de criação envolva todas as camadas até a persistência.
US20	Criar Perfil	RF20 (Must Have)	Segurança e LGPD: Requisito que exige uma Camada de Autenticação separada para lidar com credenciais e dados pessoais, garantindo a Proteção de Dados de Usuário (RN07).

4. Visão Lógica

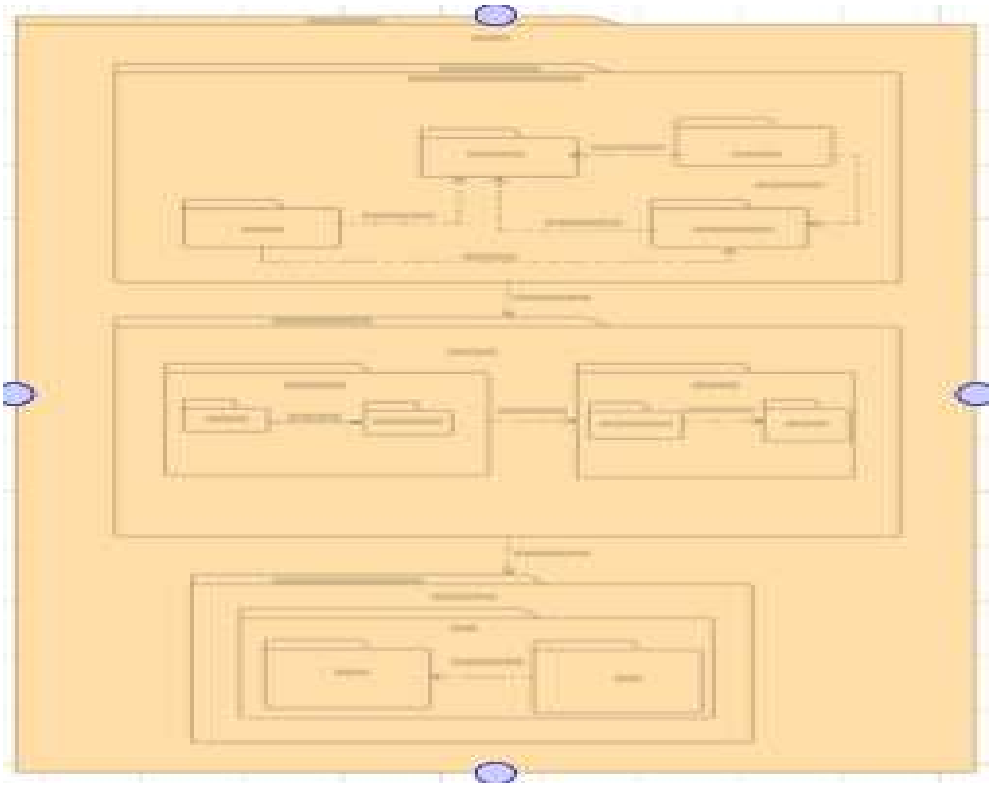
A Visão Lógica do Kioku define a estrutura interna do *back-end* e a organização do código-fonte para atingir a Manutenibilidade e a portabilidade. O estilo arquitetural adotado é o N-Camadas Distribuída.

O sistema é dividido em cinco camadas, com dependência de fora para dentro:

1. Camada Cliente: Interface do usuário (Web/Mobile).
2. Camada de Serviço (API Gateway): Controladores REST que adaptam as requisições HTTP para a aplicação.
3. Camada de Aplicação: Contém os serviços que orquestram a execução dos Casos de Uso Críticos.
4. Camada de Domínio: O Núcleo Imutável do sistema. contém as entidades, as regras de negócio e as interfaces de repositório.
5. Camada de Infraestrutura: Implementa as interfaces de domínio e lida com o acesso ao banco de dados (SQL + Relacional).

4.1 Visão Geral

O Sistema tem três camadas principais:
- Client: abrange as camadas de cliente e serviço, lidam com a interface do sistema e interação com os usuários
- Back: abrange as camadas de aplicação e dominio, lidam com as regras de negócio, casos de uso e fazem comunição com o banco de dados
- Infraestrutura: aqui temos o banco sql que implementa as entidades e seus items



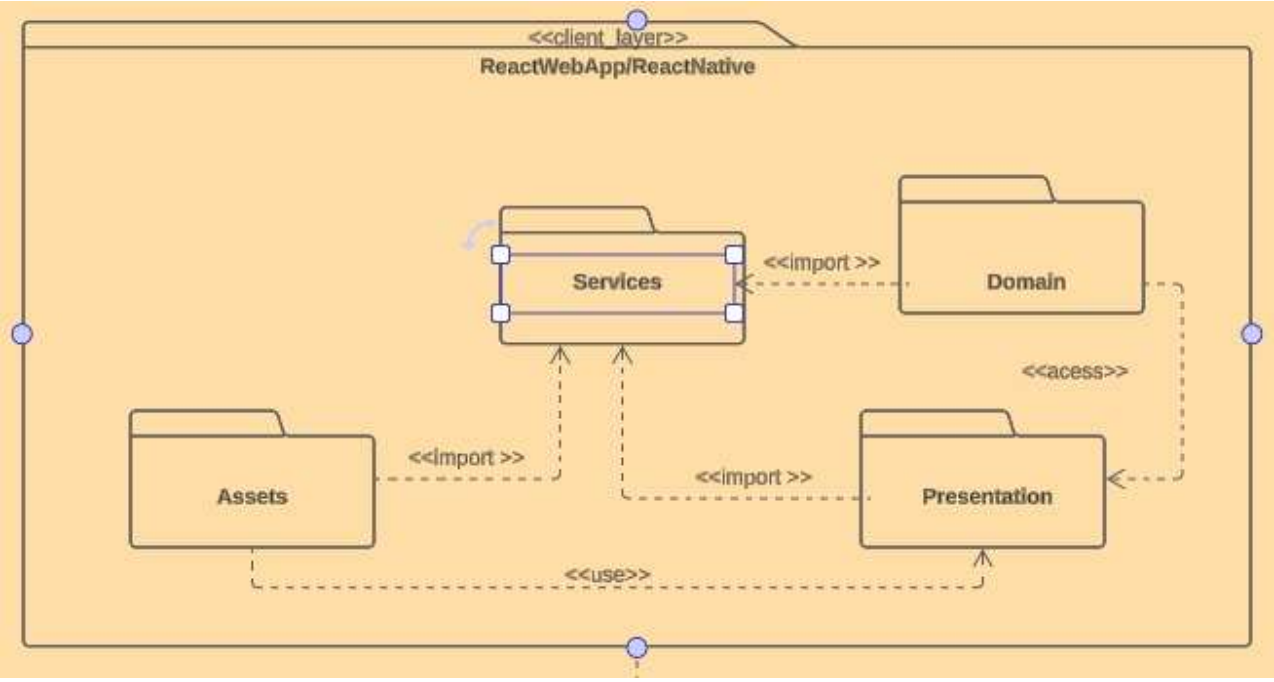
4.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura

- Services: camada que vai lidar com as requisições para o backend

- Domain: camada contem as regras de dominio e usabilidade do usuário

- Presentation: camada de apresentação, lida com a interface do sistema exibida pro usuário

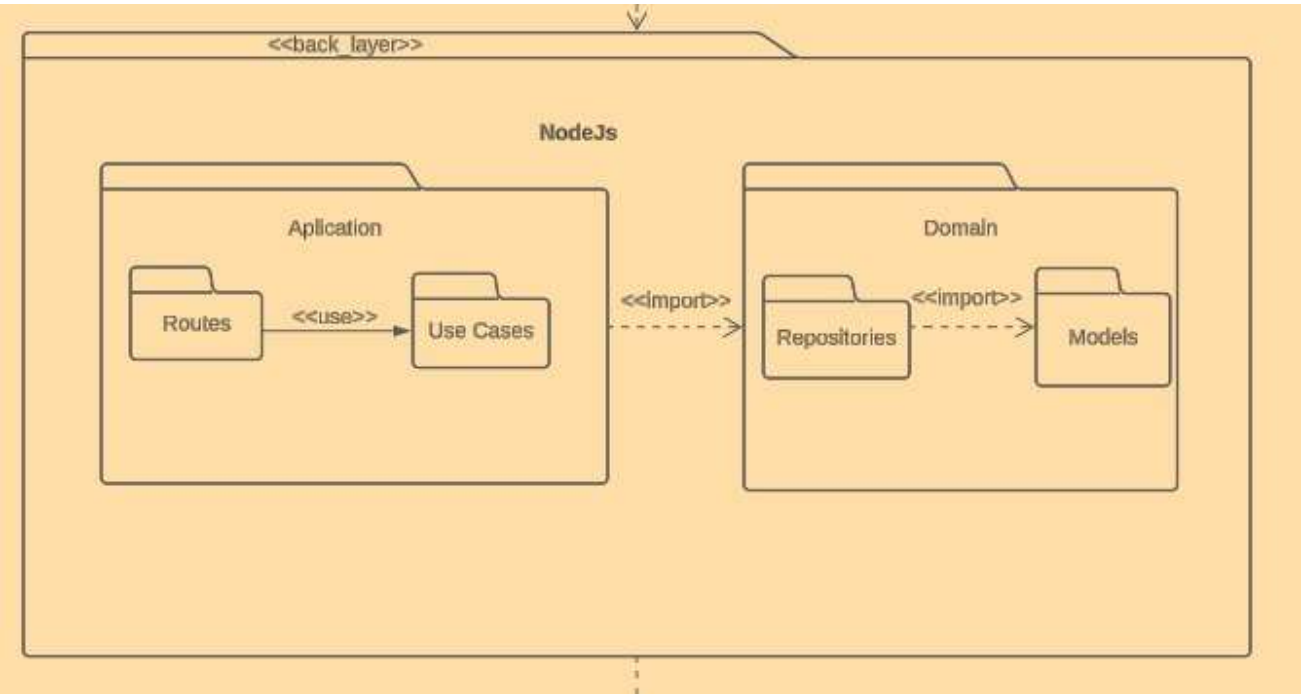
- Assets: camada que guarda as imagens de uso do front



4.3 - Camada de Back

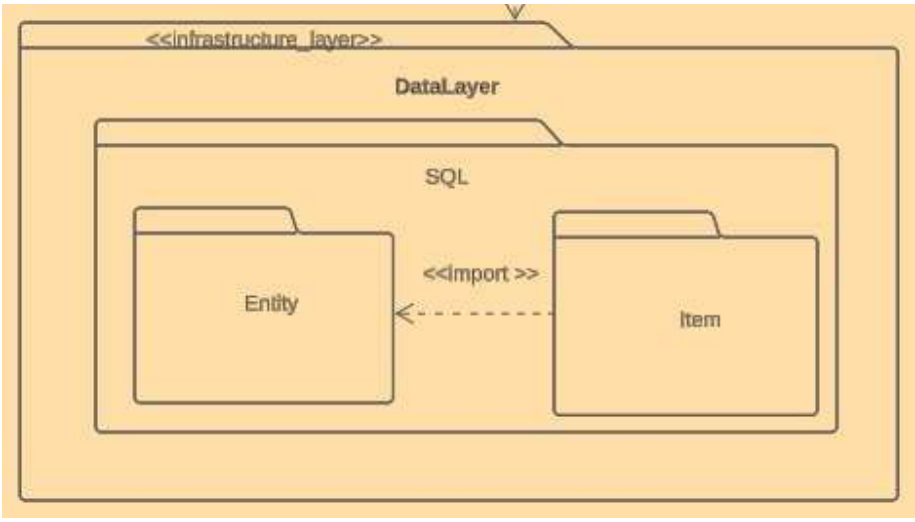
- Aplication: lida com as requisições vindas do front e redireciona pros casos de uso corretos. Tem duas sub estruturas, sendo as rotas para lidar com as requisições e a de use cases para o casos de uso

- Domain: lida com as regras de banco, então models, entidades, relacionamentos. Se divide em repositories que controla as interações com as models e as models que definem os modelos das entidades do banco relacional



4.3 - Camada de Infraestrutura

A camada de infraestrutura, que é o banco de dados relacional, aqui temos as entidades e objetos definidos pela camada de modelo da camada de backend



5. Visão de Implantação

A Visão de Implantação define a infraestrutura física que executará o Kioku, visando prioritariamente a escalabilidade e a confiabilidade.

Plataforma: AWS, utilizada para a infraestrutura de *cloud* e persistência de dados.

Estilo: Computação Distribuída em N-Camadas.

O *deployment* é estruturado em três nós físicos principais, refletindo a arquitetura lógica:

Nó Cliente (Dispositivo do Usuário): Executa as interfaces Web/Mobile e envia requisições.

Nó Servidor de Aplicação: Hospeda as Camadas de Serviço e de Negócio. É suportado por um Load Balancer para distribuir a carga e garantir a escalabilidade horizontal, suporte 1000 usuários simultâneos.

Nó Servidor de Dados: Repositório central de dados e responsável pelo Backup Diário.

Comunicação cliente servidor: Realizada via HTTPS/REST, cumprindo o requisito de Comunicação Segura.

Comunicação Interna (Servidor de Aplicação Servidor de Dados): Utiliza a Rede Privada (LAN) da nuvem para alta velocidade e segurança máxima.

6. Volume e Desempenho

Esta seção descreve as características de dimensionamento do Kioku e as restrições de desempenho desejadas que impactam diretamente a arquitetura distribuída. O cumprimento dessas restrições de desempenho válida a escolha da arquitetura N-Camadas e do *deployment* em AWS, que oferecem o isolamento de componentes e a elasticidade necessários.

Volume e Escalabilidade (Capacidade) e Desempenho e Latência (Velocidade)

Característica	Restrição	Impacto na Arquitetura
Escalabilidade (RNF03)	Suportar ao menos 1.000 usuários simultâneos sem degradação perceptível de desempenho.	Impõe a necessidade de um Load Balancer e Escalabilidade Horizontal no Servidor de Aplicação.
Volume de Dados	Conjuntos de até 1.000 cartas por usuário para teste de sincronização.	Exige um banco de dados otimizado SQL e relacional eficiente para consultas e

Característica	Restrição	Impacto na Arquitetura
		persistência na Camada de Persistência.

7. Qualidade

A arquitetura do Kioku foi projetada para garantir os seguintes atributos de qualidade, conforme detalhado no Documento de Requisitos de Software.

Item	Descrição	Solução
Escalabilidade	O sistema deve suportar ao menos 1.000 usuários simultâneos sem degradação perceptível de desempenho.	Adotar a implantação em Computação Distribuída (N-Camadas), utilizando a AWS para provisionamento de recursos elásticos. Será implementado um Load Balancer na Visão de Implantação para distribuir as requisições horizontalmente no Nó Servidor de Aplicação.
Confiabilidade	O sistema deve garantir que, em caso de *crash*, os dados sejam restaurados sem perda significativa e que os baralhos sejam idênticos após a sincronização.	Utilizar um Servidor de Dados Gerenciado (AWS RDS), com replicação e snapshots automatizados. A arquitetura exige um Backup Diário dos baralhos e progresso de estudo. A lógica de sincronização (US19) deve resolver conflitos priorizando a versão mais recente.
Disponibilidade	O serviço deve garantir uma disponibilidade mínima de 95% mensal.	Usar serviços de *cloud* com SLA (Service Level Agreement) robusto e configurar a Arquitetura Distribuída com failover entre múltiplas zonas/regiões. A separação de Camadas evita que falhas no cliente derrubem o servidor.
Portabilidade	O sistema deve funcionar em dispositivos móveis (Android e IOS), além de ser compatível com *web browsers* modernos.	Implementar o Cliente usando tecnologia Multi-plataforma (ex: React Native) e construir o *back-end* como uma API REST independente do cliente. Isso desvincula a Camada Cliente da Lógica de Negócio.
Segurança	É obrigatório o uso de hash seguro para o armazenamento de senhas, e toda a comunicação deve usar TLS 1.3. Além disso, deve haver total	Isolar a lógica de autenticação em uma Camada de Autenticação Dedicada. Impor o protocolo HTTPS para todas as interconexões e implementar um algoritmo de *hashing* robusto na Camada de Persistência para senhas.

Item	Descrição	Solução
	Proteção de Dados de Usuário (LGPD).	