DANILLO FELIPE ARAGÃO

Análise Comparativa entre Arquiteturas Serverless e Baseadas em Servidores: O Caso da Plataforma Mnemos

São Paulo
2024

DANILLO FELIPE ARAGÃO

Análise Comparativa entre Arquiteturas Serverless e Baseadas em Servidores: O Caso da Plataforma Mnemos

Versão Original

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Prof. Bruno Sofiato

São Paulo
2024

FICHA CATALOGRÁFICA

[Colocar na versão final do trabalho. Obter em:
https://www.poli.usp.br/bibliotecas/servicos/catalogacao-na-publicacao]

Nome: ARAGÃO, Danillo Felipe

Título: Análise Comparativa entre Arquiteturas Serverless e Baseadas em Servidores: O Caso da Plataforma Mnemos.

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Aprovado em: / /

Banca Examinadora

Prof(a). Dr(a). _____

Instituição: _____

Julgamento: _____

Prof(a). Dr(a). _____

Instituição: _____

Julgamento: _____

Prof(a). Dr(a). _____

Instituição: _____

Julgamento: _____

DEDICATÓRIA

Dedico este trabalho a minha esposa, Bruna, por me apoiar nessa trajetória, me inspirar a não desistir e por ajudar na elaboração deste trabalho.

RESUMO

ARAGÃO, D. F. **Análise Comparativa entre Arquiteturas Serverless e Baseadas em Servidores: O Caso da Plataforma Mnemos**. 2024. 40f. Monografia (MBA em Tecnologia de Software). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. 2024.

Este trabalho visa especificar e projetar a plataforma "Mnemos", utilizando serviços de computação em nuvem. A plataforma “Mnemos” objetiva a otimização do aprendizado com base na teoria da "curva do esquecimento" de Hermann Ebbinghaus. Em sua pesquisa, Ebbinghaus concluiu que a retenção de conhecimento pode ser aprimorada por meio de revisões periódicas do conteúdo estudado. Este trabalho realiza uma comparação entre duas arquiteturas para a implementação da plataforma: a arquitetura serverless (sem servidores), e a arquitetura baseada em servidores, analisando aspectos de desempenho, custo e viabilidade de cada abordagem.

Palavras-chave: esquecimento, curva, conhecimento, tecnologia, plataforma, revisão, cloud, serverless.

ABSTRACT

ARAGÃO, D. F. **Análise Comparativa entre Arquiteturas Serverless e Baseadas em Servidores: O Caso da Plataforma Mnemos**. 2024. 40f. Monografia (MBA em Tecnologia de Software). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. 2024.

This work aims to specify and design the "Mnemos" platform, using cloud computing services, to optimize learning tracking based on Hermann Ebbinghaus' "forgetting curve" theory. Ebbinghaus' research shows that knowledge retention can be improved through periodic content reviews, carried out within specific time intervals. The study provides a detailed comparison between two distinct architectures for the platform implementation: the serverless architecture, which offers dynamic scalability and cost optimization, and the traditional architecture, based on dedicated servers, analyzing performance, cost, and feasibility aspects of each approach.

Keywords: forgetting, curve, knowledge, technology, platform, reviewing, cloud, serverless.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

1.1. Motivações

Em 2023, a cidade de São Paulo tinha cerca de 231 mil alunos matriculados no Ensino Médio público, em situação regular, conforme os dados divulgados pelo “Histórico de matrículas por turma” fornecido pela Secretaria da Educação (2023). Ainda em 2023, o Exame Nacional do Ensino Médio (Enem) teve um total de 4,6 milhões de inscritos (Ministério da Educação, 2023), enquanto a Associação Nacional de Proteção e Apoio aos Concursos (ANPAC) estimou em 2019 que existiam cerca de 10 milhões de concurseiros no Brasil todos os anos (VALLE; Tondo, 2019). Em abril de 2024, de acordo com os dados da Pesquisa Nacional por Amostra de Domicílios Contínua (PNAD Contínua, 2024), divulgada pelo Instituto Brasileiro de Geografia e Estatística (IBGE), existiam cerca de 8.082 mil pessoas desempregadas em todo o país.

Com tantos números, é notável o mercado existente de pessoas que precisam e buscam estudar, bem como encontrar meios de reter o aprendizado, seja para provas escolares, concursos públicos ou até mesmo aprender um novo idioma e expandir as oportunidades no mercado de trabalho.

1.2. Objetivo

O objetivo deste trabalho é projetar um aplicativo de auxílio aos estudos, baseando-se na teoria da curva do esquecimento de Hermann Ebbinghaus (1880), utilizando os serviços de computação em nuvem da Amazon Web Services (AWS).

O aplicativo visa otimizar a retenção de conhecimento dos usuários, fornecendo lembretes programados para revisões de conteúdo de acordo com os princípios de repetição espaçada, a fim de melhorar a eficiência do processo de aprendizagem e memorização.

O projeto busca não apenas validar a aplicação da teoria de Ebbinghaus em ambientes digitais, mas também explorar a utilização de uma infraestrutura em nuvem, garantindo escalabilidade, disponibilidade e custo-benefício na operação do aplicativo.

1.3. Justificativas

Tendo como base o estudo da “curva do esquecimento”, o excesso de informações e a necessidade cada vez maior de estudar e reter informações, este trabalho apresenta como uma proposta de solução uma plataforma que organiza e facilita o controle dos conteúdos e revisões.

Levando em conta o cenário nacional, temos uma quantidade significativa de pessoas buscando melhores condições de vida e oportunidades de emprego, dessa forma, justifica-se a necessidade de desenvolver ferramentas e plataformas que ajudem a organizar e otimizar o estudo e as revisões de conteúdo, de forma a melhorar a retenção de conhecimento. Este trabalho se propõe a contribuir com uma solução prática e acessível para a organização dos estudos, baseada nos princípios da curva do esquecimento e nas técnicas de recordação ativa, visando melhorar a eficiência no processo de aprendizagem e, consequentemente, os resultados acadêmicos e profissionais.

A solução proposta é baseada em computação em nuvem. Segundo a AWS (2024), a computação em nuvem consiste na entrega sob demanda de recursos computacionais, como banco de dados, armazenamento, servidores de aplicação, entre outros, cujo preço se dá de acordo com a utilização dos recursos. Ao utilizarmos a computação em nuvem, deixamos de nos preocupar com algumas coisas, como manutenção e espaço físico para acomodação de servidores e redimensionamento de poder computacional, em caso de aumento ou diminuição de usuários.

1.4. Estrutura do Trabalho

O Capítulo 1 INTRODUÇÃO apresenta as motivações, o objetivo, as justificativas, método de pesquisa e a estrutura do trabalho.

O Capítulo 2 CURVA DO ESQUECIMENTO apresenta um pouco da teoria de Hermann Ebbinghaus sobre a curva do esquecimento, que serve como base para a plataforma Mnemos.

O Capítulo 3 DEFINIÇÕES define alguns termos e conceitos que serão utilizados no decorrer do trabalho.

O Capítulo 4 PROJETO aborda a idealização do projeto.

O Capítulo 5 DETALHES DA IMPLEMENTAÇÃO define de forma um pouco mais detalhada alguns modelos e interfaces.

O Capítulo 6 ARQUITETURA apresenta os modelos de arquitetura *serverless* e tradicional propostos.

O Capítulo 7 SERVERLESS X ARQUITETURA COM SERVIDORES expõe um comparativo de custo e desempenho entre os dois modelos de arquitetura.

O Capítulo 8 CONCLUSÕES FINAIS apresenta a conclusão sobre o trabalho.

REFERÊNCIAS relaciona as fontes utilizadas para a confecção do trabalho.

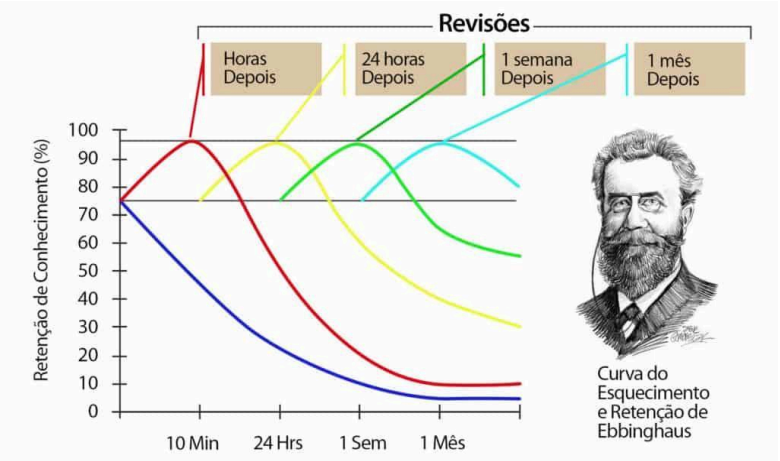
ANEXO A apresenta a configuração e precificação da arquitetura com servidores, segundo a calculadora AWS.

ANEXO B apresenta a configuração e precificação da arquitetura *serverless*, segundo a calculadora AWS.

2. CURVA DO ESQUECIMENTO

Em 1880, Hermann Ebbinghaus, psicólogo e pedagogo alemão, iniciou um estudo e publicou uma hipótese cinco anos depois chamada “Über das Gedächtnis” (traduzida para o inglês como “Memory: A Contribution to Experimental Psychology”, 1885), apresentada por meio de um gráfico, representado na Figura 1, que demonstrou como acontece a retenção de informações no cérebro humano ao longo de um período.

Figura 1 - Curva do Esquecimento e Retenção



Fonte: Diário Oficial de Brasília, 2020.

A curva de esquecimento de Hermann Ebbinghaus suporta um dos sete tipos de falhas de memória: a transitoriedade, que é o processo de esquecimento que ocorre com o passar do tempo.

Ele levantou a hipótese de que a velocidade de esquecimento depende de vários fatores, como a dificuldade do material aprendido, sua representação e outros fatores fisiológicos, como estresse e sono, e que a taxa de esquecimento basal difere pouco entre os indivíduos. Ele concluiu que a diferença no desempenho pode ser explicada por habilidades de representação mnemônica (técnicas para desenvolver a memória e memorizar coisas).

Ebbinghaus afirmou que os melhores métodos para aumentar a força da memória são:

- Melhor representação da memória (por exemplo, com técnicas de associação de ideias ou fatos difíceis de reter a outros mais simples ou mais familiares);
- Repetição baseada na recordação ativa (especialmente repetição periódica).

Sua premissa era que cada repetição na aprendizagem aumenta o intervalo ótimo antes que a próxima repetição seja necessária. Ele descobriu que a informação é mais fácil de lembrar quando é construída sobre coisas que já se conhece, e a curva do esquecimento era achatada a cada repetição. Assim, ao aplicar treinamento frequente na aprendizagem, a informação era solidificada por recordações repetidas.

As evidências sugerem que esperar de 10 a 20% do tempo até quando as informações serão necessárias é o momento ideal para uma única revisão. Como mostrado na Figura 2, após 20 minutos, uma pessoa esquece cerca de 42% do que aprendeu. Depois de uma hora, mais da metade do conteúdo já foi esquecido (56%). Passados 30 dias, 80% do conteúdo estudado é esquecido também.

Figura 2 - Como funciona a curva do esquecimento

Fonte: Blog Folha Dirigida, 2020.

3. DEFINIÇÕES

Este capítulo discute acerca de algumas definições, de forma a facilitar o entendimento do projeto.

3.1. Computação em Nuvem

A AWS (2024) define computação em nuvem como sendo a entrega sob demanda de recursos de TI, como poder computacional, banco de dados, entre outros, no lugar de comprar, manter e evoluir *datacenters* físicos. Diferentemente dos *datacenters* físicos, que têm custo do local, ar-condicionado, segurança, além dos servidores e licenças, entre outros, na computação em nuvem, esses custos são de responsabilidade do provedor, sendo cobrado apenas pela utilização dos serviços.

3.2. Front-end e *Back-end*

De acordo com a AWS, qualquer aplicação web[1] possui dois aspectos críticos: *front-end* e *back-end.* O *front-end* são as telas que o usuário interage, o que inclui elementos visuais, como botões, caixas de seleção, e mensagens de texto, entre outros. O *back-end* é a camada de processamento da aplicação, que a faz funcionar por trás da camada de exibição, processando, armazenando, consultando e manipulando os dados no banco de dados.

3.3. Arquitetura REST

A arquitetura REST*,* criada por Roy Fielding (2000), é uma forma de organizar e padronizar a interação entre componentes de sistemas na internet, estabelecendo regras básicas sobre como os sistemas devem funcionar juntos, de forma que os sistemas emissores e receptores consigam entender os dados um do outro. Algumas das regras da arquitetura REST são: a aplicação deve ter um interface uniforme, assim todos os seus recursos devem ser acessados de maneira padronizada, a aplicação não deve ter estado (*stateless*), ou seja, cada solicitação deve ser independente, não necessitando informações prévias de outras solicitações, entre outras.

[1] Nesse contexto, entendemos como aplicação web qualquer aplicação acessada via internet e que tenha uma interface para que possa ser utilizada pelos usuários.

A arquitetura REST é modelada através de recursos, que são as entidades ou objetos do sistema que serão expostos para manipulação. Os recursos podem representar algo concreto, como “usuário”, ou algo mais abstrato como “relatório”.

3.4. Protocolo HTTP

A comunicação nas APIs (Application Programming Interface) que seguem o paradigma Rest é feita com o protocolo HTTP. O protocolo HTTP foi definido na RFC 1945 (1996) como um protocolo leve, rápido, genérico, sem estado e orientado a objetos para transmissão de informações entre um cliente e um servidor, onde o cliente abre uma conexão com o servidor, executa uma requisição e espera até receber a resposta. O protocolo HTTP define os métodos de requisição, conhecidos como métodos HTTP ou verbos HTTP, que indicam a ação a ser executada para um dado recurso, entre os mais utilizados, temos[2]:

- GET - Solicita a consulta de um recurso;
- POST - Envia dados para o servidor processar;
- PUT - Envia dados para o servidor para substituição de um recurso já processado;
- DELETE - Exclui um recurso;
- PATCH - Envia dados para o servidor para atualização parcial de um recurso já processado.

A RFC 1945 também define o *Status Code* das respostas das requisições como sendo um inteiro de três dígitos que indica o resultado da requisição:

- 1xx: Respostas informativas;
- 2xx: Respostas bem-sucedidas;
- 3xx: Mensagem de redirecionamento;
- 4xx: Resposta de erro do cliente;
- 5xx: Resposta de erro do servidor.

Na RFC 7231(2014) é exposto que o alvo das requisições HTTP são os chamados recursos, porém o HTTP não limita um recurso, apenas define a

[2] Métodos GET, POST, PUT e DELETE segundo RFC 1945
Método PATCH segundo RFC 5789 (2010)

interface de interação com eles.

Cada recurso é identificado através de uma URI (Uniform Resource Identifier - Identificador Uniforme de Recurso). De forma resumida, as URIs são compostas pelo endereço do servidor, nome do recurso e, opcionalmente, alguns parâmetros. Com a junção da URI, identificando o recurso, e o método HTTP, identificando a ação, temos a função daquela requisição. Por exemplo, uma requisição do tipo PATCH, na URI “http://exemplo.com.br/clientes/154” atualiza parcialmente os dados do cliente cujo identificador é 154.

3.5. *Serverless*

Segundo a AWS (2024), as tecnologias *serverless* recebem esse nome porque sua execução ocorre em servidores totalmente gerenciados pela provedor dos serviços de nuvem, eliminando a necessidade dos responsáveis técnicos pela aplicação se preocuparem com a configuração, manutenção ou administração desses servidores, logo, para os responsáveis técnicos, é como a aplicação não possuísse um servidor. Esse modelo permite que os desenvolvedores foquem exclusivamente no código e na lógica de negócios, enquanto o provedor gerencia a infraestrutura de forma automática, garantindo, assim, alta disponibilidade e escalabilidade, além da cobrança ser realizada pelo uso de cada serviço.

3.6. FinOps

A FinOps Foundation (2024) explica que o FinOps é um *framework* e a prática cultural estratégica que visa proporcionar uma gestão financeira eficiente, permitindo que as empresas otimizem os gastos com recursos em nuvem, maximizando o valor do sistema nela hospedado. O *framework* envolve a colaboração das equipes de finanças, operações e tecnologia para garantir a alocação de recursos de forma eficaz, reduzindo os custos, além disso, o *framework* proporciona uma visão detalhada dos custos e da infraestrutura, o que permite a tomada de decisão baseada em dados de forma prática e ágil.

4. PROJETO

Esse capítulo aborda a idealização do projeto, definindo o escopo e os requisitos.

4.1. Escopo

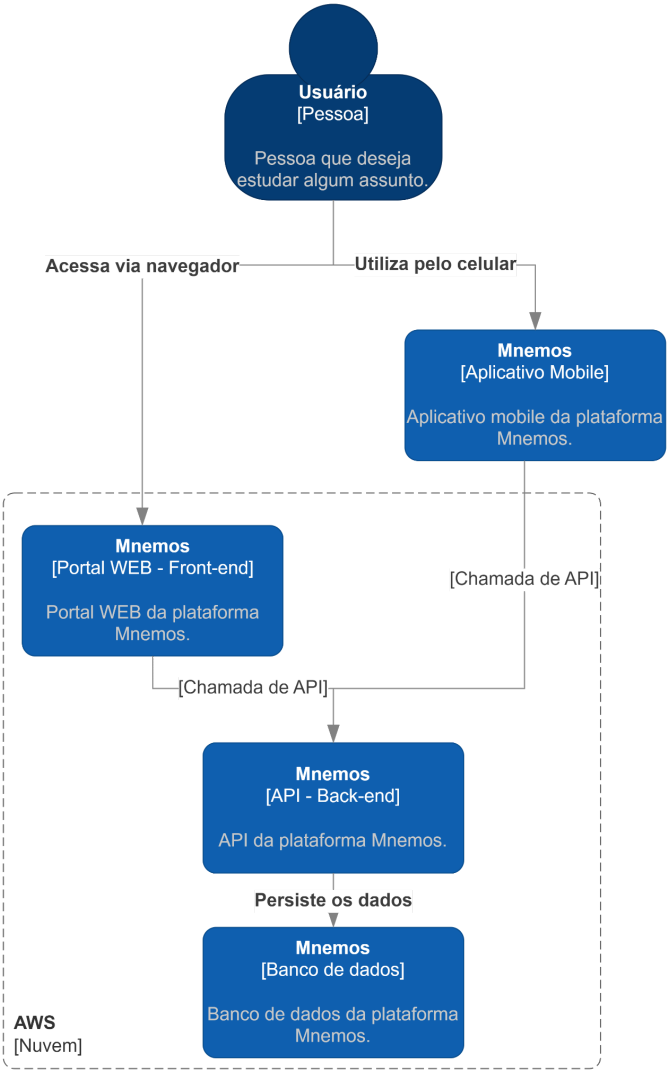
A proposta deste trabalho é projetar e estruturar a plataforma Mnemos de acompanhamento de estudo, que aplica uma sugestão de espaçamento de revisões de conteúdos criados pelos próprios usuários, derivada da pesquisa de Hermann Ebbinghaus sobre a “curva do esquecimento”.

Este trabalho tem seu foco no principal caso de uso da plataforma, que é a exibição de cartões a serem revisados de acordo com o espaçamento sugerido por Ebbinghaus, um dia, uma semana e um mês, porém, para a plataforma completa, teríamos diversos casos de uso, como o cadastro de usuários e cadastro de cartões.

4.2. Sistema

A Figura 3 mostra o diagrama de contexto do sistema, seguindo o padrão C4 Model (2024). O usuário acessa a plataforma pelo portal web, via navegador ou pelo aplicativo mobile, em dispositivos móveis. A plataforma, seja o portal web ou aplicativo, comunica-se com o *back-end,* que, por sua vez, faz as devidas operações do banco de dados, sejam elas consultas, alterações, inserções ou exclusões. Para o *front-end*, *back-end* e banco de dados, serão utilizados serviços de computação em nuvem da AWS (Amazon Web Services).

Figura 3 - Diagrama de contexto de sistema



4.3. Requisitos não-funcionais

No livro Engenharia de Software, Sommerville (2011) explica que os requisitos não-funcionais são restrições aplicadas ao sistema, incluindo restrições de tempo de resposta e restrições nos processos, são, diferentemente dos requisitos funcionais, restrições que não afetam as regras de negócio.

A ISO 25010 define características de qualidade para categorizar requisitos não-funcionais. Abaixo encontram-se características de qualidade consideradas importantes para a plataforma, bem como o racional pela sua escolha:

- Capacidade: A capacidade, ou limite máximo do produto, está relacionado à capacidade de processamento do sistema sem degradação significativa de seu desempenho. O sistema deverá ter boa resposta à variação de capacidade. Como a plataforma poderá ser utilizada por uma gama muito grande de usuários devemos prever necessidade de um possível aumento de poder computacional, devido a um aumento do número de acessos;
- Adaptabilidade: A adaptabilidade é a capacidade de um sistema de ser adaptado para um novo hardware, e a tecnologia. O sistema deverá ser implementado para plataforma web e para plataforma mobile. O acesso ao sistema via plataforma web visa facilitar a escrita e a criação dos cartões que serão revisados, uma vez que é mais fácil redigir textos mais longos com teclados físicos de um computador do que com o teclado virtual de um dispositivo mobile. A camada de apresentação deve ser desenvolvida em Flutter[3], uma vez que ele é capaz de gerar, com o mesmo código, aplicações mobile para dispositivos Android e iOS, bem como aplicações web e até mesmo aplicações desktop para sistemas Windows, macOS e Linux;
- Comportamento do Tempo e Uso de Recursos: O comportamento do tempo é o tempo de resposta de um software, e uso de recursos, a quantidade de poder computacional utilizado por uma sistema. As funções Lambda deverão ser desenvolvidas na linguagem Python: Conforme observado em um *benchmark* realizado pela Xebia (2024), as duas linguagens de programação cujas funções Lambda executam mais rápido são Rust e Python. Uma pesquisa realizada pela Jetbrains (2023) mostra Python em terceiro lugar entre as linguagens com mais desenvolvedores, enquanto Rust não aparece no top 7. A linguagem Python foi escolhida por contar com uma rápida execução em funções Lambda, aliado a uma alta disponibilidade de desenvolvedores.

[3] Site oficial: https://flutter.dev/

4.4. Requisitos funcionais

Segundo Sommerville, os requisitos funcionais são os serviços que o sistema deve fornecer, as respostas e reação do sistema de acordo com determinadas entradas. Os requisitos funcionais também são conhecidos como requisitos de negócio, ou regras de negócio.

O sistema deverá permitir a criação de novos usuários, CRUD (Create, Read, Update, Delete - Criação, Leitura, Atualização e Exclusão) de cartões, CRUD de etiquetas, recuperação de senha, arquivamento de cartões (inibição da exibição do cartão na lista de revisão) e notificação dos cartões que devem ser revisados no dia. Dentre esses requisitos, o principal para a plataforma, é a notificação dos cartões que devem ser revisados no dia, é essa funcionalidade que colocará em prática o estudo de Hermann Ebbinghaus, e auxiliará o usuário em seus estudos.

4.5. Caso de uso exibição de cartões a serem revisados

O caso de uso “Exibição de Cartões a Serem Revisados” é o principal caso de uso do sistema. Ele é responsável por notificar o usuário nas datas corretas de revisão do conteúdo do cartão, seguindo o estudo do Ebbinghaus.

A Tabela 1 define as regras de negócio relacionadas a esse caso de uso.

Tabela 1 - Regras de negócio

Regras de negócio	Descrição
RN01	As notificações de revisões do cartão deverão ocorrer um dia após a criação, uma semana após a primeira revisão, um mês após a segunda revisão, um semestre após a terceira revisão.
RN02	Após a data de revisão estipulada, o cartão continuará sendo exibido na lista de cartões a serem revisados até que o usuário o marque como revisado, como arquivado ou o exclua.
RN03	A data da próxima revisão do cartão será calculada com base na data em que o usuário efetuou a revisão atual.
RN04	Cartões arquivados não aparecerão na lista de cartões a revisar.

5. DETALHES DA IMPLEMENTAÇÃO

Esse capítulo aborda a implementação de uma forma mais detalhada, definindo algumas interfaces e modelos.

5.1. API - Application Programming Interface

Para possibilitar a comunicação entre as camadas do front-end e back-end, é necessária uma interface de aplicação que padronize a troca de mensagens, conforme define a arquitetura REST.. Para tanto, temos, na Tabela 2, a API (Application Programming Interface - Interface de Programação de Aplicativos), que define quais os recursos disponíveis e como acessá-los. Nessa definição, estão sendo seguidos os conceitos elencados no item 3.3, onde os recursos são consultados com o método GET, atualizados com o método PATCH e deletados com o método DELETE.

Tabela 2 - API de Cartões

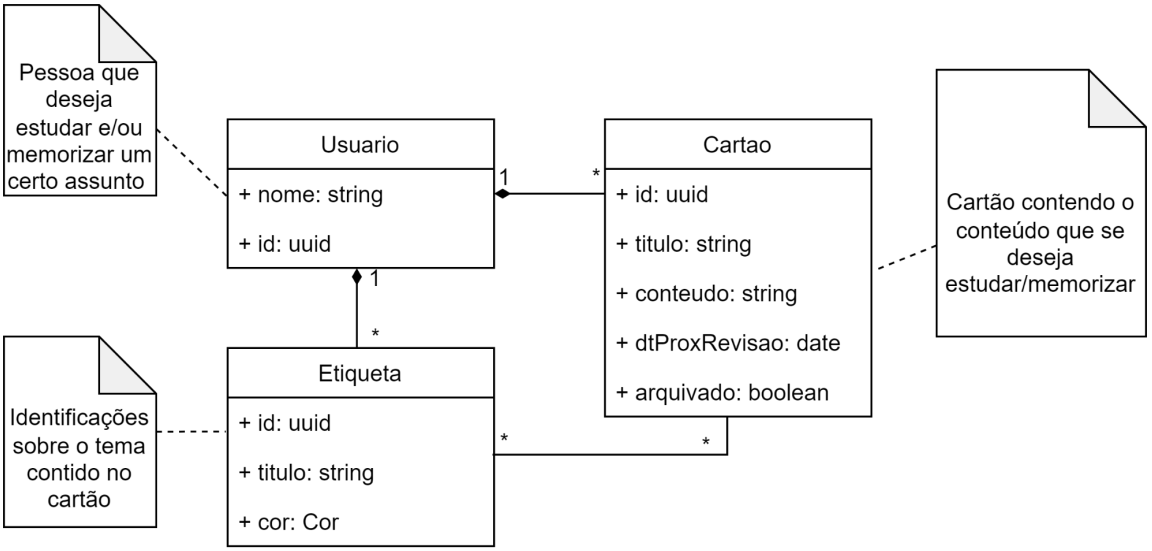
Cartões		
Endpoint	**Método**	**Ação**
api/v1/cartoes/{id}	PATCH	Marca o cartão como revisado e atualiza a data da próxima revisão
api/v1/cartoes/{id}	GET	Coleta as informações de um cartão
api/v1/cartoes/{id}	DELETE	Deleta um cartão
api/v1/cartoes	GET	Coleta as informações de todos os cartões
api/v1/cartoes/revisao	GET	Coleta as informações dos cartões a serem revisados no dia

5.2. Modelo de domínio

Na Figura 4 temos o diagrama de domínio, seguindo o padrão de diagrama de classes da UML (2024). No diagrama temos as entidades Usuario, Cartao e Etiqueta. Um Cartao e uma Etiqueta só existe, caso exista também um Usuario, caracterizando, assim uma composição[4].

[4] Segundo a UML (2024), composição é quando uma classe está contida na outra, a classe contida só existe enquanto a classe que a contém existe, caso a classe que contém seja destruída, a classe contida também será destruída.

Figura 4 - Diagrama de domínio

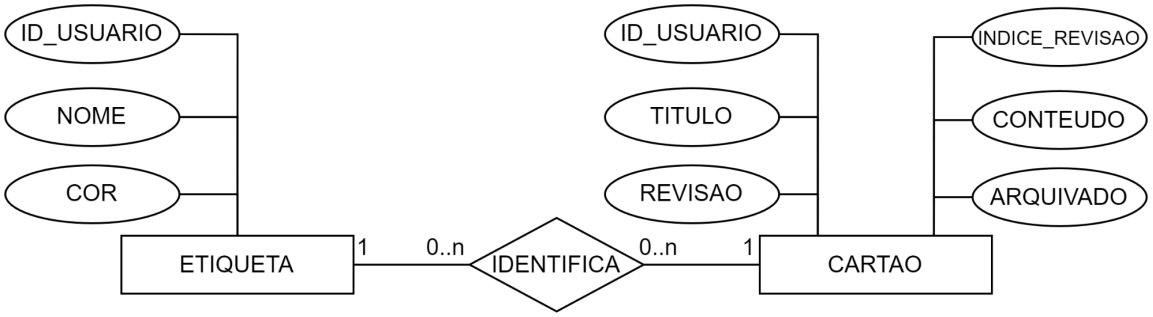


5.3. Banco de dados

Algumas aplicações não necessitam persistir dados, e, por consequência, não exigem um banco de dados, por exemplo uma aplicação que faça a integração entre dois sistemas, ou mesmo um aplicativo de celular que se utilize somente do armazenamento local.

A plataforma Mnemos poderia contar apenas com o armazenamento local do dispositivo móvel do usuário, sem utilizar um banco de dados, porém em um cenário como esse não seria possível integrar com um portal web e o usuário perderia todos os dados ao realizar limpeza ou trocar o dispositivo, além disso, o usuário poderia ter problemas com a plataforma, caso seu armazenamento estivesse cheio. Para evitar esses problemas, foi decidido pela utilização de um banco de dados para a persistência de dados. Como as informações são bem estruturadas e padronizadas, foi decidido por um banco de dados relacional, por ter um melhor desempenho nesse cenário. Caso a estrutura dos dados pudesse variar muito, dificultando a padronização e criação de tabelas, o banco de dados não relacional seria o mais indicado.

Figura 5 - Modelagem de banco de dados



Para a modelagem de banco de dados utilizamos a notação de diagrama de classes da UML, conforme ilustrado no Diagrama disposto na Figura 5. Foram definidas três tabelas, a tabela CARTAO, que contem informações sobre o cartão que o usuário cadastrou com o resumo do assunto estudado, a tabela ETIQUETA, que armazena as informações das etiquetas para categorização dos cartões, e, por fim, a tabela CARTAO_ETIQUETA, que relaciona as etiquetas aos cartões, uma vez que, ao mesmo tempo que uma etiqueta pode estar atrelada a nenhum ou vários cartões, um cartão também pode possuir nenhuma ou várias etiquetas.

Como estamos utilizando o Amazon Cognito para autenticação e autorização dos usuários, foi decidido pela não criação de uma tabela de usuários, apenas usando o ID do usuário do próprio Amazon Cognito[5] como referência nas tabelas CARTAO e ETIQUETA.

[5] Os serviços AWS utilizados estão explicados no Anexo C

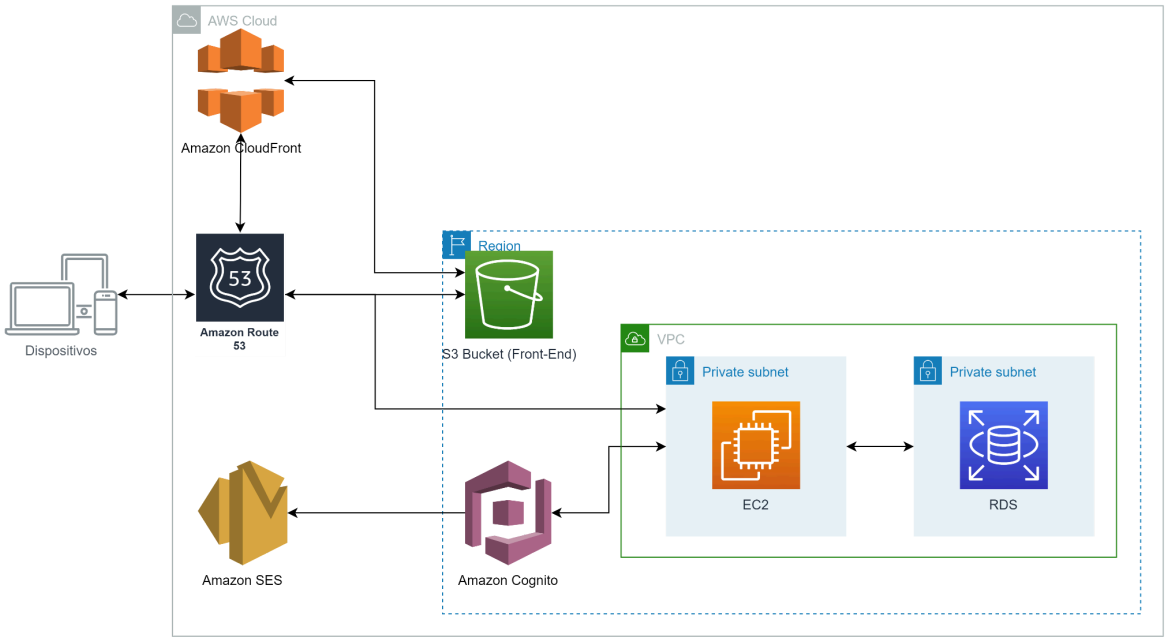
6. ARQUITETURA

Neste capítulo serão discutidos os modelos de arquitetura tradicional (com servidores) e a arquitetura *serverless*.

6.1. Arquitetura tradicional (com servidores)

Na Figura 6 temos a arquitetura AWS tradicional, com servidores. Da mesma forma que a arquitetura *serverless* os dispositivos acessam a plataforma por meio do Route53 com os arquivos do site da plataforma armazenados no S3 Bucket e distribuídos pelo CloudFront. Também tem integração com o Amazon Cognito para autenticação e com o Amazon SES para envio de e-mail de confirmação de cadastro e recuperação de senhas. Porém a API é hospedada e disponibilizada por uma máquina EC2 que atua como servidor, que por sua vez, acessa o RDS para persistência e consulta dos dados.

Figura 6 - Arquitetura AWS - Com servidores

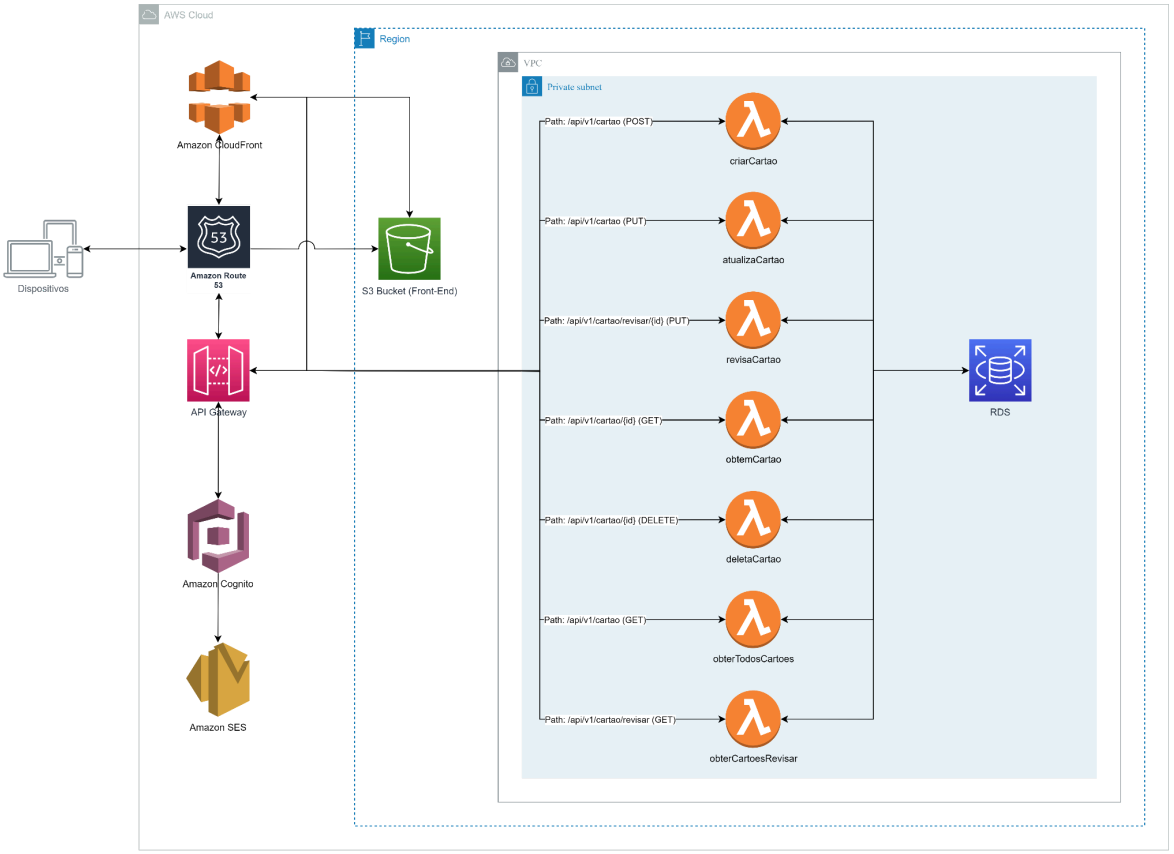


6.2. Arquitetura *Serverless*

A Figura 7 representa a arquitetura AWS *serverless*, na qual os dispositivos (navegadores web e dispositivos móveis) acessam a plataforma por meio do

Route53. Caso a plataforma esteja sendo acessada via navegador, os arquivos do site da plataforma, armazenados no S3 Bucket, são distribuídos pelo CloudFront.
O API Gateway expõe as funções Lambda por meio de rotas da API, garantindo a autenticação e autorização com o Amazon Cognito. As funções Lambda, por sua vez, acessam o RDS para persistência e consulta dos dados. O Amazon Cognito integra-se com o Amazon SES para envio de e-mail de confirmação de cadastro e recuperação de senhas.

Figura 7 - Arquitetura AWS - Serverless



6.3. *Serverless* X Arquitetura com Servidores

Em questão de desempenho, um *benchmark* realizado por Pedro Correa (2021) comparou uma aplicação sendo executada em uma arquitetura com servidores com a mesma aplicação sendo executada em uma arquitetura *serverless*, ambas utilizando os serviços Azure de computação em nuvem. Foram realizadas 825 requisições em 70 segundos no ambiente com servidores, dessas requisições, apenas 271 foram bem sucedidas, o que representa 33% das chamadas, o que nos dá 3,9 requisições/segundos com sucesso. Já no ambiente *serverless*, foram

realizadas 848 chamadas em 70 segundos, todas bem sucedidas, equivalente a 12,11 requisições/segundo com sucesso.

Em termos de custo, as duas arquiteturas têm formas distintas de cobrança, enquanto na arquitetura com servidores a cobrança é pela disponibilidade do servidor, independente de estar sendo utilizado ou não, no *serverless* ela é realizada de acordo com a utilização. Por conta disso, foi analisado um período de maior estresse da aplicação, com maior quantidade de chamadas. Nesse cenário, o custo estimado para a arquitetura com servidores seria de US$ 840/mês contra US$ 190 para a arquitetura *serverless*.

Além disso, por ser cobrado conforme consumo de recursos, utilizar a arquitetura *serverless* incentiva a otimização dos códigos, como forma de redução de custo, refletindo no desempenho do sistema.

Foi utilizada a calculadora AWS (2024) para estimar o custo da aplicação seguindo a arquitetura *serverless* apresentada na Figura 6, sua configuração e precificação estão contidas no Anexo B, e uma arquitetura com servidores, apresentada na Figura 7, cuja configuração e precificação então contidas no anexo A utilizando máquinas virtuais como servidores para o *back-end*.

De acordo com a calculadora AWS, como exibido na tabela 3, para o cenário *serverless*, não temos custo inicial e um custo mensal de US$ 356,59, enquanto para o cenário com servidores, temos um custo inicial de US$ 1.014,41 e um custo mensal de US$ 1.040,02, o um custo mensal 292% maior que o custo mensal do cenário *serverless*.

Tabela 3 - Comparação de custos - Arquitetura com servidores x *Serverless*

	Com servidores	***Serverless***
Custo inicial	US$ 1.014,41	-
Custo mensal	US$ 1.040,02	US$ 356,59
Custo primeiro ano	US$ 13.494,65	US$ 4279,08

Como exposto acima, o custo de um ambiente *serverless* é muito menor que o ambiente com servidores, variando o valor conforme o uso, sem desperdiçar recursos computacionais, indo de encontro a ideia de *FinOps*. Além dessa vantagem financeira, temos também a vantagem em relação a capacidade do sistema; ao dimensionarmos um servidor em uma arquitetura com servidores, levamos em conta

uma estimativa de acessos simultâneos, caso haja um aumento de acessos, mesmo que temporário, o servidor pode não ter recursos suficientes, podendo causar lentidão ou até mesmo indisponibilidade do sistema, já em uma arquitetura serverless, como a nuvem fornece recurso sob demanda, caso ocorra um aumento de demanda, a performance do sistema não é afetada. Por conta dessas vantagens, a arquitetura escolhida foi a *serverless*.

7. CONSIDERAÇÕES FINAIS

7.1. Conclusões

Infelizmente, a curva de esquecimento é um efeito que não pode ser impedido. A eliminação de informações da memória é uma forma do cérebro se proteger, evitando uma sobrecarga. É uma função natural cujos efeitos não podem ser bloqueados. Porém, com o uso da plataforma “Mnemos”, as pessoas terão às mãos, por meio de um aparelho celular, todo o conteúdo que elas já estudaram, um resumo dele da forma que melhor entenderam e as datas propostas para revisão. Desta forma, a sequência de estudos estará organizada e sempre acessível e disponível para consulta.

As arquiteturas *serverless* e com servidor em nuvem apresentam diversos *trade-offs* que devem ser considerados ao escolher a mais adequada para uma aplicação, por exemplo, apesar das arquiteturas com servidor apresentarem custos previsíveis, esse custo aumenta quando temos baixa utilização, por termos custos fixos de instâncias ativas de servidores. Em contrapartida, nas arquiteturas *serverless* a cobrança por consumo proporciona um custo mais eficiente para utilização intermitente, mas pode ser um pouco mais caro em utilizações intensivas ou contínuas. Enquanto a arquitetura com servidores proporciona um controle maior sobre o ambiente, a arquitetura *serverless* conta com a escalabilidade automática.

Como exposto neste trabalho, a implementação da plataforma “Mnemos” com programação em nuvem da Amazon (AWS) e uma arquitetura *serverless* é factível. A integração de serviços gerenciados AWS proporciona uma base sólida escalável, flexível e confiável para a plataforma. A combinação desses serviços não apenas garante eficiência e desempenho, mas também proporciona uma infraestrutura segura e econômica. Portanto, a arquitetura proposta para a “Mnemos”, utilizando as soluções oferecidas pela AWS, confirma a viabilidade e a capacidade da plataforma para atender às necessidades de escalabilidade, segurança e performance, posicionando-a favoravelmente para alcançar sucesso a longo prazo.

7.2. Trabalhos Futuros

Como próximos passos para este trabalho temos a criação das arquiteturas propostas, de forma a comparar, de forma mais prática, o desempenho de ambas (*benchmark*); análise e comparação dos custos a longo prazo; e estudo sobre o uso de inteligência artifical para gerenciar os custos de forma ainda mais eficiente. Um pré-requisito para os três estudos é a implementação do sistema completo nas duas arquiteturas.

Para realização do *benchmark*, deve-se realizar requisições de forma a simular acessos simultâneos para poder mensurar a taxa de sucesso e de falhas, algo similar ao *benchmark* realizado pelo Pedro Correa, mas com os serviços da AWS.

Para a realização do *benchmark*, de forma a realizar o comparativo dos custos a longo prazo, será necessário implementar uma forma de equilibrar as requisições entre as duas arquiteturas, em um teste A/B, o que pode ser realizado com ajustes na política de roteamento do serviço Route 53. Segundo a Oracle (2024), o teste A/B serve para comparar o desempenho de duas versões de um sistema, geralmente as versões divergem em termos de conteúdo, para testar qual delas é mais atrativa para os visitantes. Nesse caso as versões divergem em termos de arquitetura.

Para o estudo sobre o uso da inteligência artificial para gerenciamento de custos, será necessária a implementação de algumas ferramentas de coleta de dados, para a aquisição de dados de uso e custos, ferramentas para tratamento desses dados, além de ferramentas de análise de dados, e de modelagem de inteligência artifical.

REFERÊNCIAS

SÃO PAULO, Secretaria da Educação de. **Dados Abertos da Educação. Histórico de matrículas por turma.** Disponível em: <https://dados.educacao.sp.gov.br/dataset/histórico-de-matrículas-por-turma>. Acesso em 01 dez. 2024.

BRASIL. Ministério da Educação. **Inep registra 4,6 milhões de inscritos no Enem 2023** Disponível em: <https://www.gov.br/inep/pt-br/assuntos/noticias/enem/inep-registra-4-6-milhoes-de-inscritos-no-enem-2023>. Acesso em 01 dez. 2024.

VALLE, Patrícia. TONDO, Stephanie. **Concurseiros e mercado estão assustados com medidas do governo Bolsonaro.** Disponível em: <https://extra.globo.com/economia/concurseiros-mercado-estao-assustados-com-medidas-do-governo-bolsonaro-23579586.html#:~:text=A%20associação%20Nacional%20de%20Proteção,os%20impactos%20para%20o%20mercado>. Acesso em 01 dez. 2024.

IBGE. **Painel PNAD Contínua.** Disponível em: <https://painel.ibge.gov.br/pnadc/>. Acesso em 01 dez. 2024.

EBBINGHAUS, Hermann. ***Über das Gedächtnis*** (*Memory: A Contribution to Experimental Psychology*). Tradução de Henry A. Ruger & Clara E. Bussenius. Disponível em: <http://psychclassics.yorku.ca/Ebbinghaus/index.htm>. Acesso em 01 dez. 2024.

SCHACTER, Daniel L. ***The seven sins of memory: How the mind forgets and remembers***. Boston: Houghton Mifflin, 2001.

SONNAD, Nikhil. ***You probably won't remember this, but the "forgetting curve" theory explains why learning is hard.*** Disponível em: <https://qz.com/1213768/the-forgetting-curve-explains-why-humans-struggle-to-memorize/>. Acesso em 01 dez. 2024.

ECKEL, Bruna. ***Curva do Esquecimento: o que é e como superar.*** Disponível em: <https://noticiasconcursos.com.br/curva-do-esquecimento-o-que-e-e-como-superar/>. Acesso em 01 dez. 2024.

BRASÍLIA, Diário Oficial de. ***Curva de Esquecimento: o que você tem que saber para não desperdiçar seu estudo***. Disponível em: <https://www.diariooficialdf.com.br/curva-de-esquecimento-o-que-voce-tem-que-saber-para-nao-desperdicar-seu-estudo/>. Acesso em 01 dez. 2024.

WIKIPÉDIA. **Curva do esquecimento.** Disponível em: <https://pt.wikipedia.org/wiki/Curva_do_esquecimento>. Acesso em 01 dez. 2024.

WIKIPÉDIA. **Hermann Ebbinghaus.** Disponível em: <https://pt.wikipedia.org/wiki/Hermann_Ebbinghaus>. Acesso em 01 dez. 2024.

AWS. **Boas-vindas à documentação do AWS.** Disponível em: <https://docs.aws.amazon.com/pt_br/>. Acesso em 01 dez. 2024.

C4 MODEL. **The C4 model for visualising software architecture**. Disponível em: <https://c4model.com/>. Acesso em 01 dez. 2024.

AWS. **Qual é a diferença entre front-end e back-end no desenvolvimento de aplicações?**. Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-frontend-and-backend/#:~:text=O%20back%2Dend%20consiste%20nos,da%20aplica%C3%A7%C3%A3o%20para%20seus%20usu%C3%A1rios.>. Acesso em 01 dez. 2024.

AWS. **O que é a API RESTful?**. Disponível em: <https://aws.amazon.com/pt/what-is/restful-api/#:~:text=A%20API%20RESTful%20%C3%A9%20uma,terceiros%20para%20executar%20v%C3%A1rias%20tarefas.>. Acesso em 01 dez. 2024.

AWS. **Fundamentos da Nuvem AWS**. Disponível em: <https://aws.amazon.com/pt/getting-started/cloud-essentials/#:~:text=A%20computa%C3%A7%C3%A3o%20em%20nuvem%20%C3%A9,com%20pre%C3%A7o%20conforme%20o%20uso.>. Acesso em 01 dez. 2024.

Fielding, Roy. **Architectural Styles and the Design of Network-based Software Architectures**. Disponível em: <https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acesso em 01 dez. 2024.

mdn web docs. **HTTP**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP>. Acesso em 01 dez. 2024.

mdn web docs. **Métodos de requisição HTTP**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>. Acesso em 01 dez. 2024.

ABNT NBR ISO/IEC 25010:2011. **Engenharia de sistemas e software - Requisitos de qualidade e avaliação de sistemas e software (SQuaRE) - Modelo de qualidade de produto e modelo de qualidade em uso**. Rio de Janeiro: ABNT, 2011.

Xebia. **AWS Lambda Benchmarking**. Disponível em: <https://xebia.com/blog/aws-lambda-benchmarking/>. Acesso em 01 dez. 2024.

AWS. **Sem servidor na AWS**. Disponível em: <https://aws.amazon.com/pt/serverless>. Acesso em 01 dez. 2024.

Jetbrains. **Ecossistema de Desenvolvedores: Data Playground**. Disponível em: <https://www.jetbrains.com/pt-br/lp/devecosystem-data-playground/>. Acesso em 01 dez. 2024.

NIELSEN, J. **Usability Engineering**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

EVANS, Eric. **Domain-Driven Design: Tackling Complexity in the Heart of Software**. 1. ed. Boston: Addison-Wesley, 2004.

OMG UML, **OMG® Unified Modeling Language® (OMG UML®)**. Disponível em: <https://www.omg.org/spec/UML/2.5.1/PDF>. Acesso em 01 dez. 2024.

BERNERS-LEE, Tim; FIELDING, Roy T.; FRYSTYK, Henrik. **Hypertext Transfer Protocol – HTTP/1.0. Fremont, CA: Internet Engineering Task Force**, 1996. Disponível em: <https://datatracker.ietf.org/doc/html/rfc1945>. Acesso em 01 dez. 2024.

FIELDING, Roy T.; GETTYS, Jim; MOGUL, Jeffrey C.; NIELSEN, Henrik Frystyk; MASINTER, Larry; LEE, Paul J.; TETZLAFF, Tim Berners-. **Hypertext Transfer Protocol – HTTP/1.1. Fremont, CA: Internet Engineering Task Force**, 1999. Disponível em: <https://datatracker.ietf.org/doc/html/rfc2616>. Acesso em 01 dez. 2024.

DUSSEAULT, Lisa; SNELL, James M. **PATCH Method for HTTP. Fremont, CA: Internet Engineering Task Force**, 2010. Disponível em: <https://datatracker.ietf.org/doc/html/rfc5789>. Acesso em 01 dez. 2024.

FIELDING, Roy; RESCHKE, Julian. **Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content**, 2014. Disponível em: <https://datatracker.ietf.org/doc/html/rfc7231>. Acesso em 01 dez. 2024.

Luchidchart; **Principais diagramas da UML: modelos e exemplos**, 2024. Disponível em: <https://www.lucidchart.com/blog/pt/modelos-e-exemplos-de-diagramas-uml#class>. Acesso em 01 dez. 2024.

CORREA, Pedro; **Serverless Saga #2: Benchmarks & Custo**, 2021. Disponível em: <https://medium.com/digitalproductsdev/serverles-saga-2-benchmarks-custo-24e7063e9cc>. Acesso em 01 dez. 2024.

AWS; **Calculadora de preços da AWS**, 2024. Disponível em: <https://calculator.aws/>. Acesso em 01 dez. 2024.

FinOps Foundation; **What is FinOps**, 2024. Disponível em: <https://www.finops.org/introduction/what-is-finops/>. Acesso em 01 dez. 2024.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

Oracle; **O que é teste A/B?**, 2024. Disponível em: <https://www.oracle.com/br/cx/marketing/what-is-ab-testing/#:~:text=teste%20A%2FB%3F-,Defini%C3%A7%C3%A3o%20de%20teste%20A%2FB,delas%20atrai%20mais%20visitantes%2Fespectadores.>. Acesso em 08 dez. 2024.

ANEXO A

Na Tabela A.1 temos os serviços e valores para um ambiente com servidores para a plataforma Mnemos. A estimativa de preços foi gerada pela calculadora AWS no dia 10/11/2024, seguindo a arquitetura definida na Figura A.1, estimando um custo inicial de US$ 1.014,40, e um custo mensal de US$ 1.040,02.

Tabela A.1 - Configuração com servidores

Serviço	Custo Inicial	Mensal	Resumo da configuração
S3 Standard	US$ 0,00	US$ 0,03	Armazenamento S3 Standard (0.5 GB por mês), Solicitações PUT, COPY, POST, LIST para S3 Standard (2), GET, SELECT e todas as outras solicitações do S3 Standard (15000)
Amazon CloudFront	US$ 0,00	US$ 0,13	Número de solicitações (HTTPS) (por mês), Transferência de dados de saída para a Internet (00 GB por mês), Transferência de dados de saída para a origem (1 GB por mês), Número de solicitações (HTTPS) (0 por mês)
Amazon EC2	US$ 1.014,40	US$ 28,17	Locação (Instâncias compartilhadas), Sistema operacional (Linux), Carga de trabalho (Consistent, Número de instâncias: 1), Instância do EC2 avançada (r6g.large), Pricing strategy (Compute Savings Plans 3yr Partial Upfront), Habilitar monitoramento (desabilitada), DT Entrada: Not selected (0 TB por mês), DT Saída: Not selected (0 TB por mês), DT Intrarregião: (0 TB por mês)
Amazon RDS for MySQL	US$ 0,00	US$ 299,20	Quantidade de armazenamento (100 GB), Nós (1), Tipo de instância (db.m1.medium), Utilização (somente sob demanda) (100 %Utilized/Month), Opção de implantação (Multi-AZ), Modelo de preço (OnDemand), Armazenamento para cada instância do RDS (SSD de uso geral (gp2))
Amazon Route 53	US$ 0,00	US$ 2,20	Zonas hospedadas (1)
VPN Connection	US$ 0,00	US$ 384,50	Dias úteis por mês (22), Número de conexões do Site-to-Site VPN (), Número de associações de sub-rede (1)
VPN Connection	US$ 0,00	US$ 275	Dias úteis por mês (22)
Amazon Cognito	US$ 0,00	US$ 50,75	Taxa de otimização para solicitações de token (0), Taxa de otimização para clientes de aplicativo (0), Recursos avançados de segurança (Habilitada), Número de usuários ativos mensalmente (MAU) (1000)
Amazon Simple E-mail Service (SES)	US$ 0,00	US$ 0,03	Mensagens de e-mail enviadas do EC2 (0 por mês), Mensagens de e-mail enviadas do cliente de e-mail (300 por mês)

ANEXO B

Na Tabela B.1 temos os serviços e valores para um ambiente serverless para a plataforma Mnemos. A estimativa de preços foi gerada pela calculadora AWS no dia 10/11/2024, seguindo a arquitetura definida na Figura 10 do item 6.4, a estimativa não gerou um custo inicial, e gerou um custo mensal de US$ 356,59.

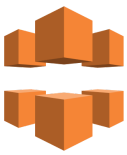
Tabela B.1 - Configuração Serverless

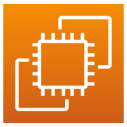
Serviço	Custo Inicial	Mensal	Resumo da configuração
S3 Standard	US$ 0,00	US$ 0,03	Armazenamento S3 Standard (0.5 GB por mês), Solicitações PUT, COPY, POST, LIST para S3 Standard (2), GET, SELECT e todas as outras solicitações do S3 Standard (15000), Dados retornados pelo S3 Select (0.5 GB por mês)
Amazon CloudFront	US$ 0,00	US$ 0,13	Número de solicitações (HTTPS) (por mês), Transferência de dados de saída para a Internet (0 GB por mês), Transferência de dados de saída para a origem (1 GB por mês), Número de solicitações (HTTPS) (0 por mês)
AWS Lambda	US$ 0,00	US$ 0,00	Arquitetura (x86), Arquitetura (x86), Modo de invocação (Em buffer), Quantidade de armazenamento temporário alocada (512 MB), Número de solicitações (900000 por mês)
Amazon RDS for MySQL	US$ 0,00	US$ 299,20	Quantidade de armazenamento (100 GB), Nós (1), Tipo de instância (db.m1.medium), Utilização (somente sob demanda) (100 %Utilized/Month), Opção de implantação (Multi-AZ), Modelo de preço (OnDemand), Armazenamento para cada instância do RDS (SSD de uso geral (gp2))
Amazon Route 53	US$ 0,00	US$ 2,20	Zonas hospedadas (1)
Amazon Cognito	US$ 0,00	US$ 50,75	Taxa de otimização para solicitações de token (0), Taxa de otimização para clientes de aplicativo (0), Recursos avançados de segurança (Habilitada), Número de usuários ativos mensalmente (MAU) (1000)
Amazon Simple Email Service (SES)	US$ 0,00	US$ 0,03	Mensagens de e-mail enviadas do EC2 (por mês), Mensagens de e-mail enviadas do cliente de e-mail (300 por mês)
Amazon API Gateway	US$ 0,00	US$ 4,25	Unidades de solicitações da API HTTP (milhões), Unidades de solicitação da API REST (milhões), Tamanho da memória do cache (GB) (Nenhum), Unidades de mensagens WebSocket (milhares), Tamanho médio da mensagem (32 KB), Solicitações (por mês), Solicitações (1 por mês)

ANEXO C

Na Tabela C.1 temos os serviços AWS que foram utilizados nas arquiteturas, bem como suas funções.

Tabela C.1 - Descrição dos serviços AWS utilizados na arquitetura

Representação	Nome	Função
	Route53	Atua como DNS (Domain Name System - Sistema de nomes de domínio), atribuindo endereço de domínio da plataforma ao portal e à API
	S3 Bucket	S3 (Simple Storage Service - Serviço de armazenamento simples) armazena arquivos na AWS, pode ser utilizado para armazenar desde arquivos para uma página web, até arquivos de backup, data lakes, entre outros
	CloudFront	O CloudFront acelera a distribuição de conteúdo web para os usuários. Quando um usuário solicita um conteúdo disponibilizado pelo CloudFront, a solicitação é roteada para o servidor com menor latência, caso o conteúdo já esteja presente nesse servidor, o CloudFront o entregará imediatamente, caso contrário, ele fará uma cópia do conteúdo da origem, e o disponibilizará. Esse serviço ajuda a diminuir a latência e melhorar o desempenho das aplicações web
	API Gateway	O API Gateway serve para criação, publicação, manutenção, monitoramento e proteção de APIs e tem integração com outros serviços de autenticação da AWS, como o Amazon Cognito
	Amazon Cognito	O Amazon Cognito gerencia, autentica e autoriza usuários
	Amazon SES	O Amazon SES (Simple E-mail Service - Serviço simples de E-mail) é uma plataforma de envio e recebimento de e-mail

Representação	Nome	Função
	Lambda	As funções Lambda são funções computacionais que permitem a execução de códigos em resposta ao acionamento de diversos outros serviços AWS, tais como API Gateway, S3, DynamoDB, entre outros
	RDS	O RDS (Relational Database Service - Serviço de banco de dados relacional) fornece banco de dados relacional, para persistência de dados
	EC2	O EC2 (Elastic Compute Cloud) são máquinas remotas em *data centers* da Amazon que fornecem capacidade computacional redimensionável, que podem ser utilizadas como servidores

Fonte: AWS (2024)