UNIVERSIDADE FEDERAL DE ALFENAS

DANIEL DA COSTA LIMA

RYAN RODRIGUES

BANCOS DE DADOS NÃO RELACIONAIS EM APLICAÇÕES MÓVEIS DE GERENCIAMENTO DE TAREFAS: APLICATIVO MESADA

ALFENAS/MG

2025

DANIEL DA COSTA LIMA
RYAN RODRIGUES

BANCOS DE DADOS NÃO RELACIONAIS EM APLICAÇÕES MÓVEIS DE GERENCIAMENTO DE TAREFAS:

APLICATIVO MESADA

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Alfenas.

Orientador: Prof. Eliseu César Miguel

ALFENAS/MG
2025

Sistema de Bibliotecas da Universidade Federal de Alfenas
Biblioteca Unidade Educacional Santa Clara

Lima, Daniel da Costa.
Bancos de Dados Não Relacionais em Aplicações Móveis de Gerenciamento de Tarefas : Aplicativo Mesada / Daniel da Costa Lima, Ryan Rodrigues. - Alfenas, MG, 2025.
16 f. : il. -

Orientador(a): Eliseu César Miguel.
Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Federal de Alfenas, Alfenas, MG, 2025.
Bibliografia.

1. Banco de dados não relancionais. 2. NoSQL. 3. Aplicativo Móvel. 4. Gerenciamento de Tarefas. 5. MongoDB. I. Rodrigues, Ryan. II. Miguel, Eliseu César, orient. III. Título.

Ficha gerada automaticamente com dados fornecidos pelo autor.

DANIEL DA COSTA LIMA
RYAN RODRIGUES

BANCOS DE DADOS NÃO RELACIONAIS EM APLICAÇÕES MÓVEIS DE GERENCIAMENTO DE TAREFAS: APLICATIVO MESADA

O Presidente da banca examinadora abaixo assina a aprovação do Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Alfenas.

Aprovada em: 10 de Dezembro de 2025

Prof. Eliseu César Miguel Assinatura:
Presidente da Banca Examinadora
Universidade Federal de Alfenas

gov.br
Documento assinado digitalmente
ELISEU CESAR MIGUEL
Data: 16/12/2025 11:24:15-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Luiz Eduardo da Silva Assinatura:
Universidade Federal de Alfenas

Prof. Romário da Silva Borges Assinatura:
Universidade Federal de Alfenas

Dedicamos este trabalho aos nossos pais e familiares, base de nossa formação, pelo amor incondicional, incentivo constante e por acreditarem em nossos sonhos mesmo nos momentos mais difíceis.

AGRADECIMENTOS

Agradecemos primeiramente a Deus, por nos dar força e saúde.

Ao nosso orientador, Prof. Eliseu César Miguel, pela paciência, pelas correções precisas e por nos guiar neste caminho acadêmico.

À Universidade Federal de Alfenas (UNIFAL-MG), em especial ao Departamento de Ciência da Computação, e a todos os professores que contribuíram para nossa formação.

Aos nossos colegas de curso e amigos, pelo companheirismo durante a graduação.

E à nossa família, por nos apoiar nessa jornada.

“Todos merecem a chance de voar.”

(Wicked - Stephen Schwartz)

RESUMO

Embora bancos de dados relacionais sejam amplamente utilizados em sistemas de gerenciamento de tarefas, ainda há pouca exploração do potencial de modelos não relacionais (*NoSQL*) nesse tipo de aplicação. Com esse contexto, este trabalho apresenta o desenvolvimento de um aplicativo móvel denominado *Mesada*, utilizado como estudo de caso para avaliar a viabilidade do uso de bancos orientados a documentos em sistemas dessa natureza. O projeto foi implementado em arquitetura cliente-servidor, empregando *.NET Core* no *back-end*, *React Native* no *front-end* e *MongoDB Atlas*® como base de dados. A modelagem orientada a documentos mostrou-se adequada para representar dados semiestruturados e acompanhar as mudanças estruturais necessárias ao longo do desenvolvimento. Os resultados obtidos reforçam o potencial das tecnologias *NoSQL* em aplicações que demandam flexibilidade e evolução contínua.

Palavras-chave: Banco de dados não relacional; NoSQL; Aplicativo móvel; Gerenciamento de tarefas; MongoDB.

ABSTRACT

Although relational databases are widely used in task management systems, the potential of non-relational (NoSQL) models in this type of application remains underexplored. In this context, this work presents the development of a mobile application called Mesada, used as a case study to evaluate the feasibility of employing document-oriented databases in such systems. The project was implemented using a client-server architecture, with .NET Core on the back end, React Native on the front end, and MongoDB Atlas®as the database. The document-oriented modeling proved suitable for representing semi-structured data and accommodating structural changes throughout the development process. The results obtained reinforce the potential of NoSQL technologies in applications that require flexibility and continuous evolution.

Keywords: Non-relational database; NoSQL; Mobile application; Task management; MongoDB.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE LISTAGENS

SUMÁRIO

1 INTRODUÇÃO

A organização e acompanhamento de tarefas constituem elementos centrais na gestão de atividades em diferentes contextos institucionais, profissionais e sociais, sobretudo diante da crescente complexidade dos fluxos de trabalho contemporâneos (Tureta; Júnior, 2020; Huang; Jin, 2023). Em ambientes corporativos, especialmente em equipes submetidas a regimes de turnos ou com alta rotatividade, ferramentas de controle de tarefas tornam-se essenciais para assegurar previsibilidade, coordenação e eficiência operativa (Ajiga *et al.*, 2024; Jr. *et al.*, 2024).

Em contextos sociais e domésticos, o gerenciamento de tarefas também desempenha papel relevante, especialmente no ambiente familiar. Práticas como a administração de mesadas e a delegação de responsabilidades domésticas têm sido associadas ao desenvolvimento de competências socioeducativas, como disciplina, senso de responsabilidade e autonomia em crianças e adolescentes (Drummond, 2014). Nesse cenário, o gerenciamento de tarefas em ambientes familiares apresenta-se como ferramenta útil para apoiar rotinas organizacionais e incentivar comportamentos estruturados desde a infância.

Com o avanço da computação móvel e a ampla disseminação de dispositivos como *smartphones* e *tablets* (Moreira *et al.*, 2024), tornou-se tecnicamente viável a criação de soluções que automatizam o gerenciamento de tarefas e promovem maior acessibilidade à informação. Aplicativos móveis têm demonstrado potencial para consolidar dados, centralizar comunicações e reduzir a sobrecarga cognitiva de usuários em atividades organizacionais cotidianas (Freitas; Bianchi, 2018). Esse cenário favorece o surgimento de sistemas voltados à organização de tarefas em domicílios, instituições educacionais e grupos sociais.

Paralelamente, a estruturação e o armazenamento dos dados manipulados por essas soluções automatizadas exigem abordagens compatíveis com a natureza dinâmica e semiestruturada das informações tratadas. Nesse contexto, bancos de dados não relacionais *(NoSQL)* surgem como alternativa viável em despeito dos modelos relacionais tradicionais, por dispensarem esquemas rígidos e permitirem o armazenamento direto de documentos com estrutura variada (Capris *et al.*, 2022; Mihai, 2020). O *MongoDB Atlas*®, sistema de gerenciamento de banco de dados orientado a documentos, tem se destacado nesse cenário por sua integração nativa com aplicações *web* e móveis, além da alta flexibilidade do modelo de dados e do suporte ao armazenamento distribuído (Capris *et al.*, 2022).

Apesar do crescente uso de bancos *NoSQL* em sistemas modernos, observa-se na literatura técnica uma predominância de estudos e implementações que adotam bancos de dados relacionais em aplicações de gerenciamento de tarefas. São escassas as abordagens que tratam da aplicação prática de bancos orientados a documentos nesse tipo de solução, o que evidencia uma possível lacuna relevante e pouco explorada. Nesse sentido, o presente trabalho propõe investigar, por meio do desenvolvimento do aplicativo *Mesada*, como um banco de dados não relacional pode atender às necessidades estruturais e operacionais de um sistema móvel voltado à organização de tarefas.

O presente artigo tem como objetivo demonstrar a viabilidade da utilização de um banco de dados não relacional em aplicações móveis voltadas ao gerenciamento de tarefas. Para isso, foi desenvolvido o aplicativo *Mesada*, concebido como um estudo de caso em âmbito familiar, que implementa um ambiente real de organização de atividades com regras dinâmicas e pontuações personalizadas. A proposta não se limita à construção de funcionalidades, mas busca demonstrar como o modelo orientado a documentos se adapta de forma mais eficiente a cenários com estrutura de dados variável, evitando a complexidade de esquemas rígidos e múltiplas junções exigidas por bancos relacionais convencionais.

A solução desenvolvida permite a criação de grupos, o registro e a atribuição de tarefas a usuários e o controle de pontuação com base em ações positivas ou negativas. A modelagem em documentos *JSON-like* viabilizou a representação direta dessas estruturas, permitindo, por exemplo, o armazenamento de listas de tarefas personalizadas por usuário, sem necessidade de múltiplas tabelas ou junções complexas. Esse nível de flexibilidade estrutural foi particularmente relevante na manipulação de dados semiestruturados, como registros de desempenho e variações de pontuação atribuídas por diferentes administradores.

A aplicação adota uma arquitetura cliente-servidor, composta por um *back-end* desenvolvido com *.NET Core*, um *front-end* em *React Native* e armazenamento persistente na plataforma *MongoDB Atlas*®. A comunicação entre as camadas é realizada por meio de uma *API REST (Representational State Transfer Application Programming Interface)*, que gerencia as requisições e garante a consistência das operações. Essa composição tecnológica demonstra, na prática, como é possível integrar ferramentas amplamente utilizadas no mercado com bancos de dados não relacionais, resultando em uma solução funcional, escalável e aderente a cenários reais de organização de tarefas.

Além do desenvolvimento da solução proposta, o trabalho também proporcionou aos autores a oportunidade de explorar tecnologias *NoSQL*, ainda pouco abordadas na matriz curricular do curso. Essa vivência ampliou a compreensão sobre modelos modernos de armazenamento e contribuiu para a formação prática e teórica em um tema relevante para sistemas distribuídos contemporâneos.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados, destacando soluções semelhantes existentes na literatura. A Seção 3 descreve os recursos tecnológicos. Já a Seção 4 detalha os requisitos, a modelagem de dados e a implementação da aplicação. Por fim, a Seção 5 apresenta as conclusões e sugestões para trabalhos futuros.

2 TRABALHOS RELACIONADOS

A literatura apresenta soluções para o gerenciamento de finanças e tarefas baseadas em diferentes arquiteturas e estratégias de armazenamento. Nesta seção, essas soluções são agrupadas de acordo com suas abordagens de persistência, de modo a facilitar a comparação com a proposta do *Mesada*.

2.1 APLICAÇÕES COM PERSISTÊNCIA LOCAL

O trabalho de Oliveira (Oliveira, 2020) apresenta uma solução *mobile* para gerenciamento de tarefas e compromissos pessoais, desenvolvida em *Java* e destinada exclusivamente à plataforma *Android*. O sistema utiliza *SQLite* como banco de dados local, priorizando uso *offline* e controle individual, com possibilidade de *backup* e compartilhamento esporádico via *Firebase*.

De forma semelhante, o aplicativo de Debastiani (Debastiani, 2015) é uma aplicação *desktop* implementada em *Delphi*, voltada à organização de atividades e rotinas domésticas. Assim como no trabalho anterior, adota *SQLite* como mecanismo de persistência local, o que limita a sincronização entre dispositivos e reduz a escalabilidade.

Esses trabalhos demonstram que é possível desenvolver aplicações funcionais utilizando tecnologias acessíveis e independentes da nuvem. Em contraste, o *Mesada* avança ao adotar um banco de dados não relacional em nuvem, explorando recursos de atualização contínua e suporte a operações em tempo real.

2.2 APLICAÇÕES COM ARMAZENAMENTO HÍBRIDO

O projeto *Flynow*, de Bitarães (Bitarães, 2020), propõe um aplicativo móvel que incentiva usuários a cumprir metas e consolidar hábitos por meio de gamificação. A arquitetura utiliza *React Native* no *front-end* e *Node.js* no *back-end*, com comunicação via *APIs RESTful*.

Para a persistência, o sistema adota uma abordagem híbrida: *PostgreSQL* armazena dados centrais, como usuários, tarefas e metas, enquanto o *MongoDB* é utilizado exclusivamente para informações de ranqueamento e notificações, organizando os dados conforme suas características operacionais.

Em comparação, o *Mesada* simplifica a arquitetura ao utilizar apenas o *MongoDB Atlas*® como solução unificada. A centralização elimina a necessidade de integração entre diferentes bancos, reduz a complexidade do desenvolvimento e facilita a manutenção.

2.3 APLICAÇÕES COM BANCOS *NOSQL* EM NUVEM

O *Study Buddy*, de Lacerda (Lacerda, 2024), busca auxiliar estudantes na organização do tempo e na redução de distrações, combinando bloqueio de aplicativos com técnicas como o método Pomodoro (Cirillo, 2006). A solução foi desenvolvida exclusivamente para *iOS*, utilizando *Swift* e *Xcode*.

A persistência na nuvem ocorre por meio do *Firestore*, banco de dados *NoSQL* do *Firebase*, que armazena apenas informações básicas, como nome de usuário e pontuação, enquanto os demais dados permanecem no dispositivo. O trabalho destaca a praticidade do *Firestore* para operações *CRUD (create, read, update* e *delete)* e sua integração simplificada com o ecossistema *iOS*.

Outro trabalho relevante é o projeto *Guilda dos Universitários*, de Ticianeli (Ticianeli, 2023), que utiliza elementos de jogo para auxiliar estudantes na organização de tarefas e rotinas acadêmicas. Desenvolvido para *Android*, o sistema emprega o *Firebase Realtime Database*, integrado à autenticação *Google*, permitindo sincronização em tempo real das tarefas entre os usuários.

Esses trabalhos evidenciam a adoção crescente de bancos de dados não relacionais em soluções móveis. O *Mesada* se insere nesse cenário ao explorar o *MongoDB* como única solução de armazenamento, destacando sua aplicabilidade para operações contínuas, em nuvem e em múltiplos dispositivos.

2.4 SÍNTESE

De modo geral, a literatura apresenta soluções que variam entre persistência local, arquiteturas híbridas e abordagens totalmente baseadas em bancos *NoSQL*. Ao adotar uma arquitetura unificada e inteiramente não relacional, o *Mesada* se diferencia por simplificar o fluxo de dados e demonstrar a viabilidade de uma solução integralmente em nuvem para o gerenciamento de tarefas e finanças no contexto móvel.

3 RECURSOS TECNOLÓGICOS

Esta seção apresenta os principais recursos utilizados no desenvolvimento do aplicativo *Mesada*, contemplando as tecnologias empregadas no *front-end, back-end*, banco de dados e ferramentas de apoio. O objetivo é contextualizar o ambiente de desenvolvimento e justificar, de forma objetiva, as escolhas realizadas ao longo do projeto.

3.1 *FRONT-END*

O aplicativo móvel foi desenvolvido utilizando *React Native*, *framework* que permite criar interfaces multiplataforma a partir de um único código-fonte. A escolha se deve à sua produtividade no desenvolvimento, à ampla disponibilidade de bibliotecas e à facilidade de integração com *APIs REST*, característica importante para o funcionamento do sistema. A interface foi estruturada com componentes reutilizáveis, visando clareza visual e consistência entre telas.

3.2 *BACK-END*

A camada de serviços foi implementada em *.NET Core*, responsável por concentrar as regras de negócio e expor os *endpoints* consumidos pelo aplicativo. O *framework* oferece organização modular, suporte nativo a *APIs REST* e ferramentas robustas para validação e tratamento de requisições, o que permitiu estruturar a lógica do sistema de forma clara e segura. A comunicação com o banco de dados ocorre por meio de operações diretas, sem abstrações complexas, mantendo o fluxo simples e alinhado ao modelo orientado a documentos.

3.3 BANCO DE DADOS

O armazenamento dos dados foi realizado no *MongoDB Atlas*®, plataforma em nuvem para bancos não relacionais. O modelo baseado em documentos se mostrou adequado às entidades do sistema, caracterizadas por estruturas flexíveis, listas internas e dados sujeitos a alterações frequentes. A instância do banco foi hospedada na *Azure*, utilizando créditos acadêmicos

concedidos por meio da parceria entre a UNIFAL e a plataforma[1]. Esses recursos permitiram o uso de uma solução em nuvem segura, gerenciada e sem custos adicionais ao projeto.

A manipulação dos documentos foi feita diretamente pela *API*, aproveitando a flexibilidade das coleções e a possibilidade de reorganização das estruturas sem necessidade de migrações. Essa abordagem atendeu bem à dinâmica do aplicativo, no qual tarefas, grupos e pontuações são atualizados constantemente.

3.4 FERRAMENTAS DE APOIO

Algumas ferramentas complementares foram utilizadas durante o desenvolvimento:

- ***Figma***®: criação dos protótipos e definição inicial das interfaces.
- ***Visual Studio Code*® *e Visual Studio***®: ambientes utilizados para o desenvolvimento do aplicativo e da *API*.
- ***Swagger***®: documentação e testes dos *endpoints* disponibilizados pela *API*.
- ***MongoDB Compass***®: visualização e acompanhamento das coleções armazenadas no banco de dados.

Essas ferramentas facilitaram o fluxo de trabalho e contribuíram para a organização das etapas de desenvolvimento, mas sem interferir diretamente nas decisões de modelagem e arquitetura descritas na Seção 4.

Para sintetizar os principais componentes tecnológicos adotados no projeto, a Tabela 1 apresenta um resumo das ferramentas, linguagens e plataformas utilizadas em cada camada do sistema.

[1] Informações sobre a parceria encontram-se detalhadas na página https://www.unifal-mg.edu.br/nti/microsoft-365

Tabela 1 – Resumo das tecnologias utilizadas na implementação do *Mesada*

Camada	**Tecnologia**	**Finalidade**
Front-End	*React Native*	Interface móvel multiplataforma (*Android*/*iOS*)
Back-End	*.NET Core 8.0*	*API REST* responsável pela lógica de negócio e acesso ao banco
Banco de Dados	*MongoDB Atlas®*	Armazenamento orientado a documentos, com estrutura flexível
Ferramentas de Apoio	*Figma®, VS Code®, Swagger®, Compass®*	Prototipação, desenvolvimento, documentação e inspeção do banco

Fonte: Autores (2025).

4 DESENVOLVIMENTO

Esta seção apresenta o processo de construção do aplicativo *Mesada*, descrevendo as etapas que envolveram a definição dos requisitos, a modelagem dos dados, a escolha da arquitetura e a implementação dos principais componentes do sistema. O objetivo é detalhar como o projeto foi estruturado, desde as decisões iniciais até a composição final da solução, evidenciando o percurso técnico adotado ao longo do desenvolvimento. As subseções a seguir descrevem cada uma dessas etapas de forma organizada.

4.1 METODOLOGIA DE DESENVOLVIMENTO

O desenvolvimento do sistema seguiu uma adaptação simplificada do modelo cascata, no qual as etapas foram executadas de forma linear, mas com espaço para pequenos ajustes conforme o projeto avançava. Essa abordagem permitiu organizar o fluxo de trabalho de forma clara, guiando o desenvolvimento desde a definição inicial até a implementação final.

O processo foi estruturado nas seguintes etapas:

1. **Definição dos requisitos**: identificação das funcionalidades essenciais e dos objetivos do sistema.
2. **Elaboração dos protótipos**: criação das telas iniciais no *Figma*®.
3. **Implementação da *API***: desenvolvimento das rotas e da lógica de comunicação com o banco de dados.
4. **Implementação do aplicativo móvel**: construção das telas e integração com a *API*.
5. **Validação e ajustes finais**: testes gerais e refinamento do sistema.

De forma geral, essa metodologia proporcionou um desenvolvimento organizado e coerente, garantindo que cada etapa fosse concluída antes da seguinte, ao mesmo tempo em que permitiu pequenos refinamentos quando necessários.

4.2 REQUISITOS DO SISTEMA

A definição dos requisitos funcionais, como apresentado na Tabela 2, priorizou operações que explorassem características dinâmicas de armazenamento e manipulação de dados,

típicas de bancos *NoSQL*. Essas características incluem a capacidade de lidar com estruturas flexíveis e atualizações frequentes sem necessidade de reestruturação do esquema.

Tabela 2 – Requisitos funcionais do sistema *Mesada*. Os requisitos priorizam a manipulação dinâmica de dados e o suporte à estrutura orientada a documentos.

Código	**Descrição**
RF01	Permitir o cadastro e autenticação de usuários, com armazenamento de informações pessoais em documentos independentes.
RF02	Viabilizar a criação de grupos familiares, associando usuários e tarefas por meio de referências diretas entre coleções.
RF03	Permitir o registro, atribuição e controle de tarefas personalizadas, com valores e regras ajustáveis por usuário.
RF04	Controlar pontuações positivas e negativas, registrando cada atualização em um histórico de interações persistido no banco de dados.
RF05	Gerar relatórios de desempenho com base em dados agregados, obtidos diretamente das coleções sem a necessidade de junções complexas.

Fonte: Autores (2025)

Além das funcionalidades descritas, foram definidos requisitos não funcionais diretamente relacionados também à viabilidade técnica do modelo proposto:

- **Escalabilidade:** O sistema deve permitir o crescimento do volume de dados sem degradação significativa de desempenho, aproveitando o particionamento horizontal nativo do *MongoDB*.
- **Consistência eventual:** As operações de atualização devem garantir consistência lógica dos dados, mesmo em cenários assíncronos de múltiplos usuários.
- **Portabilidade:** O sistema deve operar de forma integrada com dispositivos móveis *Android* e *iOS*, mantendo comunicação padronizada via *API*.

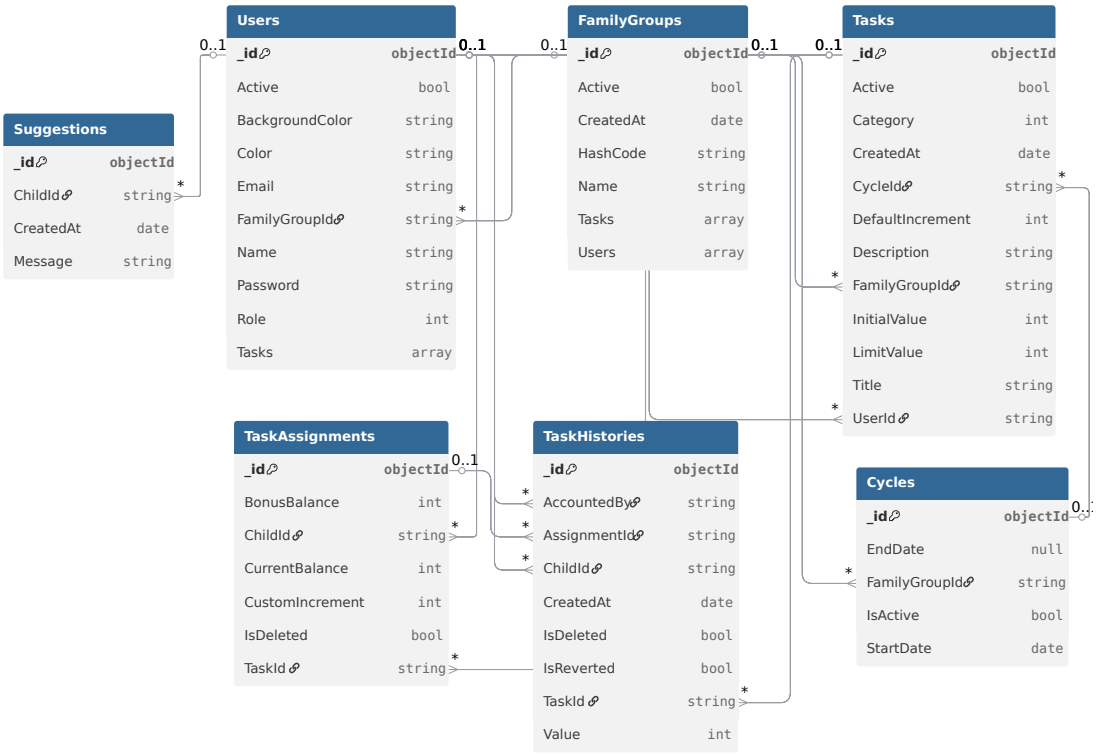
Em síntese, esses requisitos, estruturados para explorar cenários típicos de manipulação dinâmica de dados, servem como base para a modelagem e arquitetura apresentadas na subseção seguinte.

4.3 MODELAGEM DE DADOS COM *MONGODB*

A modelagem de dados do sistema foi construída de forma a explorar as características do banco *NoSQL* adotado, com ênfase na flexibilidade estrutural, na representação direta das relações e na minimização de operações de junção. As principais entidades do domínio — usuários, grupos familiares, tarefas e histórico — foram organizadas em coleções independentes, conectadas entre si por meio de identificadores do tipo *ObjectId*.

A Figura 1 apresenta a visão geral da modelagem lógica do sistema. Nela, é possível observar como as coleções se relacionam por meio de referências diretas, indicando que o vínculo entre usuários, grupos e tarefas não depende de tabelas intermediárias, mas sim de identificadores únicos que conectam os documentos.

Figura 1 – Diagrama lógico do banco de dados *MongoDB* utilizado no sistema



Fonte: Autores (2025).

Com base nessa estrutura, a Listagem 1 apresenta um exemplo real de documento da coleção `FamilyGroups`, ilustrando como o relacionamento representado no diagrama se materializa na prática. O documento reúne, em um único registro, os vínculos com usuários e tarefas do grupo, refletindo a abordagem orientada à agregação que fundamenta o modelo.

Listagem 1 – Exemplo real de documento da coleção *FamilyGroups*, ilustrando o uso de referências (*ObjectId*) para associação entre usuários e tarefas.

```
{
  "_id": { "$oid": "682e73f01a53971e263dd407" },
  "Name": "Familia Rodrigues",
  "HashCode": "#WH63J2",
  "Users": [
    "682e73b01a53971e263dd406",
    "682e73a21a53971e263dd405"
  ],
  "Active": true,
  "CreatedAt": { "$date": "2025-05-21T00:00:00.000Z" },
  "Tasks": [
    "682e73b01a5397112d1dd406",
    "682e73a26543971e263dd405"
  ]
}
```

Fonte: Autores (2025).

Esse exemplo evidencia dois aspectos centrais do uso de bancos *NoSQL*:

1. **Agregação de relacionamentos via *arrays* de referências.** As listas `Users` e `Tasks` evitam a criação de tabelas intermediárias e permitem acessar, em uma única leitura, os identificadores das entidades relacionadas. A obtenção dos dados completos cabe à aplicação, que realiza consultas adicionais quando necessário — uma operação que, em um modelo relacional, normalmente exigiria múltiplas junções.

2. **Referências diretas por *ObjectId*.** Em vez de armazenar objetos completos dentro de `FamilyGroups`, utilizam-se identificadores únicos. Isso preserva consistência, simplifica atualizações e reduz redundância.

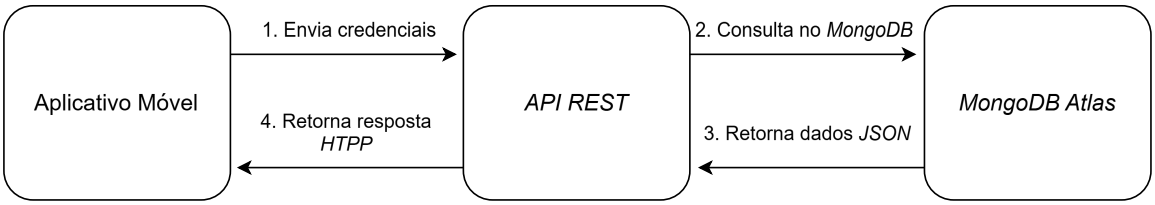
A combinação do diagrama lógico com o exemplo prático reforça a adequação do modelo orientado a documentos ao contexto do aplicativo, especialmente em um domínio caracterizado por relacionamentos muitos-para-muitos e pela necessidade de evolução contínua da estrutura de dados ao longo do desenvolvimento de maneira direta e com capacidade de adaptação a mudanças frequentes durante o desenvolvimento.

4.4 ARQUITETURA DA APLICAÇÃO

A arquitetura do sistema foi organizada seguindo o modelo tradicional de três camadas, separando responsabilidades entre o aplicativo móvel, a *API* responsável pelas regras de negócio e o banco de dados. Essa divisão visa estruturar o fluxo de comunicação de forma simples e compreensível, permitindo que cada parte da aplicação opere de maneira independente.

A Figura 2 apresenta a visão geral da arquitetura utilizada no desenvolvimento do aplicativo *Mesada*.

Figura 2 – Diagrama da arquitetura cliente-servidor do aplicativo Mesada, mostrando o fluxo de comunicação entre o aplicativo móvel, a *API* e o *MongoDB*, desde o envio das requisições até o retorno das respostas em formato *JSON*.



Fonte: Autores (2025).

O aplicativo móvel envia as ações do usuário para a *API* por meio de requisições *HTTP* seguindo o padrão *REST*. A *API* concentra a lógica de negócio, valida as informações recebidas e realiza a comunicação com o banco de dados. O armazenamento é feito em um banco não relacional, no qual a *API* executa diretamente as operações de leitura e escrita necessárias ao funcionamento do sistema.

No conjunto, essa arquitetura distribui de forma clara as responsabilidades de apresentação, processamento e armazenamento, mantendo o sistema organizado e facilitando o desenvolvimento das funcionalidades ao longo do projeto.

4.5 IMPLEMENTAÇÃO DAS FUNCIONALIDADES

Esta subseção apresenta como as principais funcionalidades do aplicativo foram implementadas a partir das decisões descritas nas seções anteriores. O objetivo é evidenciar a relação entre a modelagem dos dados, a arquitetura adotada e o funcionamento real do sistema, demonstrando como cada componente contribui para a operação final do aplicativo.

A criação e gerenciamento dos grupos familiares utilizam diretamente a estrutura do documento `FamilyGroups`. A *API* realiza operações de leitura e escrita sobre listas internas de usuários e tarefas, refletindo a modelagem orientada à agregação discutida na Seção 4.3. Essa abordagem permite que as

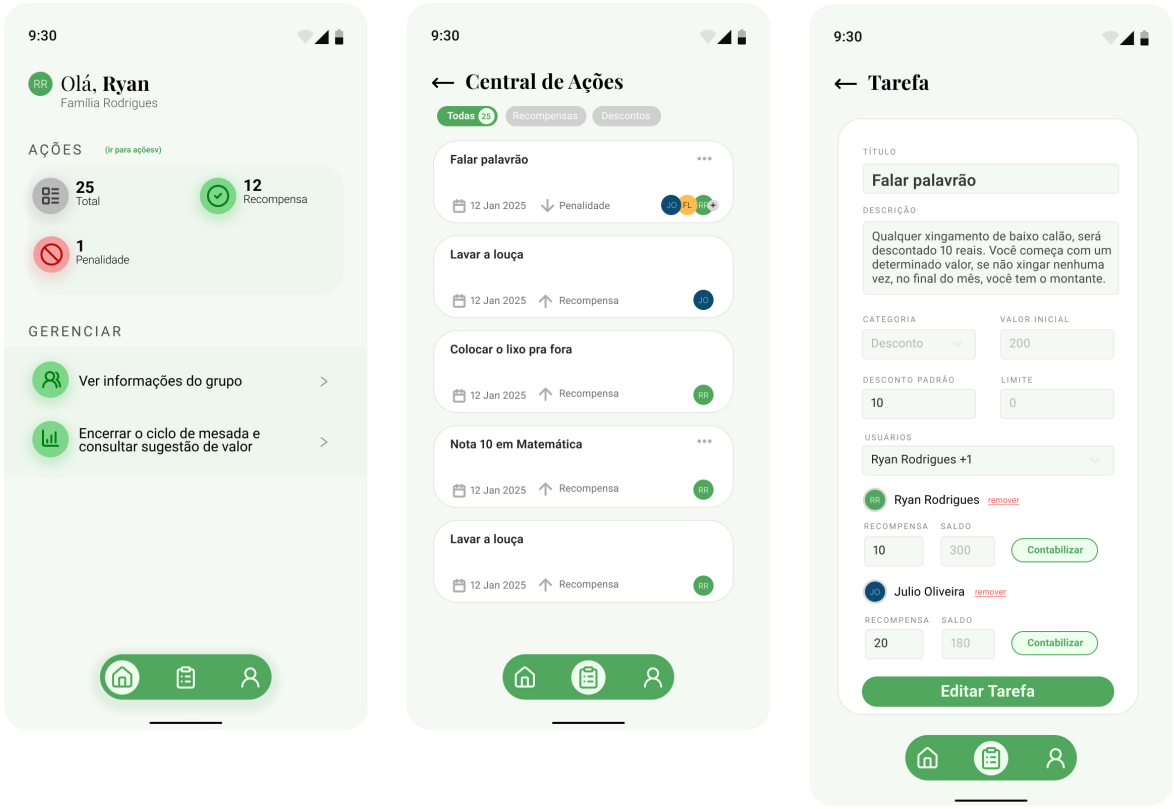
associações entre membros do grupo sejam obtidas de forma direta, uma vez que os identificadores das entidades relacionadas já estão agregados no próprio documento do grupo.

O cadastro e controle de tarefas seguem o mesmo princípio. Ao criar uma tarefa, o aplicativo envia para a *API* os atributos definidos pelo usuário, que são incorporados ao documento correspondente no banco de dados. As atualizações de pontuação, por sua vez, são registradas no histórico interno de cada tarefa, permitindo rastrear alterações sem a necessidade de coleções auxiliares.

A autenticação dos usuários e o gerenciamento das informações pessoais foram implementados utilizando documentos individuais, mantendo apenas referências nos grupos e tarefas. Isso reduz a duplicação de dados e facilita ajustes no perfil do usuário sem impacto estrutural nas demais coleções.

Para ilustrar a implementação visual das funcionalidades descritas, a Figura 3 apresenta algumas telas da versão final do aplicativo, destacando exemplos do fluxo de interação entre usuário e sistema.

Figura 3 – Exemplos de telas do aplicativo *Mesada*: (a) visão geral das ações e acesso às funcionalidades principais do grupo; (b) listagem das tarefas registradas no sistema, refletindo a estrutura de documentos utilizada na modelagem; (c) tela de edição de tarefa, evidenciando o vínculo entre usuários, valores e histórico de pontuações conforme implementado na *API*



Fonte: Autores (2025).

A implementação consolidou as decisões de projeto tomadas anteriormente, demonstrando que o modelo de dados, a *API* e o aplicativo interagem de maneira coerente e adequada ao contexto proposto. As telas apresentadas ilustram de forma clara os fluxos essenciais do sistema, contemplando a criação, listagem e edição das principais entidades.

4.6 COMPARAÇÃO ENTRE BANCOS DE DADOS E ANÁLISE DOS RESULTADOS

A adoção do *MongoDB* no desenvolvimento do aplicativo *Mesada* não pressupõe superioridade em relação ao modelo relacional. Ambos os paradigmas apresentam vantagens e limitações, e a escolha depende do domínio do problema e do ciclo de evolução do sistema. A Tabela 3 resume aspectos relevantes para o cenário do projeto, servindo como ponto de partida para os exemplos comparativos apresentados a seguir.

Tabela 3 – Comparação entre características de bancos relacionais e não relacionais

Aspecto	**Modelo Relacional (*SQL*)**	**Modelo Não Relacional (*NoSQL*)**
Estrutura dos Dados	Esquema rígido, normalização estruturada.	Esquema flexível, permitindo variação entre documentos.
Relacionamentos	Relações formais via chaves estrangeiras e junções.	Referências manuais; junções controladas pela aplicação.
Consultas	Junções complexas são tratadas nativamente.	Consultas simples são diretas; consultas avançadas exigem múltiplas operações.
Evolução do Esquema	Mudanças exigem migrações estruturadas.	Alterações incrementais sem migrações formais.
Consistência	Forte, com integridade garantida pelo banco.	Consistência forte por documento; integridade referencial depende da aplicação.
Adequação ao Projeto	Ideal para dados estáveis e fortemente estruturados.	Ideal para dados dinâmicos e que podem ser agrupados.

Fonte: Autores (2025).

A Tabela 3 sintetiza diferenças conceituais relevantes entre os modelos relacional e não relacional no contexto do projeto. Com base nesses pontos, as subseções a seguir apresentam duas situações reais do sistema, comparando consultas *SQL* e *NoSQL* aplicadas a cenários distintos da aplicação. O objetivo é ilustrar, de forma prática, como cada abordagem se comporta diante de necessidades específicas do domínio do *Mesada*.

4.6.1 Consulta estruturada a partir de documentos agregados

Na subseção 4.3 foi apresentada a modelagem orientada a documentos do sistema, onde grupos familiares concentram listas de usuários e tarefas. Este é um caso típico de estrutura agregada, que pode ser representada como um único documento no *MongoDB*. A seguir, comparamos como essa mesma necessidade é atendida nos dois modelos.

Para recuperar as tarefas de um grupo no modelo relacional, geralmente utiliza-se uma tabela associativa que representa o vínculo entre grupos e tarefas, conforme o exemplo abaixo:

Listagem 2 – Consulta *SQL* utilizando tabela intermediária

```sql
SELECT T.*
FROM Tasks T
JOIN GroupTasks GT ON GT.TaskId = T.Id
WHERE GT.GroupId = 12;
```

Fonte: Autores (2025).

No modelo não relacional adotado no *Mesada*, essa associação é representada de forma diferente. Em vez de uma tabela intermediária, o próprio documento do grupo armazena as referências para usuários e tarefas, conforme modelado na Seção 4.3:

Listagem 3 – Documento *MongoDB* resumido representando um grupo com referências

```
{
  "_id": "682e7...",
  "Name": "Familia Souza",
  "Users": ["612e3...", "616e3..."],
  "Tasks": ["612e3", "692e5", "612e5"]
}
```

Fonte: Autores (2025).

Com as referências armazenadas diretamente no documento do grupo, a aplicação pode recuperar a lista de identificadores e, em seguida, buscar os detalhes completos das tarefas associadas. No *MongoDB*, isso é feito em duas etapas: uma consulta `findOne` para obter o documento do grupo e outra `find` utilizando o operador `$in` para retornar apenas as tarefas cujos identificadores constam no *array* `Tasks`.

Listagem 4 – Consulta equivalente em *MongoDB*

```
db.FamilyGroups.findOne(
  { _id: "682e7..." },
  { Tasks: 1 }
);

db.Tasks.find(
  { _id: { $in: ["612e3...", "692e5...", "612e5..."] } }
);
```

Fonte: Autores (2025).

Discussão: Ao contrário do que pode parecer inicialmente, esse padrão não reduz o número de consultas: em vez de uma junção automática, temos múltiplas buscas controladas pela aplicação.

O ganho do *NoSQL* aqui está em outro ponto:

- A estrutura do documento acompanha o domínio lógico (um grupo contém suas entidades);
- Mudanças no formato das listas podem ser aplicadas sem migrações;
- O aplicativo controla como compor os dados, oferecendo liberdade para ajustes rápidos.

Embora esse exemplo específico não apresente redução no número de consultas, vale destacar que o *MongoDB* poderia simplificar a operação dependendo da modelagem escolhida. Caso as tarefas fossem armazenadas de forma embutida dentro do documento do grupo (e não apenas referenciadas), toda a informação poderia ser obtida em uma única leitura. Por outro lado, essa prática aumentaria o tamanho dos registros e poderia gerar atualizações redundantes, comprometendo a escalabilidade em cenários de alto volume ou com mudanças frequentes.

4.6.2 Consultas envolvendo entidades associativas

A coleção *TaskAssignments*, responsável por armazenar o progresso do usuário em cada tarefa, representa um tipo de entidade que não pertence exclusivamente a usuários ou tarefas. Trata-se de um cenário conceitualmente semelhante ao das tabelas associativas dos bancos relacionais, onde a própria entidade intermediária armazena informações adicionais. Por esse motivo, *SQL* e *MongoDB* convergem para estruturas e operações bastante próximas.

Para ilustrar essa equivalência, vejamos o exemplo da operação seguinte no modelo relacional:

Listagem 5 – Atualização de pontuação em *SQL*

```sql
UPDATE TaskAssignments
SET CurrentBalance = CurrentBalance + 10
WHERE ChildId = 5 AND TaskId = 32;
```

Fonte: Autores (2025).

No *MongoDB*, a operação equivalente segue o mesmo princípio: localizar o vínculo específico entre usuário e tarefa e atualizar o campo desejado.

Listagem 6 – Atualização equivalente em *MongoDB*

```
db.TaskAssignments.updateOne(
  { ChildId: "612e4", TaskId: "652e4" },
  { $inc: { CurrentBalance: 10 } }
);
```

Fonte: Autores (2025).

Discussão: Nesse cenário, observa-se que:

- Ambos os modelos utilizam uma entidade própria intermediária;
- A estrutura conceitual é praticamente a mesma;
- O *MongoDB* não oferece vantagem estrutural significativa;
- O *SQL* possui mecanismos nativos de integridade referencial que o *NoSQL* delega à aplicação.

Assim, observamos que nenhum dos modelos apresenta diferenças significativas para nossa aplicação que justifiquem a escolha de um em detrimento ao outro. Podemos, portanto, considerá-los adequados como elementos técnicos para nosso caso de estudo.

4.6.3 Análise dos Resultados

A análise do funcionamento do *Mesada* indica que o modelo adotado foi adequado ao fluxo operacional do aplicativo. Todas as funcionalidades previstas — criação de grupos, associação de usuários, registro de tarefas e atualização de pontuações — foram implementadas de forma consistente com a modelagem proposta, sem contradições ou necessidade de reestruturar coleções.

A avaliação é qualitativa, pois o escopo do trabalho não incluiu métricas formais de desempenho. Ainda assim, foram observados os seguintes pontos positivos na prática:

- Ajustes na estrutura dos documentos (como inclusão de novos campos) puderam ser realizados sem migrações e impacto significativo na *API*;
- A separação em coleções independentes refletiu corretamente o domínio da aplicação;
- O uso de referências diretas manteve a consistência dos vínculos entre entidades, sem duplicação excessiva de dados;
- A *API* respondeu de forma estável às operações esperadas para o cenário de uso, caracterizado por carga leve e interações frequentes.

Conclui-se, portanto, que o modelo orientado a documentos atendeu plenamente às necessidades do *Mesada* e trouxe vantagens claras de agilidade durante o desenvolvimento, principalmente por sua flexibilidade estrutural e facilidade de evolução. Um banco relacional seria igualmente viável do ponto de vista funcional, ainda que exigisse mais esforços na manutenção do esquema ao longo das iterações do projeto. Assim, a escolha entre *SQL* e *NoSQL* deve considerar as características específicas do domínio e a frequência de mudanças esperada para o sistema.

5 CONCLUSÃO

O desenvolvimento do aplicativo *Mesada* possibilitou avaliar o uso de um modelo de banco de dados não relacional em um contexto de gerenciamento de grupos familiares, tarefas e pontuações. A estrutura baseada em documentos mostrou-se adequada às necessidades do sistema, oferecendo flexibilidade para reorganizações durante o desenvolvimento e permitindo que as principais funcionalidades fossem implementadas sem dependência de operações complexas.

Nossa análise demonstrou que as decisões de modelagem e arquitetura refletiram-se de maneira consistente no funcionamento do sistema. As operações de criação de grupos, registro de tarefas e atualização de pontuações ocorreram de forma estável e previsível, indicando que o modelo adotado foi efetivo para o padrão de uso proposto. Além disso, ajustes internos nas coleções puderam ser incorporados sem necessidade de migrações estruturais, reforçando a adequação do modelo ao processo iterativo de desenvolvimento.

Embora a análise tenha tido caráter predominantemente qualitativo, dado que o escopo do trabalho não contemplou testes quantitativos de desempenho ou cenários de maior carga, as observações feitas ao longo da implementação demonstram que o modelo não relacional atendeu de forma satisfatória às necessidades da aplicação desenvolvida.

O desenvolvimento do aplicativo também possibilitou aos autores um contato mais aprofundado com tecnologias *NoSQL*, que não são amplamente exploradas na estrutura curricular da instituição. Essa vivência prática contribuiu para consolidar conhecimentos relacionados a modelos de dados modernos, reforçando o potencial dessas tecnologias em cenários aplicados.

Como trabalhos futuros, recomenda-se a realização de experimentos comparativos entre modelos relacionais e não relacionais, especialmente em cenários de maior volume de dados ou múltiplos usuários simultâneos. A incorporação de funcionalidades adicionais no sistema, como autenticação baseada em *JWT*, controle mais refinado de permissões e suporte a notificações em tempo real, também pode ampliar o entendimento sobre o comportamento do sistema e fortalecer a análise realizada. Essas extensões poderão complementar os resultados obtidos e contribuir para uma avaliação mais abrangente do uso de bancos não relacionais em aplicações móveis.

REFERÊNCIAS

AJIGA, D. *et al.* The role of software automation in improving industrial operations and efficiency. **International Journal of Engineering Research Updates**, v. 7, aug 2024.

BITARãES, R. J. R. **Desenvolvimento de um aplicativo móvel para gestão de tarefas, hábitos e metas utilizando elementos de gamificação**. Ouro Preto, 2020. 73 f. Orientadora: Valéria de Carvalho Santos.

CAPRIS, T. *et al.* **Comparison of SQL and NoSQL databases with different workloads: MongoDB vs MySQL evaluation**. 2022.

CIRILLO, F. **The Pomodoro Technique**. New York: Crown Publishing, 2006.

DEBASTIANI, J. A. **Aplicativo desktop para gerenciamento de atividades e planejamento domésticos**. Pato Branco, 2015. 41 f. Orientador: Beatriz Terezinha Borsoi.

DRUMMOND, A. F. d. **Participação de crianças e de adolescentes nas tarefas domésticas**. 125 p. Tese (Tese (Doutorado em Ciências da Reabilitação)) — Universidade Federal de Minas Gerais, Belo Horizonte, fev. 2014. Orientadora: Marisa Cotta Mancini. Coorientadora: Ana Maria Rabelo.

FREITAS, R. A. d. C. d.; BIANCHI, L. C. D. A tecnologia nos negócios: análise da influência do celular na produtividade organizacional. **Research, Society and Development**, v. 7, sep 2018.

HUANG, B.; JIN, Y. Social learning in self-organizing systems for complex assembly tasks. **Advanced Engineering Informatics**, v. 57, aug 2023.

JR. *et al.* Evaluating workflow automation efficiency in a government agency in the philippines. **International Journal of Entrepreneurship and Sustainability Studies**, v. 4, dec 2024.

LACERDA, M. C. S. d. **Desenvolvimento de um app de gerenciamento de tempo: Study Buddy**. João Pessoa, 2024. 69 f. Orientador: Carlos Eduardo Coelho Freire Batista.

MIHAI, G. Comparison between relational and nosql databases. **Annals of Dunarea de Jos University of Galati. Fascicle I. Economics and Applied Informatics**, v. 26, dec 2020.

MOREIRA, S. *et al.* Tecnologia ao toque: Desvendando o impacto dos smartphones e tablets na revolução do aprendizado contemporâneo. **Revista Foco**, v. 17, jan 2024.

OLIVEIRA, J. A. d. **Aplicativo Android para o gerenciamento de tarefas e compromissos pessoais**. Pato Branco, 2020. 51 f. Orientadora: Andreia Scariot Beulke.

TICIANELI, G. H. G. **Guilda dos universitários: um aplicativo gamificado para organização de tarefas e rotinas acadêmicas**. Bauru, 2023. 47 f. Orientador: Juliana da Costa Feitosa.

TURETA, C.; JúNIOR, C. C. Organizing professionalism: integrating institutional logics in brazilian law firms. **Management Research Review**, may 2020.