INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TRINGULO MINEIRO
CAMPUS PARACATU
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

BRUNO RÉGIS MACHADO DE OLIVEIRA

AUTOMATIZAÇÃO DE INTERAÇÃO DE USUÁRIOS NA RESOLUÇÃO DE CONFLITOS VOLUNTÁRIOS

PARACATU
2022

BRUNO RÉGIS MACHADO DE OLIVEIRA

AUTOMATIZAÇÃO DE INTERAÇÃO DE USUÁRIOS NA RESOLUÇÃO DE CONFLITOS VOLUNTÁRIOS

Trabalho de conclusão apresentado ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro - IFTM, como requisito à obtenção do Certificado de Conclusão do Curso em ADS.

Orientador: Prof. João Felipe Souza

PARACATU
2022

Ficha Catalográfica elaborada pelo Setor de Referência do IFTM – Campus Paracatu

O48a Bruno Régis Machado de Oliveira -
Automatização de Interação de Usuários na Resolução de Conflitos Voluntários / Bruno Régis Machado de Oliveira - 2022.
45 f. : il.

Orientador: João Felipe Souza.
Trabalho de conclusão de curso (graduação) – Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro, Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Paracatu, 2022.

1. Software de aplicação. 2. Back-End. 3. React native. 4. Arbitragem. I. João Felipe Souza. II. Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro - Campus Paracatu. III. Título.

CDD 005.3

Bibliotecária: Nathália de Morais Torres CRB6-3097

BRUNO RÉGIS MACHADO DE OLIVEIRA

AUTOMATIZAÇÃO DE INTERAÇÃO DE USUÁRIOS NA RESOLUÇÃO DE CONFLITOS VOLUNTÁRIOS

Trabalho de conclusão apresentado ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro - IFTM, como requisito à obtenção do Certificado de Conclusão do Curso em ADS.

Data da Defesa: 08 de Julho de 2022
Conceito: Excelente

Banca Examinadora

Prof. João Felipe Souza
Instituto Federal do Triângulo Mineiro - IFTM
Orientador

Prof. Edwar Saliba Júnior
Instituto Federal do Triângulo Mineiro - IFTM
Membro da Banca

Prof. Pedro Henrique Tomás
Instituto Federal do Triângulo Mineiro - IFTM
Membro da Banca

PARACATU
2022

Dedicado esse trabalha a uma pessoa em especial: minha mãe, exemplo de persistência, determinação e amor. Aos professores, meus mestres que tanto contribuíram para enriquecer meus conhecimentos acadêmicos. Agradeço pela minha saúde e sabedoria! A todos, meu obrigado!

"É durante nossos momentos mais sombrios que devemos nos concentrar para ver a luz."
Aristotle Onassis

RESUMO

As disputas estão se tornando cada vez mais internacionais, complexas e cada vez mais dispendiosas, enquanto há um aumento burocrático, que acaba tornando ainda mais lento e custoso. No Brasil o tempo gasto no sistema judiciário para a resolução de um conflito simples é imenso. O objetivo desse trabalho foi desenvolver um aplicativo que tornasse mais acessível, rápido e gratuito o processo de resolução de conflitos voluntários. O modelo jurídico escolhido foi a arbitragem pela sua viabilidade perante a proposta inicial. O sistema foi desenvolvido usando o React Native com o Expo para o Front-End da aplicação, o Back-End foi feito usando o Node.js e o banco de dados escolhido por sua maleabilidade foi o MongoDB. Ao final da pesquisa, pode se concluir que a ODR é uma forma viável para a resolução de conflitos, os contratos inteligentes são uma ótima opção por possuírem regras claras, facilitando a solução de conflitos. E com a virtualização da sociedade futuramente grande parte das resoluções de conflitos será online. O aplicativo de arbitragem foi concluído no prazo esperado e completamente funcional. O desenvolvimento em React Native é mais rápido que na linguagem nativa, o Node.js possui um custo mais elevado de operação, o MongoDB pode se tornar complexo em um banco de dados com muitas relações.

Palavras-chave: Arbitragem. Back-End. React Native. Front-End. MongoDB. Node.js.

ABSTRACT

Disputes are becoming increasingly international, complex and increasingly expensive, while bureaucratic increases, which end up making it even slower and more costly. In Brazil, the time spent in the judicial system to resolve a simple conflict is immense. The objective of this work was to develop an application that would make the voluntary conflict resolution process more accessible, fast and free. The legal model chosen was arbitration for its feasibility in the face of the initial proposal, the system was developed using React Native with Expo for the Front-End of the application, the Back-End was made using Node.js and the database chosen for its malleability was MongoDB. At the end of the research, it can be concluded that ODR is a viable way to resolve conflicts, smart contracts are a great option because they have clear rules, facilitating conflict resolution. And with the virtualization of society, in the future, most of the conflict resolutions will be online. The arbitration application was completed on time and fully functional. Development in React Native is faster than in native language, Node.js has a higher cost of operation, MongoDB can become complex in a database with many relations.

Keywords: Arbitration, Back-End, React Native, Front-End, MongoDB and Node.js.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidad, Consistência Isolamento Durabilidade.
ADR	Alternative Dispute Resolution
JSI	Javascript Interface
NoSQL	Not only Standard Query Language
ODR	Online Dispute Resolution
OOHDM	Object Oriented Hypermedia Design Method
RDBMS	Relational Database Management Systems
SQL	Standard Query Language
TIC	Tecnologia da informação e comunicação
UML	Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

A Revolução Digital é o constante desenvolvimento das tecnologias de comunicação, mudou a forma como nos comportamos, como interagimos, nos redefiniu como seres humanos. Pensamos de forma diferente, trabalhamos de forma diferente e nos relacionamos de formas diferentes, porém o sistema jurídico bem como outros aspectos do Estado se manteve iguais ao longo do tempo.

Com a facilidade e rapidez das comunicações, provocou um aumento das interações, elevando também de forma proporcional o número de disputas e o crescimento dos litígios.

Segundo estatísticas do Conselho Nacional da Justiça (CNJ), retratada no documento "Justiça em números", o mais completo diagnóstico disponível acerca dos Tribunais brasileiros, durante o ano de 2017 foram iniciados 29,1 milhões de processos no país, enquanto encerrados 31 milhões, ou seja, o Poder Judiciário decidiu 6,5% a mais de processos do que a demanda de casos novos (ANDREIA, 2020, p. 767).

Dado o alto custo e a burocracia dos processos judicial, as disputas privadas vem se tornando mais comuns, e como consequência da globalização as disputas estão se tornando cada vez mais internacionais, complexas e cada vez mais incluem processos multipartidários e reivindicações. Enquanto as restrições de custo permanecem rigorosas e as demandas por resolução rápida aumentam. A grande maioria das disputas privadas, são resolvidas através de arbitragem. Um método para remediar a duração excessiva e os custos da arbitragem é recorrer a ferramentas digitais para acelerar e facilitar a arbitragem. As ferramentas digitais têm, de fato, o potencial de reduzir custos e tempo exponencialmente e, assim manter um nível necessário de acesso à justiça. (KAUFMANN-KOHLER, 2005, p. 437).

Como essa modernização vem ocorrendo com diversas instituições tradicionais como bancos e serviços de entrega. É cabível que essa atualização venha a ocorrer com organizações mantidas pelo Estado, como o sistema de justiça. Essa mudança vem ocorrendo em decorrência das diversas vantagens trazidas pelos aplicativos, em relação à forma tradicional de interação com o cliente, possibilitando uma maior velocidade e maior acesso a serviços.

Um dos expoentes da ODR são as plataformas de vendas online que lidam diariamente com conflitos. Segundo Ricardo Marques, na LAWTECH CONFERENCE, a empresa mercado livre atingiu o nível de 98,9% de "de juridicização", ou seja, usando o ODR (resolução de disputas online) sem a necessidade de usar um sistema de justiça estatal.

"A ODR pode levar a uma fidelização do consumidor. Existem benefícios de custo, de eficiência e também ecológicos, pois ninguém se desloca para uma audiência. A resolução online de disputa faz sentido especialmente para as demandas de consumo, que geralmente são mais simples e podem ser resolvidas sem o judiciário" (MARQUES, 2019)

"Mais de 1.800 processos foram extintos, levando a uma economia de cerca de R$ 2 mi-

lhões" (MARQUES, 2019) Como apresentado na Figura 1 no primeiro semestre de 2021, foram gastos R$ 320 bilhões nas lojas de aplicativos da Google e Apple. Em termos de comparação ao mesmo período de 2020 foram gastos R$ 265 bilhões, esse aumento mostra a importância dos aplicativos no mundo atualmente. E justifica a criação de um aplicativo ODR.

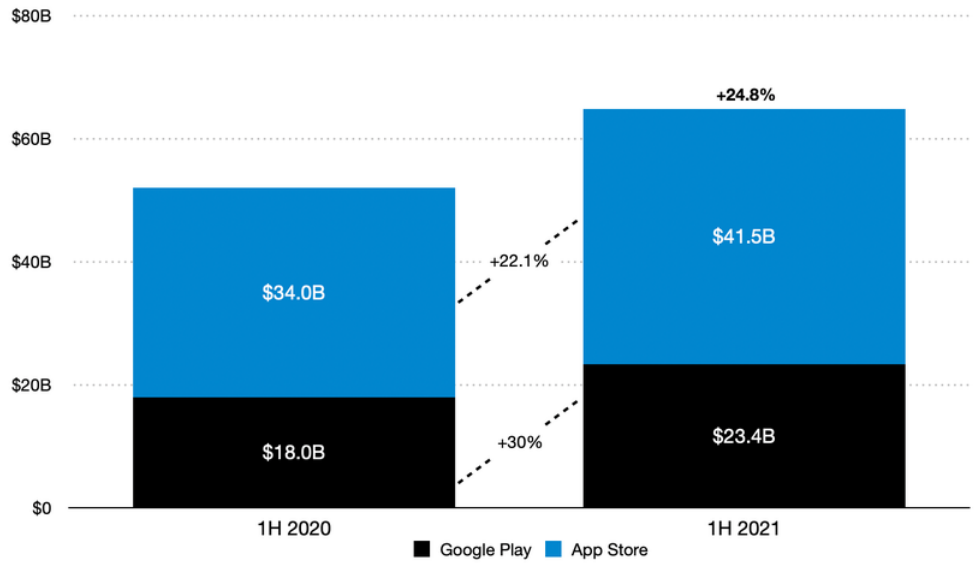


Figura 1 – Gastos globais do consumidor em aplicativos para dispositivos móveis no primeiro semestre de 2020 em comparação com ao mesmo período de 2021
Fonte: SENSORTOWER, 2021

1.1 Problema

A arbitragem não é o domínio mais inovador em termos de tecnologia uma vez que se manteve em um grau de inércia por um longo período, tornando-se obsoleto quando se trata de avanços tecnológicos.

A pandemia foi uma das responsáveis por acelerar a digitalização dos meios de arbitragem e justiça. Utilizando sistemas de Online Dispute Resolution(ODR) que vem tornaram-se revolucionários na arbitragem, como a Índia, Emirados Árabes Unidos, Hong Kong e Cingapura sendo essa última pioneira em seu uso (SHAH, 2004, p 4). Os ODRs prometem resoluções democratizadas, acessíveis, econômicas e rápidas. No entanto, essa revolução foi prevista décadas atrás, quando iniciativas como o Virtual Magistrate Project em 1996 (SHAH, 2004, p 3) e o Net Case em 2005 foram definidas para atuar como catalisadores para isso. Eles tiveram pouco sucesso. Dito isso, muitas disputas nunca chegam ao mundo da arbitragem formalizado, como tal, ainda pode haver espaço para o crescimento do ODR.

O ODR pode ser útil hoje, em pequenas transações online, mas frequentes, como compras no comércio eletrônico. Esse tipo de disputa, no entanto, provavelmente não precisará de advogados, árbitros ou especialistas.

Portanto, o principal problema tanto da arbitragem como da justiça, atualmente é o excesso de burocracia, complexidade e alto custo de advogados, árbitros e juízes. O que inviabiliza completamente pequenas disputas de baixo custo.

1.2 Justificativa

Esse trabalho justifica-se pelo valor elevado e tempo gasto nas disputas litigiosas, bem como a burocracia atrelada a esses procedimentos. Criando se a necessidade de uma alternativa acessível à resolução de conflitos de pequena intensidade.

1.3 Objetivos

1.3.1 Geral

Desenvolver um aplicativo de resolução de conflitos usando React Native, Node.js e MongoDB, a fim de tornar a resolução de conflitos acessível ao público em geral.

1.3.2 Específicos

- Debater métodos de resolução de conflito atuais.
- Analisar a viabilidade da automação em um sistema de arbitragem.
- Avaliar o uso das tecnologias React Native, Node.js e MongoDB na criação de um aplicativo para Android.
- Desenvolver um aplicativo de arbitragem.

2 REFERENCIAL TEÓRICO

Certamente, pode-se dizer que o uso da tecnologia aumentou o acesso à justiça. Existem várias maneiras pelas quais a tecnologia pode ajudar a tornar a arbitragem mais eficiente, oportuna e econômica. Já se dedica ao arquivamento e entrega de documentos eletrônicos, bem como à divulgação assistida por tecnologia, audiências telefônicas e evidências de vídeo-link. “As possibilidades oferecidas pelas (Tecnologia da informação e comunicação) TICs possuem poder cada vez maior de manipular e, em certo grau, mesmo remover os obstáculos de tempo e espaço” (KATSH, 2012).

Em destaque as cinco áreas que nos parecem ter uma necessidade particular de resolução de disputas online (ODR) e-commerce, saúde, mídia social, trabalho e tribunais (KATSH, 2017, p 173).

A resolução de disputas online pode ser definida da seguinte forma: “Refere-se a uma ampla classe de processos alternativos de resolução de disputas que aproveitam a disponibilidade e o desenvolvimento crescente da tecnologia da Internet.” (NENSTIEL, 2006, p. 15).

Resolução de disputas online, ou ODR, refere-se a um amplo conjunto de tecnologias destinadas a complementar ou substituir as maneiras pelas quais as pessoas tradicionalmente resolvem suas disputas. O ODR compartilha e se baseia nas características fundamentais da resolução alternativa de conflitos, ou ADR, enfatizando métodos mais fáceis e eficientes de lidar com conflitos.

O processo de arbitragem é menos formal do que uma audiência no tribunal ou julgamento (e geralmente menos dispendioso), porém mais formal do que a mediação ou negociação (KAUFMANN-KOHLER, 2005, p. 443).

“Até certo ponto, isso aconteceu; mediação, arbitragem e outras abordagens extrajudiciais tornaram-se opções de resolução de disputas usadas com muito mais frequência” (KATSH, 2017, p. 14).

Os desenvolvedores de software geralmente têm três maneiras principais de criar um aplicativo. Eles podem optar por codificar um aplicativo nativo, um aplicativo híbrido ou um aplicativo Web progressivo. Os desenvolvedores criam aplicativos nativos para funcionar em uma plataforma específica, geralmente iOS ou Android. Eles criam esses aplicativos usando Swift ou Objective C para iOS. Para Android, eles usam JAVA, Kotlin ou várias outras linguagens. Os desenvolvedores também criam aplicativos híbridos usando linguagens como JavaScript e HTML.

O desenvolvimento híbrido começa com a criação de um Back-End comum e, em seguida, o skinning do aplicativo para iOS e Android. As principais linguagens híbridas são o React Native, Ionic e Flutter. Os aplicativos híbridos podem ser usados tanto no iOS ou Android, agilizando o desenvolvimento.

Por simplificar em várias etapas o desenvolvimento o React Native permite facilitar e agilizar o desenvolvimento como citado no artigo de HANSSON 2013: “O desenvolvimento de um aplicativo React Native foi surpreendentemente fácil e o processo rápido, o que resultou na conclusão do aplicativo antes do esperado” (HANSSON, 2016, p. 53).

React Native é uma biblioteca JavaScript para o desenvolvimento de aplicativos móveis desenvolvida pelo Facebook para seu desenvolvimento interno de aplicativos. Capaz de construir a camada de visualização de aplicativos da web e móveis. Construindo uma estrutura mais robusta para seus aplicativos, reutilizando componentes nativos (HANSSON, 2016, p. 45).

Geralmente é usado para desenvolvimento de aplicativos em grande escala, especialmente para plataformas de streaming de vídeo, aplicativos dinâmicos de página única e outros aplicativos da web. O Node.js é orientado por eventos, o que a torna a escolha certa para os aplicativos de tempo real que usam muitos dados (DARSKI, 2016, p. 7).

MongoDB é um banco de dados orientado a documentos sem esquema. O nome MongoDB vem de “humongous”. O banco de dados foi projetado para ser escalonável e escrito em C++. O principal motivo para abandonar o modelo relacional é tornar o dimensionamento mais fácil. (KHAN, 2013, p. 2)

3 METODOLOGIA

Para o desenvolvimento desse trabalho foi empregado duas metodologias, sendo elas uma pesquisa exploratória e uma pesquisa bibliográfica. Com o objetivo de identificar os melhores métodos de resolução de disputa, bem como a eficiência das tecnologias usadas no desenvolvimento do mesmo.

A pesquisa exploratória, tem a finalidade de desenvolver uma solução, realizando testes no software apresentado e nas tecnologias utilizadas. “A pesquisa exploratória busca apenas levantar informações sobre um determinado objeto, delimitando assim um campo de trabalho, mapeando as condições de manifestação desse objeto. Na verdade, ela é uma preparação para a pesquisa explicativa.” (SEVERINO, 2017)

Já para a pesquisa bibliográfica, foi analisando os dados referentes a livros, artigos e monografias. Além disso, foi realizado o desenvolvimento de um software usando o método Ágil, que emprega uma forma de desenvolvimento incremental. As tecnologias usadas foram o React native, Node.js e MongoDB.

3.1 Resolução de disputas online

As origens do ODR seguem de perto a história das interações digitais, particularmente as transações comerciais. À medida que o volume de interações aumentava, também aumentava o volume de disputas e a necessidade de reparação nativa da própria Internet. Embora a Internet remonte ao final dos anos 60. Dessa forma, as queixas muitas vezes podiam ser resolvidas off-line, já que as partes, provavelmente estavam nos mesmos círculos do mundo real. Atualmente, podemos realizar interações com pessoas em qualquer parte do mundo através da internet, dessa forma a ferramenta ODR torna-se imprevisível.

Segundo os dados apresentados na Figura 2 (STATISTA, 2013), o número de litígios oriundos do comércio eletrônico necessariamente aumentará nos próximos anos. Eles também mostram que a Internet faz parte do nosso dia a dia. Considerando essas duas observações em conjunto, não há dúvida de que o ODR tem um papel crescente a desempenhar. Essa função dependerá do tipo de disputa e do método de resolução de disputa.

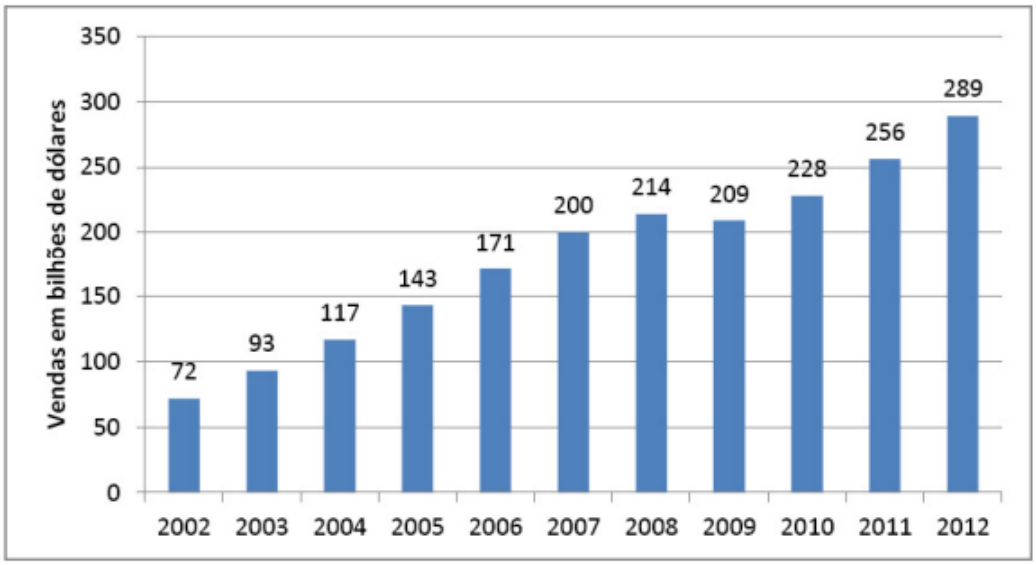


Figura 2 – Vendas anuais do e-comemerce nos Estados Unidos entre 2002 e 2012
Fonte: STATISTA, 2013

Disputas decorrentes de grandes transações comerciais internacionais, que constituem a maior parte do número de casos de arbitragem tradicional, provavelmente não serão encaminhadas ao ODR. Essas disputas irão gradativamente assimilar as tecnologias para a melhora da administração das arbitragens. Os valores em jogo não servirão de incentivo para substituir as audiências ao vivo por e-mails e chats.

> "É provável que métodos não vinculativos de resolução de disputas continuem a prevalecer no ODR - um reflexo do que o filósofo jurídico francês Mireille Delmas-Marty chama de 'véritable triomphe du mou, du flou, du doux' (suavemente, 'o verdadeiro vitória do soft law ') 59 Esta é uma tendência geral do direito contemporâneo e uma das explicações para o sucesso do ADR, o que certamente é uma reação às ineficiências da justiça tradicional, inclusive da arbitragem clássica. Mas é ao mesmo tempo também o reflexo de uma mudança na função do juiz ou de qualquer juiz, cada vez mais chamado a avaliar, aconselhar, conciliar e não apenas a tomar decisões" (KAUFMANN-KOHLER, 2005, p. 455).

3.2 Arbitragem

A arbitragem é um método de resolução de disputas fora do tribunal. As partes encaminham suas disputas a um árbitro que analisa as evidências, ouve as partes e, em seguida, toma uma decisão.

> "A mediação e a arbitragem são mais flexíveis ao permitir que o terceiro neutro determine como gerenciar a comunicação e usar as informações. A comunicação e o processamento de informações, as principais capacidades tanto

> dos computadores quanto dos profissionais de resolução de disputas, devem estar no centro da solução de problemas com base na tecnologia" (KATSH, 2017, p. 16).

3.2.1 Arbitragem Obrigatória

Na arbitragem obrigatória, a decisão do árbitro é final. Não pode ser revisado ou anulado por um tribunal, exceto em circunstâncias muito limitadas, como quando há envolvimento de fraude ou abuso de poder.

3.2.2 Arbitragem Voluntária

Na arbitragem voluntária, ambos os lados em uma disputa concordam em submeter sua discordância à arbitragem depois que ela surgir e após terem avaliado outras opções para resolvê-la. A maioria dos defensores do consumidor considera esta a abordagem preferida e imparcial para a arbitragem, permitindo que seja uma escolha em vez de uma necessidade.

Na arbitragem não obrigatória, qualquer uma das partes pode rejeitar a sentença arbitral e exigir um julgamento em seu lugar. As partes geralmente tratam as decisões não obrigatórias como uma avaliação independente dos pontos fortes e fracos de um processo judicial em potencial, com o objetivo de promover um acordo. Mas, mesmo em tais casos, a convenção de arbitragem frequentemente estipulará que a sentença pode se tornar obrigatória se as partes concordarem com ela ou esperar mais do que o tempo determinado para solicitar que o caso seja devolvido ao tribunal.

3.2.3 Escolha de Árbitros

Frequentemente, uma convenção de arbitragem designará uma das grandes agências de arbitragem como a National Arbitration Forum ou o JAMS para lidar com a arbitragem. Os árbitros nas listas desses grupos geralmente são juízes aposentados ou advogados em exercício, embora alguns não sejam advogados, possuem especialização em uma área como construção ou custódia de crianças, possuindo mais conhecimento em uma área específica que advogados generalistas.

Essas agências normalmente cobram um determinado valor ou porcentagem do valor em disputa como uma taxa de depósito e um valor ou porcentagem adicional como uma "taxa de serviço do caso". Por exemplo, se o valor em disputa for de 75 mil a 150.000 mil dólares uma organização de arbitragem normalmente cobraria cerca de 2.250 dólares em taxas para iniciar o processo. Essas disputas por serem internacionais a moeda mais usada na arbitragem é o dólar.

Muitas convenções de arbitragem exigem que as disputas sejam arbitradas por um dos principais grupos ou associações de arbitragem, outras permitem que as partes vão a um árbitro

independente. O aplicativo terá como principal diferencial a escolha dos árbitros independentes, que será realizada de forma aleatória a fim de atingir uma maior imparcialidade.

3.3 Desenvolvimento do Software

O método de desenvolvimento utilizado é o método Ágil, que consiste em uma abordagem iterativa e com limite de tempo para entrega de software, que constrói software de forma incremental desde o início do projeto. Requisitos, planos e resultados são avaliados continuamente para que as equipes tenham um mecanismo natural para responder às mudanças rapidamente (ABRAHAMSSON, 2017, p. 101).

Estudos mostram que as metodologias tradicionais de desenvolvimento de software baseadas em planos não são usadas na prática. Tem sido argumentado que as metodologias tradicionais são mecanicistas demais para serem usadas em detalhes. Como resultado, os desenvolvedores de software industrial tornaram-se céticos sobre as "novas" soluções que são difíceis de entender, e, portanto, permanecem sem uso. "Os métodos ágeis de desenvolvimento de software, foram oficialmente iniciados com a publicação do manifesto Agile, buscam uma mudança de paradigma no campo da engenharia de software." (ABRAHAMSSON, 2017, p. 105)

3.4 Tecnologias e Ferramentas Utilizadas

As principais tecnologias usadas atualmente para o desenvolvimento de aplicativos são o React Native, por ter um desenvolvimento mais ágil e o node.js como Back-End, por possibilitar o envio de dados em tempo real. Os bancos de dados Not only Standard Query Language(NoSQL) por possuírem maior flexibilidade na criação das tabelas, vem se tornando uma opção viável para sistemas que exigem escalabilidade.

3.5 UML

Em 1997 o Object Management Group (OMG) lançou a Unified Modeling Language (UML) (GUEDES, 2009, p. 17). Cujo objetivo era fornecer à comunidade de desenvolvimento uma linguagem de design estável e comum que pudesse ser usada para desenvolver e construir softwares. O Unified Modeling Language (UML) trouxe uma notação de modelagem padrão unificada que os desenvolvedores desejavam há anos. Usando UML, os programadores agora podem ler e disseminar a estrutura do sistema e planos de design assim como os trabalhadores da construção têm feito há anos com plantas de edifícios.

"Um diagrama de classes UML é usado para representar graficamente um modelo conceitual como uma visão estática que mostra uma coleção de elementos estáticos do domínio. Seguindo o OOHDM" (Rossi et al., 2000). O diagrama de classes foi usado para exemplificar

as diferentes entidades e suas relações. Em outros termos, mostrando as estruturas estáticas do sistema. Um diagrama de classes também pode ser usado para exibir classes lógicas, auxiliando no desenvolvimento do sistema.

3.6 React Native

O React Native foi escolhido para esse projeto por sua facilidade de desenvolvimento, pela quantidade de conteúdo disponível, pela facilidade de entendimento da documentação e pelo uso de linguagem comuns no desenvolvimento Web com o HTML, CSS e JavaScript.

Além de ser um dos Freamwork mais usados no desenvolvimento de dispositivos moveis, em 2015, o Facebook disponibilizou o React Native ao público. No repositório do React Native no Github em novembro de 2021, havia 99,6 mil estrelas e 21,5 mil bifurcações (FACEBOOK, 2021).

Nos últimos 4 anos, o framework React Native cresceu para uma comunidade de mais de 2.000 colaboradores, com uma média de mais de 300.000 downloads semanais por meio do npm. (FACEBOOK, 2015) Algumas das maiores empresas do mundo adotaram o React Native, incluindo Facebook, Instagram, Uber Eats, Pinterest e Discord. Sua ampla adoção é impulsionada principalmente pela conveniência de sua natureza multiplataforma e pela abordagem tecnológica exclusiva usada para esse fim.

Há um equívoco comum de que o React Native compila JavaScript de forma nativas, como Swift ou Java, mas esse não é o caso.

React Native é escrito, principalmente com JavaScript e classificado como uma estrutura "híbrida", o que significa que é independente de plataforma. Isso o separa de aplicativos tradicionais escritos em linguagens nativas, como Java ou Kotlin para Android e Swift ou Objective-C para Apple. Em vez disso, os aplicativos híbridos têm uma única base de código que produz um aplicativo que será executado em dispositivos Android e iOS. Os benefícios são óbvios, menos código e logística relacionada quando comparado a escrever um par de aplicativos nativos usando idiomas diferentes. As desvantagens dos híbridos são que eles não têm um desempenho tão bom quanto seus equivalentes nativos e, às vezes, não têm a capacidade de fazer uso total dos recursos de um dispositivo.

Não usando WebViews, mas sim um sistema que permite renderizar componentes nativos de seu código JavaScript base.

A maioria das estruturas híbridas como Ionic, Cordova e Phonegap, contam com o que é conhecido como WebView para realizar seus recursos de plataforma cruzada. Basicamente, eles incorporam uma página da web dentro de um aplicativo nativo e se conectam a ela para se integrar ao dispositivo subjacente. O problema com isso é que os WebViews, especialmente ao hospedar aplicativos complicados, enfrentam problemas de desempenho e têm outras limitações.

3.6.1 Desempenho

React Native não é a melhor escolha para aplicativos com requisitos de alto desempenho especificamente aqueles com cargas de trabalho graficamente intensas ou com muitos dados. Embora existam atenuações para isso como módulos nativos ou o Javascript Interface (JSI).

Um aplicativo React Native não tem um desempenho tão bom quanto um aplicativo Android nativo. No entanto, as diferenças nos testes foram pequenas e o aplicativo React Native foi capaz de desafiar o aplicativo Android de uma maneira notável (HANSSON, 2016, p. 53).

Devido à natureza desconectada e assíncrona deste meio de comunicação, podem surgir alguns problemas de desempenho como mostrado na Figura 3. As filas podem ficar cheias, caso o usuário navegue rapidamente por uma lista longa e complexa, atualizações de UI ocorrerão junto a navegação. Por um motivo semelhante, as animações também podem ser um motivo de cuidado. Na maioria das vezes, esses tipos de déficits de desempenho são insignificantes para o usuário, no entanto, eles ainda são algo que os programadores precisam estar cientes no desenvolvimento do aplicativo.

Em relação ao seu desempenho no Android, a diferença média na carga da CPU entre o React Native e o iOS nativo é pequena, cerca de 1-3

O React Native usa mais memória no início do cenário, mas também que a diferença se torna menor quanto mais o aplicativo nativo é usado. (HANSSON, 2016, p. 53).

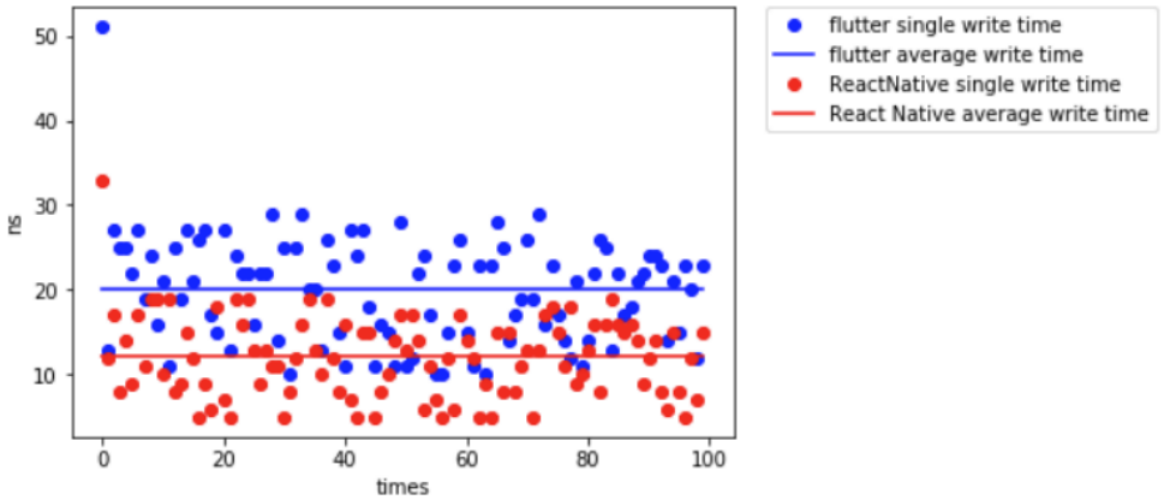


Figura 3 – Gráfico de Flutter (azul) e nativo de reação (vermelho), tempo de codificação de arquivo
Fonte: WU, 2018, p. 24

3.6.2 Comunidade

A própria documentação fornece muitos exemplos de como adicionar certas funcionalidades e junto com uma comunidade que produz muitos guias e componentes para serem usados,

o desenvolvimento é facilitado mesmo que o framework seja recente (HANSSON, 2016, p. 53).

3.6.3 Benefícios

Para o desenvolvimento de protótipos rápidos. Mesmo que você esteja pensando em usar o desenvolvimento nativo ao lançar um produto no mercado, o React Native é uma excelente opção para se desenvolver um protótipo rapidamente. Isso pode ser um teste valioso da viabilidade do framework visto que, para o seu caso de uso talvez o nativo não seja necessário afinal.

Com uma estrutura hibrida, um único código é escrito para ambas as plataformas e pelos mesmos desenvolvedores, o que significa que pode ser construído aplicativos nativos do React para Android e iOS usando os mesmos códigos. A estrutura híbrida oferece vários benefícios logísticos, uma única base de código e processo de desenvolvimento de suporte, do qual normalmente resulta uma redução nos custos.

3.7 MongoDB

Como definição, MongoDB é um banco de dados de código aberto que usa um modelo de dados orientado a documentos e uma linguagem de consulta não estruturada. É um dos bancos de dados NoSQL mais poderosos disponíveis atualmente. Que com base em uma pesquisa de 2013(Figura 4) consegue realizar consultas mais rápido que o MySQL (Khan, 2013, p. 3).

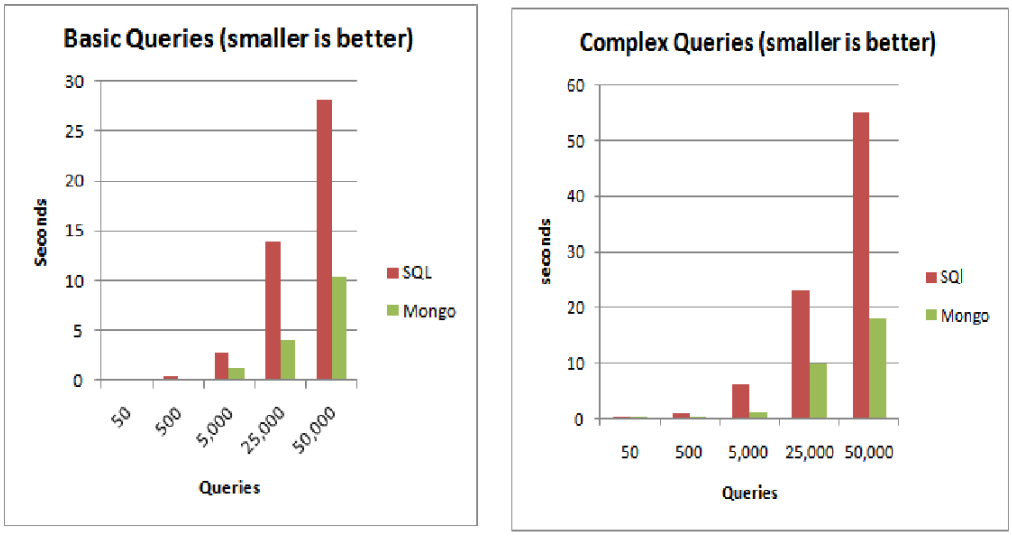


Figura 4 – Comparação de velocidade de consulta
Fonte: Khan, 2013, p. 4

No mundo dos bancos de dados, os sistemas de banco de dados mais comuns e populares são Relational Database Management Systems (RDBMS). Agora, se queremos desenvolver uma

aplicação que lida com um grande volume de dados, então precisamos escolher um banco de dados que forneça sempre soluções de armazenamento de dados de alto desempenho. Para isso, podemos atingir o desempenho da solução em termos de armazenamento e recuperação de dados com precisão, rapidez e confiabilidade. Agora, se categorizarmos as soluções de banco de dados, então existem principalmente dois tipos de categoria de banco de dados disponíveis, ou seja, RDBMS ou banco de dados relacional como SQL Server, Oracle etc. E outro tipo é banco de dados NoSQL como MongoDB, CosmosDB, entre outros.

O banco de dados NoSQL é na verdade uma alternativa ao banco de dados Standard Query Language (SQL) também convencional, este tipo de banco de dados fornece todos os tipos de recursos que estão normalmente disponíveis nos sistemas RDBMS. Atualmente, os bancos de dados NoSQL se tornam muito populares em comparação com o passado devido ao design simples, provisão para escalonamento horizontal, vertical e para controle fácil e simples sobre os dados armazenados. Esse tipo de banco de dados basicamente quebra a tradição normal de estrutura de armazenamento de dados do banco de dados relacional. Ele dá aos desenvolvedores a possibilidade de armazenar dados no banco de dados de acordo com os requisitos reais de seu programa. Essa funcionalidade, não está presente nos bancos de dados tradicionais RDBMS.

Sendo uma ferramenta NoSQL significa que ela não possui linhas e colunas, sendo uma característica muito associada aos bancos de dados relacionais. É uma arquitetura que se baseia em coleções e documentos. Ele permite que os documentos tenham diferentes campos e estruturas. Este banco de dados usa um formato de armazenamento de documentos denominado BSON, que é um estilo binário de documentos JSON.

O modelo de dados que o MongoDB segue é altamente elástico, que permite combinar e armazenar dados de tipos multivariados sem comprometer as poderosas opções de indexação, acesso a dados e regras de validação. Não há tempo de inatividade quando você deseja modificar os esquemas dinamicamente. Isso significa que se pode concentrar mais em fazer os dados trabalharem mais, em vez de gastar mais tempo preparando os dados para o banco de dados.

Podemos escolher MongoDB em vez de MySQL por causa de dois fatores: facilidade de uso e desempenho (KHAN, 2013, p. 2).

3.7.1 Benefícios

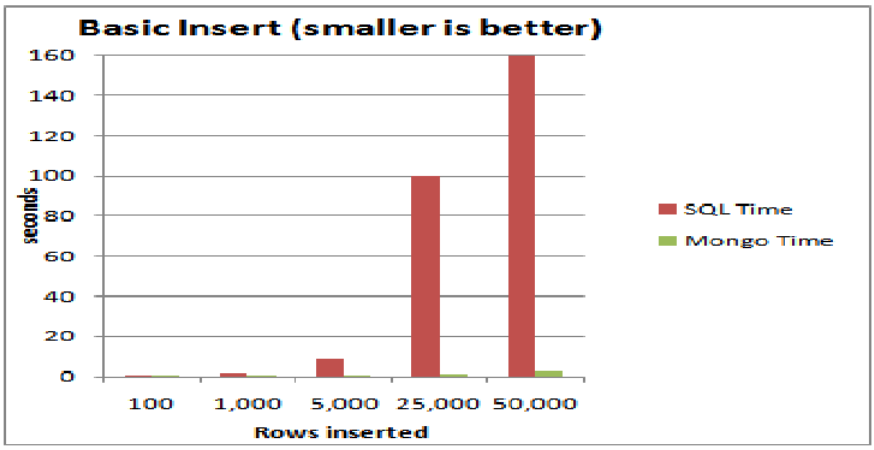
Essa tecnologia superou uma das maiores armadilhas dos sistemas de banco de dados tradicionais, ou seja, a escalabilidade. Com as necessidades em constante evolução das empresas, seus sistemas de banco de dados também precisaram ser atualizados. O MongoDB tem escalabilidade excepcional. Facilita a busca de dados e fornece integração contínua e automática. Junto com esses benefícios, existem vários motivos pelos quais você precisa do MongoDB:

Um modelo de dados flexível com esquema dinâmico e ferramentas GUI e de linha de comando poderosa, que torna mais rápido para os desenvolvedores construírem e desenvolve-

rem aplicativos. O provisionamento automatizado permite integração e entrega contínuas para operações produtivas.

O MongoDB armazena dados em documentos flexíveis do tipo JSON, o que facilita a persistência e a combinação dos dados. Os objetos em seu código de aplicativo são mapeados para o modelo de documento, por isso trabalhar com dados se torna mais fácil. Tornando a inserção de dados mais ágil como mostrado na figura 5.

Figura 5 – Comparação de velocidade de inserção



Fonte: Khan, 2013, p. 4

Não é preciso dizer que os controles de governança de esquema, o acesso a dados, as agregações complexas e a rica funcionalidade de indexação não são comprometidos de forma alguma.

Sem tempo de inatividade, pode-se modificar o esquema dinamicamente. Devido a essa flexibilidade, um desenvolvedor precisa se preocupar menos com a manipulação de dados.

Bancos de dados NoSQL e Relacionais utilizam paradigmas diferentes e, por sua vez, possuem finalidades diferentes, mas com o mesmo propósito de persistir dados. Segundo os testes de performance (Figura 6), para uma aplicação com alta carga de consultas a base de dados, como serviços Web, por exemplo, o banco MongoDB (POLITOWSKI, 2014, p. 9).

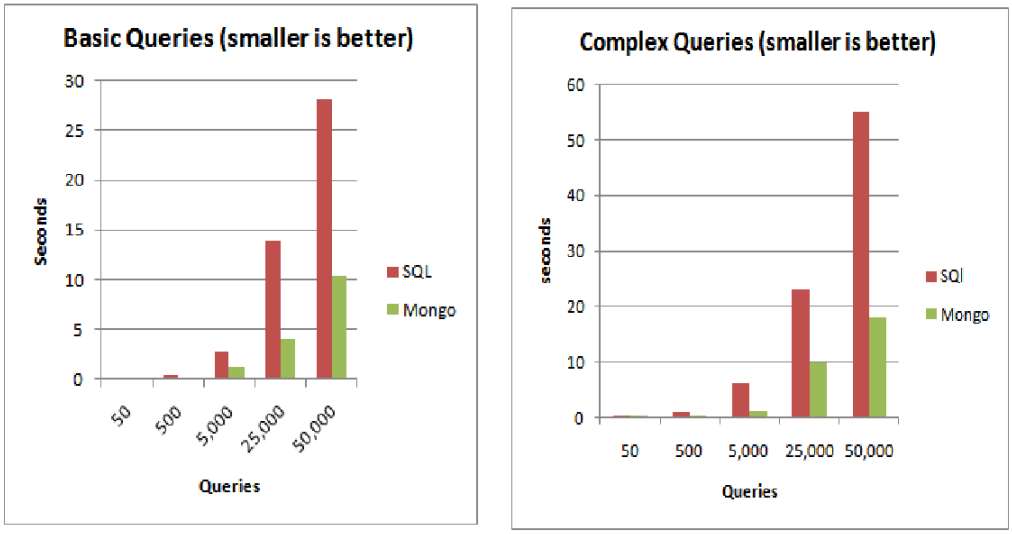


Figura 6 – Comparação velocidade de consulta
Fonte: Khan, 2013, p. 4

Possuindo velocidades de inserção maior que sistemas de banco de dados tradicionais como o PostgreSQL como é mostrado na figura 7.

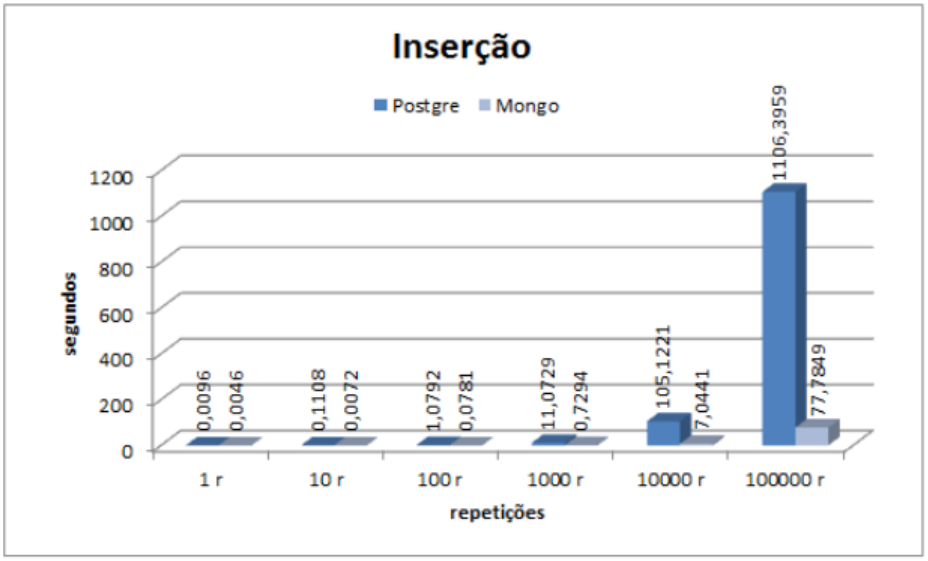


Figura 7 – Comparação de velocidade de inserção: PostgreSQL e MongoDB
Fonte: Politowski, 2014, p. 8

3.7.2 Mongoose

Uma ferramenta encarregada de conectar no banco de dados MongoDB, criar os modelos das tabelas e realiza as operações do banco de dados de forma assíncronas, usando chamadas de retorno e promessas. Instalada com o comando npm install mongoose.

3.7.3 MongoDBcompass

O software usado para gerenciar o banco de dados foi o MongoDBcompass, sendo hospedado no MongoDB Atlas, que é uma solução de banco de dados em nuvem. Ele também garante disponibilidade, escalabilidade e conformidade com os mais rigorosos requisitos de privacidade e segurança de dados.

> "Se um desenvolvedor deseja construir um aplicativo da web que seja rápido e flexível, o MongoDB é a escolha certa. Se a principal preocupação do designer do aplicativo é a relação entre os dados e ter um banco de dados normalizado que usa transações Atomicidad, Consistência Isolamento Durabilidade (ACID), então a escolha certa é um banco de dados relacional clássico." (TRUICĂ, 2013, p. 348)

3.8 Node.js

Node.js é uma estrutura poderosa desenvolvida no motor V8 JavaScript do Chrome que compila o JavaScript diretamente no código de máquina nativo. É uma estrutura leve usada para criar aplicativos da web do lado do servidor e estende a API JavaScript para oferecer funcionalidades usuais do lado do servidor.

O Node.js recebe grande suporte da comunidade de desenvolvedores do Javascript. Possuindo o maior número de solicitações pull no Github (figura 8). Na maioria das situações, soluções podem ser encontradas a partir dessas comunidades ao encontrar problemas. "O Node.js é um framework de código aberto é usado por muitos desenvolvedores terceirizados e utiliza a linguagem mais popular de acordo com o GitHub" (MAO, 2018,p. 38)

Figura 8 – Dez linguagens de programação mais populares no número de solicitações pull
Fonte: GITHUB, 2021

Node.js é um ambiente de tempo de execução JavaScript de código aberto, multiplataforma, que executa código JavaScript fora de um navegador da web. Node.js é uma estrutura da

web leve e popular para iniciantes e é usado por grandes empresas como Netflix e Uber.

Sendo orientado a eventos assíncronos que ajuda a lidar com solicitações simultâneas o qual é provavelmente, o ponto mais significativo do Node.js. Esse recurso significa basicamente que, se uma solicitação for recebida pelo Node para alguma operação de entrada ou saída, ele executará a operação em segundo plano e continuará com o processamento de outras solicitações. Isso é bem diferente de outras linguagens de programação.

O Node usa o mecanismo V8 JavaScript em tempo real, aquele que é usado pelo Google Chrome. O Node tem um wrapper sobre o mecanismo JavaScript que torna o mecanismo de tempo de execução muito mais rápido, e portanto o processamento de solicitações no Node também fica mais rápido.

Tratamento de solicitações simultâneas é outra funcionalidade importante do Node é a capacidade de lidar com conexões simultâneas com uma sobrecarga mínima em um único processo.

A biblioteca Node.js usa JavaScript este é outro aspecto importante do desenvolvimento em Node.js. Uma grande parte da comunidade de desenvolvimento já é bem versada em JavaScript, logo o desenvolvimento em Node.js se torna mais fácil para um desenvolvedor que conhece JavaScript.

3.8.1 Comunidade

Há uma comunidade ativa e vibrante para a estrutura Node.js. Por causa da comunidade ativa, sempre há atualizações importantes disponíveis para a estrutura. Isso ajuda a manter a estrutura sempre atualizada com as últimas tendências em desenvolvimento web.

3.8.2 Desempenho

> "Em relação à latência o Node.js apresentou um melhor desempenho em todas as dez situações de carga que ele foi submetido, sendo superior ao Tomcat, nessas amostras em 23% (vinte e três por cento) com nível de confiança de 99% (noventa e nove por cento) obtido de amostras de 500.000 uploads de carga de teste (figura 9). Também em relação à variabilidade dessa latência o Node.js apresentou uma razão de 1,54 contra uma razão de 2,76 no Tomcat, nessas dez amostras sintetizadas. Somente em uma única amostra, a mostra dez, o Node apresentou uma variabilidade da latência maior que o Tomcat." (DARSKI, 2016, p. 24)

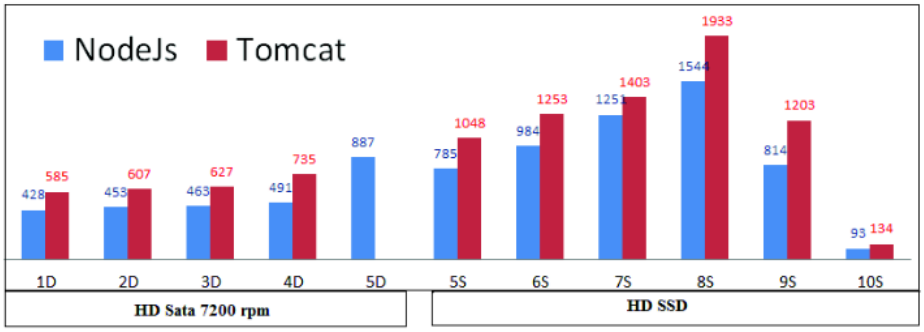


Figura 9 – Comparativo do Tempo Médio de Resposta (ms)
Fonte: Darski, 2016, p. 21

O Node.js pode ser usado para muitos aplicativos com vários propósitos. O único cenário em que não deve ser usado é quando há longos tempos de processamento, o que é exigido pelo aplicativo.

O nó é estruturado para ser de thread único. Se um aplicativo precisar realizar alguns cálculos de longa execução em segundo plano, ele não será capaz de processar nenhuma outra solicitação. Conforme discutido acima, o Node.js é mais usado onde o processamento precisa de menos tempo de CPU dedicado.

3.8.3 NPM

Para gerenciar os módulos do node.js foi usado o Node Package Manager (NPM) que como o nome sugere, é responsável por instalar o gerenciador de todos os pacotes, que também são chamados de módulos.

3.8.4 Express.js

O Node fornece um módulo http que você pode usar para verificar rotas, analisar as cargas úteis e fornecer conteúdo. O Express JS simplifica esse processo de construção de rotas, fornecendo uma forma de criação de rota eficaz e dá suporte para middlewares. Ele fornece todas as ferramentas necessárias para a criação de rotas, análise de cargas úteis, criação de páginas de exibição HTML, trabalho com funções de middleware e conexão com bancos de dados.

O framework Express funciona fazendo uso de middlewares. As rotas criadas com Express.js possuem a opção de fornecer um middleware. O middleware é como uma função simples que pode realizar uma determinada tarefa. Para acessar a rota o middleware fornece uma rota vai realizar uma função, como a função de autenticação de usuário. Com middleware, pode-se tornar um código modular a que pode ser reutilizado em todo o aplicativo, o que facilita o desenvolvimento.

3.8.5 JWT

O JSON Web Token (JWT) é uma biblioteca responsável pelo armazenamento e transmissão de forma compacta e segura de objetos JSON, e por isso é usado o padrão definido pela RFC7519. Nessa aplicação ele será utilizado para gerar o Token de autenticação, usando uma chave privada.

3.9 TESTES DE SOFTWARE

Para garantir a qualidade dos itens desenvolvidos, os testes serão divididos em quatro etapas, que são:

3.9.1 Teste de Unidade

Esta abordagem básica de teste de software é seguida pelo programador para testar a unidade do programa. Ajuda os desenvolvedores a saber se a unidade individual do código está funcionando corretamente ou não (CRESPO, 2004, p. 3).

3.9.2 Teste de integração

Esse tipo de teste enfoca a construção e o design do software. Para saber se as unidades integradas estão funcionando sem erros ou não (CRESPO, 2004, p. 6).

3.9.3 Teste de sistema

Neste método, o software é compilado como um todo e depois testado como um todo. Esta estratégia de teste verifica a funcionalidade, segurança, portabilidade, entre outros.

3.9.4 Teste de software

O Teste de Programa é um método de execução de um programa de software real com o objetivo de testar o comportamento do programa e encontrar erros (CRESPO, 2004, p. 10). O software é executado com os dados do caso de teste para analisar o comportamento do programa ou a resposta aos dados de teste. Um bom teste de programa é aquele que tem grandes chances de encontrar bugs.

Na fase de desenvolvimento do Back-End, foi utilizado o software Insomnia para a realização dos testes de request no servidor node.js.

4 DESENVOLVIMENTO DO APLICATIVO

4.1 Diagrama Entidade Relacionamento

O diagrama de classes será usado para mostrar como as diferentes entidades do aplicativo se relacionam, em outras palavras, mostra as estruturas estáticas do sistema. Um diagrama de classes pode ser usado para exibir classes lógicas, e podem ser usados para mostrar as classes de implementação, que são coisas com as quais os programadores normalmente lidam. Um diagrama de classes de implementação também mostrará algumas das mesmas classes que o diagrama de classes lógicas. O diagrama de classes de implementação não será desenhado com os mesmos atributos, pois possui referências a variáveis como Vetores e HashMaps.

Para a redução da complexidade do aplicativo, evitando a necessidade de criar muitas tabelas, as tabelas foram planejadas com diferentes tipos, com diferentes características (figura 10), que seriam gerenciadas no Back-End.

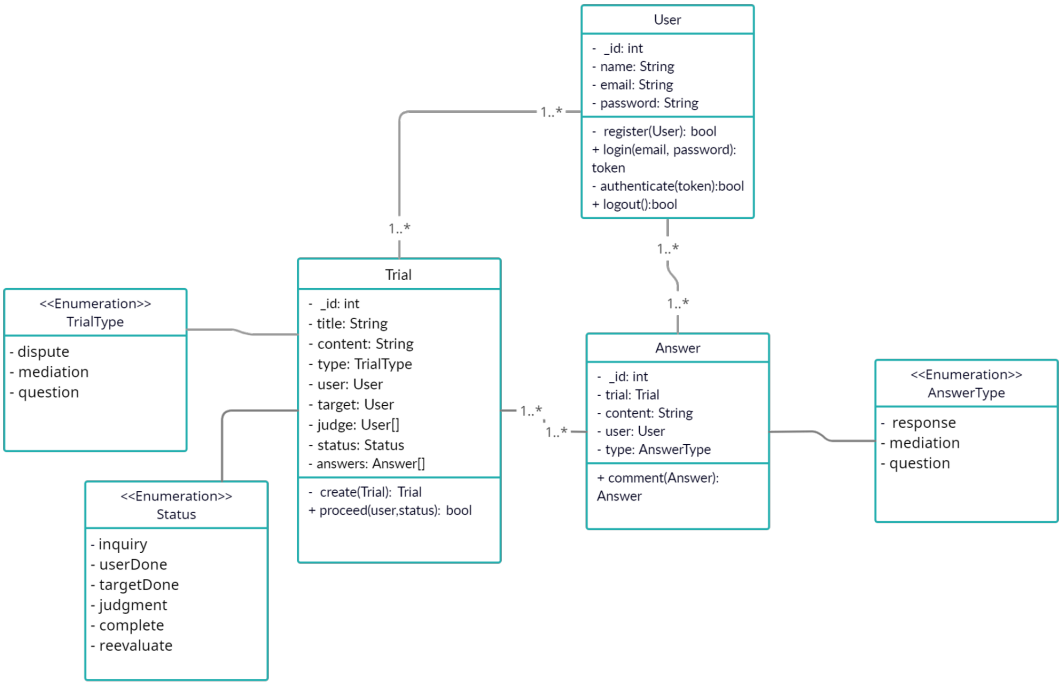


Figura 10 – Diagrama de classe
Fonte: Autor

4.1.1 Classe User

Essa classe representa o usuário que possuirá e-mail, nome, e senha criptografada usando o modulo do React Native cypher.

E possuirá as funções de registro do usuário, login recebendo o e-mail, senha, retornado um Token para acesso ao sistema. Função de autenticação para verificar se o usuário está logado,

ou seja, possui um Token válido e pode acessar os recursos do sistema. E uma função para sair da conta logada.

4.1.2 Classe trial

Essa classe representa os processos, com título, uma breve descrição do que se trata o caso, usuário que iniciou o processo, o usuário alvo do processo, os possíveis juízes, as respostas que são uma referência à classe Answer, o tipo de processo que é um enumerador categorizando os como disputa, mediação e questão. Com as funções para criação do processo e de avanço nos estágios do processo.

E importante salientar a possibilidade de usar array em uma tabela no MongoDB, que torna as classes do banco de dados mais próximas a um objeto.

4.1.3 Classe Answer

Essa classe representa as respostas nos processos, que podem ser do usuário que criou o processo, do alvo do processo, ou do jures. As respostas são vinculas a um processo pelo id, possuem um conteúdo, o usuário que as criou e o tipo de resposta classificado em um enumerador como argumento das pessoas envolvidas, mediação do júri e parecer do júri. Com a função postar uma resposta em um processo.

4.2 Diagrama de arquitetura de software

O Diagrama de arquitetura de software mostra visualmente a estrutura básica do software, incluindo os componentes e as relações do software bem com as propriedades de cada um. E também mostra as relações com os componentes externos, como serviços, clientes do sistema e o banco de dados. O diagrama apresentado na Figura 11 foi criado para facilitar a descrição da estrutura do aplicativo.

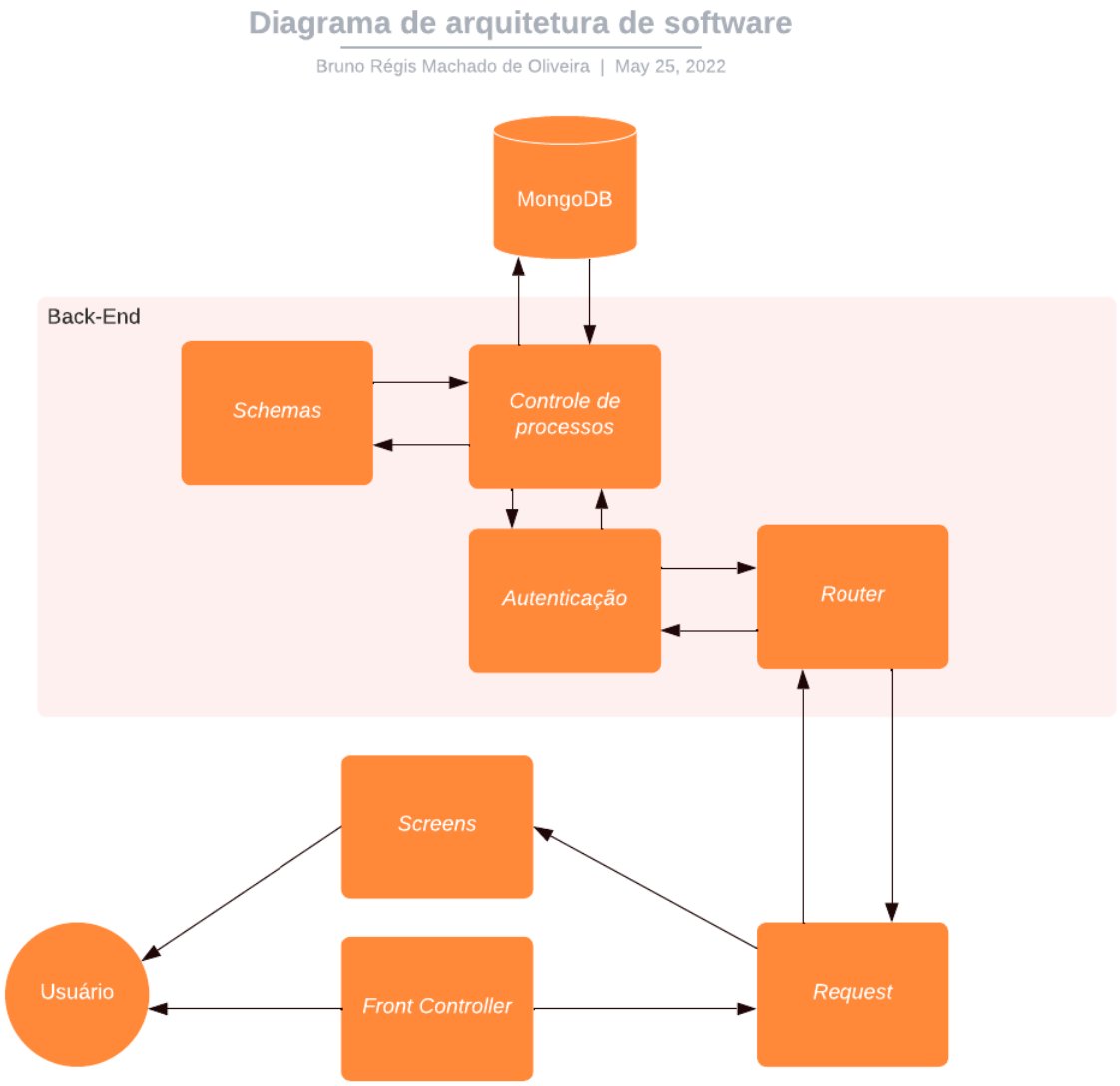


Figura 11 – Diagrama de arquitetura de software
Fonte: Autor

4.3 Black-End

4.3.1 Schemas

Inicialmente foi criado os módulos para os Schemas do banco de dados, trial, Answer e User, com base no diagrama de classe UML, usando para isso a biblioteca mongoose para criar os esquemas e os modelos do banco de dados. Destacando a capacidade do MongoDB de criar enumeradores diretamente no modelo da tabela, e parâmetros como valor padrão, parâmetro para checar se o valor e necessário, parâmetro que permite colocar o conteúdo em letras minusculas.

4.3.2 Autenticação

Esse módulo exporta duas rotas de posts, o post para registro de usuário que recebe um post contendo as informações de registro de usuário, nome, e-mail e senha. Checa se já existe um usuário já criado, e caso a validação dos dados retorne verdadeiro e criado o usuário, retornando um Token gerado usando a chave privada de biblioteca JWT.

O post de login, recebe e-mail e senha, verifica se existe um usuário com o e-mail recebido, caso verdadeiro, criptografa a senha recebida e compara com a senha do usuário encontrado, retornando o Token caso seja válido.

Foi criado um módulo para realizar a autenticação do Token, fazendo checagem se o Token possui os parâmetros adequados, e se estiver formatado adequadamente, retornando erros conforme o problema encontrado.

4.3.3 Controle de processos

O controle de processos só pode ser acessado pelo Middleware, que verifica se o usuário possui um Token válido. Nesse módulo ficam as rotas para o gerenciamento dos processos e respostas, um GET para o requerimento dos processos públicos, um para a seleção de um processo específico, o post para a criação do processo, outro para a remoção e um para a edição, que também é responsável por alterar o estágio do processo. E por fim a rota responsável por iniciar o estágio de julgamento, selecionado os juízes, de forma aleatória excluindo os usuários envolvidos no processo.

4.4 Front-End

4.4.1 Screens

Para o desenvolvimento do Front-End em React Native foi utilizado o Freamwork expo, que facilita a compilação, o desenvolvimento e o lançamento do aplicativo. Trazendo muitos recursos convenientes que simplificam o desenvolvimento. A principal vantagem de usar o Expo é a obtenção de recursos como acesso à câmera, localização, notificações, sensores, haptics com a configuração pronta para o uso. Em que no caso do React-Native-CLI, é necessário a construção deses recursos.

Expo também oferece a conveniência adicional de ter um aplicativo móvel dedicado em Android e iOS para a visualização do aplicativo e as alterações acontece assim que são salvas no código. Tornando o desenvolvimento e principalmente os testes mais rápidos.

Para a instalação do Expo e usado o comando npm install –global expo-cli e para a criação do aplicativo o comando expo init <Nome do aplicativo>.

Com o objetivo de melhorar a responsividade do aplicativo foi usado a tag SafeAreaView para impedir que os conteúdos da tela sejam ocultados pela tela do celular, como exemplo os celulares que possuem um notch onde fica localizado a frontal do telefone. Também foi usado a tag KeyboardAvoidingView para evitar que o teclado sobreponha os campos de texto.

Para realizar a navegação das telas foi usado o react navigation, instalado com o comando npm install @react-navigation/native.

4.4.2 Front controller

Na tela de login, foi utilizado o useState para atualizar as variáveis e-mail e senha quando ocorrer alguma alteração dos inputs de texto. E usando uma função assíncrona que realiza o post, esperando a resposta do servidor, e dependendo da resposta apresenta um erro de e-mail ou senha invalida, ou envia o usuário para a home e usando a biblioteca react-native-async-storage para armazenar o Token, para a autenticação do usuário. O async storage, salva os dados, mesmo quando o usuário fecha a API ou mesmo o dispositivo. Recebendo os dados de forma assíncrona.

Foi usado a biblioteca BCrypt para realiza encriptado e decriptando a senha, realizando de forma automática o processo de hash e o salt da senha, aumentando a segurança e agilizando o processo que em outras linguagens geralmente e feito manualmente.

Para a verificação se o usuário já está logado é feito uma verificação usando o useEffect, que checa na renderização da tela a existência da variável Token no async-storage, realizando a autenticação do Token. O mecanismo de logout simplesmente o asyncStorage, esquecendo o Token e envia o usuário a tela de login. A tela de registro é similar a tela de login, acessando o post de registro do WebServer, enviando os dados do novo usuário e verificando a resposta, enviando o usuário a home caso o sucesso no registro do novo usuário ou mostrando o campo que não foi preenchido corretamente, podendo ser o e-mail caso já exista um usuário com o mesmo e-mail.

A implementação de abas no topo da tela foi extremamente fácil, com o auxílio do createMaterialTopTabNavigator.

5 RESULTADO

Os resultados alcançados são considerados bastante satisfatórios, considerando que foi possível realizar a finalização do desenvolvimento do aplicativo, com as principais funcionalidades. Sendo capaz de realizar a autenticação de usuário, criação, inquérito e julgamento dos processos. O nome escolhido para o aplicativo foi decido que significa decisão em esperanto. As funcionalidades do aplicativo serão detalhadas nos tópicos a seguir.

5.1 Autenticação

Na figura 12 que representa a tela de login apresenta os logo do aplicativo decido, os campos de e-mail e senha, necessários para acessar o aplicativo, e caso o usurário não possua uma conta poderá ir para a tecla de cadastro acessando o botão com mesmo nome.

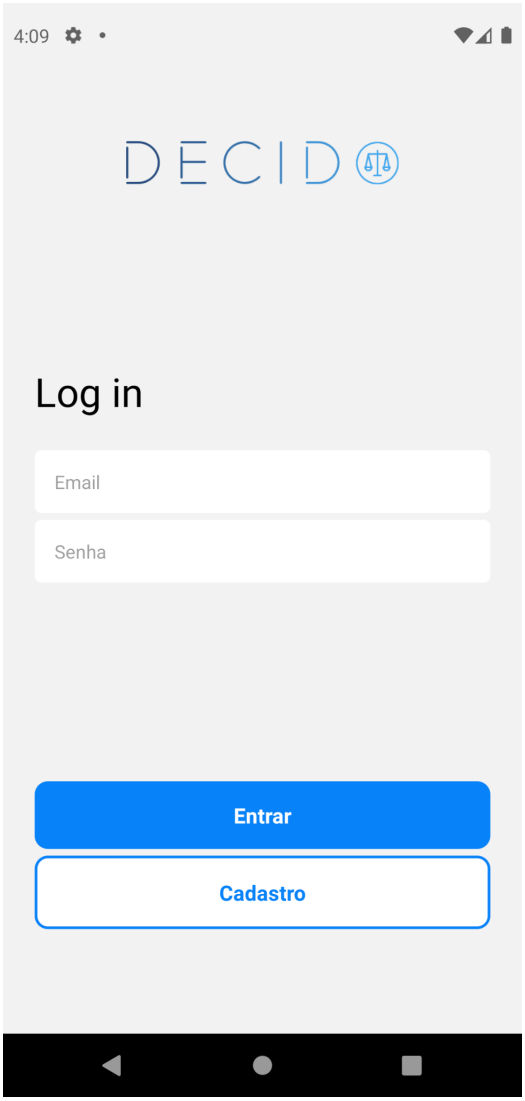


Figura 12 – Tela de Login
Fonte: Autor

5.2 Cadastro

Na figura 13 que representa a tela de registro é similar a tela de login, com os campos de nome e-mail e senha necessários para realizar o cadastro do usuário, e o botão para finalizar o cadastro, recebendo mensagem caso algum campo tenha sido inserido de forma incorreta. Caso o cadastro seja realizado com sucesso o usuário será redirecionado a home.

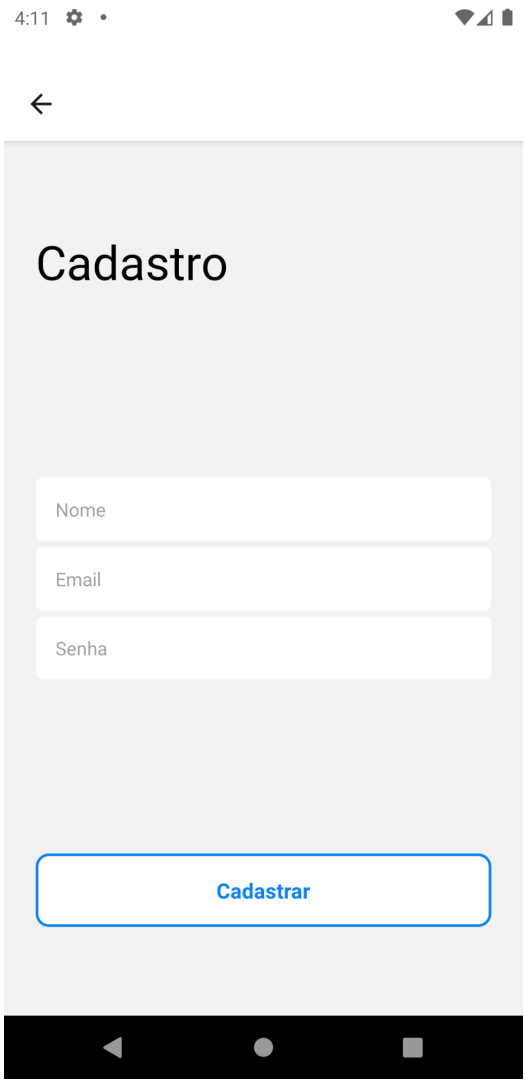


Figura 13 – Tela de Registro
Fonte: Autor

5.3 Home

Na figura 14 que representa a tela Inicial é mostrado os processos já concluídos e públicos, e as abas para as telas te processos a serem julgados, e tela de usuário. O botão verde com símbolo de adição redirecionara o usuário a tela de criação de processos. O usuário poderá usar as abas superiores para realizar a navegação no aplicativo.

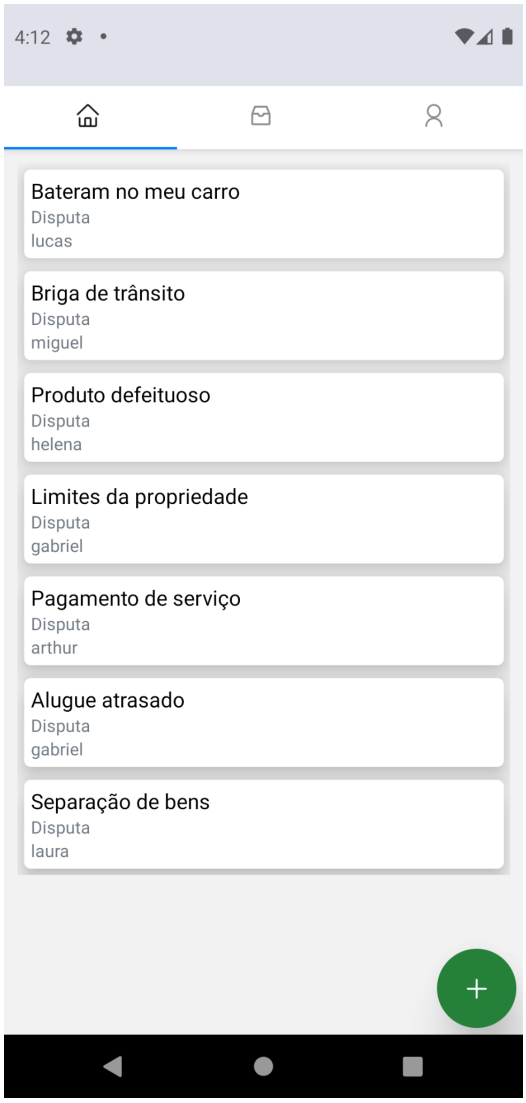


Figura 14 – Tela de Tela Inicial
Fonte: Autor

5.4 Notificações

Na figura 15 que representa a tela de Processos a serem julgados é mostrado os processos que o usuário está inserido como juiz. Esperando a avaliação do processo pelo usuário.

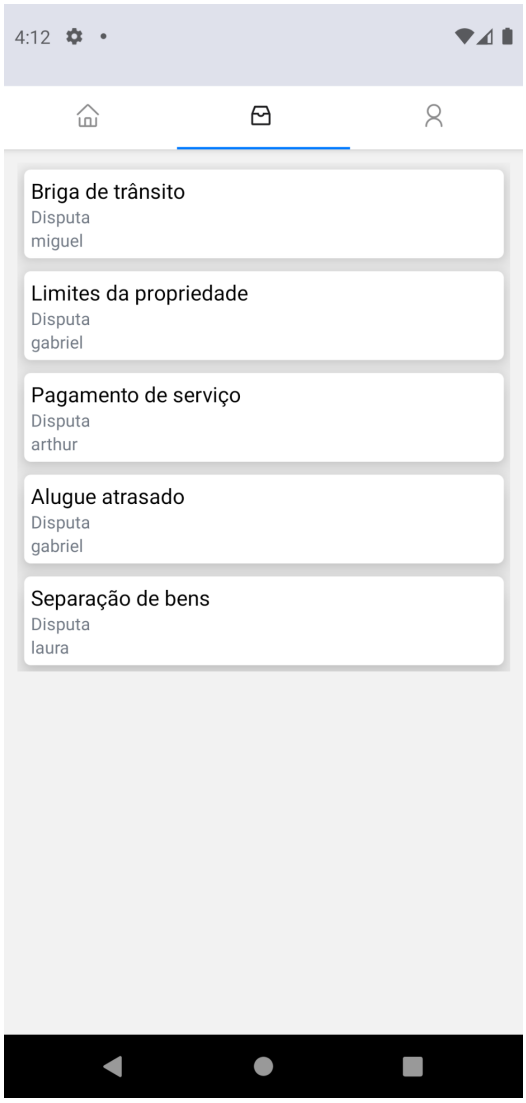


Figura 15 – Tela de Processos a serem julgados
Fonte: Autor

5.5 Usuário

Na figura 16 que representa a tela de usuário é mostrado as informações do usuário, nome, e-mail, avaliação calculada com base em sua participação nos processos, e os processos em que ele está envolvido, além do botão para logout, que desautentica o usuário e o retorna a tela de login.

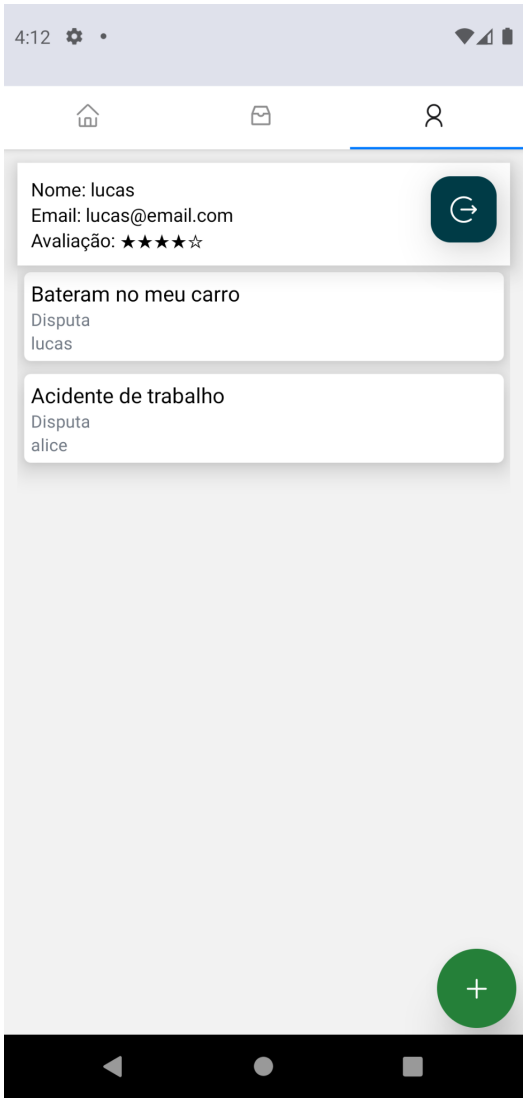


Figura 16 – Tela de Usuário
Fonte: Autor

5.6 Início do processo

Na figura 17 representa a tela de criar processo é iniciado o processo, com o usuário informando o título, uma descrição breve sobre a sua queixa, e o usuário responsável ou envolvido no caso. Podendo receber como resposta do servidor, de erro caso algum campo não tenha sido preenchido corretamente ou se o usuário envolvido não foi encontrado. Caso o processo tenha sido criado com sucesso o usuário e redirecionado a tela do processo, podendo assim dar continuação a sua queixa.

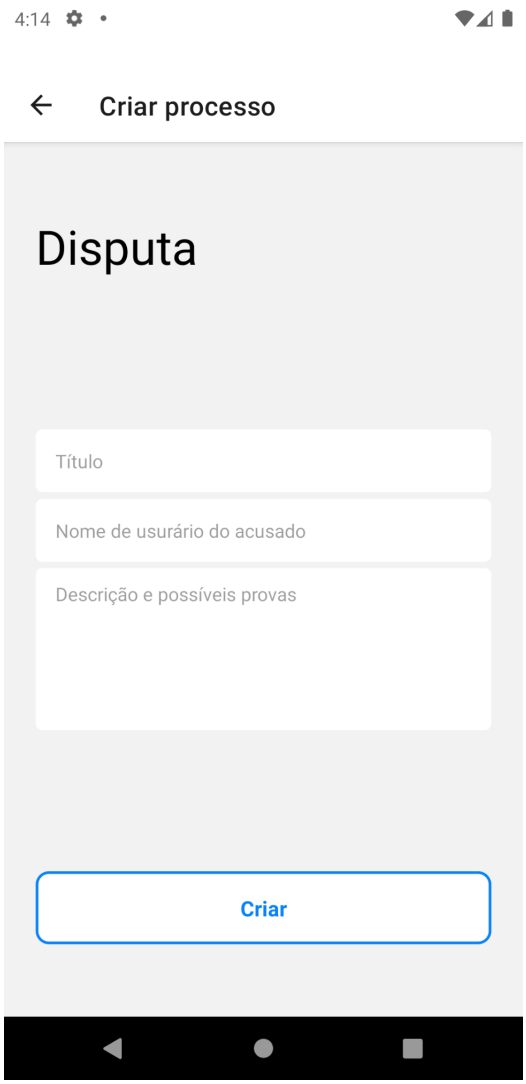


Figura 17 – Tela de criação do processo
Fonte: Autor

5.7 Processo

Na figura 18 que representa a tela de processo é onde ocorre a arbitragem. No cabeçalho e mostrado informações do processo, em seguida as mensagens enviadas e o campo de envio. O processo ocorrerá em duas etapas, a etapa de inquérito e discussão, e a etapa de julgamento. Na fase de inquérito os usuários envolvidos apresentam seus pontos de vista, quando finalizado essa etapa, os usuários devem solicitar e confirmar o início do julgamento, dado que esse modelo de arbitragem é voluntário e não obrigatório, os envolvidos podem seguir a decisão do juiz ou pedir a reavaliação do caso, reiniciando a fase de julgamento.

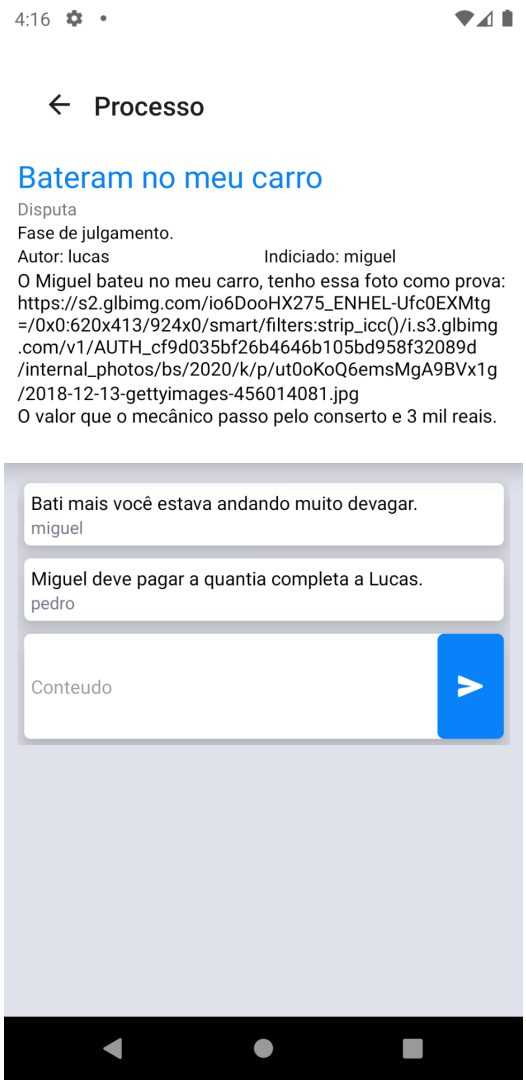


Figura 18 – Tela do Processo
Fonte: Autor

6 CONCLUSÃO

Foi possível finalizar o desenvolvimento do aplicativo, concluindo todas as fases de desenvolvimento, dentro do prazo estipulado. Tendo todas as funcionalidades estipuladas. Sendo capaz de realizar a autenticação de usuário, criação, inquérito e julgamento dos processos.

E com o aplicativo funcional, e aplicável em situações práticas, o aplicativo foi publicado na Google Play com o mesmo nome.

Dentre as tecnologias estudadas React native é o que possui o pior desempenho em relação as suas contrapartes, possuindo como benefício a facilidade e velocidade de desenvolvimento em conjunto com a qualidade e quantidade de conteúdo e documentação disponível online. Em contrapartida, ao React Native, existem tecnologias promissoras como o flutter que é compilado nativamente, possuindo assim uma maior velocidade de compilação, ultrapassando o React Native também em sua popularidade no Github.

Os resultados do MongoDB foram surpreendentes, além de ser superior em sua flexibilidade, possui o desempenho superior às tecnologias convencionais usadas no gerenciamento de banco de dados, mais rápido que o PostgreSQL e SQL. É disponibilizando gratuitamente hospedagem para teste, através do MongoDB atlas. Tendo como lado negativo a sua própria simplicidade, não possui funções de gerenciamento integradas presentes nos bancos de dados relacionais.

Além disso, foram observados resultados positivos em relação ao node.js que é mais rápido que o apache e o Tomcat, porém por ser uma tecnologia nova e voltada a grandes volumes de dados, não possui muitas opções de hospedagem acessíveis.

E importante salientar a dificuldade de apuração do código encontrada em relação a esses novos Framework, sendo necessário usar ferramentas como o Insomnia para testar os request do Back-End em Node.js. E a falta de detalhamento nos relatórios de erros no React Native, que na maioria das vezes apontava para erros que abrangiam muitos tipos de problemas, esporadicamente apontando a uma linha de código especifica.

E com a virtualização da sociedade futuramente grande parte das resoluções de conflitos será online. Os contratos inteligentes também são uma opção promissora, por possuírem regras claras, facilitando a solução de conflitos.

REFERÊNCIAS

ABRAHAMSSON, Pekka et al. Agile software development methods: Review and analysis. **arXiv preprint arXiv:1709.08439**, 2017.

ANDREIA, Chiquini Bugallho; DA SILVEIRA, Sebastião Sérgio; ZANFERDINI, Flávia de Almeida Montingelli. RESOLUÇÃO ONLINE DE CONFLITOS COMO FERRAMENTA DE CIDADANIA E FACILITAÇÃO DO ACESSO À JUSTIÇA. In: **Anais do Congresso Brasileiro de Processo Coletivo e Cidadania.** 2020. p. 766-783.

AMORIM, Benjamim Siqueira; JACOMINI, Alessandro. ARBITRAGEM COMO MEIO DE SOLUÇÃO DE CONFLITOS ENVOLVENDO A TECNOLOGIA BLOBKCHAIN E SMART CONTRACTS. **Revista Vertentes Do Direito,** v. 6, n. 1, p. 279-294, 2019.

CRESPO, Adalberto Nobiato et al. Uma metodologia para teste de Software no Contexto da Melhoria de Processo. In: Anais do III Simpósio Brasileiro de Qualidade de Software. SBC, 2004. p. 204-218.

DANIELSSON, William. React Native application development: A comparison between native Android and React Native. 2016.

DARSKI, Rinaldo. Busca de indicadores de eficiência no uso de recursos em aplicações de transmissão por Upload em um estudo comparativo de um servidor Node. js para Middleware versos um Apache Tomcat. 2016.

DIPINA DAMODARAN, B.; SALIM, Shirin; VARGESE, Surekha Marium. Performance evaluation of MySQL and MongoDB databases. **Int. J. Cybern. Inform.(IJCI),** v. 5, 2016.

GUEDES, Gilleanes TA. UML 2. **Uma Abordagem Prática",** São Paulo, Novatec, 2009.

HAMANN, Evandro Vieira et al. Custos para o julgamento de um processo no Tribunal de Justiça do Distrito Federal-TJDF. In: **Anais do Congresso Brasileiro de Custos-ABC.** 2014.

HANSSON, Niclas; VIDHALL, Tomas. Effects on performance and usability for cross-platform application development using React Native. 2016.

KATSH, M. Ethan; RABINOVICH-EINY, Orna. **Digital justice: technology and the internet of disputes.** Oxford University Press, 2017.

KATSH, Ethan; RIFKIN, Janet; GAITENBY, Alan. E-commerce, E-disputes, and E-dispute Resolution: In the Shadow of eBay Law. **Ohio St. J. on Disp. Resol.**, v. 15, p. 705, 1999.

KAUFMANN-KOHLER, Gabrielle. Online dispute resolution and its significance for international commercial arbitration. **Dalam Global Reflections on International Law, Commerce and Dispute Resolution yang dipublikasikan oleh ICC Publising pada November,** 2005.

KHAN, Sanobar; MANE, Vanita. SQL support over MongoDB using metadata. **International Journal of Scientific and Research Publications,** v. 3, n. 10, p. 1-5, 2013.

LAWTECH CONFERENCE, 2019, StartSe. **Law Innovation Online,** 2019. Disponível em: https://eventos.startse.com.br. Acesso em: 23 maio 2019.

LIMA, Gabriela Vasconcelos; FEITOSA, Gustavo Raposo Pereira. Online dispute resolution (ODR): a solução de conflitos e as novas tecnologias. **Revista do Direito,** v. 3, n. 50, p. 53-70, 2016.

MAO, Xiaoli. Comparison between Symfony, ASP. NET MVC, And Node. js Express for Web Development. 2018.

NENSTIEL, AndraLeigh. Online Dispute Resolution: A Canada-United States Initiative. Can.-USLJ, v. 32, p. 313, 2006.

POLITOWSKI, Cristiano; MARAN, Vinıcius. Comparaç ao de Performance entre PostgreSQL e MongoDB. **X Escola Regional de Banco de Dados. SBC,** p. 1-10, 2014.

SENSORTOWER. In: **Estimated consumer spending between January 1, 2020 and June 30, 2021. [S. l.], 2021.** Disponível em: https://app.sensortower.com/. Acesso em: 30 maio 2022.

SEVERINO, Antônio Joaquim. **Metodologia do trabalho científico**. Cortez editora, 2017.

SHAH, Aashit. Using ADR to resolve online disputes. **Richmond Journal of Law & Technology,** v. 10, n. 3, p. 25, 2004.

STATISTA. Annual U.S. e-Commerce sales from 2002 to 2012. fev. 2013. Disponível em: . Acesso em: 25 de mai. 2013.

TRUICĂ, Ciprian Octavian; BOICEA, Alexandru; TRIFAN, Ionut. CRUD operations in MongoDB. In: **International Conference on Advanced Computer Science and Electronics Information.** 2013. p. 347-348.

WU, Wenhao. React Native vs Flutter, **Cross-platforms mobile application frameworks.** 2018.