UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIAS EM SOFTWARE LIVRE

CARLOS AUGUSTO MARTINS CLARO

# PERSISTÊNCIA POLIGLOTA EM BANCOS DE DADOS E FERRAMENTAS DE DESENVOLVIMENTO DE APLICATIVOS

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2017

CARLOS AUGUSTO MARTINS CLARO

# PERSISTÊNCIA POLIGLOTA EM BANCOS DE DADOS E FERRAMENTAS DE DESENVOLVIMENTO DE APLICATIVOS

Monografia de Especialização, apresentado ao Curso de Especialização em Tecnologia em Software Livre, do Departamento Acadêmico de Eletrônica, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Especialista.
Orientador: Prof. Leandro Batista de Almeida

CURITIBA
2017

Ministério da Educação
**Universidade Tecnológica Federal do Paraná**
**Câmpus Curitiba**
Diretoria de Pesquisa e Pós-Graduação

# TERMO DE APROVAÇÃO

## Título da Monografia

## PERSISTÊNCIA POLIGLOTA EM BANCOS DE DADOS E FERRAMENTAS DE DESENVOLVIMENTO DE APLICATIVOS

por

**Carlos Augusto Martins Claro**

Esta monografia foi apresentada às 19 horas do dia 23 de fevereiro de 2017 como requisito parcial para a obtenção do título de ESPECIALISTA EM TECNOLOGIA E SOFTWARE LIVRE, do Programa de Pós-Graduação da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após a deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Msc Christian Carlos Souza Mendes
UTFPR

_____
Msc Leandro Batista de Almeida
UTFPR

_____
Msc Fabiano Kuss
SERPRO

_____
Prof. Msc Lincoln Herbert Teixeira

“As aplicações do futuro aproveitarão a natureza poliglota do mundo das linguagens. Devemos abraçar essa ideia. Enquanto um vai fazer algumas tarefas mais difíceis, outros as mais fáceis. É tudo questão de escolha da ferramenta certa para o trabalho. Cada vez mais, vamos começar a adicionar linguagens específicas do serviço. Os tempos de escrever um aplicativo em uma única linguagem de uso geral terminaram.” (FORD, 2006).

# RESUMO

CLARO, Carlos Augusto Martins. Persistência poliglota em bancos de dados e ferramentas de desenvolvimento de aplicativos. 2017. 37 f. Monografia (Curso de Especialização em Tecnologias em Software Livre), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Com o aparecimento de novos aparelhos, inúmeras plataformas, formas de compartilhar e exibir conteúdos, as empresas devem estar cada vez mais preocupadas em estar presente. É primordial buscar rapidez, performance e usabilidade, mantendo uma base consolidada para consulta e comunicação com bancos de dados. Nossa proposta é apresentar um protótipo de aplicativo, com compatibilidade entre aparelhos e linguagens, aplicando um formato de consulta centralizado, utilizando novas tecnologias, através de REST API. Para otimizar ainda a entregaa de conteudo, estudar, explicar como cada ferramenta abordada funciona e medir o desempenho de bancos de dados MySQL e NoSQL, sugerindo qual a melhor ferramenta para cada serviço prestado pela entrega imóveis, através de pesquisa em portais imobiliários. Exemplificando ao fim do projeto como podemos aplicar e otimizar a consulta de imóveis com MongoDB, buscando sugestões mais acertiva com base em pesquisas relacionadas em bancos de dados graphos – Neo4J. E mostrando, como utilizando persistência poliglota podemos otimizar os serviços e economizar recursos de servidor. Comparando a aplicação de recursos, pré e pós persistência poliglota.

**Palavras chaves:** Banco de dados. MySQL. MongoDB. Neo4J. Android. Cordova.

# ABSTRACT

CLARO, Carlos Augusto Martins. Poligloty persistence in databases and application development tools. 2017. 37 f. Monography (Specialization Course in Free Software Technologies), Electronic Academic Department, Federal Technological University of Paraná. Curitiba, 2017.

With the emergence of new gadgets, numerous platforms, ways to share and display content, companies must be increasingly concerned about being present effectively in all. It is essential to seek speed, performance and usability, maintaining a consolidated basis for consultation and communication with databases. For this we will study the development of applications, with compatibility between gadgets and languages, using a centralized query format, using new technologies, through REST API. To further optimize these deliveries, study, explain how each tool covered works and measure the performance of MySQL and NoSQL databases, suggesting the best tool for each service provided by real estate delivery, through real estate portal research.Exemplifying at the end of the project how we can apply and optimize the real estate query with MongoDB, searching for more accurate suggestions based on related searches in graphos databases Neo4J. And showing how using persistence polyglot we can optimize services and save server resources. Comparing resource application, pre and post polyglot persistence.

**Keywords:** Databases.  MySQL. MongoDB. Neo4J. Android. Cordova.

# Índice de figuras

# LISTA DE TABELAS

# Sumário

# 1 Introdução

Aplicativos são cada vez mais presentes e necessários na vida de usuários e empresas, seja em busca de competitividade, interatividade ou diversão, as ferramentas mobile tem uma busca por otimização cada vez mais evidente gerando melhores interações. Como a cada dia são lançados novos produtos e sensores, cada vez mais portáteis e presentes nos bolsos dos usuários.

A tecnologia tem avançado rapidamente e as formas de integrar o conteúdo em computadores pessoais, notebooks, tabletes, celulares, wearable, smartTV deve evoluir na mesma velocidade, gerando uma sensação de ambientação e adaptação fácil em qualquer gadget que o usuário deseje interagir.

Para exibir resultados com mais eficiência e utilizando menos recursos, tanto do servidor, como do gadget do usuário. A utilização de bancos de dados NOSql e Sql integradas a linguagens de programação específica para cada serviço, otimiza as rotinas e melhora as entregas. A performance dessas ferramentas se torna necessário e deve nos definir um caminho melhor para entrega e gerenciamento de conteúdo.

A utilização de persistência poliglota, que é o uso de diferentes linguagens de programação e diferentes bancos de dados, torna o caminho até a entrega muito mais rápido, intuitivo e favorável ao crescimento do mercado. Hoje em dia com o uso de diferentes equipamentos para consultar, gerenciar ou apenas utilizar o conteúdo, cada um com um background de linguagem específica, seja WEB, com o simples e fácil PHP, aplicativos mobile, com o uso de java ou apenas JavaScript, TypeScript e HTML. A obtenção de conhecimento diversificado para programadores, se sentirem livres para trocar a linguagem quando necessário, mantendo suas linhas de projeto e excelência no produto final.

Os estudos aqui apresentados na área de banco de dados e desenvolvimento de aplicativos, deve servir para melhorar a entrega de imóveis, pelos portais imobiliários da POW Internet, e também para futuros estudos de persistência poliglota e bancos de dados NoSQL.

## 1.1 A POW e os buscadores de imóveis

Desenvolvendo soluções para o mercado imobiliário e conhecendo os desafios e competitividade envolvido em um ramo populoso e que recebe grandes investimentos para desenvolvimento de ferramentas de entrega de busca de imóveis. A POW Internet, desde 1999 sempre buscou inovação e disponibilizou ótimas soluções, visando atender clientes que buscam sites, portais e sistemas para internet. Sempre mantendo a inovação e excelência em ferramentas de nicho. Para isto o software livre sempre esteve presente, seja nos servidores Linux, ou na linguagem de programação escolhida, o PHP, e o banco de dados MySQL.


**Figura 1: Serviços POW internet**

A modernização das ferramentas vigentes, passa pela pesquisa de ferramentas disponíveis no mercado para otimizar a exibição de seus produtos.

Primeiramente a otimização de entrega online das pesquisas de imóveis e integração com as formas de busca pelo usuário, seja em desktop, mobile navegador responsivo ou aplicativo, é primordial pensarmos na velocidade, pesquisa de persistência poliglota visa sugerir opções de bancos de dados e linguagem para cada modo de pesquisa.

A pesquisa de ferramentas de desenvolvimento de aplicativos multiplataforma e de entregas eficientes baseadas em geolocalização, nos leva a testar diferentes formas de desenvolvimento, utilizando Cordova com ionic 2, para compatibilidade em vários dispositivos que se apresentam no mercado, como Android e IOS.

A comunicação e integração entre diferentes plataformas deve ser feita dinamicamente e sem necessidade de duplicar a fonte de dados. Sendo aplicado o conceito de API's, facilitando também o tratamento destes dados.

## 1.2 Objetivos

### 1.2.1 Estudos e aplicações.

Estudar o desenvolvimento Javascript, TypeScript Apache Cordova com IONIC 2 para compatibilidade com diferentes Sistemas operacionais. Também a utilização da API SDK do google maps.

Estudo e análise da melhor função para cada banco de dados com base em performance e utilização de recursos, entre Bancos de dados relacionais (MYSql) e não relacionais (MongoDB, Neo4J) para armazenamento de dados, integração de dados e aplicação de geolocalização.

Desenvolver REST SERVICE para integração entre sistema atual, baseado em PHP, ao Apache Cordova e Android com respostas JSON.

Apresentar resultado de pesquisa e performance de cada aplicação utilizada, através de métricas de tempo de extração de dados e hardware utilizado.

### 1.2.2 Objetivos Específicos

- Analisar o uso de banco de dados após apresentação de cada ferramenta, exemplificando cada aplicação e comunicação. Com a intenção de entrega de dados para aplicativo, como imagens e dados dos imóveis e de suas anunciantes. Relação de opções de escolha e sugestões inteligentes com base nas buscas realizadas.
- Analisar querys e chaves de indexação de geolocalização e resultado de otimização nos bancos de dados.
- Criar protótipo de aplicativo utilizando api’s de código aberto, ferramenta Apache Cordova e Framework IONIC, exibindo as vantagens do método de desenvolvimento.

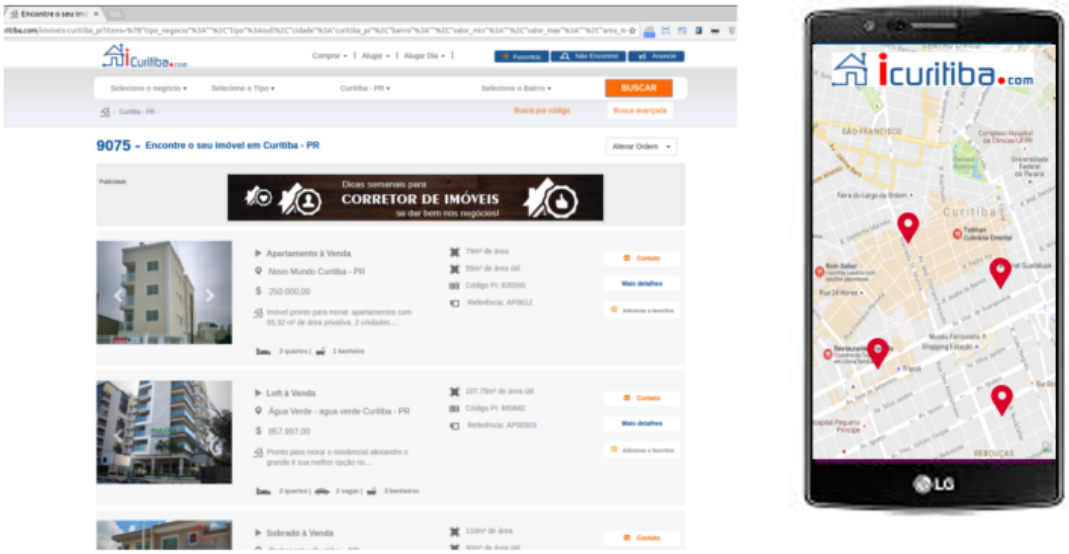


**Figura 2: Persistência poliglota(Acervo pessoal)**

## 1.3 Justificativa

Com cada vez mais pessoas conectadas e realizando buscas de seus objetivos, sejam compras, diversão ou outros interesses, utilizando diversas formas de conexão a internet, o estudo de performance de aplicação de melhores praticas com base em ferramentas disponíveis no mercado, para armazenamento e acesso a estes dados, pode se tornar um diferencial tanto econômico como de visibilidade de usuários. As principais ferramentas de busca da internet (Google, Yahoo, Bing) tem uma preocupação cada vez maior com o tempo de processamento de sites e aplicativos, e os mais otimizados e leves, levam uma vantagem de relevância na entrega deste conteúdo.

Com base nas pesquisas de performance de banco de dados e ferramentas de desenvolvimento, analisar os ganhos, vantagens e desvantagens das ferramentas disponíveis no mercado. Aplicando nas ferramentas ativas da POW Internet.

Para otimizar ainda mais as novas ferramentas disponíveis no mercado, o resultado da pesquisa de aplicativos, nos levará a entregar um produto com mais compatibilidade, performance e portabilidade, estando disponível para vários tipos de dispositivos.

## 1.4 Estrutura do trabalho

**Capítulo 1 - Introdução:** inicia com uma breve apresentação da POW internet, empresa objeto e analise do estudo deste trabalho, que contem uma aplicação web para busca de imóveis e necessita de uma otimização de suas ferramentas, seja em performance como no uso de novas tecnologias. Mostra como será conduzido o estudo, através do objetivo, justificativa e Também exibe os objetivos e justificativa para a aplicação de persistência em aplicações multi-plataforma.

**Capítulo 2 – Fundamentação Teórica:** fundamenta através de estudos e pesquisa de utilização de cada ferramenta escolhida e de que forma cada uma serve para o propósito de otimizar o sistema através do uso de persistência poliglota. Exibe como cada ferramenta se comporta e pode colaborar para uma estrutura mais rápida, econômica e confiável para entrega de conteúdo.

**Capítulo 3 – Estudo de caso:** Apresenta a melhor aplicação para cada ferramenta e com base na capacidade de uma, sugere um formato que otimize os recursos de servidor e mão de obra no desenvolvimento para entrega de conteudo atrativo, dinamico e rápido.

**Capítulo 4 – Considerações finais:** Apresenta os benefícios da persistência poliglota, com base no desempenho de servidor e economia de recursos, rapidez na entrega e como isto pode ser vantajoso para sistemas web.

**Capítulo 5 – Conclusão:** Expõe de forma quantitativa os resultados alcançados com a pesquisa e aplicação das ferramentas sugeridas, e os benefícios do modelo de persistência poliglota com banco de dados e desenvolvimento WEB.

# 2 Fundamentação teórica

## 2.1 Persistência Poliglota

Desde o início dos tempos da informática, o desenvolvimento de sistemas e o armazenamento de dados é fundamental, e com isto, o aparecimento de inúmeras linguagens de programação e banco de dados tem como objetivo resolver diferentes problemas que foram surgindo. Apesar disto, hoje em dia é comum se deparar com sistemas complexos e profissionais que adotam e aplicam apenas uma linguagem com o objetivo de otimizar o tempo de aprendizagem e adaptação dos profissionais.

Mas em 2006, Neal Ford lançava um texto em seu blog, que mostrava o que vinha pela frente no mundo da programação, o aparecimento de diferentes dispositivos, novas tecnologias e problemas diferentes. Sugeriu:

> “As aplicações do futuro aproveitarão a natureza poliglota do mundo das linguagens. Devemos abraçar essa ideia. Enquanto um vai fazer algumas tarefas mais difíceis, outros executa as mais fáceis. É tudo questão de escolha da ferramenta certa para o trabalho e debuga-lo corretamente. Os testes ajudam com problemas de depuração. SQL, Ajax e XML são apenas o começo. Cada vez mais, vamos começar a adicionar linguagens específicas do serviço. Os tempos de escrever um aplicativo em uma única linguagem de uso geral terminaram.” (FORD, 2006).

Logo em 2011, Fowler explica e exemplifica como a persistência poliglota em bancos de dados pode ser util para grandes aplicações que desejam escalabilidade e performance, criando um mix de uso e justificando cada um deles.

> “A ideia. de que as aplicações devem ser escritas em um mix de linguagens para aproveitar o fato de que diferentes linguagens são adequados para enfrentar diferentes problemas. Aplicações complexas combinam diferentes tipos de problemas, então escolher o idioma certo para o trabalho pode ser mais produtivo do que tentar encaixar todos os aspectos em uma única linguagem.” (FOWLER, 2011)


**Figura 3: Persistência poliglota em banco de dados(FOWLER,2011)**

Ainda no mesmo texto, ele explicou porque devemos utilizar modular sistemas, e cada modulo com sua linguagem adequada, em formato de web services que utilizam um formato de dados para se comunicar, seja XML, JSON ou outros formatos.

> “Este efeito poliglota será aparente mesmo dentro de uma única aplicação. Um aplicativo corporativo complexo usa diferentes tipos de dados e, normalmente, integra informações de fontes diferentes. Cada vez mais, veremos que tais aplicativos gerenciam seus próprios dados usando diferentes tecnologias, dependendo de como os dados são usados. Esta tendência será complementar à tendência de dividir o código de aplicação em componentes separados que se integram através de web service. Um limite de componente é uma boa maneira de envolver uma determinada tecnologia de armazenamento escolhida para a forma como seus dados são manipulados.” (FOWLER, 2011)

Com o aparecimento de cada vez mais bancos de dados não relacionais, é importante lembrar o trecho a seguir, que consta na apresentação da ferramenta NEO4J em seu site.

> “Enquanto bancos de dados relacionais são ferramentas poderosas para o caso de uso certo e a arquitetura certa, as necessidades atuais de usuários e empresas estão mudando. A complexidade atual e os dados mundiais estão aumentando em volume, velocidade e variedade, e os relacionamentos de dados - que são muitas vezes mais valioso do que os dados em si - estão crescendo a um ritmo ainda mais rápido.” (NEO4J, 2016)

É importante utilizar o potencial de cada linguagem ou ferramenta, e buscar diversificar as metodologias, em torno de um plano maior que é a entrega do conteúdo, de forma rápida e eficaz. Como a maioria dos dados devem ser compartilhados e gerenciados em diferentes plataformas e de diferentes localidades, sua comunicação entre o dispositivo e a base de dados deve ser intermediada através de protocolos de internet, como o HTTP, para isto uma sugestão é a REST API ou Web services.

## 2.2 REST Services

O conceito de REST API, é mostrado e traduzido na editoria de programação e desenvolvimento Metal Java Full de 2011, e nos explica como funciona as requisições, inserções e edição de dados utilizando o protocolo HTTP.

> “REST é o acrônimo para *Representational State Transfer* (Representação de estado de transferencia). É um estilo de arquitetura para sistema de media distribuída e foi apresentada por Roy Fielding em 2000” ( METAL JAVA FULL, 2011).

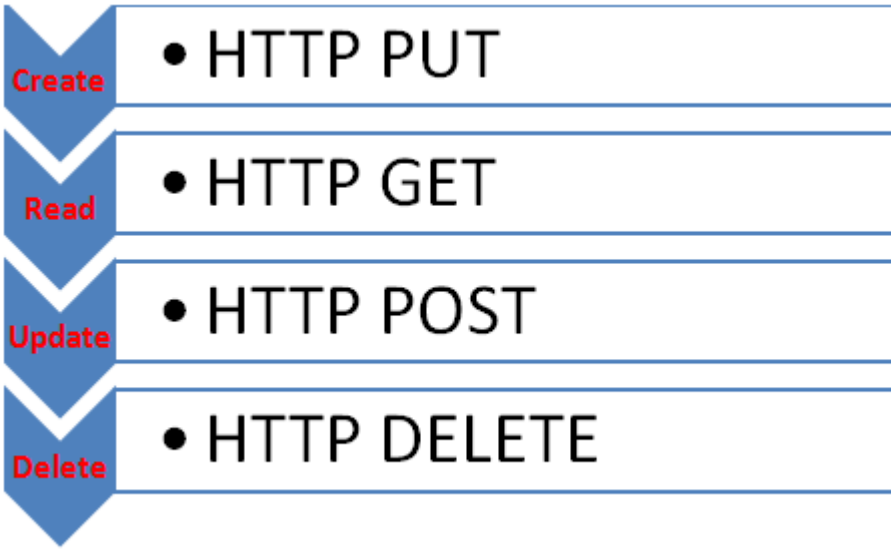


**Figura 4: CRUD**

> "CRUD, é um termo utilizado (convencionado) para designar as quatro operações básicas em Banco de Dados (Essa é a conceitualização/exemplificação mais comum): Create, Read, Update e Delete em Inglês." (METAL JAVA FULL, 2013)

Complementando o conhecimento de API o livro de Pereira, em 2016, Construindo APIS REST com NodeJS, uma forma de montar e manipular dados de diversos aparelhos diferentes, inclusive da aplicação web tradicional através de navegadores.

> “Atualmente, vivemos em uma fase na qual a maioria dos usuários utilizam diversos tipos de devices para se conectarem à internet. Os mais populares são smartphones, tablets e notebooks. Desenvolver sistemas para diversos tipos de devices requer o trabalho de construir web services, também conhecidos pelo nome de APIs (Application Program Interface).” (PEREIRA, 2016).

Com a utilização de diferentes bancos de dados e diferentes linguagens de programação que são especificas para cada aparelho ou protocolo, precisamos de uma interface que simplifique a comunicação entre eles. API REST trabalha com requisições e respostas utilizando protocolo HTTP, e o formato de dados de JSON, que é leve e facil de tratar em qualquer linguagem.

Caio Pereira também incentiva o uso de NodeJs e Javascript como linguagem de programação server-side, pois alem de reduzir o tempo de aprendizagem, por ser simples e de uso comum para quem já desenvolveu para WEB e facilitará por ser mesma linguagem do client-side.

> “o Node.js usa JavaScript como linguagem de programação server-side. Essa característica permite que você reduza e muito sua curva de aprendizado, afinal, a linguagem é a mesma do JavaScript client-side. Seu desafio nesta plataforma será de aprender a fundo como funciona a programação assíncrona para se tirar maior proveito dessa técnica em sua aplicação. Outra vantagem de se trabalhar com JavaScript é que você vai manter um projeto de fácil manutenção” (PEREIRA,2016).

A utilização de NodeJS dispensa o uso de frameworks de serialização de JSON no back-end, pois é o mesmo JSON utilizado no client-side é o mesmo do server-side. Alem de ser objeto utilizado em muitos bancos de dados, como MongoDB.

## 2.3 Bancos de Dados:

O crescimento cada vez maior da internet e dos sistemas de gerenciamento, se da pelo crescimento do uso de banco de dados, que nos traz formatos mais simples e dinamicos de armazenar e gerenciar dados. Elmaria define banco de dados da seguinte forma.


**Figura 5: Bancos de dados disponíveis no mercado(internet)**

> "Um banco de dados é uma coleção de dados relacionados. Com dados, queremos dizer fatos conhecidos que podem ser registrados e possuem significado implícito. É uma coleção logicamente coerente de dados com algum significado inerente. Projetado, construído e populado com dados para uma finalidade específica. Pode ser gerado e mantido manualmente ou pode ser computadorizado. SGBD (Database ManagementvSystem)" (ELMARIS, 2011).

Para que possamos utilizar, confiar e entender o fluxo de dados armazenados no banco de dados precisamos de SGBD's, sistemas de gerenciamento de banco de dados.

> "Os SGBDs relacionais oferecem aos usuários processos de validação, verificação e garantias de integridade dos dados, controle de concorrência, recuperação de falhas, segurança, controle de transações, otimização de consultas, dentre outros. A utilização de tais recursos facilitou a vida dos desenvolvedores de aplicações, possibilitando que estes pudessem se preocupar exclusivamente com o foco da aplicação. "(BRITO,2013)

Bancos de dados se caracterizam pelo seu potencial para garantir padrões, tempo reduzido para desenvolvimento de aplicações, flexibilidade, disponibilidade de informações atualizadas, economia de escalas.

### 2.3.1 Banco de dados Relacional

Entre os inumeros formatos de bancos de dados modernos que temos hoje em dia, os precursores, são os bancos de dados relacionais, que iniciaram sua história nos anos 70.

> "Desenvolvida originalmente pela IBM, o SQL é uma linguagem declarativa para banco de dados relacional inspirada na álgebra relacional. Sua simplicidade e alto poder de expressão fizeram do SQL a linguagem de consulta de dados mais utilizada no mundo e ajudou a

> consolidar a posição dominando do modelo relacional. Tendo surgido como sucessor dos modelos hierárquico e de rede, o modelo relacional tornou-se padrão para a grande maioria dos SGBDs (Sistemas Gerenciadores de Banco de Dados), tais como o SQL Server, Oracle, PostgreSQL, MySQL, etc. Seus elementos básicos são as relações (ou tabelas), as quais são compostas de linhas (ou tuplas) e colunas (ou atributos). ” (BRITO,2013)

Apesar de estar entre os mais antigos, os bancos de dados relacionais, se mostram eficientes e atualizados para sistemas conservadores de armazenamento e agrupamento de dados relacionais. Tendo como uma das principais ferramentas o SGBD MySQL.

#### 2.3.1.1 MySQL

Contando com código aberto, o MySQL é entre os mais utilizados, principalmente para aplicações web e sistemas na nuvem.

> “MySQL é o mais popular Gerenciados de sistema de banco de dados SQL de código aberto, é desenvolvido, distribuído e suportado pela oracle Corporation.” (MySQL,2016)

> “Sistema de gerenciamento de banco de dados relacional, open source, o servidor MySQL é muito rápido, confiável, escalável e muito fácil de uso.”(MySQL, 2016)

> “Os SGBDs relacionais oferecem aos usuários processos de Validação, verificação e garantias de integridade dos dados, Controle de concorrência, recuperação de falhas, segurança, controle de transações, otimização de consultas, dentre outros.”(BRITO,2013)

> “Outra característica importante dos SGDBs relacionais consiste na possibilidade do sistema se recuperar de forma adequada de possíveis falhas. O sistema tem a capacidade de retornar ao ponto anterior à falha ocorrida, garantindo que o banco permanecerá em um estado consistente.”(BRITO, 2013)

São estas características que justificam sua posição de liderança no mercado de banco de dados. A flexibilidade e o aperfeiçoamento a cada versão lançada ao mercado, buscando associar os novos padrões apresentados pelas ferramentas de NOSQL as ferramentas de eficacia comprovada dos bancos de dados relacionais.

### 2.3.2 Banco de dados não relacional

Apesar de não ser tão antiga quanto os bancos de dados relacionais, os bancos de dados NoSQL, vem tornando-se uma alternativa cada vez mais viável, com isto criando adeptos e seguidores.

> “Como os Patriotas, que se rebelaram contra os britânicos devido as altas taxas, NoSQLers começaram a buscar como acabar com a tirania da lentidão, caros bancos de dados relacionais, buscando mais eficiência e caminhos mais baratos de manusear dados.” (Computerword,2009)

Esta paixão de seus seguidores, em sua maioria programadores, normalmente se justifica, pelo seu manuseio simples e via shell, linha de comando.

> “NoSQL, é um derivado do sistema de banco de dados RDB. Como o próprio nome indica, NoSQL não é um banco de dados SQL, mas sim uma ferramenta de nível shell. Dados NoSQL contem arquivos ASCII regulares UNIX, e por isso pode ser manipulado por utilitários UNIX regulares, por exemplo, ls, wc, mv, cp, cat, head, more, less, editores como 'vi', etc., bem como pelos sistemas de controle de versão poderosos, como RCS e CVS. Baseado em arquivo, contendo tabelas, linhas e colunas. Para extrair informações, utilize um ou mais operadores UNIX INPUT/OUTPUT “ ( Carlo Strozzi, 2007)

No site MongoDB, que é uma das principais ferramente NoSQL para manipulação e armazenamento de dados, explica que é uma ferramenta versátil que pode atender inúmeras soluções.

> “NoSQL engloba uma variedade de diferentes tecnologias em banco de dados que pode atender diferentes tipos de aplicações modernas. Desenvolvedores trabalham com aplicações para criar um grande volume dados, alterando rapidamente seus tipos, estrutura, semiestruturada e polimorfismo.” (MongoDB, 2016)

Principalmente problemas de quantidade de dados, que hoje são captados e armazenados por qualquer dispositivo, e organizar estes dados, ou fazer alterações, pode ser uma tarefa bem complicada e pesada utilizando bancos de dados relacionais, uma situação que se faz simples com o uso de NoSQL, e deixa o sistema livre para crescer e ter escalabilidade.

> ”A grande motivação para NoSQL é resolver o problema de escalabilidade dos bancos tradicionais. Pode ser muito caro ou/e complexo escalar um banco SQL.
> Esse movimento está bastante enraizado no open source. Dá para perceber isso até mesmo pelos curiosos nomes dos projetos: Voldemort, MongoDB, Tokyo Tyrant e CouchDB.” (Nico Steppat, 2011)

Apesar da excentricidade dos nomes adotados, o grande foco dos bancos de dados NoSQL é a performance e a grande quantidade de dados manipulados, e este movimento ganhou muita força com a ajuda do google e da amazon, como Nico Steppat citou.

> ‘Apesar de grande quantidade desses bancos serem open source, o movimento ganhou muita força com a publicação de dois papers sobre implementações proprietárias: o Google Bigtable e o Amazon Dynamo. Não por acaso são duas empresas que lidam com uma quantidade enorme de informações.’ (Nico Steppat, 2011)

> ‘Big Data é um sistema de armazenamento distribuído, para gerenciar estruturas de dados desenhadas para escala e grande tamanho, petabytes de dados, espalhados por milhares de servidores. Muitos projetos da Google armazenam bigdata, incluindo GOOGLE Earth, Google Finance.’ (Google,2006)

Hoje o termo bigdata esta cada vez mais em pauta, pois com o surgimento de wearables e acessórios para casa que se comunicam entre si ou com sistemas de serviços ou outros, a quantidade de dados se multiplica e armazenar e analisar estes dados se torna um desafio.

#### 2.3.2.1 MongoDB

Desde 2007 o MongoDb vem se destacando entre seus parceiros e concorrentes SQL e NoSQL, pela sua versatilidade de performance de resposta a busca em bancos de dados.

> “Mongo DB foi fundada em 2007 pelo pessoal por trás de DoubleClick, Shopwiki and Gilt Group, A companhia foi criada para dar poder e mais eficiência a nuvem, para escalar horizontalmente, e tornar mais fácil e escalável o desenvolvimento. Hoje, mongoDB operou mais de 15 milhões de downloads. É um sistema de manipulação de banco de dados Open Source”(MONGODB, 2016)

Como escalabilidade é um dos objetivos principais do MongoDB, sua definição no site se encaixa perfeitamente.

> “Um banco de dados para ideias gigantes, ” (MongoDB,2016)

### 2.3.2.2 Neo4J

Entre os bancos de dados não relacionais, o que mais se diferencia, na minha opinião é o NEO4J, pelas suas caracteristicas de montagem de redes e ligações entre dados, seja por ação ou proximidade. Teve seu inicio em 2003 como é explicado no seu site.

> “Patrocinado pela Neo Technology, o Neo4j é um banco de dados de código aberto NoSQL implementado em Java e Scala. Com o desenvolvimento a partir de 2003, está disponível publicamente desde 2007. O código-fonte e o acompanhamento de problemas estão disponíveis no GitHub, com suporte prontamente disponível no Stack Overflow e no grupo Neo4j do Google. Neo4j é usado hoje por centenas de milhares de empresas e organizações em quase todas as indústrias. Casos de uso incluem matchmaking, gerenciamento de rede, análise de software, pesquisa científica, roteamento, gerenciamento organizacional e de projetos, recomendações, redes sociais e muito mais.” (Neo4J, 2016)

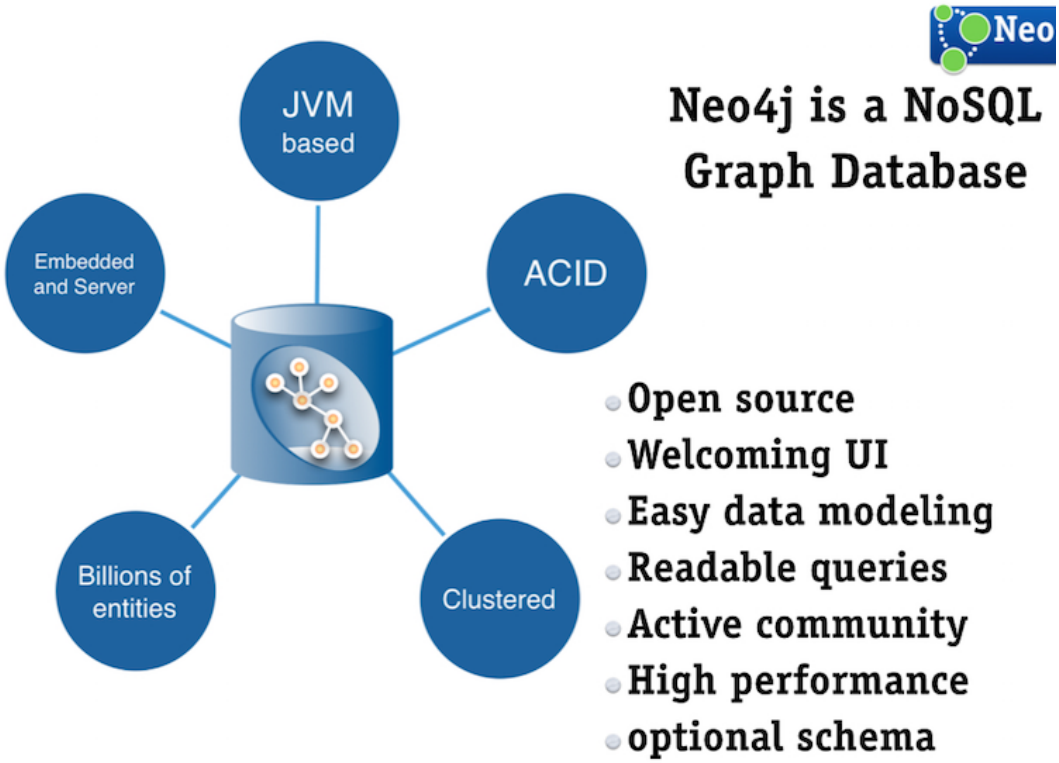


**Figura 6: estrutura NEO4J(NEO4J, 2016).**

“Cypher é a linguagem de consulta gráfica aberta para Neo4j. Cypher’s entrega uma sintaxe familiar podendo linkar vários nós e relações em um gráfico.” (NEO4J, 2016)

```
MATCH (actor:Person)-[:ACTED_IN]->(movie:Movie)
WHERE movie.title STARTS WITH "T"
RETURN movie.title AS title, collect(actor.name) AS cast
ORDER BY title ASC LIMIT 10;
```

**Figura 7: Estrutura de consulta NEO4J**

Com o surgimento e popularização das redes sociais, seu uso ficou mais evidente e potencializado, pois ele é adotado pelas principais redes do mercado, e isto comprova o seu verdadeiro potencial e uso.

> "Quando os bancos de dados de gráficos começaram a se popularizar com aplicações sociais para a web (Facebook, LinkedIn, Twitter), seus casos de uso se estendem muito além do espaço social. As organizações empresariais de hoje usam a tecnologia de banco de dados de gráficos de diversas maneiras, incluindo estes seis casos de uso mais comuns:
>
> • Detecção de fraude
>
> • Motores de recomendação em tempo real
>
> • Gerenciamento de dados mestre (MDM)
>
> • Operações de rede e TI
>
> • Gerenciamento de identidade e acesso (IAM)
>
> • Pesquisa baseada em gráficos" (NEO4J, 2016)

Com estas características esperamos poder entregar um conteudo mais atrativo e que ajude o usuário a permanecer e encontrar os produtos que deseja em nossos sistemas.

## 2.4 Desenvolvimento de aplicativos:

Com o surgimento dos aparelhos celulares, existiu uma explosão de demanda para o desenvolvimento de aplicativos, pois cada vez mais as pessoas querem a comodidade e personalização dos seus serviços favoritos. E um dos principais sistemas operacionais é o Android.

### 2.4.1 Sistema Operacional Android

Por ser Open Source, o android, foi adotado por inumeras marcas de celulares do mercado,

> “Android é o sistema operacional que move mais de um bilhão de smartphones e tablets. Já que esses dispositivos tornam nossas vidas tão doces, cada versão do Android recebe o nome de uma sobremesa. Seja para receber rotas ou jogar na Internet, cada versão do Android traz alguma novidade.” - (History Google, 2016) -

### 2.4.2 Aplicativos com Apache Cordova

Buscando facilidade, flexibilidade e utilizando o conceito de persistência poliglota, o framework apache cordova, nos traz muito mais possibilidade de desenvolvimento e fácil manutenção de aplicativos.

> “O Framework de desenvolvimento móvel, permite utilizar padrões web HTML5, CSS3 e Javascript para desenvolvimento Cross-plataform, cada plataforma tem seu renderizador que permite utilizar os recursos do aparelho. ” (Cordova, 2016)

| | android | blackberry10 | ios | Ubuntu | wp8 (Windows Phone 8) | windows (8.1, 10, Phone 8.1) | OS X |
|---|---|---|---|---|---|---|---|
| cordova CLI | ✔ Mac, Windows, Linux | ✔ Mac, Windows, Linux | ✔ Mac | ✔ Ubuntu | ✔ Windows | ✔ | ✔ Mac |
| Embedded WebView | ✔ (see details) | ✘ | ✔ (see details) | ✔ | ✘ | ✘ | ✔ |
| Plugin Interface | ✔ (see details) | ✔ (see details) | ✔ (see details) | ✔ | ✔ (see details) | ✔ | ✔ |
| | Core Plugin APIs | | | | | | |
| Accelerometer | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ |
| BatteryStatus | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ * Windows Phone 8.1 only | ✘ |
| Camera | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ |
| Capture | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ |
| Compass | ✔ | ✔ | ✔ (3GS+) | ✔ | ✔ | ✔ | ✘ |
| Connection | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ |
| Contacts | ✔ | ✔ | ✔ | desktop only | ✔ | partially | ✘ |
| Device | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| Events | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ |
| File | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| File Transfer | ✔ | ✔ * Do not support onprogress nor abort | ✔ | ✘ | ✔ * Do not support onprogress nor abort | ✔ * Do not support onprogress nor abort | ✘ |
| Geolocation | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ |
| Globalization | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ |
| InAppBrowser | ✔ | ✔ | ✔ | ✔ | ✔ | uses iframe | ✘ |
| Media | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | |
| Notification | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ |
| Splashscreen | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ |
| Status Bar | ✔ | ✘ | ✔ | ✘ | ✔ | ✔ Windows Phone 8.1 only | ✘ |
| Storage | ✔ | ✔ | ✔ | ✔ | ✔ localStorage & indexedDB | ✔ localStorage & indexedDB | ✘ |
| Vibration | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ * Windows Phone 8.1 only | ✘ |

**Figura 8: Plataformas suportadas pelo Cordova, e acesso via plugins (CORDOVA, 2016)**

Como podemos verificar na figura 8, A aplicação cordova tem compatibilidade para diferentes sistemas operacionais, como Android, Blackberry, IOS, Ubuntu, Windows e OS X. E na maioria das aplicações temos acesso a marioria das funcionalidades dos aparelhos, como status da bateria, acelerômetro, câmera, conexão, entre outros. Esta facilidade de implementação e comunicação com diversos aparelhos, faz com que muito programadores e desenvolvedores de aplicativos, optem pelo uso desta plataforma hibrida, que utiliza apenas uma linguagem para atender diferentes a todos.

Caracterizado pelo uso de elemento html, css e javascript em sua base, facilita a vida de programadores naturais a WEB e traz comunicação simples os seus componentes internos do celular. Sejam câmeras, localização via GPS ou acesso a rede, utilizando o conceito de web views.

> "Web View é o modo de entrada com interface para usuários, podendo se comunicar com aplicações híbridas e mesclar com componentes nativos. Web App, é onde o código da aplicação reside, utilizando html, css e javascript, pode-se personalizar tudo que é necessário para rodar seu app. Plugins, faz parte do ecossistema do Cordova, realizando a interação com os componentes nativos do aparelho, Tudo isto utilizando Javascript" (Cordova, 2016)

A figura a seguir, exemplifica como acontece a comunicação entre o web view, os componentes e plugins

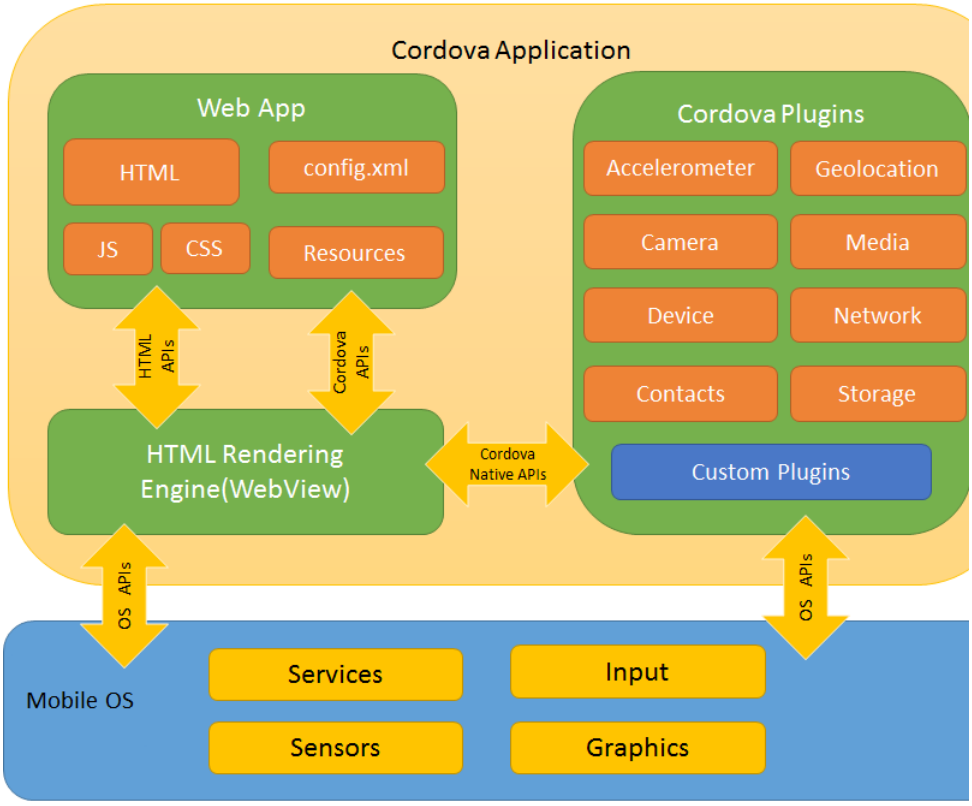


**Figura 9: Arquitetura Apache Cordova (CORDOVA,2016)**

Com base nos estudos e nesta fundamentação teórica, esperamos conseguir apresentar o estudo de caso e seus resultados ao fim deste trabalho.

# 3 Estudos de caso

Uma aplicação WEB baseada em PHP, MySQL, JavaScript e HTML5, onde provemos uma aplicação de portais imobiliários, com 260 ramificações em todo o território nacional, hospedado em uma nuvem, totalmente gerenciada, utilizando como base o conhecimento adquirido na TECSOL.

Esta plataforma é alimentada, 80% por meio de integradores, que se comunicam diretamente e diariamente com o sistema CMS dos clientes cadastrados.

Com a expansão dos serviços e prospecção de novos clientes no território nacional, uma tabela do banco vem apresentando um crescimento médio de 20% ao mês, está sendo a tabela principal do projeto, tendo consultas e updates diariamente pelos integradores, e a qualquer momento pelos clientes diretos, que utilizam os painéis para atualização. No mês de outubro 2016, apresenta 165k de registros, utilizando 430mb de espaço, sendo 155mb de chaves de indexação.

A tabela imóveis ainda se complementa pela tabela de imóveis imagens, que foi criada para que os clientes possam usufruir de um número ilimitado de imagens em suas vitrines de produtos. Simulando que cada imóvel tenha uma média de 20 imagens, podendo variar. Temos uma ocupação de disco de 1.2GB sendo 500Mb de chaves de indexação. Atualmente com 4.100 milhões de registros.

O tamanho total deste banco de dados, gira em torno de 8gb.

Ainda acessando este banco de dados temos toda a parte de financeiro, cadastro de clientes, administração de projeto. Sendo necessário um servidor dedicado ao banco de dados MySQL, com 8 cores e 14 GB de memória. Utilizando tunning para otimizar memória, buffer, pool e outras funcionalidades de concorrência e cache.

Observando esta estrutura ainda temos um sistema de logs, que registra as intenções dos clientes e suas pesquisas. Este tratado de maneira unitária diariamente no banco de dados usual, e otimizado, agrupado e salvo em um banco de dados paralelo. Gerando diariamente cerca de 1 milhão de registros únicos em 50 MB de espaço, agrupados em 30 MB e 300mil registros.

Com esta estrutura, em uma interface que utiliza PHP com frameworks que otimizam a performance e exclusivamente banco de dados MySQL, em um servidor

web dedicado, temos tempo de resposta de páginas em média de 2.4s uma vez que apenas a consulta de imóveis no SGBD MySQL gasta em média 1.1s.

Visando a otimização do tempo de resposta e prover cada vez mais serviços, como aplicativo, estatística otimizada, entregas de sugestões de relacionamento mais qualificadas, sentimos a necessidade de fragmentar os serviços e ampliar o número de ferramentas de auxílio, tanto na parte de banco de dados, como de aplicação. Onde entram as ferramentas de banco de dados NoSQL e Web services.

O Web service baseado em NodeJS, buscará centralizar a aquisição de dados junto as diferentes ferramentas de armazenamento, provendo dados para aplicativo, páginas web e sistemas de gerenciamento de sistema.

Resultando em um sistema de persistência poliglota, utilizando linguagens de programação diversificadas, Java para aplicativo android, javascript e html5 utilizando cordova para compatibilidade com outros equipamentos diferentes de Android, API REST em nodeJS para consulta e entrega de dados, PHP com Codeigniter na administração de serviços como cadastro de clientes, gerenciamento de projetos e rotinas administrativas.

Os bancos de dados adotados:

- MySQL para serviços administrativos e que demandam atualização constante e consistência de dados.

- MongoDB para armazenamento de imóveis e entrega de forma mais rápida e baixa concorrência.

- Neo4j para monitoramento de comportamento e sugestão de produtos ao cliente.

Um dos principais objetivos é retirar o peso sofrido pelo banco de dados MySQL e dividi-lo entre os demais, diminuindo a necessidade de máquina dedicada de alto consumo, podendo assim reduzir custos e esforço computacional.

Com a ampliação do escopo do projeto com persistência poliglota, devemos descrever quais são as tarefas, suas características e a ferramenta responsável, como será a integração destes dados e atualização.

Dividimos o escopo em 4 áreas:

Administrativa: Cadastro de empresas, gerenciamento de ocorrências, cadastro de contatos realizados, cadastro de interessados em imóveis, cadastro de logradouros.

Utilizando tabelas com relacionamento de chaves primarias e secundárias.

Produtos: Cadastro de Imóveis

Relacionamento: Cadastro e Sugestão de produtos relacionados.

Controle de acesso e pesquisa: Armazenamento de log de acesso, armazenamento de log de pesquisa e interesse.

## 3.1 MySQL

O Banco de dados MySQL com suas características de ACID (Atomicity, Consistency, Isolation, Durability).

> *“Atomicidade, significa que n*a transação ou se faz tudo, ou nada, sem meio termo. Pensando que em uma transação podemos ter mais de uma operação.
>
> *Consistência, t*em por objetivo garantir que o banco de dados antes da transação esteja consistente e, que após a transação o banco permaneça consistente, sem problemas de integridade.
>
> *Isolamento, o*bjetiva garantir que nenhuma transação seja interferida por outra até que ela seja completada.
>
> *Durabilidade, c*omo o nome já pode nos remeter, esta propriedade garante que a informação gravada no banco de dados dure de forma imutável até que alguma outra transação de atualização, ou exclusão afete-a.” (NISHIMURA, 2013)

Consistência e integridade dos dados, são essenciais para sistemas financeiros, gerenciamento de projetos e dados que são constantemente atualizados, e precisam ser mostrados com exatidão e atomicidade. Ganhasse muito com o uso de chaves primárias e secundárias, deixando o sistema mais confiável.

> “Indexes são usadas para encontrar rows específicas com valores nas colunas, rapidamente, Sem index, mysql percorrerá todos os registros para encontrar o valor buscado. Em tabelas grandes, isto gera mais custo. Se na tabela existir uma index para a coluna em questão, o MySQL pode rapidamente determinar a busca a partir de certo ponto ao invés de todo o documento.”(MySQL, 2016)
>
> “Chaves Primarias, únicas, index e fulltext, são armazenadas em B-tree, indexes spaciais utilizam R-tree.” (MySQL, 2016)

### 3.1.1 Estrutura de dados

Vamos abordar e focar nos dados necessários para que a aplicação entregue a lista de imóveis, com galeria de imagens e empresa responsável pelo anúncio, cadastro de usuários com interesse em imóveis, tipos de imóveis, logradouros, cidades.

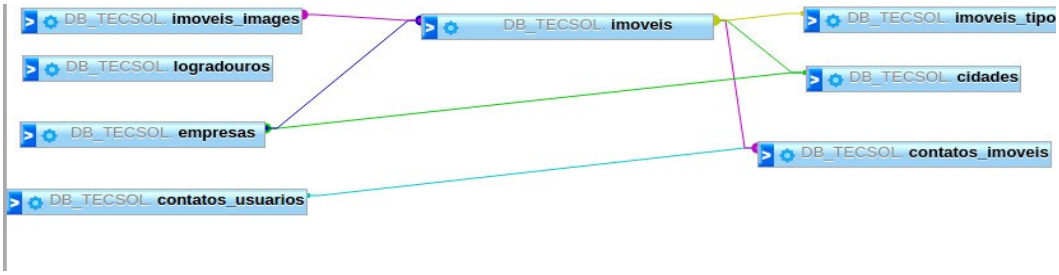


**Figura 10: Requisito de estrutura de dados MySQL(acervo)**

## 3.2 MongoDB

Na busca de velocidade, performance e utilizar índices spaciais simples, MongoDB é uma ferramenta que apresenta o conceito de BASE (Basically Available, Soft state, Eventual consistency). Principalmente pelo baixo índice de alterações, A parte de edição e inclusão de dados será feito no banco de dados MySQL, e por parte de integração de dados de clientes. E buscando a facilidade para o programador tratar os dados em JSON.

Um ponto atraente no MongoDB e que nos convence que é a escolha certa, são seus índices espaciais, que através de localização de latitude e longitude, agrupados em um campo location, podemos ter uma versatilidade de resultados, com rapidez.

### 3.2.1 Principais características.

> “ O MongoDB é um banco de dados documental que armazena os dados em formato BSON, uma implementação leve, rápida e altamente examinável. Como o JSON, MongoDB BSON suporta incorporação de documentos, arrays com objetos e outras arrays. MongoDB pode até mesmo atingir o interior do objeto BSON para criar indexes e objetos correspondentes. Isto deixa o MongoDb fácil e flexível com JSON documents, velocidade e leveza do formato binário.”(MongoDb, 2016)

> “Com MongoDB, é possível criar aplicações que não eram possíveis com banco de dados relacionais, rápidas e iterativo desenvolvimento, modelo

> de dados flexível, escalabilidade, recursos integrados.” (mongodb-architecture, 2016)

> “Esse conceito é estendido para o paradigma chamado BASE (Basically Available, Soft state, Eventual consistency) que se caracteriza por ser basicamente disponível, ou seja, o sistema parece funcionar o tempo todo; em estado leve, o sistema não precisa ser consistente o tempo todo; e eventualmente consistente, o sistema torna-se consistente no momento devido” (BRITO, 2013)

### 3.2.2 Estrutura de dados

MongoDB dispensa Schemaless, porém ele deve existir e ser gerenciado pelo código e programador, portanto, abaixo temos um modelo de schema no formato JSON, que é a base da estrutura BSON.

**{ "_id" : (int), "nome" : (string), "preco" : (double), "data_atualizacao" : (string), "logradouro" : (string), "video" : (string), "quartos" : (int), "garagens" : (int), "banheiros" : (int), "área" : (double), "área_terreno" : (double), "área_util" : (double), "bairro" : (string), "bairros_link" : (string), "imoveis_tipos_titulo" : (string), "imoveis_tipos_english" : (string), "imoveis_tipos_link" : (string), "id_tipo" : (int), "cidades_link" : (string), "tipo_negocio" : "venda", "id_empresa" : (int), "id_cidade" : (int), "cidade_nome" : (string), "uf" : (string), "imobiliaria_nome" : (string), "logo" : (string), "imobiliaria_logradouro" : (string), "imobiliaria_bairro" : (string), "imobiliaria_cidade" : (string), "imobiliaria_numero" : (string), "empresa_email" : (string), "descricao" : (string), "referencia" : (string), "ordem" : (int), "uso" : (string), "suites" : (int), "mobiliado" : (int), "comercial" : (int), "residencial" : (int), "lazer" : (int), "ddd" : (int), "cidade" : (string), "estado" : (string), "cidade_link" : (string), "email_corretor" : (string), "nome_corretor" : (string), "sms_corretor" : (string), "imagens" : { { "status" : bolean, "arquivo" : (string), "titulo" : (string), "id" : (int) } }, "data_update" : ISODate(), "location" : [(double),(double)] }**

**Estrutura Schema MongoDB**

### 3.2.3 Spatial Index

> “MongoDb oferece um grande número de mecanismos de index e querys para utilizar informações geoespaciais.“ (MONGODB, 2016)

O armazenamento de dados de localização, depende do tipo de query, superfície, e index que escolher trabalhar.

> “GeoJSON objects sempre calculam em uma esfera. O sistema de referência de coordenadas padrão do GEOJSON é o WGS84 datum.” (MONGODB, 2016)

Objetos como Point, LineString, Poligono, MultiPonto, Multi LineString, MultiPolygon, GeometryCollection. No caso, point seria o mais indicado para o momento de buscar pontos em localização.

Operadores de consulta, por proximidade, utilizando o operador “$near” para utilizar o index 2d, calculando geometria plana.

Estes pontos estão disponíveis a partir da versão 2.4

```
[
        {
                "v" : 1,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "imoveis.imoveis"
        },
        {
                "v" : 1,
                "key" : {
                        "location" : "2d"
                },
                "name" : "location_2d",
                "ns" : "imoveis.imoveis"
        }
]
```


**Figura 11: Estrutura de index mongoDB 2d**

## 3.3 Neo4J

### 3.3.1 Principais características

> “O maior valor que os gráficos trazem para o desenvolvimento é a sua capacidade de armazenar relacionamentos e conexões como entidades de primeira classe. Os primeiros adeptos da tecnologia gráfica reimaginaram seus negócios pelo valor das relações de dados. E hoje são lideres em suas áreas.” (NEO4J, 2016).

Para pensarmos um negócio em forma de graphos, precisamos primeiro entender o conceito envolvido e quais as melhores aplicações de graphos.

> “Um gráfico é composto por dois elementos: um nó e um relacionamento. Cada nó representa uma entidade (uma pessoa, um lugar, uma coisa, uma categoria ou outro pedaço de dados) e cada relação representa como dois nós estão associados. Por exemplo, os dois nós "bolo" e "sobremesa" teriam "é um tipo de" relação de apontando de "bolo" para "sobremesa”. Esta estrutura tem o propósito geral de permitir modelar todos os tipos de cenários - a partir de um sistema de estradas, a uma rede de dispositivos, à história médica de uma população ou a qualquer relacionamento.”

As relações são compostas de nós, relacionamentos e atributos. Os relacionamentos podem ser discriminados e inseridos através de tags, “[:COMPROU]”, “[:OBSERVOU]” e servem de ligação entre nós e atributos, indicando o sentido de seu relacionamento através do sinal “-[:COMPROU]->”.

Como o SQL, os bancos de dados graphos tem sua linguagem de interpretação que é o Cypher.

> “Cypher combina os padrões em um grapho imediatamente em torno da ancora baseada em informações de relação com nós vizinhos.” (Neo4J, 2016).

E existem alguns comandos básicos que podem ajudar o programador a organizar e alcançar os resultados esperados, os principais, MATCH, RETURN, fazem o relacionamento e retornam os dados. WHERE, trata os critérios e filtros para MATCH e RETURN. CREATE and CREATE UNIQUE, criam nós e relações. MERGE, garante que existem padrões no grapho e reutiliza ou cria nós e relações. DELETE/REMOVE. SET, carrega propriedades de chaves e valores. ORDER BY, ordena o resultado. SKIP LIMIT, limita os resultados. FOREACH, otimiza as ações de update para todos os elementos de uma lista. UNION, une resultados de 2 ou

mais queries. A maioria dos comandos é familiar aos usuários de bancos de dados relacionais, facilitando assim a migração de dados.

Mais um ótima ferramenta de compatibilidade para colaborar e agilizar o trabalho dos programadores, é a importação de dados via LOAD para arquivos .csv.

> “No Neo4j, LOAD CSV é uma chave de Cypher que habilita você a carregar arquivos CSV, de HTTP ou URL’s para dentro de seu banco de dados, você pode criar e recarregar nós e relações para seu grapho.”(NEO4J, 2016).

Podemos utilizar várias formas de comunicação, podendo obter e inserir dados, tanto via “Drivers” para diversas linguagens como via REST API incorporada.

> “Se você preferir acessar Neo4j, pode fazer isso com a REST API que permite:
>
> • POST um ou mais declarações Cypher com parâmetros por solicitação ao servidor
>
> • Manter transações abertas em vários pedidos
>
> • Obter diferentes formas de resultados
>
> • Executar opções de gerenciamento ou introspeccionar o banco de dados.” (NEO4J, 2016)

### 3.3.2 Modelo de aplicação.

Seguindo o conceito de relações nós, teremos dois nós principais, o Usuário, e o Imóvel, e nós secundários de tipo de negócio, tipo de imóvel, características.

Os primários serão identificados por ID comparado a chave primária. O usuário está habilitado a Clicar, telefonar e enviar e-mail. E pode PESQUISAR tipo de negócio, tipo de imóvel e características. O imóvel SET seu tipo de negócio, tipo de imóvel e características.

Os dados que pretendemos extrair, com base neste fluxo de nós e relacionamento, é o sentido das pesquisas de imóveis no portal, e com base nas características, quais são os imóveis com maior índice de cliques e de contatos, seja via telefone ou e-mail, com base na pesquisa. Esta analise e resultado, será util para sugestões mais assertivas de e-mail mkt, sugestões em forma de migalha de pão no Portal, aumentando o tempo que o usuário utiliza o sistema e gerando assim aumento das conversões em imóveis com características similares.

Como fluxograma abaixo:

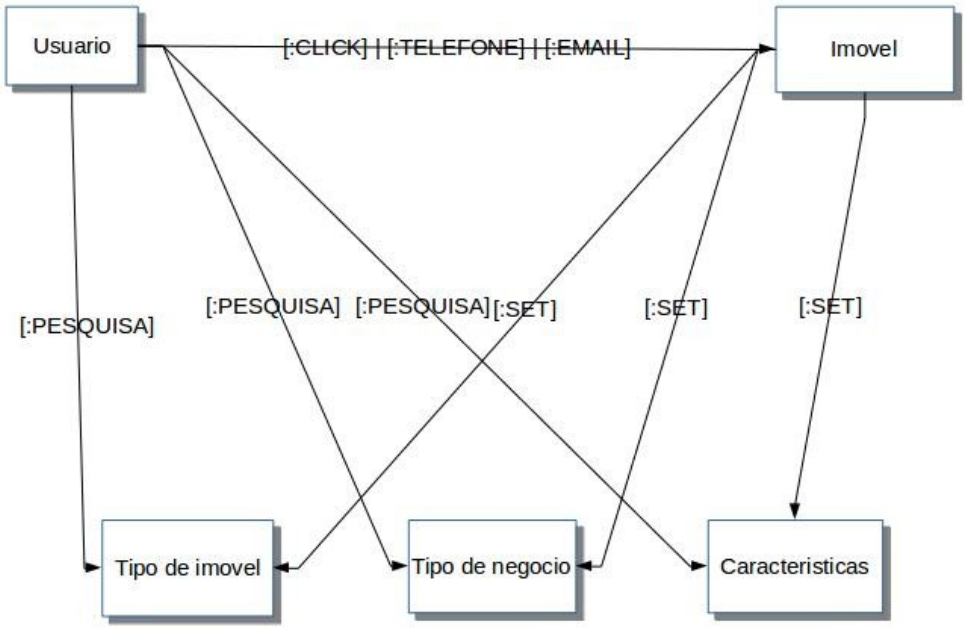


**Figura 12: fluxograma de dados em neo4j.**

A utilização de cookies e sessão para os usuários, ajudará a consolidar os dados, uma vez que na entrada do Portal, não temos a identificação do usuário, apenas após um contato convertido. Tratando primeiramente os dados com identificação de IP de rede e/ou MAC, e realizando um update após a conversão.

## 3.4 REST API

O desenvolvimento para multiplataformas e diferentes modelos de equipamento, uma realidade que se consolida cada vez mais, nos leva a necessidade de armazenamento de dados e distribuição de dados para as ferramentas nativas de cada equipamento, quando possível, por exemplo, para WEB e bem compatível com navegadores modernos, o javascript, junto com NodeJS, pode prover um serviço competente, podendo ser utilizado desde a interface, até a comunicação com múltiplos bancos de dados.

> “Node foi idealizado para servir redes de aplicação escalável, utiliza javascript assíncrono em tempo de execução.

> Node tem um design similar e influenciado por linguagens como Ruby e Python. Node leva o modelo um pouco mais longe, mantendo o event loop em tempo de execução do construtor em vez de bibliotecas. Node simplesmente executa o event loop depois da entrada do script, e sai do event loop quando não é mais necessário. E o event loop é invisível ao usuário.” (NODEJS, 2016)

Para auxiliar no compartilhamento e versionamento das aplicações o NodeJS possui um repositório chamado “NPM”, que mantêm seus pacotes organizados e atualizados.

> “Assim como o RubyGems do Ruby ou o Maven do Java, o Node.js também possui seu próprio gerenciador de pacotes, que se chama NPM (Node Package Manager). Ele se tornou tão popular pela comunidade que, desde a versão 0.6.X do Node.js, ele foi integrado no instalador principal do Node.js, tornando-se o gerenciador padrão desta plataforma. Hospeda mais de 250.000 módulos Node.js criados por terceiros e comunidades. Diariamente são efetuados mais de 180 milhões de downloads e, mensalmente, são cerca de três bilhões de downloads de diversos módulos.”(PEREIRA,2016)

Com o aumento do consumo de dados em várias plataformas diferentes, o padrão de dados JSON, vem sendo cada vez utilizado e suportado por várias plataformas.

> “ JSON (JavaScript Object Notation) é um formato leve de intercâmbio de dados. É fácil para os seres humanos ler e escrever. É fácil para as máquinas analisar e gerar.”(JSON,2016)

Com bibliotecas e conversões na maioria das linguagens, vem aparecendo cada vez no mercado, sendo utilizado como base para alguns bancos de dados NoSQL, como mongoDB e Neo4j, tornando-se uma ferramenta de fácil manipulação e interpretação, além de leve para trafegar na rede.

### 3.4.1 Estrutura e dependências:

Para o desenvolvimento em NodeJS é importante utilizar os pacotes disponíveis no NPM e que facilitem e criem atalhos no desenvolvimento da API. Abaixo vamos explicar algumas bibliotecas muito uteis.

body-parser – Gerência as transações, realizadas através do corpo do documento, liberando estas informações para serem gerenciadas pela API.

Express – Framework, responsável pelo roteamento, gerenciamento de HTTP, redirecionamento, cache e negociação de conteúdo.

Mongodb – Drive nativo de comunicação com MongoDB

neo4j – Drive de comunicação com Neo4J

sequelize – Driver para comunicação com bancos de dados, MySQL e SQLite3

passport – Responsável pela autenticação e segurança do sistema, utiliza chaves de criptografia para gerenciamento de sessões.

Várias outras bibliotecas podem ser utilizadas, como gerenciamento de pacotes, https e qualquer funcionalidade necessária para a aplicação.

### 3.4.2 INPUT e OUTPUT

Na utilização de REST API baseado em NodeJS, é possível requisitar dados e receber respostas dos dados via curl, para aplicações HTTP ou PHP, que contenham parâmetros via body, utilizando métodos GET, POST, UPDATE, DELETE.

Sendo GET, Roteamento para requisição de dados, contendo parametros em body e roteamento em URL, ex: http://localhost:3000/imoveis/ e o body com parametros em JSON: {location:[0,0], raio: 8}, muito utilizado no aplicativo Android, que requisitará os dados utilizando bibliotecas JAVA, podendo requisitar dados, no banco de dados MongoDB e salvar a pesquisa no banco de dados Neo4J, na mesma requisição.

POST, utilizado para inserção de dados através de roteamento e parametros. Normalmente utilizando no roteamento a base específica como imóveis ou empresas. Gerenciando qual o banco de dados a ser utilizado. O mesmo conceito se aplica aos métodos de UPDATE e DELETE, que realizam updates e exclusão de itens. Lembrando que toda transação realizada tem como formato de dados JSON.


**Figura 13: Exemplo de comando e resposta cURL.**

## 3.5 Aplicativo Android

Para compatibilidade e melhor ajustes de funcionalidades de aplicativos em celulares, tablets e pequenos portáteis, é indicado se utilizar o formato nativo de desenvolvimento e comunicação com periféricos. Utilizando a linguagem indicada, no caso de Android, Java.

Para o melhor desenvolvimento para Android, disponibiliza o IDE Android Studio, que atualmente está na versão 2.2.2, alguns pontos importantes sobre o IDE abaixo.

> “Instant run, Envie alterações de códigos e recursos ao aplicativo executado em um dispositivo ou emulador e veja imediatamente as alterações em ação. O Instant Run acelera drasticamente os ciclos de edição, compilação e execução, mantendo você “no fluxo”.
>
> Editor de código inteligente, Escreva código melhor, trabalhe mais rápido e seja mais produtivo com um editor de código inteligente que orienta você a cada etapa do caminho. O Android Studio é baseado no IntelliJ e oferece conclusão de código, refatoração e análise de códigos avançados.
>
> Emulador rápido com recursos completos, Instale e execute aplicativos mais rapidamente que em um dispositivo físico e teste o aplicativo em praticamente qualquer configuração de dispositivo Android: telefones Android, tablets Android e dispositivos Android Wear e Android TV. O novo Android Emulator 2.0 oferece uma rapidez sem precedentes e permite redimensionar dinamicamente o emulador e acessar um conjunto de controles de sensor.
>
> Sistema de compilação robusto e flexível, Configure facilmente o projeto para incluir bibliotecas de código e gere diversas variações de compilação para um único projeto. Com o Gradle, o Android Studio oferece automação de compilação de alto desempenho, gerenciamento de dependências robusto e configurações de compilação personalizáveis.” (Android Developer, 2016).

Os componentes de android estão no SDK, bibliotecas e formas de comunicação entre aparelho e seus periféricos.

### 3.5.1 Google Maps API

Disponível para aplicativos e versão web, o google disponibiliza para os desenvolvedores, uma API completa, com todas as funcionalidades necessária para marcar pontos no mapa, traçar rotas, interagir com museus, e ruas com o google street view.

Para nossa aplicação vamos utilizar o marcador de pontos com janelas de informação do imóvel, buscando o conteúdo de imóveis na API REST, utilizando a localização extraída do GPS ou localização do celular, e tratando as informações por proximidade, tipo.

Para ativar o serviço de mapas, é necessário criar uma chave que ativa e registra as movimentações em seu aplicativo.

## 3.6 Aplicativo em Cordova

Apesar dos benefícios de se desenvolver aplicativos em suas linguagens nativas, temos uma dificuldade para compatibilizar todos os aparelhos e sistemas operacionais.

Para isto uma boa solução é o desenvolvimento utilizando Apache Cordova, um framework, que utiliza HTML, CSS e JS. Gerando compatibilidade entre os principais sistemas operacionais do mercado: Android, Blackberry 10, iOS, OS X, Ubuntu, Windows, WP8.

> “Mobile apps com HTML, CSS & JS, Alcance múltiplas plataformas com apenas um código, livre e de código aberto.” (CORDOVA,2016)

### 3.6.1 Ionic Framework

> “Ionic é um Framework HTML5 de código aberto para criação de projetos multiplataformas híbrido para aplicativos e websites moveis, com HTML, JavaScript , TypeScript e CSS. Tem como característica, a facilidade e agilidade no desenvolvimento”(DRIFTY, 2016)

Na busca por performance, fácil ambientação, compatibilidade e baixa manutenção, afinal estamos lidando com uma ferramenta que com poucos ajustes atende a diversos interpretadores de aplicativos, como android, IOS, blackberry, windows mobile, entre outros.

Para criar uma ferramenta mais robusta, Ionic foi construído em cima de AngularJS para que não pareça apenas bonito mas seja fácil de criar aplicativos ou websites, utilizando um poderoso SDK adequado com uma boa arquitetura e documentação para desenvolvedores.

O que nos ajuda com a escolha do Ionic, é a base de nodeJS e componentes de NPM, como a aplicação REST que reportamos anteriormente, utilizando JavaScript e sua facilidade e versatilidade de desenvolvimento.

Para comunicação com o aparelho celular, o Ionic utiliza as bibliotecas de Apache Cordova

### 3.6.2 Aplicação

Aproveitando os componentes de TABS e NAVS do Framework Ionic, Pretendemos ter 3 modelos de navegação:

- Busca por proximidade diretamente no mapa, com tipo de negócio e tipo de imóvel. Buscando imóveis, sempre a partir do ponto que o usuário definir.

- Busca por filtros, de tipo de imóvel, tipo de negócio, bairro e filtros avançados.

- Visualização aleatória de fichas de imóvel.

A comunicação será feita através de cURL, diretamente a API REST implementada, recebendo as respostas em JSON e processando as respostas dependendo da forma de visualização definida acima.

Janelas no caso do mapa, listas para resultado de filtro e ficha do imóvel. Podendo ser alterado o modo e continuar vendo a mesmo resultado de filtro.

Opção de favoritar um imóvel e visualizá-lo posteriormente ou compartilhar através de outros aplicativos.

# 4 Considerações Finais

Os temas abordados neste trabalho, buscaram otimizar as rotinas de desenvolvimento, diminuição de tempo de manutenção, otimização de uso de equipamento de armazenamento e processamento na nuvem, a utilização de Software livres na administração, armazenamento, comunicação, segurança entre outros serviços.

Esta otimização ficou visível, tanto no desenvolvimento deste trabalho como nas implementações realizadas na empresa POW, na questão de migração de servidores, administração, desenvolvimento de aplicativos, e manipulação de bancos de dados, tanto relacionais, como não relacionais e grafos.

Nas aplicações que nos propomos a otimizar, obtivemos um resultado visível e sensível, tanto a equipe de desenvolvimento como aos usuários finais que buscam seus imóveis na web. A facilidade de uso das aplicações e a velocidade foram os pontos mais relevantes e que devem nos inspirar a continuar a busca por novas tecnologias e estudo que colaborem com a continuidade da otimização.

## 4.1 Utilização e benefícios da persistência poliglota em banco de dados.

Com a divisão de tarefas do banco de dados MySQL, para os bancos MongoDB, Neo4J, tivemos uma diminuição de equipamento empregado para utilização e gerenciamento do banco de dados MySQL, para um servidor com 4 cores e 6gb e o impacto do uso de MongoDb e Neo4j, é praticamente imperceptível, uma vez que estão alocados no mesmo servidor WEB, web service.

A REST Api baseada em NodeJS, nos facilita a comunicação entre aplicação WEB, aplicativo e os bancos de dados, nos economizando também em horas trabalhadas na aplicação pois centralizamos as requisições e respostas e apenas um código. O uso de objetos JSON, simplifica o tratamento de dados no momento do usuário, seja no aplicativo, como na aplicação web.

## 4.2 Considerações na pesquisa de imóveis.

Com a pesquisa de imóveis sendo realizada no MongoDB, utilizando arrays JSON para armazenamento, dispensamos inúmeras consultas e relacionamentos, utilizando chaves de indexação, para conter apenas uma chamada ao banco, retornando todos os dados necessários para o preenchimento da ficha de imóvel.

Com isto tivemos um aumento de performance, com o carregamento da página caindo de 2.4s para 0.61s sendo tempo de resposta do banco de dados de 0.010s no MongoDB contra 1.13s do MySQL.

## 4.3 Aplicativo para corretores de imóveis

Com os estudos que acabamos de realizar, podemos planejar como próximos passos em nossas aplicações, um aplicativo de uso direto pelos corretos de imóveis que desejem utilizar os sistemas POW internet, em especial, portais imobiliários. Onde através do aplicativo, poderá realizar o cadastro do imóvel no momento da avaliação ou primeira visita ao imóvel.

Algumas das principais características devem ser no sentido de localização exata do imóvel, cadastro de vídeos e imagens, utilizando o potencial do aparelho celular que usam no dia a dia.

Sendo utilizado também o gerenciamento, apresentaremos estatistas do imóvel, baseado em Grafos com neo4J, que exibirá a performance de visualização, cliques e contatos para cada imóvel.

# 5 Conclusão

## 5.1 Software Livre

O uso de Software livre em servidores na nuvem gerenciáveis, nos ajuda a controlar, otimizar e entregar melhores serviços, seja por meio de aplicativos, aplicações web, web services, api entre outros serviços.

O controle de estado do servidor e upgrades em tempo de execução, nos ajuda a otimizar custos, podendo ser realizados testes de eficiência para caso de uso.

## 5.2 Persistência poliglota

Utilizando ferramentas de bancos de dados relacionais e não relacionais, alcançamos os objetivos citados no tópico inicial, que era de melhorar as entregas de imóveis nos portais imobiliários, e-mail marketing e aplicativos.

Testamos a performance dos bancos de dados MySQL e MongoDB, com base em um grupo de dados que compõe a ficha de imóveis e listagem, com base em 20 imóveis com 20 imagens e obtivemos uma grande otimização com o uso de mongoDB, como pode ser verificado na tabela abaixo:

| | MySQL | | Servidor WEB | |
|---|---|---|---|---|
| | Pré | Pós | Pré | Pós |
| Nucleos | 8 | 4 | 8 | 6 |
| Memória | 14 | 6 | 10 | 8 |
| | Query MySQL * | Query Mongo * | Carregamento Página ** | |
| Tempo médio | 1,135s | 0,010s | 2.4s | 0,61s |

* Query padrão para requisição de 20 imóveis, com 20 images
** Com base no aplicativo google pagespeed

**Tabela 1: Vantagens da persistência poliglota**

Com uma economia de 37,5% de núcleos de processamento e 41,7% de memória, e contando com o suporte e gerenciamento de servidores que no momento é por conta da equipe POW, conseguimos diminuir os custos de operação, manutenção e armazenamento de servidores em 29%, o que soma ao fim de 12

meses cerca de R$ 8,500,00 (oito mil e quinhentos reais). O que valida o verdadeiro beneficio de otimizar e utilizar persistência poliglota em sistemas web.

Além da melhora no tempo de entrega, tivemos uma redução nos recursos dedicados a servidor WEB e MySQL. O que nos faz concluir que com a modernidade e aumento na quantidade de ferramentas específicas para cada serviço, podemos entregar produtos melhores com uso de menos recursos.

O Status deste estudo, é de 80% aplicado as ferramentas vigentes, devendo ser um projeto futuro a análise de dados com graphos utilizando o banco de dados Neo4J, uma vez que os estudos estão concluídos 100%.

## 5.3 Aplicativo

No desenvolvimento de aplicativos, podemos concluir, com base no baixo conhecimento de desenvolvimento Java pelo responsável pela programação e na intenção de reaproveitar os estudos realizados para implementação de REST API web service utilizando nodeJS e Javascript, a melhor opção para nosso aplicativo é utilizar o framework Apache Cordova com auxílio de Ionic para desenvolvimento e comunicação com a API, uma vez que ambos se comunicam utilizando JSON e a comunicação com componentes que se faz necessário é totalmente compatível com o framework, não restando necessidade de programação na linguagem nativa.

## REFERÊNCIAS

BRITO, Ricardo W. Bancos de Dados NoSQL x SGBDs Relacionais: Análise Comparativa. Disponível em: <http://www.infobrasil.inf.br/userfiles/27-05-S4-1-68840-Bancos%20de%20Dados%20NoSQL.pdf>. Acesso 20 fevereiro 2017.

CHANG, Fay. DEAN, Jeffrey. GHEMAWAT, Sanjay. HSIEH, Wilson C. WALLACH, Deborah A. BURROWS, Mike. CHANDRA, Tushar. FIKES, Andrew. GRUBER, Robert. Bigtable: A Distributed Storage System for Structured Data – 2006. Disponível em <http://static.googleusercontent.com/media/research.google.com/pt-BR//archive/bigtable-osdi06.pdf>. Acesso em 20 fevereiro 2017.

CORDOVA. Cordova Documentation, 2016. Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>. Acesso em 20 fevereiro 2017.

DRIFTY. What is Ionic? Sem data. Disponível em: <https://www.npmjs.com/package/ionic-sdk>. Acesso em 20 fevereiro 2017

ELMARIS, Ramez; Navathe, Sistemas de banco de dados. Ed 6, 2011. Pearson Education - Br

FORD, Neal. Polyglot Programming. Meme agora, Dezembro,2006. Disponível em: <http://memeagora.blogspot.com.br/2006/12/polyglot-programming.html>. Acesso em: 01 outubro 2016.

FOWLER, Martin. Polyglot Persistence, Martin Fowler.com, Novembro, 2011 Disponíveis em <http://martinfowler.com/bliki/PolyglotPersistence.html> acesso em 01 outubro 2016.

Google Maps API.  O melhor do Google Maps para cada aplicativo para Android. Disponível em: <https://developers.google.com/maps/documentation/android-api/>. Acesso em 20 fevereiro 2017.

LAI, Eric. Twitter growth prompts switch from MySQL to 'NoSQL' database. ComputerWorld. Fevereiro, 2010, Disponível em: <http://www.computerworld.com/article/2520084/database-administration/twitter-growth-prompts-switch-from-mysql-to—nosql--database.html>. acesso em 20 fevereiro 2017.

LAI, Eric. No to SQL? Anti-database movement gains steam. Computerworld. Julho, 2009. Disponível em: <http://www.computerworld.com/article/2526317/database-administration/no-to-sql—anti-database-movement-gains-steam.html>. Acesso 20 fevereiro 2017

NEO4J. Neo4j: The World’s leading graph database. Sem data. Disponível em: <https://neo4j.com/product/>.  acesso em 20 fevereiro 2017.

NISHIMURA, Roberto Yukio. Os conceitos da propriedade ACID. PapoSQL. Maio 2013. Disponível em <http://paposql.blogspot.com.br/2013/05/os-conceitos-da-propriedade-acid.html>. acesso em 20 fevereiro 2017.

NODEJS. API Reference Documentation. Sem data. Disponível em <https://nodejs.org/en/docs/>. acesso em 20 fevereiro 2017.

METAL JAVA FULL. CRUD - O que é mesmo?. JavaFree. Março 2013, disponível em <http://javafree.uol.com.br/artigo/886032/CRUD-O-que-e-mesmo.html>. acesso em 20 fevereiro 2017.

MYSQL. MySQL 5.7 reference Manual. Disponível em: <https://dev.mysql.com/doc/refman/5.7/en/>. acesso em 20 fevereiro 2017.

MONGODB. The MongoDB 3.4 Manual. Disponível em <https://docs.mongodb.com/manual/>. acesso em 20 fevereiro 2017.

MONGODB. Top 5 Considerations when evaliating NoSQK Databases. A mongoDB White papers. Disponível em: <https://webassets.mongodb.com/_com_assets/collateral/10gen_Top_5_NoSQL_Considerations.pdf>. Acesso em 20 fevereiro 2017.

PEREIRA, Caio Ribeiro. Contruindo API REST com NodeJS. 2016. Casa do Código.

PLANSKY, Ricardo. Definição, restrições e benefícios do modelo de arquitetura REST, Imasters. outubro 2014. Disponível em: <http://imasters.com.br/desenvolvimento/definicao-restricoes-e-beneficios-modelo-de-arquitetura-rest/> acesso em 20 fevereiro 2017.

STEPPAT, Nico. Banco de dados não relacionais e o movimento NoSQL. Disponível em: <http://blog.caelum.com.br/bancos-de-dados-nao-relacionais-e-o-movimento-nosql/>. Acesso em 20 fevereiro 2017.

STROZZI, Carlos. NoSQL. A Relational database management system. 2010. Disponível em: <http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/Home%20Page>. Acesso 20 fevereiro 2017.

STUDIO, Android. Android Studio O IDE oficial do Android. Sem data. Disponível em: <https://developer.android.com/studio/index.html?hl=pt-br>. acesso em 20 fevereiro 2017