DOI: https://doi.org/10.29367/issn.1809-3957.15.2020.174.39

Volume 16 – Nº 174 – Junho / 2020
XLII International Sodebras Congress
28 a 30 de maio de 2020 – Web Conference.

# OPERAÇÕES CRUD UTILIZANDO BANCOS DE DADOS RELACIONAIS E NÃO RELACIONAIS

## *CRUD OPERATIONS USING RELATIONAL AND NON-RELATIONAL DATABASES*

NEPOMUCENO, César; LOPES, Diego; BASTOS, Moisés; SOARES, André; VALENZUELA, Walter

UNIVERSIDADE DO ESTADO DO AMAZONAS

*camn.lic@uea.edu.br; diml.eng@uea.edu.br; mpbastos@uea.edu.br; alsoares@uea.edu.br; wvalenzuela@uea.edu.br*

***Resumo** – Bancos de dados são parte crucial de uma aplicação que utiliza permanência de informações, na atual fase da tecnologia computacional existe uma ampla possibilidade de escolha entre bancos de dados disponíveis para atender as necessidades de um software. Este trabalho apresenta a implementação da persistência de dados utilizando operações CRUD em um banco SQL e um banco NoSQL. Os SGBD escolhidos foram o MongoDB e o MySQL que possuem ampla adesão mundial e para realizar as operações CRUD foi implementada uma interface com Angular que possibilita o cadastro de localizações geográficas em um mapa da cidade de Manaus-AM. Essas localizações serão calculadas por uma API utilizando dados de latitude e longitude. O trabalho inclui uma explicação das estruturas básicas e os processos iniciais da sua implementação do MongoDB e do MySQL. Com o intuito de ampliar as informações acerca do assunto abordado, também demonstra uma análise qualitativa entre o MongoDB e o MySQL, destacando características relevantes acerca desses SGBDs.*
***Palavras-chaves:** Persistência, SQL, NoSQL*

***Abstract** - Databases are a crucial part of an application that uses information permanence, in the current phase of computational technology there is a wide choice between databases available to meet the needs of a software. This work presents the implementation of data persistence using CRUD operations in an SQL and a NoSQL database. The DBMS chosen were MongoDB and MySQL, which have wide worldwide adhesion and to carry out CRUD operations, an interface with Angular was implemented, which allows the registration of geographic locations on a map of the city of Manaus-AM. These locations will be calculated by an API using latitude and longitude data. The work includes an explanation of the basic structures and the initial processes of its implementation of MongoDB and MySQL. In order to expand the information on the subject addressed, it also demonstrates a qualitative analysis between MongoDB and MySQL, highlighting relevant characteristics about these DBMSs.*
***Keywords:** Persistence, SQL, NoSQL*

## I. INTRODUÇÃO

No ramo da tecnologia os bancos de dados e seus sistemas de gerenciamento desempenham papel inquestionável, sendo a base para o funcionamento de qualquer sistema que necessite de armazenamento ou manipulação de dados. Nesse contexto, os avanços da tecnologia e de suas possibilidades de utilização exigem que gerenciamento das estruturas de dados seja feito da maneira mais eficiente possível. (RAMEZ, 2005)

Essa eficiência inicia pela escolha correta entre bancos de dados relacionais (*Structured Query Language*, SQL) e os não relacionais (*No Structured Query Language*, NoSQL) para a utilização em conjunto com uma aplicação *web*. Esses dois tipos de estruturas de dados possuem características, estruturação e manipulação distintas, mas visam o mesmo objetivo de gerenciar dados. (RAMEZ, 2005)

Além da escolha mais assertiva de um tipo de banco de dados, existe também a necessidade de uma interface de manipulação de dados onde o usuário poderá realizar operações de inserção, edição, consulta e exclusão de registros. Uma interface implementada com boas práticas, alinhada com um banco de dados bem estruturado são o ponto de partida para uma aplicação funcional. (Ramez, 2005)

Neste trabalho serão apresentadas operações básicas implementadas para uma interface CRUD (acrônimo do inglês *Create*, *Read*, *Update and Delete*) persistindo registros de localização no mapa da cidade de Manaus em um banco de dado relacional, utilizando o MySQL, e não relacional, utilizando o MongoDB. O estudo também apresenta uma análise qualitativa entre os bancos de dados utilizados.

## II. REFERÊNCIA TEÓRICA

### *2.1 Bancos de dados relacionais*

O SQL foi a primeira solução adotada em larga escala para a estruturação de registros em um banco de dados e por um longo período permaneceu como a única opção viável. Criado em 1974, pela IBM, buscava utilizar o modelo relacional criado por Edgar Frank Codd. O SQL recebeu revisões e consequentemente novas versões para garantir e melhorar o desempenho da sua implementação. O esquema de organização no modelo relacional organiza os registros de dados em tabelas, onde as linhas são registros e as colunas são uma característica daquele registro. Essas características são determinadas por quem realizar a implementação. O

quadro 1 apresenta um exemplo da organização dessas tabelas.

Quadro 1 - Exemplo de Organização de registro em tabela SQL

| Funcionários | | | | |
|---|---|---|---|---|
| ID | Nome | Sobrenome | Cargo | Depto |
| 1 | Carlos | Sanchez | Gerente | Vendas |
| 2 | João | Batista | Porteiro | Adm |
| 3 | Carla | Guimarães | Faxineira | Serviços |

*Fonte: Autor*

No exemplo apresentado no quadro 1, os registros de dados na tabela possuem as seguintes características: ID, Nome, Sobrenome, Cargo e Depto. Toda tabela SQL precisa ser identificada por um nome que será definido pelo programador e que visa indicar quais registros serão destinados àquela estrutura, na Tabela 1 o nome escolhido foi *Funcionários* indicando que esta tabela irá possuir os registros dos funcionários de uma empresa. Cada campo armazena uma informação importante do registro (linha), entretanto a primeira linha da tabela indica o tipo computacional de cada campo (DONAHOO, 2005).

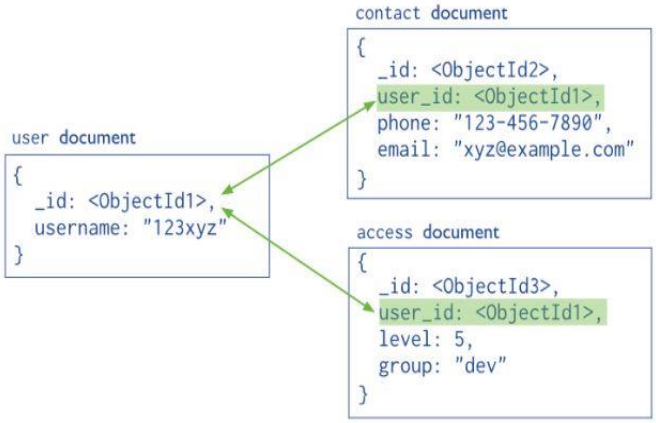
Como exemplo de sistema de gerenciamento de banco de dados (SGBD) para bancos relacionais é possível citar o MySQL, que é *open source* e possui um vasto número de desenvolvedores.

*2.2 Bancos de dados não relacionais*

Com a ampliação do uso das tecnologias nos ramos industrial e comercial, os tamanhos do fluxo e da quantidade de dados a serem manipulados por sistemas de gerenciamento de banco de dados (SGBD), os bancos que utilizavam SQL demonstraram um aumento do custo computacional em seu funcionamento, bem como aumento da complexidade de suas estruturas. Diante disso, uma nova solução foi implementada, a estrutura NoSQL, ou bancos não relacionais, com o objetivo de suprir a necessidade de gerenciamento de grandes quantidades de dados sem que a complexidade das estruturas aumente de forma proporcional. O termo NoSQL foi utilizado pela primeira vez por Carlo Strozzi em 1998 para designar um banco de dados de sua autoria, que não utilizava uma interface SQL. Apesar desse primeiro momento, os bancos não relacionais só ganharam notoriedade quando grandes empresas da Internet, como Google e Amazon, divulgaram informações sobre seus projetos de armazenamento de dados que buscavam suprir as necessidades e reduzir a complexidade que eram apresentadas nos bancos SQL (DIANA, 2010).

Os registros em uma estrutura não relacional não são organizados em tabelas e não necessitam de interligações entre os tipos existentes para possibilitar consultas múltiplas. Na figura 1 temos um exemplo da organização dos dados em documentos livres, um exemplo de dados não relacionais.

Figura 1 - Persistência de dados NoSQL




Fonte: https://www.mongodb.com/

O MongoDB é um exemplo de banco de dados NoSQL, sendo uma opção multiplataforma e de código aberto. Sua implementação é orientada a documentos livres e visa possibilitar um mundo onde a velocidade e a escalabilidade funcionam; onde não existe a necessidade de configurações ou instalações complicadas (PLUGGE; HAWKINS; MEMBREY, 2010)

*2.3 CRUD – Create, Read, Update and Delete*

A realização de operações de gerenciamento básicas em um banco de dados pode ocorrer em interfaces conhecidas como CRUD, que é uma sigla para as operações *Create, Read, Update and Delete*. Tais operações possibilitam ao usuário uma manipulação completa dos dados armazenados em um banco de dados.

## IIL. METODOLOGIA

Neste estudo foi desenvolvida uma interface CRUD visando realizar a persistência de dados obtidos via formulário *web* em banco de dados relacional e não relacional. Tal sistema permite um estudo prático e comparativo entre os tipos de bancos de dados supracitados. Como banco de dados relacional foi utilizado o MySQL e não relacional o MongoDB.

*3.1 Interface CRUD*

Para obter os dados a serem persistidos nos bancos de dados foi necessário o desenvolvimento de uma interface CRUD, isto é, um meio de interação com o usuário visando apenas cadastrar e listar os dados. Tal interface foi desenvolvida em TypeScript com o uso do framework Angular. Antes de conceituar ambos, é necessário um entendimento do que é JavaScript ou JS que representa uma linguagem de programação multiparadigma, com *script* de alto nível, interpretada e estruturada. Agora em uma rápida apresentação, podemos conceituar o TypeScript como uma ferramenta adicional ao JavaScript sendo capaz de adicionar uma *tipagem estática* ao mesmo, que originalmente possui *tipagem dinâmica*. Essa conversão ocorre apenas no ambiente de desenvolvimento, ao passo que no processo conhecido como *build* de produção, toda a linguagem implementada em TypeScript será convertida em JavaScript. Em relação ao Angular, é um *framework* destinado a criar interfaces utilizando linguagens de programação *web*, com ferramentas nativas que permitem implementar o que são chamadas de


ISSN 1809-3957

*Single-page Application*, que basicamente são páginas *web* que funcionam como uma aplicação completa, um conceito parecido com as aplicações *mobile*. A interface desenvolvida é apresentada na figura 2.

Figura 2 - Interface de cadastro dos pontos no mapa


Fonte: Autores

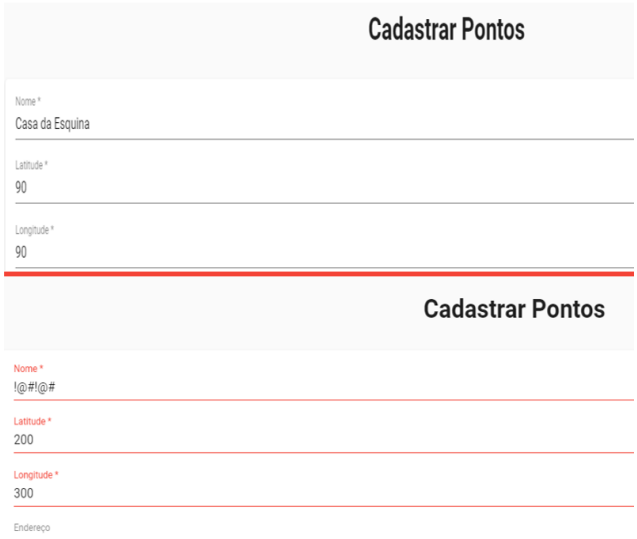
Como pode ser visto na figura 2 temos os seguintes campos:

- Campo “Nome”: destinado a uma identificação textual personalizada ao ponto do mapa a ser registrado
- Campo “Latitude”: recebe um valor real presente entre o intervalo -90 e 90, que são os valores mínimo e máximo na latitude do globo.
- Campo “Longitude”: recebe um valor real presente entre o intervalo -180 e 180, que são os valores mínimo e máximo de longitude do globo.
- Campo “Endereço”: destinado a informações geográficas acerca do ponto a ser registrado, essas informações serão geradas automaticamente pela Application Programming Interface (API) do GoogleMaps utilizando os valores dos campos Latitude e Longitude.
- Campo “Informações Adicionais”: recebe as características personalizadas sobre o ponto a ser registrado, sendo o único campo no formulário que não possui preenchimento obrigatório.

No desenvolvimento de uma aplicação que possui persistência em banco de dados é importante utilizar medidas que garantam a integridade das informações fornecidas pela interface CRUD e uma dessas medidas é implementar validações para os campos a serem preenchidos pelo usuário. O Angular possui ferramentas nativas que atuam na validação de dados ainda no front-end, antes mesmo de enviar os dados ao banco.

Na figura 3 é apresentado um exemplo da validação com Angular implementada. Ao ocorrer o preenchimento dos campos Nome, Latitude e Longitude com caracteres ou valores que não são aceitos pelo banco de dados, os campos da interface alteram sua coloração para vermelho, visualmente indicando ao usuário que a informação fornecida no campo é inválida e dessa forma impossibilitando a realização do envio do formulário ao banco de dados. Também é possível implementar no Angular métodos de validações personalizadas caso as ferramentas nativas não atendam aos requisitos da aplicação.

Figura 3 - Validação de campos via Angular




Fonte: Autores

### *3.2 Persistência com o banco de dados não relacional*

O exemplar de banco de dados NoSQL escolhido para este trabalho foi o MongoDB, uma aplicação lançada em 2009 e que dispensa várias estruturas obrigatórias no esquema SQL e que é utilizado por grandes empresas que manipulam grandes quantidades de informação tais como Facebook, Google, Ebay. O MongoDB conta ainda com *features* direcionadas a facilitar sua utilização como um servidor mantido pela empresa e administrado pela comunidade de usuários, que já está pronto para a integração com aplicações e pode ser baixado do site oficial, além da grande adesão de usuários em ambientes como o GitHub e universidades.

O MongoDB é um banco de dados com armazenamento de registros orientado a documentos com estrutura baseada em JSON (*JavaScript Object Notation*) (TRUICA, 2013) e com modelagem direcionada aos objetos no código da aplicação a ser integrado, sendo esses documentos armazenados em Coleções, que são equivalentes as tabelas SQL. Possui uma versão *online* e uma versão de funcionamento local, a escolha entre as duas é pessoal e deve levar em conta qual opção atenderá as necessidades da sua aplicação. A organização de dados via documento em JavaScript pode ser representada da seguinte maneira:

```
{
    nome: “Carlos” ,
    sobrenome: “Sanchez” ,
    cargo : “Gerente” ,
    depto : “Vendas”
}
```

O MongoDB é *case sensitive*, o que significa ser capaz de diferenciar letras maiúsculas e minúsculas, detalhe importantíssimo ao realizar qualquer operação com o mesmo. O único pré-requisito para iniciar a utilização do *software* após a sua instalação é verificar se o MongoDB já está em funcionamento. Isso pode ser feito acessando a pasta *bin* dentro da pasta de instalação do MongoDB com o *terminal* (*prompt* de comando do sistema operacional utilizado). É importante frisar que adicionar a pasta de instalação do MongoDB à variável de ambiente *PATH* permite executar os comandos a partir de qualquer diretório.


ISSN 1809-3957

Através do comando *mongo* o MongoDB estabelecerá uma conexão com o seu principal serviço, o *mongo shell*, que poderá ser uma conexão local, via Host Remoto (passível de ser com ou sem autenticação), via TLS/SSL ou com uma Réplica MongoDB. Também é possível, designar a porta pela qual a conexão irá ocorrer, conforme a figura 4.

Figura 4 - Inicializando o serviço do MongoDB

```
mongo --port 28015
```

Fonte: Autores

A próxima ação é criar um banco de dados, que pode ser realizada através do comando *use NomeDoBanco*, que também funciona como comando para alternar entre bancos de dados já existentes, seguindo a figura 5.

Figura 5 - Criando um banco de dados no MongoDB

```
>>> use localizacoes
switched to db localizacoes
```

Fonte: Autores

Para verificar qual banco de dados está em uso basta executar o comando *db*, como indicado na figura 6, e na linha seguinte será impresso o nome do banco de dados:

Figura 6 - Verificando qual banco de dados está em uso

```
>>>>db
localizacoes
```

Fonte: Autores

A operação de inserção de dados em um banco gerenciado pelo MongoDB pode ser realizada utilizando o simples comando *db.NomeDaColeção.insert({})*, como demonstrado na figura 7.

Figura 7 - Inserção de dados em uma coleção

```
db.localizacao.insert ({
   id: "1",
   nome: "Padaria",
   latitude: "-3.083278",
   longitude: "-60.042094",
   endereco: "Av. Dom Pedro I-José Bonifácio, Manaus - AM, 69043-160",
   info: ""
   })
```

Fonte: Autores

Caso a coleção "Funcionários" ainda não exista no banco de dados, o comando utilizado para inserir o JSON também criará a coleção automaticamente. Apesar de funcionarem de maneira semelhante às tabelas SQL, as coleções no MongoDB não se limitam em armazenar apenas um tipo de dado, mas possibilitando que registros com estrutura similar possam ser armazenados em uma mesma coleção.

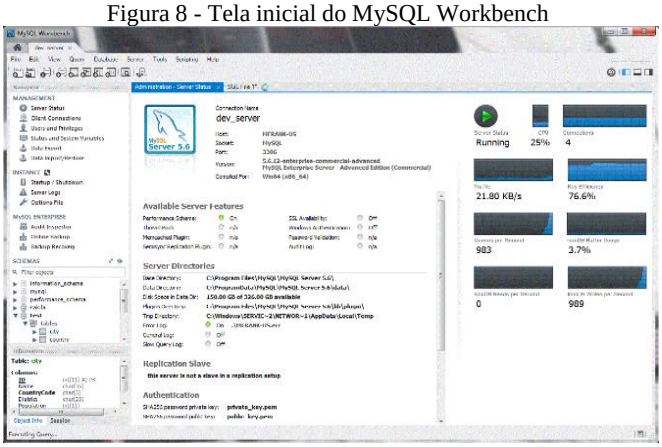
Uma característica importante sobre o MongoDB é que cada registro nas coleções possui um id único que é gerado automaticamente pelo *software* que pode ser utilizado para referenciar o registro ao qual está atrelado, mas diferente do MySQL que possui funções para atribuir um identificador único autoincrementavel conforme solicitação do usuário, o MongoDB não possui uma função nativa para essa ação, ficando a cargo do usuário implementá-la, se necessário, de maneira que requisitos de sua aplicação sejam atendidos.

### *3.3 Persistência com o banco de dados relacional*

O MySQL é o SGDB baseado no sistema SQL com código livre e com maior adesão mundial, sendo por muito tempo considerado melhor ferramenta para aplicações que utilizam bancos de dados. Foi o escolhido porque possibilita efetuar simulações em uma base de dados paralela, definida de acordo com as necessidades, detectar e resolver problemas na configuração incorreta das variáveis em sua base real, fornecendo a configuração do *storage engine InnoDB* e possui ferramentas de otimização como o *Query Cache* que armazena os comandos SQL que foram usados recentemente. (SILVA; ALVES, 2012). O MySQL é um dos principais banco de dados relacionais. Sua arquitetura é composta por três camadas: conexão com o cliente, análise da instrução SQL e execução da instrução (SCHWARTZ, 2012).

A estrutura básica do MySQL é composta por um servidor e uma aplicação cliente onde a comunicação ocorre por meio do SQL. O *software* possui uma interface gráfica que atua como cliente conhecida por MySQL Workbench, por meio da qual são implementadas as entidades, seus atributos e os relacionamentos. A interface é apresentada na figura 8.

Figura 8 - Tela inicial do MySQL Workbench




Fonte: https://mysql.com

O MySQL necessita que toda a sua estrutura esteja bem definida e criada previamente para que suas operações possam ser realizadas. Na figura 9 mostra-se a criação de um banco MySQL utilizando o comando CREATE.

Figura 9 - Criando um banco de dados no MySQL

```
mysql> create database mapa;
```

Fonte: Autores

Após criado o banco é necessário acessá-lo, utilizando o comando USE, da maneira feita na figura 10.

Figura 10 - Acessando um banco de dados MySQL

```
mysql> use mapa;
```

Fonte: Autores



ISSN 1809-3957

A partir daí, já é possível criar uma tabela SQL com um nome, todos os seus dados e respectivos tipos (para o cenário em questão foram utilizados os tipos: INT, VARCHAR e REAL), e que é demonstrado na figura 11.

Figura 11 - Criando uma tabela no MySQL

```
create table events (
   id INT AUTO_INCREMENT,
   nome VARCHAR(255) NOT NULL,
   latitude REAL,
   longitude REAL,
   PRIMARY KEY (id),
);
```

Fonte: Autores

É obrigatório que toda tabela SQL possua uma chave primária única. O comando AUTO_INCREMENT indica que o valor do campo id será gerado automaticamente e será incremental, o comando NOT NULL indica que o valor do campo nome não poderá ser vazio.

Para executar qualquer operação em um banco MySQL é necessário estar conectado a ele por meio da função de CONNECT, conforme a figura 12, onde são passados: host, nome de usuário, senha e banco de dados ao qual se deseja ter acesso. A seguir, é possível verificar um trecho de código onde a operação de conexão é feita utilizando a linguagem JavaScript e o servidor Node Express em sua versão 4.17.1.

Figura 12 - Utilizando o comando CONNECT

```
const connection = mysql.createConnection({
 host     : 'localhost',
 user     : 'mapa',
 password : 'password',
 database : 'mapa'
});
connection.connect();
```

Fonte: Autores

A linguagem SQL possui uma operação de inserção de dados em uma tabela, o INSERT. Na figura 13, é apresentado um trecho de código que demonstra a operação de inserção, onde são passados: a tabela em que se deseja incluir uma nova linha, os campos que estão sendo passados e, por fim, seus respectivos valores.

Figura 13 - Utilizando o comando INSERT

```
router.post('/mapa', (req, res, next) => {
   db.query(
'INSERT INTO mapa
(nome, latitude, longitude)
VALUES (?,?,?)',
[req.body.nome,
req.body.latitude,
req.body.longitude],
    });
 });
```

Fonte: Autores

É possível também utilizar a operação de listagem de dados utilizando o SELECT. A figura 14, está um trecho de código mostrando como utilizar tal operação, onde são passados os campos a serem buscados e a tabela que se deseja acessar.

Figura 14 - Utilizando o comando SELECT

```
router.get('/mapa, function (req, res, next) {
   db.query(
'SELECT id,
nome,
latitude,
longitude
FROM mapa,
   );
 });
```

Fonte: Autores

Na operação de listagem é possível utilizar alguns filtros por meio da cláusula WHERE (por exemplo, SELECT FROM * mapa WHERE latitude > 0 AND longitude > 0), além disso, também é possível ordenar dados utilizando o comando ORDER BY (por exemplo, SELECT FROM * mapa ORDER BY nome), os dois comandos também podem ser utilizados juntos (por exemplo, SELECT FROM * mapa WHERE latitude > 0 AND longitude > 0 ORDER BY nome).

A linguagem SQL possui outros comandos conhecidos como CRUD, são eles: DELETE e UPDATE. O comando DELETE exclui uma ou mais linhas de uma determinada tabela dependendo das condições passadas. Já o comando UPDATE, assim como o DELETE, necessita de uma ou mais condições para atualizar uma ou mais linhas de uma tabela de banco de dados.

## IV. RESULTADOS

O principal objetivo desse trabalho é apresentar a implementação prática de operações CRUD utilizando banco de dados NoSQL e SQL, conforme demonstrado ao longo deste artigo. Contudo visando ampliar os resultados do estudo foi desenvolvida uma análise qualitativa entre MySQL e MongoDB. Tal análise é apresentada no quadro 2.

Quadro 2 - Análise qualitativa entre MySQL e MongoDB

| Banco de Dados | MySQL | MongoDB |
|---|---|---|
| Estrutura | Relacional | Não-relacional |
| Linguagem de programação | C/C++ | C, C++ e JavaScript |
| Escalabilidade | Afeta a complexidade e o desempenho | Escalabilidade horizontal, pronta para grandes fluxos de dados |
| Flexibilidade | Mudanças na estrutura das tabelas podem afetar toda a base de dados | Estrutura dinâmica e adaptável à alterações após implementação |
| *Features* | - *Triggers*; | -Alta Performance; |



ISSN 1809-3957

| | | |
|---|---|---|
| Importantes | - *Query Cache*;<br>- *SubSelects*; | - Operações CRUD;<br>- Data Aggregation; |
| Compatibilidade | Disponível para os principais Sistemas Operacionais e variáveis do Sistema Ubuntu | Disponível para os principais Sistemas Operacionais |

*Fonte: Autores*

A implementação da interface para dois tipos de bancos de dados diferentes também possibilitou uma comparação entre os processos que envolvem a preparação do ambiente computacional onde ocorrerá a manutenção do banco de dados.

## V. CONCLUSÃO

O planejamento da arquitetura de uma aplicação que utilize persistência de informações em um banco de dados deve envolver uma análise atenciosa das opções de SGDB disponíveis com o intuito de implementar uma estrutura que atenda os requisitos da aplicação.

Bancos de dados relacionais como o MySQL possuem estruturas com maior complexidade e custo computacional, mas oferecem ferramentas consolidadas que são altamente integradas com suas estruturas e que proporcionam ao usuário um ambiente familiar à outras etapas de um processo de desenvolvimento de *software*. Em contraste com o ambiente relacional, e ainda de certa forma representante de um novo processo que envolve a implementação de um banco de dados não-relacional, como o MongoDB, provém uma estrutura concisa para atender as necessidades da persistência de dados. A escalabilidade de bancos de dados desse tipo são um atrativo para empresas que lidam com grandes fluxos de informação, que na atual fase da tecnologia tornou-se o principal item com potencial de mercado.

Com este trabalho foi possível perceber as diferentes implementações com bancos de dados relacionais e não relacionais ao persistirem os mesmos dados, no caso provenientes de uma mesma interface CRUD.

Por fim, após uma análise qualitativa entre os bancos de dados utilizados neste trabalho, MongoDB e MySQL, é perceptível que a utilização de cada ferramenta pode ser definida de acordo com a aplicação e seus respectivos requisitos.

## VI. REFERÊNCIAS BIBLIOGRÁFICAS


C.-O. TRUICA, A. BOICEA, AND I. TRIFAN, “CRUD Operations in MongoDB,” no. Icacsei, pp. 347–350, 2013.

E. PLUGGE, P. MEMBREY, AND T. HAWKINS, The Definitive Guide to MongoDB. Apress, 2010.

M. DE DIANA AND M. A. GEROSA, “Nosql na Web 2.0: Um estudo comparativo de bancos Não-Relacionais para Armazenamento de Dados na Web 2.0,” IX Work. Teses e Diss. em Banco Dados - WTDBD, p. 8, 2010.

M. DONAHOO AND G. SPEEGLE, *Critical Acclaim for SQL: Practical Guide for Developers*.

MongoDB official webpage. [Online]. Available:

https://www.mongodb.com/. Acessado em 15/03/2020

MySQL official webpage. [Online]. Available: https://www.mysql.com/. Acessado em 15/03/2020

N. S. B. ELMASRI, Ramez, *Sistema de Banco de Dados*, vol. 4. 2005.

SCHWARTZ, B.; ZAITSEV, P.; TKACHENKO, V. High performance MySQL: optimization, backups, and replication. Sebastopol, USA: O’Reilly Media, Inc, 2012.

SILVA, G. F. A. AND ALVES, A. G. (2012) “Simulador de análise e desempenho para banco de dados MySQL”, Anais... Congresso Nacional de Extensão Universitária, 6., Encontro de Atividades Científicas da UNOPAR, 15., 2012. Londrina: UNOPAR.


## VII. AGRADECIMENTOS


Este trabalho conta com o apoio da Samsung Eletrônica da Amazônia Ltda. através da Lei de Informática, AGIN - Agência de Inovação da UEA e FUEA - Fundação Universitas de Estudos Amazônicos.


## VIII. COPYRIGHT