mongoDB®

André Tricano

MongoDB

NoSQL na prática

Dedicatória

Dedico esta obra a minha esposa que sempre esteve ao meu lado em minhas decisões e a minha filha que chega para selar o meu projeto de vida chamado família.

Sumário

Sumário

Prefácio

Instalar

Serviço

Segurança

Java

Importar

Conectar

Inserir

Listar

Localizar

Atualizar

Remover

Copiar

Restaurar

Desinstalar

Considerações

Agradecimentos

Sobre o autor

Prefácio

Bom, como costumeiramente acontece na minha vida, o meu primeiro contato com o mundo NoSQL surgiu devido a uma necessidade quando eu estava projetando um aplicativo web para comércio eletrônico no smartphone.

No universo mobile, tanto o recurso disponível quanto a paciência do usuário são extremamente curtas. Talvez devido a pouca idade das pessoas com acesso aos recursos tecnológicos mais atuais. Isso se agrava quando se trata de um aplicativo web em comparação a um aplicativo nativo, devido limitações técnicas do universo web mobile.

A primeira coisa que eu desconsiderei na fase de arquitetura do projeto, foi a utilização de um banco de dados relacional. Não que o modelo relacional não funcione. Muito pelo contrário, o modelo relacional funciona sim, e é utilizado até hoje por uma variedade de empresas. Inclusive, por grandes empresas de tecnologia da Califórnia (estado americano considerado a Meca da tecnologia).

Até então eu havia trabalhado apenas com banco de dados relacional como: Oracle, SQL Server, Sybase, SQLite, Firebird e MySQL.

No caso específico do projeto de comércio eletrônico, a decisão inicial era utilizar o banco de dados MySQL. Durante a fase de definição de arquitetura, com base em pesquisas, surgiu a opção de utilizar o banco de

dados NoSQL chamado MongoDB.

Nessa decisão, o que pesou foi a velocidade por acesso à informação e a flexibidade de esquema.

Lembrando que, na verdade, não existiu uma régua para comparar o MongoDB com o MySQL. O que existiu foi uma maneira de analisar qual será o melhor momento para aplicar um ou o outro. Até porque são tecnologias completamente diferentes e complementares.

No meu caso, optei por utilizar o MySQL no produto backOffice (cloud software de gestao) e o MongoDB no produto shopMobile (cloud e-commerce para smartphone). Ambos produtos desenvolvidos pela empresa Vynx.

No caso do produto backOffice, como a provisão de acesso de usuario é menor e as chances de mudança na estrutura de dados também são menores, foi possível optar pelo MySQL.

Agora, no caso do produto shopMobile, o cenário é completamente o inverso. A provisão de acesso de usuários é incerta e será necessário adicionar recursos conforme audiência. Geralmente, esses recursos serão adicionados com o aplicativo em curso. Seria mais ou menos como trocar a turbina do avião com a aeronave em pleno voo.

Bom, após explicar a razão pela qual optamos por MongoDB, vamos colocar a mão na massa. Neste livro você vai encontrar exemplos do MongoDB versão 3, aplicados através do sistema operacional Ubuntu versão 14 na plataforma Java versão 8.

Instalar

Neste capítulo, vamos ver como instalar e colocar o MongoDB para funcionar com as configurações do fabricante.

1° Passo: abrir o terminal de comando do Ubuntu.

CTRL + ALT + T

2° Passo: linha de comando para importar a chave pública utilizada pelo sistema de gerenciamento de pacote.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
7F0CEB10
```

3° Passo: linha de comando para criar um arquivo de lista com os comandos apropriados para a versão do Ubuntu.

```
echo "deb http://repo.MongoDB.org/apt/ubuntu trusty/MongoDB-org/3.0
multiverse" | sudo tee /etc/apt/sources.list.d/MongoDB-org-3.0.list
```

4° Passo: linha de comando para atualizar o banco de dados de pacotes do Ubuntu.

```
sudo apt-get update
```

5° Passo: linha de comando para instalar a última versão estável do MongoDB.

```
sudo apt-get install -y MongoDB-org
```

O MongoDB-org é um meta pacote que instala os seguintes pacotes de componentes do MongoDB.

- MongoDB-org-server

  Este pacote contém o daemon mongod e configuração associada e scripts de inicialização.

- MongoDB-org-mongos

  Este pacote contém o daemon de mongos, que são serviço de roteamento para configurações de fragmento do MongoDB.

- MongoDB-org-shell

  Este pacote contém o shell mongo.

- MongoDB-org-tools

  Este pacote contém as seguintes ferramentas de MongoDB: mongoimport bsondump, mongodump, mongoexport, mongofiles, mongooplog, mongoperf, mongorestore, mongostat e mongotop.

É possível também especificar pacote e versão para instalar conforme linha de comando a seguir:

```
sudo apt-get install -y MongoDB-org=3.0.7 MongoDB-org-server=3.0.7 MongoDB-org-shell=3.0.7 MongoDB-org-mongos=3.0.7 MongoDB-org-tools=3.0.7
```

Serviço

Neste capítulo, vamos ver como iniciar o serviço do MongoDB.

Segue abaixo o comando a ser executado e o detalhe sobre cada parâmetro utilizado.

sudo mongod --port 27017 --dbpath /var/lib/MongoDB --fork --logpath /var/log/MongoDB/mongod.log --smallfiles

--port

Parâmetro para especificar a porta TCP na qual a instância do MongoDB fará a escuta de conexões clientes.

--dbpath

Define o local onde o MongoDB irá armazenar os dados.

--fork

Habilitar a execução da instância do MongoDB no modo background.

--logpath

Informar ao MongoDB onde enviar todas as informações de diagnóstico de log.

--smallfiles

Define para o MongoDB utilizar um tamanho padrão de arquivo de dados menor. O parâmetro reduz o tamanho inicial do arquivo de dados e limita ao tamanho máximo de 512 megabytes.

Observação: o parâmetro smallfiles foi utilizado apenas para executar o MongoDB em uma distribuição Live USB com espaço de armazenamento limitado. Nos demais casos, talvez não seja necessário a utilizaçao do parâmetro.

Para conferir se o MongoDB foi iniciado com sucesso, edite o arquivo /var/log/MongoDB/mongod.log e verifique se existe a seguinte linha:

[initandlisten] waiting for connections on port 27017

Segurança

Neste capítulo vamos ver como habilitar o módulo de segurança do MongoDB.

No capítulo de instalação, vimos como colocar o MongoDB para funcionar. Porém, na configuração do fabricante não vem habilitado o módulo de segurança. Desta forma, qualquer pessoa pode ter acesso aos dados do MongoDB instalado.

Para evitar problema com acesso indevido de informação, vamos habilitar o sistema de segurança do MongoDB. Após isso, apenas usuários autenticados terão acesso ao banco de dados.

1° Passo: conectar ao serviço do banco de dados MongoDB.

```
mongo  --port 27017 --dbpath /var/lib/MongoDB
```

2° Passo: trocar o acesso para o banco de dados “admin”.

```
>use admin
```

3° Passo: criar usuário “root” como administrador para o banco de dados “admin”.

```
>db.createUser({user: "root",pwd: "abc@123", roles:[{role: "userAdminAnyDatabase",db: "admin"}]})
```

4° Passo: definir permissão “__system” para usuário “root” no banco de dados “admin”.

```
>db.grantRolesToUser("root",[{role:"__system",db: "admin" }])
```

5° Passo: desconectar do banco de dados MongoDB.

```
>exit
```

6° Passo: parar o serviço do banco de dados MongoDB.

```
sudo mongod --shutdown --port 27017 --dbpath /var/lib/MongoDB
```

7° Passo: iniciar o serviço do banco de dados MongoDB habilitando o sistema de segurança com o parâmetro --auth.

```
sudo mongod --auth --port 27017 --dbpath /var/lib/MongoDB --fork --logpath /var/log/MongoDB/mongod.log --smallfiles
```

8° Passo: conectar a instância do MongoDB com o sistema de segurança habilitado.

```
mongo test --port 27017 -u "root" -p "abc@123" --authenticationDatabase "admin"
```

Bom, se todas as etapas ocorreram sem problemas, temos então o banco de dados MongoDB com o sistema de segurança habilitado e pronto para ser utilizado.

Java

Neste capítulo, vamos ver como conectar ao banco de dados MongoDB através da plataforma Java.

Faremos a conexão na plataforma Java através do MongoDB Java Driver, que está documentado no endereço http://MongoDB.github.io/mongo-java-driver/.

MongoDB Java Driver é uma api que fornece interações síncronas e assíncronas com o banco de dados MongoDB através da plataforma Java.

Para os exemplos Java dos próximos capítulos, criamos um projeto Maven com o arquétipo maven-archetype-quickstart.

Após criação do projeto Maven, adicione a dependência do MongoDB Java Driver no arquivo pom.xml conforme segue:

<dependencies><dependency><groupId>org.MongoDB</groupId>
<artifactId>MongoDB-driver</artifactId><version>3.2.0</version>
</dependency></dependencies>

Importar

Antes de começar interagir com o banco de dados MongoDB, vamos carregar uma coleção de documentos no database test.

Para iniciar, baixe o arquivo dataset.json no endereço abaixo.

https://raw.githubusercontent.com/MongoDB/docs-assets/primer-dataset/dataset.json

Neste arquivo, encontramos uma coleção de restaurantes que ficam localizados na cidade de Nova York.

Após baixar o arquivo dataset.json, será preciso importar as informações para o banco de dados test conforme comando abaixo:

mongoimport --authenticationDatabase "admin" --username root --db test --collection restaurants --drop --file dataset.json

--authenticationDatabase

Define qual banco de dados será utilizado para autenticar usuário.

--username

Define o usuário conectado.

--db

Define o banco de dados conectado.

--collection

Define o nome da coleção importada.

--drop

Define que se existir uma coleção de restaurantes no banco de dados, a coleção será descartada para que uma nova seja importada.

Conectar

Neste capítulo, vamos ver como conectar ao banco de dados através do MongoDB Shell e MongoDB Java Driver.

No caso do MongoDB Java Driver, MongoDB versão 3 incluiu o suporte para o mecanismo de autenticação chamado SCRAM-SHA-1, no qual muda a maneira como o MongoDB usa e armazena credenciais do usuário.

SCRAM-SHA-1 é um padrão criado pelo IETF através da RFC 5802, que define métodos de boas práticas de implementação de mecanismos para autenticação de usuários com senha.

Nas versões anteriores, o MongoDB utilizava o mecanismo de autenticação chamado MongoDB-CR. O fabricante recomenda fortemente a utilização do mecanismo SCRAM-SHA-1 em vez MongoDB-CR.

Segundo o fabricante, o mecanismo SCRAM-SHA-1 representa melhorias significativas em segurança com relação ao mecanismo MongoDB-CR.

O fabricante destaca ainda que a função de hash é criptograficamente mais forte (SHA-1 em vez de MD5), com autenticação do servidor para o cliente tanto quanto do cliente para o servidor.

MongoDB Shell:

```
mongo test --port 27017 -u "root" -p "abc@123" --authenticationDatabase "admin"
```

MongoDB Java Driver:

```
import java.util.Arrays;
import com.mongodb.MongoClient;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoCredential;

public class MongoDBConectar {
   public static void main(String[] args) {
            MongoClient mongoClient = null;
             try {
                        ServerAddress serverAddress = new
ServerAddress("localhost", 27017);
                        MongoCredential mongoCredential =
MongoCredential.createScramSha1Credential("root",
"admin","abc@123".toCharArray());
                        mongoClient = new MongoClient(serverAddress,
                                    Arrays.asList(mongoCredential));
                        MongoDatabase database =
mongoClient.getDatabase("test");
                        for (String name : database.listCollectionNames()) {
                                    System.out.println(name);
                        }
             } finally {
                        mongoClient.close();
```

```
            }
        }
    }
```

Inserir

Neste capítulo, vamos ver como inserir um documento no banco de dados através do MongoDB Shell e MongoDB Java Driver.

No caso abaixo, vamos adicionar o registro do restaurante Forneria Della Garfagnana, localizado na cidade de Vitoria do Espirito Santo.

MongoDB Shell:

```
>db.restaurants.save({
  "address": {
            "building": "260",
            "coord": [ -20.250218, -40.267869 ],
            "street": "Avenida Judith Leao Castello Ribeiro",
            "zipcode": "29090720"
  },
  "borough": "Jardim Camburi",
  "cuisine": "Brasileira",
  "grades": [
            { "date": ISODate("2015-03-03T00:00:00Z"), "grade": "A", "score": 2 },
            { "date": ISODate("2015-04-03T00:00:00Z"), "grade": "A", "score": 6 }
  ],
  "name": "Forneria Della Garfagnana",
  "restaurant_id": "830075445"
})
```

MongoDB Java Driver:

```
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Arrays;
```

```
import java.util.Locale;
import static java.util.Arrays.asList;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

public class MongoDBInserir {

   static MongoClient mongoClient = null;

   public static void main(String[] args) throws ParseException {
            try{
                     MongoDatabase database = getDatabase("test");
                     DateFormat format = new SimpleDateFormat("yyyy-
MM-dd'T'HH:mm:ss'Z'",Locale.ENGLISH);
                     database.getCollection("restaurants").insertOne(new
Document("address",new Document()
                                                .append("street","Avenida Judith
Leao Castello Ribeiro")

.append("zipcode","29090720").append("building", "260")

.append("coord",asList(-20.250218,-40.267869)))
                                             .append("borough","Jardim Camburi")
                                             .append("cuisine","Brasileira")
                                             .append("grades",asList(
                                                          new
Document().append("date",format.parse("2014-10-01T00:00:00Z"))
                                                      .append("grade",
"A").append("score", 11),
                                                new
Document().append("date",format.parse("2014-01-16T00:00:00Z"))
                                                      .append("grade",
"B").append("score", 17)))
```

```
                                        .append("name","Forneria Della
Garfagnana")
                                        .append("restaurant_id","830075445"));
            }finally{
                    mongoClient.close();
            }
    }

    public static MongoDatabase getDatabase(String database){
            ServerAddress serverAddress = new
ServerAddress("localhost",27017);
            MongoCredential mongoCredential = MongoCredential

.createScramSha1Credential("root","admin","abc@123".toCharArray());
            mongoClient = new
MongoClient(serverAddress,Arrays.asList(mongoCredential));
            return mongoClient.getDatabase(database);
    }

}
```

Listar

Neste capítulo, vamos ver como listar documentos no banco de dados através do MongoDB Shell e MongoDB Java Driver.

No caso vamos listar todos os documentos da coleção restaurants no banco de dados test.

MongoDB shell:

```
>db.restaurants.find()
```

MongoDB Java Driver:

```
import java.util.Arrays;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoDatabase;
import com.mongodb.Block;

public class MongoDBListar {
```

```
    static MongoClient mongoClient = null;

    public static void main(String[] args) {
            try{
                    MongoDatabase database = getDatabase("test");
                    FindIterable<Document> restaurants =
database.getCollection("restaurants").find();
                    restaurants.forEach(new Block<Document>() {
                            public void apply(final Document restaurant) {
                                System.out.println(restaurant);
                            }
                    });
            }finally{
                    mongoClient.close();
            }
    }

    public static MongoDatabase getDatabase(String database){
            ServerAddress serverAddress = new
ServerAddress("localhost",27017);
            MongoCredential mongoCredential =
MongoCredential.createScramSha1Credential("root","admin","abc@123".toC
            mongoClient = new MongoClient(serverAddress,Arrays
                    .asList(mongoCredential));
            return mongoClient.getDatabase(database);
    }

}
```

Localizar

Neste capítulo, vamos ver como localizar documentos no banco de dados através do MongoDB Shell e MongoDB Java Driver.

No caso, o filtro utilizado localiza todos os restaurantes no distrito de Manhattan da cidade de Nova York.

MongoDB shell:

```
>db.restaurants.find({"borough":"Manhattan"})
```

MongoDB Java Driver:

```
import java.util.Arrays;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoDatabase;
import com.mongodb.Block;
import static com.mongodb.client.model.Filters.eq;

public class MongoDBLocalizar {

   static MongoClient mongoClient = null;

   public static void main(String[] args) {
            try{
                     MongoDatabase database = getDatabase("test");
```

```
                    FindIterable<Document> restaurants =
database.getCollection("restaurants").find(eq("borough", "Manhattan"));
                    restaurants.forEach(new Block<Document>() {
                            public void apply(final Document restaurant) {
                                System.out.println(restaurant);
                            }
                    });
            }finally{
                    mongoClient.close();
            }
    }

    public static MongoDatabase getDatabase(String database){
            ServerAddress serverAddress = new
ServerAddress("localhost",27017);
            MongoCredential mongoCredential = MongoCredential

.createScramSha1Credential("root","admin","abc@123".toCharArray());
            mongoClient = new
MongoClient(serverAddress,Arrays.asList(mongoCredential));
            return mongoClient.getDatabase(database);
    }

}
```

Atualizar

Neste capítulo, vamos ver como atualizar um documento no banco de dados através do MongoDB Shell e MongoDB Java Driver.

No caso, vamos modificar o grau de pontuação do restaurante Forneria Della Garfagnana. Para localizar o restaurante, vamos utilizar o campo restaurant_id igual ao número identificador 830075445.

Grau de pontuação antes da atualização:

```
{ "date":ISODate("2015-03-03T00:00:00Z"), "grade" : "A", "score" : 2 }
{ "date":ISODate("2015-04-03T00:00:00Z"), "grade" : "A", "score" : 6 }
```

Grau de pontuação após a atualização:

```
{ "date":ISODate("2015-01-01T00:00:00Z"), "grade":"C", "score" : 20 }
{ "date":ISODate("2015-06-16T00:00:00Z"), "grade":"B", "score" : 17 }
{ "date":ISODate("2015-12-31T00:00:00Z"), "grade":"A", "score" : 11 }
```

MongoDB Shell:

```
>db.restaurants.update(   { "restaurant_id" : "830075445" },   { $set: { "grades" : [ { "date" : ISODate("2015-01-01T00:00:00Z"), "grade" : "C", "score" : 20 }, { "date" : ISODate("2015-06-16T00:00:00Z"), "grade" : "B", "score" : 17 }, { "date" : ISODate("2015-12-31T00:00:00Z"), "grade" : "A", "score" : 11 } ] } } )
```

MongoDB Java Driver:

```
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Locale;
import static java.util.Arrays.asList;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

public class MongoDBAtualizar {

   static MongoClient mongoClient = null;

   public static void main(String[] args) throws ParseException {
            try{
                     MongoDatabase database = getDatabase("test");
                     DateFormat format = new SimpleDateFormat("yyyy-
MM-dd'T'HH:mm:ss'Z'",Locale.ENGLISH);
                     database.getCollection("restaurants").updateOne(new
Document("restaurant_id","830075445"),
                                       new Document("$set", new
Document("grades",asList(
```

```
                                        new
Document().append("date",format.parse("2014-10-01T00:00:00Z"))
                                .append("grade", "A").append("score", 11),
                            new Document().append("date",format.parse("2014-01-
16T00:00:00Z"))
                                .append("grade", "B").append("score", 17),
                            new Document().append("date",format.parse("2013-12-
31T00:00:00Z"))
                                .append("grade", "C").append("score", 20)) ));
            }finally{
                    mongoClient.close();
            }
    }

    public static MongoDatabase getDatabase(String database){
            ServerAddress serverAddress = new
ServerAddress("localhost",27017);
            MongoCredential mongoCredential =
MongoCredential.createScramSha1Credential("root","admin","abc@123".toC
            mongoClient = new
MongoClient(serverAddress,Arrays.asList(mongoCredential));
            return mongoClient.getDatabase(database);
    }

}
```

Remover

Neste capítulo, vamos ver como remover um documento no banco de dados através do MongoDB Shell e MongoDB Java Driver.

MongoDB Shell:

```
> db.restaurants.remove({"restaurant_id":"830075445"})
```

MongoDB Java Driver:

```
import java.util.Arrays;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoDatabase;
import static com.mongodb.client.model.Filters.eq;

public class MongoDBRemover {

   static MongoClient mongoClient = null;

   public static void main(String[] args) {
            try{

                    MongoDatabase database = getDatabase("test");

database.getCollection("restaurants").deleteOne(eq("restaurant_id","8300754
            }finally{
                    mongoClient.close();
            }
   }
```

```
    public static MongoDatabase getDatabase(String database){
            ServerAddress serverAddress = new
ServerAddress("localhost",27017);
            MongoCredential mongoCredential = MongoCredential

.createScramSha1Credential("root","admin","abc@123".toCharArray());
            mongoClient = new
MongoClient(serverAddress,Arrays.asList(mongoCredential));
            return mongoClient.getDatabase(database);
    }


}
```

Copiar

Neste capítulo, vamos ver como copiar um banco de dados MongoDB.

mongodump --host localhost --port 27017 --authenticationDatabase "admin" --username "root" --password "abc@123" --db "test" --out /tmp/mongodump

--host

Parâmetro para especificar o nome do servidor MongoDB ao qual se conectar.

--port

Parâmetro para especificar a porta TCP na qual a instância do MongoDB fará a escuta de conexões clientes.

--authenticationDatabase

Parâmetro para especificar qual banco de dados será utilizado para autenticar o usuário.

--username

Parâmetro para especificar o usuário conectado.

--password

Parâmetro para especificar a senha do usuário conectado.

--db

Parâmetro para especificar o banco de dados conectado.

--out

Parâmetro para especificar o diretório onde o mongodump vai criar os arquivos BSON (formato de serialização para armazenar documento e fazer chamada de procedimento remoto no MongoDB).

Restaurar

Neste capítulo, vamos ver como restaurar um banco de dados MongoDB.

mongorestore --host localhost --port 27017 --authenticationDatabase "admin" --username "root" --password "abc@123" --db "test" /tmp/mongodump/test/

--host

Parâmetro para especificar o nome do servidor MongoDB ao qual se conectar.

--port

Parâmetro para especificar a porta TCP na qual a instância do MongoDB fará a escuta de conexões clientes.

--authenticationDatabase

Parâmetro para especificar qual banco de dados será utilizado para autenticar o usuário.

--username

Parâmetro para especificar o usuário conectado.

--password
Parâmetro para especificar a senha do usuário conectado.

--db
Parâmetro para especificar o banco de dados conectado.

Desinstalar

Neste capítulo, vamos ver como proceder para remover a instalação do MongoDB.

Atenção! Esse processo vai remover completamente o MongoDB, suas configurações e todos os bancos de dados. O processo não é reversível.

1° Passo: pare o serviço do MongoDB.

```
sudo mongod --shutdown --port 27017 --dbpath /var/lib/MongoDB
```

2° Passo: remova todos os pacotes instalados referentes ao MongoDB.

```
sudo apt-get purge MongoDB-org*
```

3° Passo: remova o diretório de log do MongoDB.

```
sudo rm -r /var/log/MongoDB
```

4° Passo: remova o diretório de dados do MongoDB.

```
sudo rm -r /var/lib/MongoDB
```

Considerações

Os arquivos de código fonte java usados nos exemplos do livro estão disponíveis no endereço https://www.dropbox.com/s/pfwmj7g41i1mxzu/mongodb-java.tar.gz?dl=0.

Agradecimentos

A Deus, pela saúde que me concede e me permite acordar por mais um dia de vida e perceber que eu tenho mais uma oportunidade de aprender.

Sobre o autor

André Tricano é casado e nasceu no Rio de Janeiro, mas vive na cidade de Vitória no estado do Espirito Santo.

No cenário acadêmico, André Tricano é graduado em Sistema de Informação e pós graduado em Gerência de Projeto pela Universidade de Vila Velha.

No cenário profissional, André Tricano atuou no setor de tecnologia de empresas do seguimento de energia elétrica, saneamento básico e extração de petróleo.

Profissional certificado na plataforma Java pela multinacional americana Oracle Corporation.

Atualmente André Tricano é diretor da empresa Vynx.

A empresa Vynx é especializada em desenvolver soluções de produto e projeto para gestão de negócio na nuvem.

andre.tricano@vynx.com.br

www.vynx.com.br