UTILIZAÇÃO DA FERRAMENTA FLYWAY PARA GERIR E APLICAR ATUALIZAÇÕES DE BANCO DE DADOS

Endrigo Antonini[1], Luiz Camargo[2]

Resumo: *Este trabalho visa mostrar a importância da adoção e implementação de um procedimento de desenvolvimento onde haja atenção com o principal ativo de um software, a informação. Como esta em sua maioria das vezes é armazenada em um banco de dados toda e qualquer alteração que deva ser realizada no banco de dados deve ser efetuada em conjunto com a evolução do software. Para facilitar a adoção desse processo no ciclo de desenvolvimento foi selecionada a utilização da ferramenta Flyway, cuja ferramenta objetiva realizar a gestão de aplicação das alterações necessárias no ambiente em questão, assim garantindo que uma determinada alteração seja executada uma única vez no sistema. Para isso foi criado um estudo de caso através da documentação da ferramenta para gerar um projeto que utiliza a mesma para aplicar scripts de atualização ao banco de dados.*

Palavras-chave: *Flyway, Banco de dados, Java, Database migration.*

1 INTRODUÇÃO

Através da constante evolução dos processos de desenvolvimento de software e o aumento da difusão de metodologias ágeis, onde o desenvolvimento aproxima-se do cliente e criasse ciclos menores com pequenos entregáveis, fez com que o código fonte de um software sofresse alterações em sua lógica e também na estrutura de forma mais rápida e mais constante. Devido a essa situação, constatou-se que alterações em software são mais dinâmicas e menos difíceis de serem mantidas e efetuadas. Mas por outro lado, alterações destinadas a estrutura de dados persistidos, considerando em sua maioria banco de dados são mais difíceis de serem tratadas pois como o banco de dados possui mecanismos de garantir a integridade de informação isso faz com que algumas alterações que aparentemente são simples, sejam um pouco mais complexas de serem executadas, como a adição de uma coluna onde seu valor não seja nulo e essa deve ser populada através da informação que encontra-se em outra coluna.
Devido a dificuldade em paralelizar as alterações relacionadas a software e banco de dados e ao mesmo tempo gerenciar a aplicação das alterações foram adotadas em várias situação a criação de manual explicando ao administrador do ambiente o procedimento que deverá ser executado

[1] Pós-graduando em Engenharia de Software pela UNISOCIESC. E-mail: flyway@endrigo.com.br.

[2] Professor nos cursos de graduação e pós-graduação na UNISOCIESC. E-mail: camargho@gmail.com.

para realizar a atualização de banco de dados, mas o processo continua sendo manual e suscetível a falhas, principalmente as falhas humanas já que esse processo é totalmente manual. Algumas ferramentas foram desenvolvidas para auxiliar esse processo, mas poucas integram-se facilmente com o procedimento de desenvolvimento bem como com a expedição do software.
Por esse motivo selecionou-se a ferramenta Flyway, que cujo um dos objetivos é atuar junto ao desenvolvimento da aplicação fazendo com que a gestão dos *scripts* a serem executados pela ferramenta sejam controlados junto ao controle de fontes da aplicação. Isso faz com que o processo de desenvolvimento seja menos falho pois tanto a gestão de fontes da aplicação quanto a gestão de *scripts* são controladas em conjunto através da mesma ferramenta SCM (Sistema de controle de versão, do inglês, *Source Control Management*).
Com isso, o objetivo desse trabalho é de criar um projeto onde será aplicada a ferramenta Flyway para efetuar a gestão do banco de dados.

2 ENGENHARIA DE SOFTWARE

Segundo Pressman (2011) não era possível prever que o software seria utilizado e incorporado em várias áreas que variam desde transporte passando pelo uso militar a até mesmo a medicina. Com isso, não foi pensado que existiria milhões de programas e esses algum dia teriam que ser corrigidos, adaptados e melhorados para atender a novas necessidades e demandas oriundas desses e outros setores. Com isso, o autor também destaca que a realização de tais atividades para manutenir e adequar os softwares consumiriam mais pessoas e recursos do que a energia gasta para a criação de um novo software.
Por esse motivo, percebeu-se a necessidade de melhorar e criar processos de engenharia de software para que fosse possível gerenciar todo o ciclo produtivo de um software.
Vale ressaltar que software não é fabricado no senso de clássico da palavra (linha de produção), esse é desenvolvido ou passa por processo de engenharia. Apesar de que ambas possuem a conotação de gerar um "produto" ao fim de seu ciclo, estas possuem diferentes meios para alcançar seu objetivo. No caso do software, os custos concentram-se no processo de engenharia fazendo com que seus projetos não possam ser geridos como se fossem projetos de fabricação (Pressman, 2011).

3 GESTÃO DA VERSÃO DO BANCO DE DADOS

3.1 Contextualização do problema

Devido as constantes alterações, correções e melhorias realizadas em um projeto de software bem como a adoção de metodologias ágeis dentro do processo de desenvolvimento de software, onde a cada ciclo produtivo (exemplo: a Sprint do Scrum) faz-se necessário a realização de um produto ou componente funcional, faz com que existam casos onde a estrutura do banco de dados precise ser atualizada e seus dados manipulados.
Essas tarefas de manipular a estrutura do banco de dados e seus dados podem ser desde rotinas simples, como a adição de uma coluna à até tarefas mais complexas como a criação de uma

tabela e inserção de registros nesta levando em conta dados já existentes no banco de dados e ao mesmo tempo executando operações matemáticas.
Conforme já citado na introdução onde destacamos que estamos na era da informação e que essa é o produto mais importante, faz com que precisemos adotar práticas, ferramentas e cuidados para ao atualizar um software não corromper esse bem tão preciso. Bem como garantir que a aplicação de *scripts* sejam realizadas de forma síncrona e ordenada, onde cada *script* seja executado no seu devido momento.
A falta de utilização de uma ferramenta que automatize esse processo faz com que acabe gerando um trabalho extra para equipes cujo o foco é voltado ao controle de banco de dados bem como para equipes que tem por finalidade realizar o *deploy* da ferramenta em ambiente de produção. Fazendo com que esses precisem analisar os *scripts* e aplica-los na sequência desejada.
A realização desse trabalho de forma manual acaba introduzindo mais um fator de risco ao atualizar o software que seria o fator humano. Esse fator acaba sendo de grande impacto no sucesso da realização do processo de atualização pois o simples fato de aplicar os *scripts* de forma errônea ou de até mesmo a não aplicação dos mesmos acarretar no comprometimento do bom funcionamento da aplicação como também no corrompimento do banco de dados.
É possível relacionar esse problema com a visão de Nygard (2011) que vê a primeira *release* do produto não sendo como o fim do projeto, mas sim o inicio da vida do projeto e que a qualidade de vida pessoal (pessoas envolvidas com o projeto) depende das escolhas que fez antes desse marco do produto. Ele ainda destaca que você precisa lembrar que esse produto é algo que pode carregar / alavancar seus negócios para o futuro e sendo assim deve ser feito com cuidado e de forma organizada.
Visto isso, realizar a escolha de uma ferramenta e um processo para a manutenção de sua base de dados é de extrema importância para que seu projeto tenha sucesso nas próximas releases bem como no ciclo de vida do mesmo.

3.2 Data migration

A necessidade de realizar alterações no banco de dados de acordo com a evolução e/ou correção do projeto não é recente. Essa já é discutida e implementada por vários fabricantes através de diferentes abordagens.
Janssen (2014) define *Data Migration* como sendo o processo de transportar dados entre computadores, dispositivos de armazenagem ou formato dos dados. Também destaca que esse processo é algo primordial para ser considerado na implementação, durante sua atualização e consolidação do mesmo.
Em um projeto, sendo na área de inovação do produto bem como na área de manutenção, acabam surgindo algumas questões como:

- Em que estado está o banco de dados nesta máquina?
- Este *script* já foi aplicado ou não?
- A correção para o problema X foi aplicada em produção?
- Quais todos os passos para eu criar uma nova instância do banco de dados?

E, infelizmente, muitas das vezes acabamos recebendo a reposta como sendo "Não sabemos".

Por isso é extremamente importante um processo e ferramenta para garantir a integridade da estrutura de informação bem como seus dados tanto em um processo onde a aplicação esteja sendo atualizada como também garantir que ao instalar a aplicação em um ambiente novo sua estrutura seja idêntica a um ambiente que foi atualizado.

3.3 Ferramentas selecionadas

Através da situação já explicada foi selecionada a ferramenta Flyway para a realização deste processo junto com a ferramenta Maven que faz todo o controle de compilação e empacotamento do projeto.

3.3.1 Maven

A historia do Maven começa através do projeto Jakarta Turbine. Este era constituído por vários outros projetos e sub projetos e cada um desses possuía seu *script* ANT, sendo que cada um dos *scripts* possuía peculiaridades distintas. Devido a essa confusão e dificuldade de garantir que a compilação em máquinas distintas se darão de forma igual, surgiu a necessidade de padronizar o processo de compilação e empacotamento através de um processo fácil, simples e de fácil compreensão do que o projeto consiste, bem como facilitar a forma de publicação de informações do projeto e compartilhar os objetos compilados do mesmo.
Através dessas necessidades surgiu-se o Maven, que segundo o língua Iídiche significa acumulador de conhecimento, onde este acabou se tornando uma ferramenta que pode controlar e construir (compilar e empacotar) qualquer projeto baseado em Java.
Atualmente já foram desenvolvidos diversos outros plugins fazendo com que a ferramenta extrapole as fronteiras de estar somente relacionada a projetos Java podendo também ser utilizado para a compilação de projetos desenvolvidos utilizando linguagens como Adobe Flex, C , C++ …
Seu funcionamento se dá através de um arquivo descritor chamado de POM (Modelo de Objeto do Projeto, do inglês: Project Object Model) que é responsável por conter as informações do projeto como: bibliotecas dependentes, parâmetros de compilação, formato de empacotamento, plugins para o processo de compilação, plugins para geração de métricas relacionadas ao fonte e várias outras informações que podem não ser de interesse a parte de compilação, mas de interesse e facilitação a compreensão humana como descrição do projeto, lista de desenvolvedores, site do projeto e outra informações.
Mesmo com todas as facilidades que a ferramenta gera no processo de construção do projeto, isso não isenta a necessidade da equipe possuir conhecimento sobre os mecanismos, processos e funcionamentos para a construção do projeto (MAVEN, 2014).

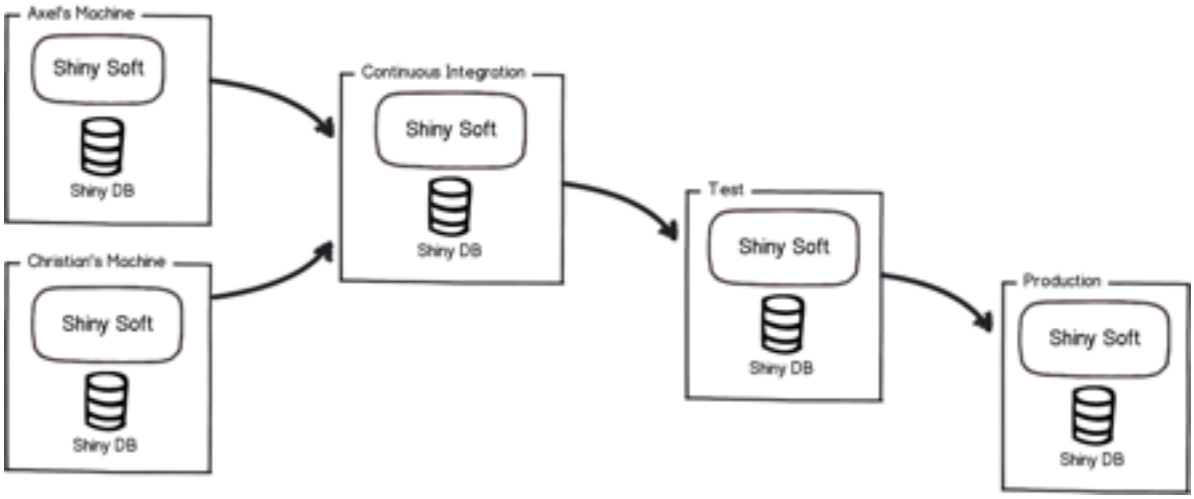
3.3.2 Flyway

Através de uma breve troca de mensagens com o autor da ferramenta, o mesmo relatou que sua motivação aconteceu pois questões relacionadas a evolução do banco de dados sempre foram um dos principais problemas que ele enfrentou em diversos projetos e isso sempre gerou muito

desgaste pessoal. Por isso muitas equipes optaram por tentar evitar ao extremo situações onde seja necessário realizar qualquer tipo de alteração no banco de dados.
Através de pesquisas com as ferramentas existentes no mercado, relata o autor, nenhuma delas encaixava com suas idéias e necessidades que seria de que a ferramenta precisa ser simples, transparente e compatível com SQL. Algumas das ferramentas eram baseadas em XML e outra delas o desenvolvimento estava estagnado. Por esse motivo ele acabou decidindo juntar os fontes que possuía de projetos particulares de publica-los. Nesse momento surge o Flyway em abril de 2010.

Uma das situações que a ferramenta se propõe atuar, conforme a Figura 1, é que ela pode ser utilizada em todo o ciclo de criação de um produto. Ou seja, atuar na criação e atualização dos bancos de dados dos desenvolvedores, bem como nas etapas de integração contínua, testes e por fim na produção. Fazendo com que muitas das vezes o tratamento de criação e evolução do banco de dados não ocorra em um único ambiente mas sim em vários ambientes que podem estar em execução paralela.

Figura 1 Visão do ambiente de desenvolvimento

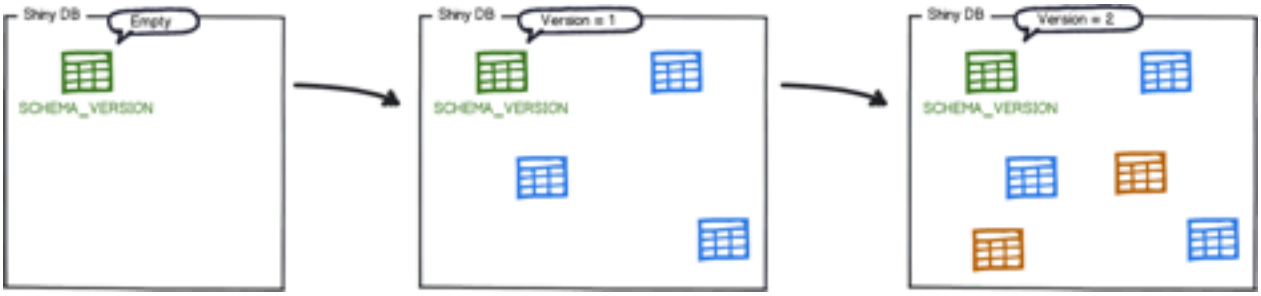


Fonte: FLYWAY, 2014

Conforme pode ser visto na descrição da ferramenta Maven pode-se verificar que o processo do controle de código fonte, compilação e expedição já encontram-se mais maduros e difundidos no mercado. Já no quesito banco de dados a criação de um processo que abranja a evolução continua do mesmo e a facilidade em descobrir em que estado o banco de dados encontra-se no momento é algo que ainda não está tão difundido.
Em relação ao funcionamento do Flyway, sabe-se que ele efetua o controle do estado do banco de dados através de uma entidade (tabela) que será gerada junto as tabelas do banco de dados (por padrão) que deseja-se gerenciar. Esta entidade chama-se SCHEMA_VERSION. Se a ferramenta não encontrar essa entidade, a ferramenta irá cria-la.

Esta tabela contem informações como a versão da alteração que foi aplicada, a sequência em que foi aplicado a descrição da alteração, o tipo de fonte (SQL ou Java), quando foi realizada, tempo que levou para executar entre outras informações pertinentes ao controle do estado do banco de dados.
Logo após a ferramenta ser iniciada, ela procura por padrão arquivos com extensão SQL localizados no classpath dentro da pasta "db/migration", bem como classes Java localizadas também no classpath no pacote "db.migration". Em ambos os casos o arquivo e classe deve obrigatoriamente começar com a letra "V" (em maiúsculo) e logo em seguida a versão que este destina-se seguido de dois sublinhados e em seguida a descrição utilizando o sublinhado no local de espaço. Exemplo: o significado do arquivo: "V1_1__Correcao_tabela_usuario.sql" contém as seguintes informações: versão: "1.1", Descrição: "Correcao tabela usuario". É aconselhado a não utilização de acentos e caracteres especiais nos nomes dos arquivos.
A figura 2 exemplifica a execução de *scripts* de atualização até a chegada na versão 2, sendo que o banco de dados encontrava-se vazio.

Figura 2 Forma de controle de aplicação das atualização pelo Flyway



Fonte: FLYWAY, 2014

No caso de arquivos do tipo SQL as instruções devem ser feitas em SQL ANSI ou utilizando a instruções que são relacionadas ao próprio banco de dados como o PL/SQL. Um arquivo pode conter mais de uma instrução e estas devem estar separadas por ";". Então caso necessite realizar um DDL (Data Definition Language) ou um DML (Data Manipulation Language) basta apenas criar um arquivo SQL seguindo as regras acima.
A utilização de classes Java nesse processo é aconselhada somente em casos complexos, onde é necessário por exemplo realizar um processamento da informação antes de atribuir a mesma a base de dados, ou atualizar alguns registros com base em dados oriundos de arquivos de configuração local, web services entre outras possibilidades.

4 ESTUDO DE CASO - IMPLANTAÇÃO DA FERRAMENTA FLYWAY

Para efeito de facilitar a utilização e possibilitar uma maior iteração com a ferramenta os fontes gerados e utilizados para o artigo a seguir estão disponíveis de forma pública através de repositório git. A url para acesso via web do mesmo é: https://github.com/antonini/exemplo-

flyway/. Esse projeto e passos foram gerados através da leitura e interpretação da documentação da ferramenta.

4.1 Ambiente

Para a execução deste experimento em seu ambiente é necessário possuir, instalar e configurar em sua máquina, as seguintes ferramentas:

- Java 7 ou superior;
- Maven 3.2.1 ou superior;
- GIT client 1.8 ou superior;

4.2 Quanto ao experimento

O objetivo desse experimento é de criar um projeto modelo desde o início, utilizando a ferramenta Flyway para a gestão e aplicação de alterações no banco de dados.
Consiste em um projeto Java onde será criada uma entidade para realizar o cadastro de pessoas. Em primeiro momento será importado uma quantidade de nomes completos e para um desses nomes será atribuído o nome errado a uma pessoa e essa será corrigida através de uma query que será executada através da ferramenta.
Em outro passo será criado uma nova coluna a essa entidade que deverá conter o sobrenome da pessoa. A criação dessa coluna e a divisão entre nome completo e nome e sobrenome será feita através de *scripts*.

4.3 Criação da estrutura do projeto e adição da dependência Flyway

Para criação do projeto foi executado no console o comando que encontra-se na figura 3. Através desse será criado um projeto maven básico em uma subpasta chamada "exemplo-flyway" com os artefatos básicos do projeto.

Figura 3 Comando para criar o projeto

```
mvn archetype:generate -B -DarchetypeGroupId=org.apache.maven.archetypes \
 -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.1 \
 -DgroupId=br.com.endrigo -DartifactId=exemplo-flyway -Dversion=1.0-SNAPSHOT \
 -Dpackage=br.com.endrigo
```

Fonte: O Autor

Dentro dessa pasta existe um arquivo chamado "pom.xml" foi utilizado um editor de texto qualquer para altera-lo e adicione as dependências no bloco « dependencies". A figura 4 exibe o trecho de código que teve de ser adicionado ao bloco. Com isso o projeto passa a ser dependente do Flyway e também a do H2 Database, um banco de dados extremamente enxuto para a execução dos exemplos.

Figura 4 Bloco de dependências

```
<dependency>
        <groupId>com.googlecode.flyway</groupId>
        <artifactId>flyway-core</artifactId>
        <version>2.3</version>
</dependency>
<dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <version>1.3.170</version>
</dependency>
```

Fonte: O Autor

Para que o maven possa efetuar o download das bibliotecas do projeto o comando que encontra-se na figura 5 foi executado.

Figura 5 Comando para limpar, compilar e empacotar o projeto

```
mvn clean install
```

Fonte: O Autor

Isso fez com que o maven efetuasse o download dos artefatos dependentes do projeto e que o projeto fosse compilado, empacotado (geração de um jar) e arquivado dentro do repositório local do maven.

4.4 Criação dos *scripts* e classe de execução

Foi criada uma classe onde esta irá apenas executar o Flyway. Vale ressaltar que em caso de que a ferramenta seja adotada por algum outro projeto é altamente aconselhado que a execução do Flyway seja uma das primeiras coisas a serem executadas. Evitando assim a execução de qualquer pesquisa, inserção, alteração no banco de dados pelo sistema antes de as alterações e atualizações do produto serem efetuadas.
Para isso dentro do diretório diretório "src/main/java/br/com/endrigo/" foi criado uma classe chamada *TesteFlyway* cujo pacote é *br.com.endrigo* .
O fonte fonte Java que encontra-se na figura 6 define o acesso ao banco de dados instancia o Flyway e executa o processo do mesmo de migração. É importante ressaltar que o fato de executar o método migrate não significa que os *scripts* sempre serão executados novamente. O método migrate irá comparar o estado do banco de dados com o estado dos arquivos locais e com base nisso irá ou não aplicar os arquivos pertinentes.

Figura 6 Comando para limpar, compilar e empacotar o projeto

```
private static final String DB_DRIVER = "org.h2.Driver";
private static final String DB_URL = "jdbc:h2:file:target/exemplo-flyway";
private static final String DB_USER = "sa";
private static final String DB_PASSWORD = null;

public static void main(String[] args) throws SQLException, ClassNotFoundException {
        Flyway flyway = new Flyway();
        flyway.setDataSource(DB_URL, DB_USER, DB_PASSWORD);
        flyway.migrate();
}
```

Fonte: O Autor

Os fontes disponibilizados através do repositório github possuem explicações detalhadas sobre a necessidade de cada linha e sua função.
Os *scripts* da tabela 1 foram criados e atribuídos cada um ao arquivo definido conforme a coluna "nome do arquivo". Todos esses foram criados dentro do diretório "src/main/resources/db/migration".

Tabela 1 - *Scripts* e seus arquivos

Nome do arquivo	*Script*
V1__Criar_tabela_pessoa.sql	create table PESSOA (CHAVE int not null, NOME varchar(100) not null);
V2__Adicionar_Pessoa.sql	INSERT INTO PESSOA (CHAVE, NOME) VALUES (1, 'Adriana Silva'),(2, 'Andre Soares'),(3, 'Bruna da Silva'),(4, 'Claudio Antonio'),(5, 'Elaine Faria'),(6, 'Fernanda Matos'),(7, 'Gilmara Souza'),(8, 'Henrique Silva'),(9, 'Zeeeee Almeida'),(10, 'Larissa Faria'),(11, 'Lucimara Abreu'),(12, 'Maria Aparecida'),(13, 'Patricia Poeta'),(14, 'Ronaldo Benke'),(15, 'Sandra Brito'),(16, 'Sandro Pires'),(17, 'Sergio Petry');
V2_1__Corrige_nome_jose.sql	UPDATE PESSOA SET NOME = 'Jose Xavier' WHERE CHAVE = 9;
V3__Adiciona_sobrenome_e_altera_dados.sql	ALTER TABLE PESSOA ADD COLUMN SOBRENOME CHAR(100); UPDATE PESSOA SET SOBRENOME = TRIM(SUBSTRING(NOME, INSTR(NOME,' '))); UPDATE PESSOA SET NOME = TRIM(SUBSTRING(NOME, 0, INSTR(NOME,' ')));

Fonte: O Autor, 2014

5 RESULTADOS OBTIDOS

Através do comando que encontra-se na figura 7 foi possível executar a ferramenta e verificar a saída onde é exibido o nome e sobrenome de cada pessoa inserida no banco de dados.

Figura 7 Comando para executar o exemplo

```
mvn clean install package exec:java -Dexec.mainClass=br.com.endrigo.TesteFlyway
```

Fonte: O Autor

mvn clean install package exec:java -Dexec.mainClass=br.com.endrigo.TesteFlyway

Com isso verificou-se que os scripts foram aplicados de forma síncrona e ordenada, fazendo com que o banco de dados efetuasse as atualizações conforme o planejado.
Também foi possível notar que a utilização da ferramenta facilitou o processo de desenvolvimento pois os scripts são aplicados automaticamente, então se entrar na equipe de desenvolvimento um novo colaborador, este ao executar a aplicação pela primeira vez em seu ambiente cru perceberá que todos os scripts da evolução do produto foram aplicados. Através disso, o desenvolvedor tem uma preocupação a menos que seria a de manter o banco de dados de deste atualizado.

5 CONCLUSÃO

A adoção de um processo de desenvolvimento ou de expedição que englobe procedimentos de atualização do banco de dados é algo primordial e não deveria ser esquecido pois como visto e frisado neste artigo, o principal ativo de um software não é o código fonte do mesmo mas sim a informação que esse armazena, gera ou disponibiliza para o usuário. E o fato de não ter ou dar a devida atenção a um procedimento que abranja a questão de gestão da atualização da informação no banco de dados pode acabar gerando problemas futuros tanto na questão de possuir informações desatualizadas a até mesmo corromper a base de dados.
Através da utilização da ferramenta Flyway no processo de desenvolvimento verificou-se a facilidade em gerir as alterações referente ao banco de dados bem como a "tranquilidade" que uma ferramenta como essa gera para a equipe de desenvolvimento devido a garantia que esta traz em aplicar as alterações ao banco de dados de forma síncrona e na sequência correta. A vantagem que esta ferramenta traz para o processo de desenvolvimento é que tanto o código fonte da aplicação quanto os *scripts* podem ser geridos dentro do mesmo projeto no sistema de controle de versão, fazendo com que a gestão de artefatos a serem expedidos sejam controlados através de um único ponto e não através de diversas ferramentas e sistemas de gestão.
Para dar continuidade a este trabalho aconselha-se a verificação de novas versões da ferramenta, pois durante o processo de criação desse artigo uma nova versão foi expedida dando suporte a mais plataformas sendo um exemplo a plataforma Android (um sistema operacional para dispositivos móveis desenvolvidos pela Google).

Com isso, foi possível verificar que a implementação e adoção da utilização da ferramenta não é complexa nem é algo que tomaria muito tempo para ser realizada dentro de um projeto acadêmico ou corporativo, mas sim a mudança da visão da equipe para adotar um processo em que essa seja responsável por criar e manter os *scripts* de atualização em quanto elas estão desenvolvendo a aplicação.

REFERÊNCIAS

FLYWAY. **Flyway** - The agile database migration framework for Java. Disponível em: <http://www.flywaydb.org/>. Acesso em: 11 fev. 2014.

JANSSEN, Cory. **What is Data Migration** - Definition from Technopedia. Disponível em: <http://www.techopedia.com/definition/1180/data-migration>. Acesso em: 10 fev. 2014.

MAVEN. **Maven** - What is Maven?. Disponível em: <http://maven.apache.org/what-is-maven.html>. Acesso em: 11 fev. 2014.

NYGARD, Michael T.. **Release It!**: Design and Deploy Production-Ready Software. Estados Unidos da América: Pragmatic Bookshelf, abr 2007.

PRESSMAN, Roger S. **Engenharia de Software:** Uma abordagem profissional. 7. ed Porto Alegre: AMGH, 2011.

USING FLYWAY TOOL TO MANAGE AND APPLY DATABASE UPGRADES

Abstract: *The purpose of this document is to show the importance of the adoption and implementation of a development process that contains a minimum concern with the most important asset of a software, the information. As the information is usually stored on database any change that must be applied on the it must be done while the software evolves. To make the adoption process easier on the development process it was chosen the use of the Flyway tool, whose goal is to manage the applicability status of those changes on the desired environment, so it will guarantees that a change script will be apply just a single time on the system. It was done a study case using the tool documentation to create a project that uses the given tool to apply the changes on the database.*

Key words: *Flyway, Database, Java, Database migration.*