ISSN 2316-2872
T.I.S. São Carlos, v. 4, n. 3, p. 216-222, set-dez 2015


# VERSION BASE - Controle de Versão de Banco de Dados para Aplicações Móveis

**Jones dos Santos Carvalho, Marilde Terezinha Prado Santos**

**Resumo:** Hoje em dia, os smartphones representam a nova era móvel de informação. Em todos os modelos, existe uma central de aplicativos que os distingue em várias categorias. O versionamento de banco de dados é uma solução que ajuda na manutenção dos sistemas e permite aos usuários trabalhar sempre na versão corrente do sistema de banco de dados. Falar em versionamento, significa dizer que é possível trabalhar sem perda de dados em várias etapas de um processo de desenvolvimento do sistema do banco de dados. Este trabalho apresenta o aplicativo Version Base, uma nova solução para o versionamento de banco de dados móvel em nível de esquema e instâncias. Os testes realizados confirmam que a solução proposta atende aos requisitos estabelecidos e é mais completa do que os concorrentes existentes no mercado.

**Palavras-Chave:** Controle de Versão, Versionamento, Banco de Dados, Desenvolvimento de Sistemas.

***Version Base – Database version control for mobile applications***

***Abstract:*** *Nowadays, smartphones represent the new era of mobile information. All models have an application center that makes them different in several categories. The versioning of database is a solution that helps maintaining the systems and allows users to work in the current version of the database system. Talking about versioning means that it is possible to work without losing data in various stages of the development process of a database system. This paper presents Version Base application, a new solution for the versioning of mobile database in schema and instance level. The tests confirm that the proposed solution meets the requirements and is more complete than the existing competitors in the market.*

***Keywords:*** *version control, versioning, database, systems development.*

## I. Introdução

Um sistema de versionamento de esquema pode ser usado em rede e permite que muitas pessoas usem o mesmo sistema em diversos dispositivos. Nele é feito todo controle de dados e as respectivas alterações feitas pelos usuários. Existem muitos sistemas de controle de versão, e alguns deles também são softwares de gerenciamento de configurações, estes sistemas são responsáveis por gerenciar códigos-fonte e muitos têm características especificas para cada modo de desenvolver um software (SVNBOOK,2015a).

Também existem softwares de controle de versão para banco de dados, como o SQL Lite Asset, que se propõe a realizar um controle de banco de dados para aplicativos móveis usando Android. A ferramenta visa facilitar desde a criação do banco de dados inicial até o controle das versões seguintes. O versionamento pode permitir que recupere versões antigas de seus dados alterados. Algumas pessoas pensam em um controle de versão como uma "máquina do tempo" (SQL LITE ASSET,2015a), porém a ferramenta SQL Lite Asset não realiza todos os aspectos de versionamento, não sendo possível voltar versões ou avançar sem ter que compilar novamente a aplicação.

Este trabalho apresenta um experimento que emprega a solução proposta Version Base, através do qual é possível controlar versões de banco de dados, restaurar e realizar backup remotamente, sem danificar a aplicação. O cenário que justifica a necessidade do controle de versão em smartphones refere-se a um sistema corporativo, em que o esquema do banco de dados deve estar sempre sincronizado, e garantir que no momento em que o banco de dados é atualizado, o sistema que funciona em todos os smartphones da corporação deve também ser atualizado, sem perder seus dados.

Android é um sistema operacional Linux, desenvolvido pela Google e que é, atualmente, usado na maioria dos smartphones (GOOGLE ANDROID). Android é o mais utilizado por ser gratuito e independente de plataforma. A linguagem de programação para desenvolver aplicativos para este ambiente é JAVA. Existem várias versões do SO Android, cada qual indicada para determinadas características do dispositivo. No caso da API SQL Lite, já vem instalado em todas as versões.

Os principais procedimentos de atualização podem ser controlados pelo Version Base, sendo que as vantagens são inúmeras, podendo desde permitir uma recuperação de falha

Departamento de Computação - Universidade Federal de São Carlos (UFSCar)
Caixa Postal 676 – 13.565-905 – São Carlos – SP – Brasil
**Autor para correspondência:** jonescarva@gmail.com, marilde@dc.ufscar.br

na aplicação até permitir a manipulação do banco de dados, sem perder os dados pré-existentes. Outra grande vantagem em utilizar um versionamento é a possibilidade de acesso remoto, ou seja, este sistema não precisa de instalação nos dispositivos, ou mesmo não exige um alto desempenho dos mesmos. Além disso, por uma página web é possível que o usuário tenha acesso às manutenções de qualquer ambiente através de um computador, tablet ou celular com acesso à internet.

Este artigo é organizado como se segue: na Seção 1 será detalhado o sistema Version Base, um concorrente de um plugin de versionamento para banco de dados móveis aplicado à plataforma Android. A Seção 2 apresenta a conceituação de WebService, que pode ser entendida como uma plataforma web de comunicação entre sistemas. Na Seção 3 é apresentado em detalhes o sistema Version Base, com suas definições e características. A Seção 4 apresenta os trabalhos correlatos e as melhorias do sistema e por fim as considerações finais deste trabalho.

## II. WEBSERVICE

Um Webservice pode ser definido como uma arquitetura de software que relaciona componentes do sistema em um ambiente distribuído através de serviços e acessados de modo dinâmico por uma rede. De um modo geral, os WebServices são serviços que estão disponibilizados na internet e são acessados por meio de protocolos que podem ser lidos em qualquer linguagem de programação, qualquer sistema operacional e rodando em qualquer dispositivo (SILVA, 2004).

REST ou Representational State Transfer é “um conjunto de princípios arquiteturais que quando aplicadas como um todo enfatiza a escalabilidade da interação entre componentes para reduzir a latência de interação, garantir segurança e encapsular sistemas legados”, conforme Roy Fielding (apud OLIVEIRA, 2009, p.51). Os seguintes contextos são relacionados ao termo REST: Cliente/Servidor; apoio a sistemas de cache; sistemas em camadas (suporte à escalabilidade); estado nulo (cada requisição de um cliente ao servidor deve conter todas as informações necessárias para entender a requisição); stateless (comunicação sem controle de estado); sistema uniforme utilizando 4 métodos padronizados: GET – operação de leitura; PUT – relacionada e inserção ou alteração; POST – cria um objeto no servidor; e DELETE – operação de remoção.

Conforme Tilkov (OLIVEIRA, 2009), o REST é “um conjunto de princípios que definem como os padrões Web, como o HTTP e URIs devem ser utilizados”. Tilkov ainda afirma que “a principal promessa do REST é que se você aderir aos princípios durante o projeto de sua aplicação, você irá acabar com um sistema que explora a arquitetura Web em seu benefício”.

No modelo REST, o mais importante são as URLs do sistema e os recursos (resources). Um resource ou entidade são representados em XML ou JSON. A URL é a mesma para ambos os tipos, porém, o método HTTP é específico para cada um deles, ou seja a implementação dos métodos POST, GET, PUT, DELETE é distinta para cada um dos tipos, assim como os parâmetros de request para o retorno da requisição. O padrão SOAP utiliza text/XML e o padrão REST text/JSON, que é o modelo usado na solução Version Base, proposta neste artigo, visto que o retorno em JSON, diminui o tráfego de dados entre cliente/servidor para aplicativo móvel.

## 3. Solução Version Base

O cenário de um sistema de banco de dados cliente servidor, com aplicativos móveis realizando downloads de novas versões do esquema de banco de dados para o SQLLite no dispositivo móvel, mas poderá perder a compatibilidade com a aplicação móvel se o banco de dados for versionado.

A solução proposta neste trabalho, o Version Base, foi baseada na percepção que o atual versionador de banco de dados móvel, o Android SQL Lite Asset, não dispõe de recursos úteis para permitir o versionamento de banco de dados em nível de esquema e de instâncias.

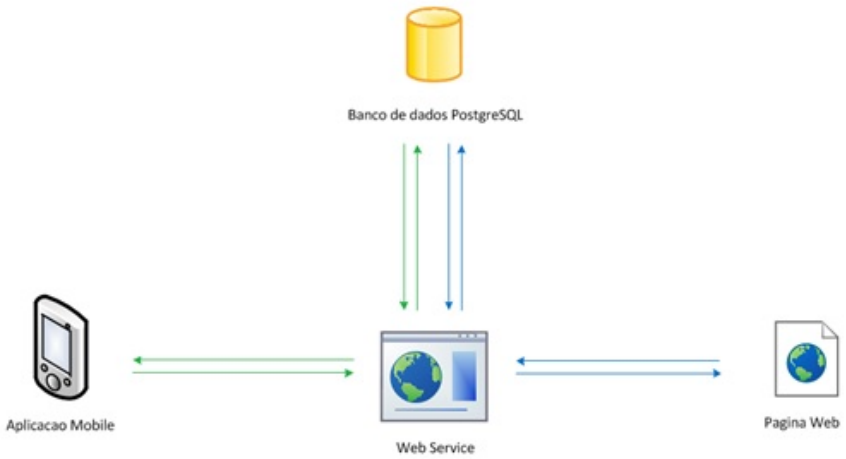


Figura 1 – Topologia do sistema Version Base

A figura 1 apresenta a topologia proposta para a solução Version Base, cuja solução incorpora um Webservice e uma aplicação Android, que foram desenvolvidos na linguagem Java e uma página Web em HTML com framework Jquery. O sistema gerenciador de banco de dados PostgreSQL foi incorporado para armazenar o esquema das versões no banco de dados interno da Version Base

*A) Página Web*

A Página Web é onde é feita toda manutenção do Version Base, nela o usuário administrador do banco de dados pode atualizar os esquemas e mudar as versões das aplicações. O WebService é responsável por fazer todo tráfego das informações da página web e da aplicação Android até o SGBD PostgreSQL. A Aplicação Android restringe-se a atuar como terminal de verificação da própria versão, e caso seja diferente da versão padrão gravada no banco de dados, a aplicação faz um cópia (backup) do banco atual e envia uma nova solicitação ao Webservice recuperando um novo esquema versionado e compatível com sua estrutura atual. Ao realizar essa operação busca reutilizar os dados inseridos anteriormente para ser restaurado junto com a versão padrão escolhida pelo usuário. Conforme pode ser observado na Figura 2, no diagrama de sequência é detalhada a solicitação da página web.

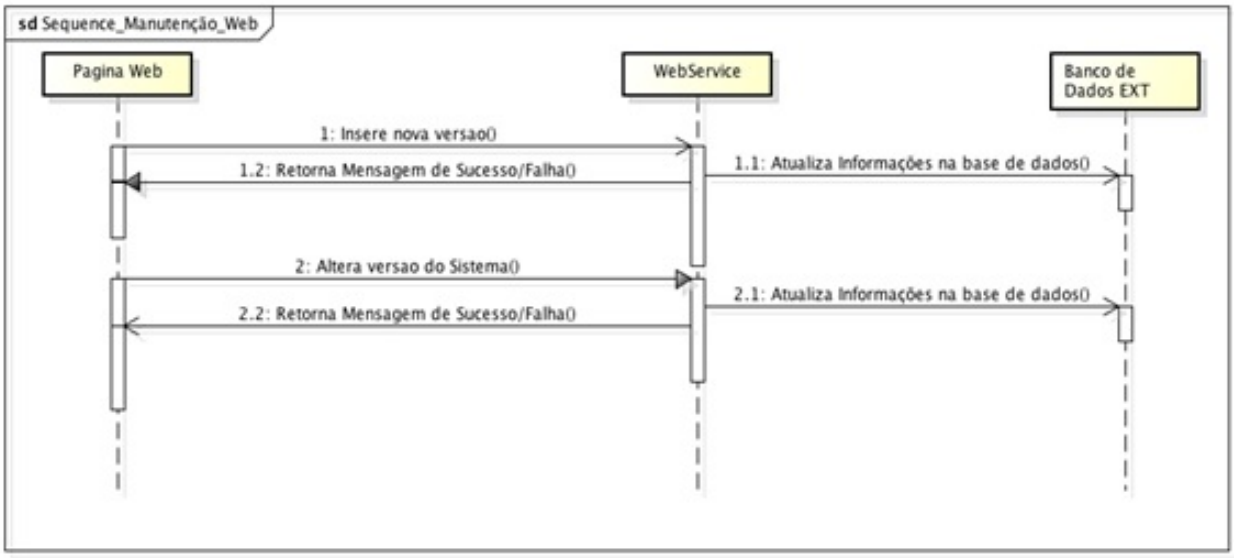


Figura 2 – Diagrama de sequência da página web

*B) Webservice*

O Webservice é responsável por toda transação de dados entre página web e o aplicativo Android, até o banco de dados no PostgreSQL. No Webservice se concentra toda regra de negócio do controle de versão. A figura 3 apresenta o diagrama de classes referente ao esquema do banco de dados interno do Version Base, apresentando os dados e os métodos que são utilizados no sistema.

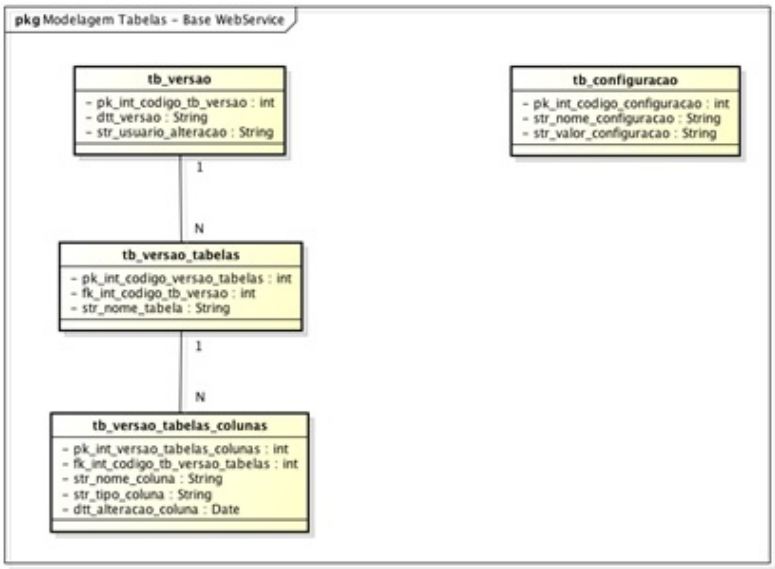


Figura 3 – Diagrama de classes Version Base

As classes representadas no diagrama de classes da Figura 3 mantém dados sobre o fluxo de controle que são armazenados e os esquemas conforme as versões do sistema, são elas:

• tb_versao – Armazena o código da versão, data e usuário que criou.

• tb_versao_tabelas – Armazena os nomes das tabelas, o código dela, e sua respectiva referencia com a tabela de versão, tb_versao.

• tb_versao_tabelas_colunas – Armazena os nomes das colunas, o código dela, o nome da coluna, tipo, data de alteração da coluna e sua respectiva referencia com a tabela tb_versao_tabelas.

• tb_configuracao – Armazena o código da versão atual, este campo será consultado pelo webservice e retornara para o aplicativo móvel o código da versão, assim se for diferente, faz o processo de versionamento.

As regras de negócio para a realização do controle de versão contemplam:

• Diferenças entre dados – Quando aplicar a versão, o sistema verifica a versão anterior com a nova e gera uma lista dos campos e tabelas que são diferentes entre as versões.

• Exclusão de campos anteriores da tabela – Opção na página web para que o usuário administrador das versões possa definir a exclusão total ou parcial da versão que vai ser sobrescrita.

• Manter dados antigos e adicionar os novos campos – Opção na página web para permitir ao usuário administrador das versões, manter os dados antigos e adicionar os novos fazendo um merge dos dados entre eles.

• Fazer backup da base de dados sempre que fizer alteração de versão – Sempre que fizer qualquer alteração que envolve o banco de dados local, fazer backup e atualizar o servidor HTTP.

• Caso verifique conflitos e precise de um merge dos campos e dados, gerar nova versão – Quando aplicar a versão e houver conflitos de campos se o usuário quiser manter os dados antigos, sugerir uma nova versão, fazendo merge da versão nova aplicada com a versão em execução.

• Nomenclatura de backup: bk_codigoVersaoApp (bk_3) – Nomenclatura padrão para backup de banco de dados SQL Lite.

• Verificações: Incluem checagens sobre o tamanho da propriedade da tabela; o tipo da propriedade e nome da propriedade. Caso seja aplicado, não se pode alterar os dados da versão depois do uso.

*C) Aplicação Android*

A aplicação Android é responsável por verificar a versão no webservice e aplicar o novo esquema do banco caso detecte diferenças entre as versões. Na Figura 4 pode ser observado o diagrama de sequência referente à solicitação da aplicação Android da Version Base. O aplicativo mobile faz uma requisição ao webservice enviando o código da versão atual da base mobile, o webservice verifica o código requisitado com o código atual cadastrado no servidor, se for diferente, recupera o esquema novo pelo código cadastrado, e também o esquema do código requisitado. Após isso, o webservice chama um método que verifica as diferenças entre as versões (diferenças de tabelas, campos e dados), prepara um merge das duas versões até encontrar algum conflito que possa impactar a perda de dados e gera uma nova versão. Como pode ser observado na Figura 4, a partir do subitem 1.2.1, o fluxo para verificação do aplicativo Android irá sempre requisitar o webservice para verificar se há diferença da versão da base mobile.

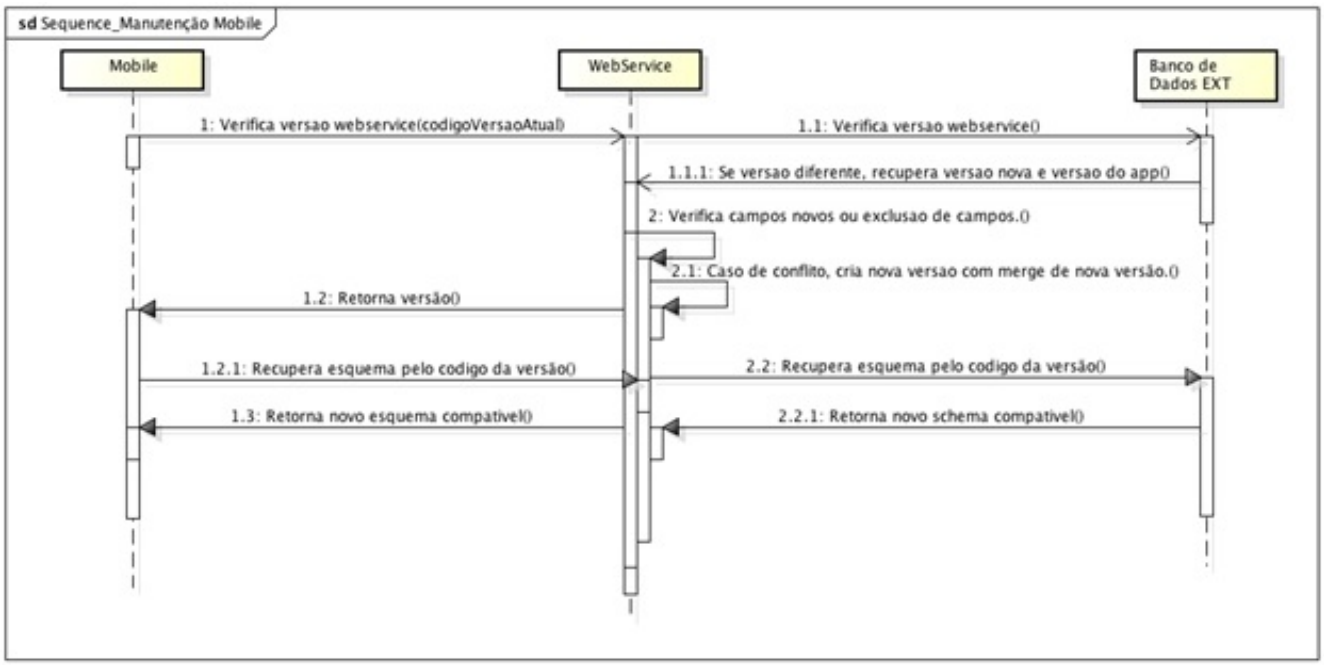


Figura 4 – Diagrama de sequência da aplicação Android

Exemplificando o funcionamento, tem-se a seguinte situação na versão 1: Conforme ilustra a Figura 5, a tabela de configuração com o valor da versão atual, nesta tabela fica armazenado somente um registro, o que identifica a versão atual que a aplicação Android deve verificar.

| | pk_int_codigo_tb_configuracao [PK] integer | str_nome_configuracao character varying(255) | str_valor_configuracao character varying(255) |
|---|---|---|---|
| 1 | 1 | VERSAO_ATUAL | 1 |
| * | | | |

Figura 5 – Representação tabela de configuração (Fonte: próprio autor)

A Figura 6 apresenta a representação da tabela que armazena as versões. Nesta Figura está exemplificada a situação onde tem-se registrado duas versões, nela é contido o código para referência, a data que foi criada a versão e o usuário que criou.

| | pk_int_codigo_tb_versao [PK] integer | dtt_versao date | str_usuario_alteracao character varying(255) |
|---|---|---|---|
| 1 | 1 | 2015-06-22 | Thamires Tabacchi |
| 2 | 2 | 2015-06-30 | Jones Carvalho |
| * | | | |

Figura 6 – Representação tabela de versão

A figura 7 apresenta os valores da tabela de versão. Nesta tabela são armazenados todos os registros de nome de tabelas referenciando sua versão.

| | pk_int_codigo_tb_versao_tabelas [PK] integer | str_nome_tabela character varying(255) | fk_int_codigo_tb_versao integer |
|---|---|---|---|
| 1 | 1 | tb_pessoa | 1 |
| 2 | 2 | tb_venda | 1 |
| 3 | 3 | tb_venda_item | 1 |
| * | | | |

Figura 7 – Representação tabela de versão tabelas

A Figura 8 apresenta os campos de cada tabela com seu tipo específico e referencia a tabela "tb_versao_tabelas", nome "tb_versao_tabelas_colunas", responsável por armazenar os dados de referência da tabela versionada.

| | pk_int_versao_tabelas_colunas [PK] integer | dtt_alteracao_coluna date | str_nome_coluna character varying(255) | str_tipo_coluna character varying(255) | fk_int_codigo_tb_versao_tabelas integer |
|---|---|---|---|---|---|
| 1 | 1 | 2015-09-01 | vch_nome | varchar(250) | 1 |
| 2 | 2 | 2015-09-01 | vch_telefone | varchar(20) | 1 |
| 3 | 3 | 2015-09-01 | pk_int_codigo_pessoa | int | 1 |
| 4 | 4 | 2015-09-01 | vch_endereco | varchar(200) | 1 |
| 5 | 5 | 2015-09-01 | pk_int_codigo_venda | int | 2 |
| 6 | 6 | 2015-09-01 | dtt_venda | datetime | 2 |
| 7 | 7 | 2015-09-01 | fk_int_codigo_pessoa | int | 2 |
| 8 | 8 | 2015-09-01 | pk_int_codigo_venda_item | int | 3 |
| 9 | 9 | 2015-09-01 | fk_int_codigo_venda | int | 3 |
| 10 | 10 | 2015-09-01 | vch_produto | varchar(200) | 3 |
| 11 | 11 | 2015-09-01 | dec_valor | decimal | 3 |
| * | | | | | |

Figura 8 – Representação tabela de versão tabelas – versão 1 (Fonte: próprio autor)

Neste exemplo, a versão 1, está pronta para ser aplicada. Como pode ser observado na Figura 6, a versão 2 foi criada pelo usuário Jones Carvalho. Na Figura 9, pode ser observado a representação do esquema da versão 2 e que há uma tabela adicionada na versão dois. Na Figura 10 pode ser observado as outras alterações da nova versão

| | pk_int_codigo_tb_versao_tabelas [PK] integer | str_nome_tabela character varying(255) | fk_int_codigo_tb_versao integer |
|---|---|---|---|
| 1 | 1 | tb_pessoa | 1 |
| 2 | 2 | tb_venda | 1 |
| 3 | 3 | tb_venda_item | 1 |
| 4 | 4 | tb_pessoa | 2 |
| 5 | 5 | tb_venda | 2 |
| 6 | 6 | tb_venda_item | 2 |
| 7 | 7 | tb_produto | 2 |
| * | | | |

Figura 9 – Alterações na versão tabela – versão 2

| | pk_int_versao_tabelas_colunas [PK] integer | dtt_alteracao_coluna date | str_nome_coluna character varying(255) | str_tipo_coluna character varying(255) | fk_int_codigo_tb_versao_tabelas integer |
|---|---|---|---|---|---|
| 1 | 12 | 2015-10-01 | vch_nome | varchar(250) | 4 |
| 2 | 13 | 2015-11-01 | vch_telefone | varchar(20) | 4 |
| 3 | 14 | 2015-11-01 | pk_int_codigo_pessoa | int | 4 |
| 4 | 15 | 2015-10-01 | vch_endereco | varchar(200) | 4 |
| 5 | 16 | 2015-10-01 | pk_int_codigo_venda | int | 5 |
| 6 | 17 | 2015-10-01 | dtt_venda | datetime | 5 |
| 7 | 18 | 2015-10-01 | fk_int_codigo_pessoa | int | 5 |
| 8 | 19 | 2015-10-01 | pk_int_codigo_venda_item | int | 6 |
| 9 | 20 | 2015-10-01 | fk_int_codigo_venda | int | 6 |
| 10 | 21 | 2015-10-01 | bit_cpf | boolean | 5 |
| 11 | 22 | 2015-10-01 | cpf | varchar(11) | 4 |
| 12 | 23 | 2015-10-01 | pk_int_codigo_produto | int | 7 |
| 13 | 24 | 2015-10-01 | dec_desconto_promocao | decimal | 7 |
| 14 | 25 | 2015-10-01 | vch_produto | varchar(200) | 7 |
| 15 | 26 | 2015-10-01 | dec_valor | decimal | 7 |
| 16 | 27 | 2015-10-01 | fk_int_codigo_produto | decimal | 6 |
| * | | | | | |

Figura 10 – Alterações na versão – versão 2

Itens Movidos
Itens Adicionados

Como pode ser notado, foi adicionada uma tabela de Produto, foram movidos os campos referentes a produto para a tabela, adicionado o campo CPF na tabela de Pessoa, e uma sinalização de CPF, caso tenha CPF na Venda. Caso algum dado já tenha sido informado e utilizado no banco de dados móvel, o Version Base, solicita uma confirmação do administrador para gerar uma nova versão com um merge entre os campos já utilizados, como representado na Figura 11.

| | pk_int_codigo [PK] integer | dtt_versao date | str_usuario_alteracao character varying(255) |
|---|---|---|---|
| 1 | 1 | 2015-06-22 | Thamires Tabacchi |
| 2 | 2 | 2015-06-30 | Jones Carvalho |
| 3 | 3 | 2015-10-03 | Usuario do sistema - Merge Versao |
| * | | | |

Figura 11 – representação de alterações na versão

Como pode ser notado na Figura 12, o sistema Version Base, gerou o merge dos campos das tabelas representadas na Figura 13.

| | pk_int_versao_tabelas_colunas [PK] integer | dtt_alteracao_coluna date | str_nome_coluna character varying(255) | str_tipo_coluna character varying(255) | fk_int_codigo_tb_versao_tabelas integer |
|---|---|---|---|---|---|
| 1 | 28 | 2015-10-01 | vch_nome | varchar(250) | 8 |
| 2 | 29 | 2015-11-01 | vch_telefone | varchar(20) | 8 |
| 3 | 30 | 2015-11-01 | pk_int_codigo_pessoa | int | 8 |
| 4 | 31 | 2015-10-01 | vch_endereco | varchar(200) | 8 |
| 5 | 32 | 2015-10-01 | pk_int_codigo_venda | int | 9 |
| 6 | 33 | 2015-10-01 | dtt_venda | datetime | 9 |
| 7 | 34 | 2015-10-01 | fk_int_codigo_pessoa | int | 9 |
| 8 | 35 | 2015-10-01 | pk_int_codigo_venda_item | int | 10 |
| 9 | 36 | 2015-10-01 | fk_int_codigo_venda | int | 10 |
| 10 | 37 | 2015-10-01 | bit_cpf | boolean | 5 |
| 11 | 38 | 2015-10-01 | cpf | varchar(11) | 8 |
| 12 | 39 | 2015-10-01 | pk_int_codigo_produto | int | 12 |
| 13 | 40 | 2015-10-01 | dec_desconto_promocao | decimal | 12 |
| 14 | 41 | 2015-10-01 | vch_produto | varchar(200) | 12 |
| 15 | 42 | 2015-10-01 | dec_valor | decimal | 12 |
| 16 | 43 | 2015-10-01 | fk_int_codigo_produto | decimal | 10 |
| 17 | 44 | 2015-09-01 | vch_produto | varchar(200) | 10 |
| 18 | 45 | 2015-09-01 | dec_valor | decimal | 10 |
| * | | | | | |

Figura 12 –Representação de alterações na versão

| | pk_int_codigo_tb_versao_tabelas [PK] integer | str_nome_tabela character varying(255) | fk_int_codigo_tb_versao integer |
|---|---|---|---|
| 1 | 8 | tb_pessoa | 3 |
| 2 | 9 | tb_venda | 3 |
| 3 | 10 | tb_venda_item | 3 |
| 4 | 11 | tb_pessoa | 3 |
| 5 | 12 | tb_produto | 3 |
| * | | | |

Figura 13 – Representação final das alterações com merge

## IV. Trabalhos Futuros e Considerações Finais

Pretende-se, como trabalho futuro, melhorar a Version Base, utilizando reflection do JAVA, para ser possível tornar dinâmicos os modelos de acesso às tabelas do banco de dados no aplicativo, eliminando a dependência em relação à aplicação Android. Com reflection, é possível permitir a criação de cadastros genéricos baseados no esquema da versão, deixando o sistema mais genérico, dependendo somente do layout do aplicativo. Assim o sistema modela dinamicamente as tabelas conforme o esquema da versão cadastrada. Mas o uso mais adequado é somente a cadastros básicos, e não a cadastros onde a orientação a objeto fica mais complexa, como por exemplo, quando a regra de um determinado cadastro é muito específica.

Originalmente, para o SQL Lite existia o plugin - o SQL Lite Asset, que se apresenta como um versionador. Porém, esse plugin somente faz uma carga de uma base já existente, facilitando a criação de uma base na primeira execução da aplicação. Existem outros versionadores de banco de dados como, por exemplo, o Liquibase (LIQUIBASE, 2006), porém não se aplica ao uso de banco de dados móveis.

O Version Base permite o acompanhamento das diversas versões de banco de dados. Nesta solução é possível atualizar, mas também permite restaurar versões anteriores, sem depender de publicações na central de aplicativos da plataforma do smartphone.

O sistema Version Base pode ser instalado em uma máquina com configuração básica somente exigindo acesso à internet. Tendo acesso à pagina web de manutenção do sistema é possível controlar diretamente as versões do aplicativo. Lembrando que o sistema gerencia o banco de dados existente dentro do aplicativo Android, o mesmo necessita de uma conexão de rede com o servidor, seja por internet ou rede local, para recuperar as informações necessárias para aplicação das versões.

O SQL Lite Asset apenas ajuda a implantar o banco de dados. A manutenção pode ser feita através do Version Base pela página web, deixando assim todo banco do aplicativo sempre sincronizado com a versão atualizada no sistema Version Base.

## Referências


ANDROID , História do Android, Disponível em: < https://www.android.com/intl/pt-BR_br/history/>. Acesso em 16/09/2015.

GOOGLE ANDROID, Aprenda a criar aplicações para dispositivos móveis com o Android SDK , Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=NrVUAwAAQBAJ&oi=fnd&pg=PA21&dq=Android&ots=QaPADcRlZt&sig=N639QQgxqV162cHZEXtxU4mORN4#v=onepage&q=Android&f=false>. Acesso em 16/09/2015.

LIQUIBASE, 2006, “Documentation Major Concepts”, Disponível em:
<http://www.liquibase.org/documentation/>. Acesso em 28/09/2015.

OLIVEIRA, L. E. Estado da arte de banco de dados orientados a documento, 2009. Monografia (Conclusão do Curso de Graduação em Ciências Tecnológicas) – Universidade de Fortaleza –UNIFOR, Ceára, 2009.

SILVA, G. K. de C.; PEREIRA, P. M.; MAGALHÃES, G.. Disponibilização de Serviços Baseados em Localização via Web Services. In: Simpósio Brasileiro de GeoInformática, GeoInfo. 2004.

SQL LITE ASSET, An Android helper class to manage database creation and version management using an application's raw asset files, Disponível em: <https://github.com/jgilfelt/android-sqlite-asset-helper>. Acesso em 11/08/2015

SOAP, 2003, “Simple Object Access Protocol” (27 de abril, 2007); Disponível em
<http://www.w3.org/TR/soap12>. Acesso em 14/09/2015.

SVNBOOK, Controle de Versão com Subversion, Disponível em: <http://svnbook.red-bean.com/en/1.7/svn.intro.whatis.html>, Acesso em: 11/08/2015