UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ESPECIALIZAÇÃO EM BANCO DE DADOS

LUÍS FERNANDO GIACOMINI DUTRA

APLICANDO O CONCEITO DE BANCO DE DADOS RELACIONAL E TEMPORAL NO APLICATIVO CALENDAR TATOO

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2017

LUÍS FERNANDO GIACOMINI DUTRA

APLICANDO O CONCEITO DE BANCO DE DADOS TEMPORAIS NO APLICATIVO CALENDAR TATOO.

Trabalho de Conclusão de Curso, apresentado ao II Curso de Especialização em Banco de Dados, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof. Fabiano Carniel.

PATO BRANCO
2017

TERMO DE APROVAÇÃO

APLICANDO O CONCEITO DE BANCO DE DADOS RENACIONAL E TEMPORAL NO APLICATIVO CALENDAR TATOO

por

LUIS FERNANDO GIACOMINI DUTRA

Este Trabalho de Conclusão de Curso foi apresentado em 06 de abril de 2017 como requisito parcial para a obtenção do título de Especialista em Banco de Dados. O(a) candidato(a) foi arguido(a) pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Fabiano Carniel
Prof.(a) Orientador(a)

_____
Beatriz Terezinha Borsoi
Membro titular

_____
Eden Ricardo Dosciatti
Membro titular

"O Termo de Aprovação assinado encontra-se na Coordenação do Curso"

RESUMO

DUTRA, Luís Fernando Giacomini. Aplicando o conceito de banco de dados temporais no aplicativo Calendar Tattoo. 2017. 41 f. Monografia (II Curso de Especialização em Banco de Dados) - Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

Com a evolução dos aparelhos móveis, a utilização de aplicativos e a conexão em rede têm aumentado de maneira expressiva, gerando grande quantidade de dados que tem desafiado os limites dos servidores disponíveis. Essa quantidade de dados disponíveis e manipulados, faz com que, a organização passe a ser essencial na vida de quem utiliza serviços disponibilizados de maneira digital. Considerando esse cenário, verificou-se a possibilidade de realizar um estudo sobre bancos de dados temporais juntamente com banco de dados relacionais. Para a realização do estudo foi utilizado o aplicativo Calendar Tatoo, desenvolvido para Android com o banco de dados MySQL, que gerencia o processo relacional. Optou-se pela ferramenta de banco de dados Oracle para implementação do processo pelos recursos que o mesmo possui para dados temporais. Neste trabalho será apresentado o desenvolvimento dos dois tipos de banco de dados utilizados.

Palavras-chave: Aplicativo. Banco de dados. Android. Dados temporais. MySQL. Oracle.

ABSTRACT

DUTRA, Luís Fernando Giacomini. Applying the concept of temporal databases to the Calendar Tattoo system. 2017. 41 f. Monography (II Specialization Course in Database) - Federal University of Technology - Parana. Pato Branco, 2017.

With the evolution of mobile devices, the use of applications and networking have increased significantly, generating large amounts of data that has challenged the limits of available servers. This amount of data available and manipulated, makes the organization become essential in the lives of those who use services available in a digital way. Considering this scenario, it was possible to carry out a study on temporal databases together with relational databases. For the accomplishment of the study was used the application Calendar Tatoo, developed for Android with the database MySQL, that manages the relational process. The Oracle database tool was used to implement the process for the resources it has for temporal data. In this work the development of the two types of database will be presented.

Palavras-chave: Software. Database. Android. Temporal data. MySQL. Oracle.

LISTA DE SIGLAS

BD	Banco de Dados
BDT	Banco de Dados Temporal
BDR	Banco de Dados Relacional
SGDB	Sistema Gerenciador de Banco de Dados
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
XML	eXtensible Markup Language

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações inicias, com uma visão geral do trabalho, os objetivos, a sua justificativa e a organização do texto.

1.1 CONSIDERAÇÕES INICIAIS

Bancos de dados podem armazenar grandes quantidades de dados e das mais diversas áreas, como comércio, dados bancários, imagens, dados georrefenciados, entre outros. É comum que os bancos de dados utilizados sejam do tipo convencional. Neles a informação que importa é sempre a mais atual, a última inserida no sistema, sobrescrevendo qualquer informação que possuía, descartando dados de valor relevante para o histórico do desempenho e desenvolvimento da empresa. Esse modelo, então, possui problemas como perda do histórico de dados armazenados.

Para esse tipo de problema a solução que se encontra é a utilização de Banco de Dados Temporais (BDT). BDT está relacionado ao tempo, oferecem a possibilidade de armazenar informações históricas a respeito de um determinado objeto que nele está sendo mantido (EDELWEISS, 1998, TANSEL, 1997).

Com o intuito de melhorar a troca de informações entre estúdios de tatuagens e seus clientes existe um aplicativo, Calendar Tattoo, para que os funcionários do estúdio possam controlar horários, agendamentos de clientes, criar sua própria galeria de imagens, e acessar um chat, onde podem conversar diretamente com o cliente.

O aplicativo ajuda a reduzir a lentidão dos processos de agendamento de sessões, também diminui as possibilidades de erros, as dificuldades de comparação de dados e a quantidade de papel.

O aplicativo está dividido em partes, as quais são agenda, histórico de sessões, galeria, configurações e comunicação entre os interessados realizados por meio de um *chat* interno no aplicativo.

Para controlar todas essas informações, será apresentado o desenvolvimento referente aos dois bancos de dados utilizados. O MySQL com o conceito relacional e o ORACLE com o conceito temporal. Também será apresentada a conexão realizada entre o aplicativo e os bancos de dados.

1.2 OBJETIVOS

O objetivo geral apresenta o resultado principal do trabalho realizado e os objetivos específicos o complementam, no sentido de valores agregados.

1.2.1 Objetivo Geral

O objetivo deste trabalho é implementar o conceito dos bancos de dados temporais e relacionais no aplicativo Calendar Tattoo, para ter um controle adequado quanto às informações nele inseridas.

1.2.2 Objetivos Específicos

- Desenvolver dois modelos de banco de dados para armazenar o histórico de agendamentos, informações de clientes e *chats* no modelo relacional e no banco de dados temporal ficará salva a relação entre estúdios, artistas, clientes e sessões.
- Obter os benefícios que uma organização das informações pode trazer.
- Possuir uma evolução dos dados por meio da organização proposta com os bancos de dados temporais, como histórico de agendamentos, ausências nas sessões e também definir qual período de maior ocorrência de agendamentos.

1.3 JUSTIFICATIVA

Atualmente utilizar um banco de dados temporal se faz necessário, com a quantidade de informações que circulam e também com a velocidade com a qual se alteram. Apesar de necessitar mais espaço de armazenamento, pois ao contrário do banco de dados relacional que sobrescreve os dados, o banco de dados temporal armazena todos os dados. Contudo, os benefícios são maiores, levando em conta a consistência da informação, sendo possível recuperar dados históricos e compará-los no tempo.

Com a utilização dessa ferramenta é possível ampliar a capacidade do aplicativo e também utilizar como *marketing*, pois poucos aplicativos se aproveitam disso.

Com base nesse estudo pretende-se desenvolver uma aplicação mais confiável, mais organizada e acima de tudo que forneça as informações que os usuários necessitam.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos, dos quais este é o primeiro e apresenta a ideia e o contexto do banco de dados, incluindo os objetivos e a justificativa.

O Capítulo 2 contém o referencial teórico que fundamenta a proposta conceitual do banco de dados temporal relacionado ao aplicativo escolhido para o estudo. O referencial teórico está centrado em bancos de dados temporais.

No Capítulo 3 estão os materiais e o método utilizados no desenvolvimento deste trabalho, incluindo a elaboração da monografia e a modelagem e implementação do banco de dados temporal para o aplicativo.

O Capítulo 4 contém o banco de dados temporal, com exemplos de documentação da modelagem e de implementação. A modelagem é exemplificada por documentos de análise e projeto. A implementação é exemplificada pela apresentação do banco com telas e descrição de suas funcionalidades e ainda por partes das tabelas do banco.

No Capítulo 5 está a conclusão com as considerações finais.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico utilizado para fundamentar o banco de dados proposto. O texto está focado em bancos de dados temporais.

2.1 BANCOS DE DADOS TEMPORAIS

Um sistema de banco de dados é um sistema que através do computador manipula e organiza registros, tornando-os disponíveis ao usuário (DATE, 1990, p. 2). Usuários podem realizar operações, como consultar, alterar, excluir apagar e inserir novos dados, em bancos de dados.

Dados históricos podem ter grande valor em tomadas de decisão. No entanto, os Sistemas Gerenciadores de Banco de Dados (SGBD) convencionais não possuem uma forma de trabalhar esses dados relacionados ao tempo. É possível trabalhar com o presente momento do dado inserido, por exemplo, caso seja alterado o endereço de um cliente, sempre será apresentada última alteração e ao acessar o dado será exibida essa versão final. As informações mais antigas ficam armazenadas em logs caso seja necessário recuperar.

Para que fosse possível guardar os dados passados, presentes e futuros no mesmo banco de dados seria necessário programar atributos adicionais para cada entidade.

Visando uma forma de atender essa necessidade de trabalhar com as informações ao longo do tempo surgiu o modelo de banco de dados temporais, que conseguem representar esse tipo de informação. Os bancos de dados temporais são diferentes, pois não armazenam apenas os dados, eles atrelam o tempo a eles.

2.2 CLASSIFICAÇÕES DE BANCOS DE DADOS TEMPORAIS

São adotas adotadas as seguintes classificações para os bancos de dados temporais: Instantâneos, de Tempo de Transação, de Tempo de Validade e Bitemporais.

2.2.1 Instantâneos

São os bancos de dados mais usados pelas empresas, são os bancos de dados convencionais utilizados comercialmente. Neles não é encontrada nenhuma forma de tratar os dados temporais nativamente, para se tratar esse tipo de informação se faz necessária a criação de atributos para definir as datas atreladas aos dados inseridos. Esse tipo de banco de dados trata apenas a informação no presente ou a informação associada algum tempo que seja tratado pelo aplicativo, sistema ou usuário. Ocorre que toda vez em que seja feita alguma alteração o registro perderá o que foi inserido antes, a informação passada. Exemplos de SGDBs instantâneos: SQL Server, PostGreSQL e MySQL.

A Figura 1 apresenta a ideia de funcionamento de bancos de dados temporais. Nesse caso os dados antigos do registro que foram alterados são perdidos.

Figura 1 - Exemplo de alteração em banco de dados instantâneos
Fonte: Desenvolvido pelo autor (2017).

Como apresentado na Figura 1, conforme forem ocorrendo alterações nos registros, os dados anteriores não são mantidos, eles se perdem e assim o último valor fica sendo o utilizado para consulta. Este tipo amplamente utilizado insere, altera e consulta os dados utilizando SQL do modo tradicional, sendo assim, para cada mudança feita no banco, os dados anteriores são sobrescritos.

2.2.2 Tempo de Transação

São bancos de dados que a cada dado inserido é vinculada uma data, utilizando o conceito de transações no banco e assim é adicionado como rótulo temporal. A cada inserção e atualização esse controle da data é feito pelo próprio SGDB. Esse tipo de banco de dados não trata apenas a informação presente, não é mais substituída pela última informação inserida, a cada alteração é adicionada a data em que foi realizada a operação.

Com esse tipo de controle são mantidos históricos das informações e assim o usuário tem mais controle sobre todas as operações efetuadas, podendo filtrar por determinada data para visualizar a informação que estava vigente naquele período.

Como apresentado na Figura 2, conforme forem ocorrendo alterações nos registros, os dados anteriores são mantidos e, assim, o usuário pode realizar consultas baseadas no tempo de transação, muito utilizado para fazer comparações entre períodos de datas.

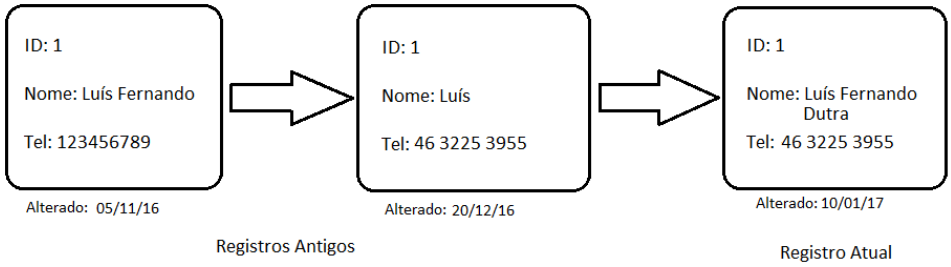

Figura 2 - Exemplo de alteração em banco de dados de tempo de transação
Fonte: Desenvolvido pelo autor (2017).

Hübler (2000) define o comportamento deste tipo de banco de dados para adicionar e alterar os dados. Adiciona-se ao registro o ponto no tempo em que o registro foi inserido/atualizado na tabela. Desse modo, ao se efetuar uma busca nos registros utilizando uma data, pode-se obter a informação válida na data informada.

2.2.3 Tempo de Validade

São bancos de dados que se baseiam na ideia do tempo de transação, associando uma data de alteração, mas o diferencial nesse caso é que ao realizar a operação é adicionada uma data de validade para o registro. Essa data de validade fica a critério do usuário, ele pode informar a data que desejar, todo controle da validade fica por conta do usuário.

Dessa forma é possível acessar as informações que foram inseridas no passado, tornando possível a sua correção em caso de ter sido inserida incorretamente, ficando disponível apenas a informação atual. Essa data inserida não significa que ela está atrelada ao momento em que o registro foi adicionado ou alterado.

Como apresentado na Figura 3, conforme forem ocorrendo adições e alterações nos registros, os dados recebem uma data de validade, atribuindo assim ao usuário a possibilidade de definir a partir de qual momento aquela informação se tornará válida e assim o usuário pode realizar consultas baseadas no tempo de validade.

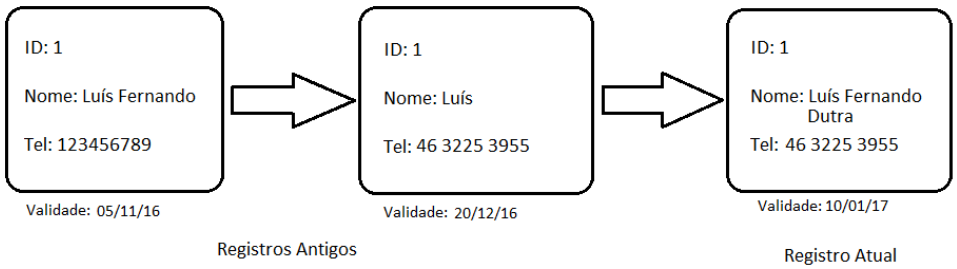

Figura 3 - Exemplo de alteração em banco de dados de tempo de validade
Fonte: Desenvolvido pelo autor (2017).

Hübler (2000) define o comportamento deste tipo de banco de dados durante a adição dos registros. Neste tipo de BD não é armazenado o instante de tempo em que é efetuada a inserção. Essa operação cria uma coluna na tabela para informar a data a partir da qual o registro torna-se válido.

O que ocorre nesse tipo de banco de dados é que se torna possível incluir informações antigas. Por exemplo, pode se inserir um valor com a validade sendo 01/10/16, esse registro passa a ser o mais antigo e passa a ser mostrado no começo da lista, mantendo a sequência da data de validade.

2.2.4 Bitemporais

São bancos de dados que se baseiam no conceito de transação e tempo de validade. Desse modo, é possível armazenar a data de validade e também a data em que a transação ocorreu, assim fica mais ampla a possibilidade de acesso ao histórico de modificações. Além de ter a possibilidade de filtrar pela data de validade fica aberta a possibilidade de filtrar pela data da transação, ou até mesmo utilizar as duas para buscar as informações desejadas. Nesse modelo é possível verificar as informações que eram tratadas como válidas, mas acabaram sendo modificadas.

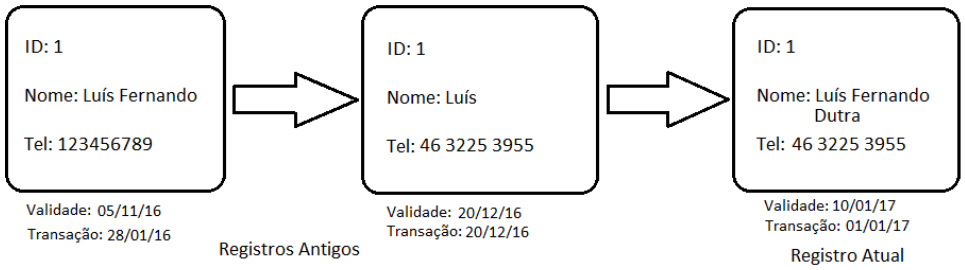

Figura 4 - Exemplo de alteração em banco de dados Bitemporal
Fonte: Desenvolvido pelo autor (2017).

Como apresentado na Figura 4, conforme forem ocorrendo alterações nos registros, os dados anteriores são mantidos e assim o usuário pode realizar consultas baseadas no tempo de transação, muito utilizado para fazer comparações entre períodos de datas.

2.3 BANCOS DE DADOS RELACIONAIS

Um banco de dados relacional é um mecanismo de armazenamento que permite a persistência de dados e opcionalmente programar funcionalidades.

São usados para armazenar a informação requerida por aplicações construídas usando tecnologias orientadas a objeto.

Também pode ser controlado por um SGDB e trabalham da mesma forma que um BDT, mas os relacionais não possuem uma forma de trabalhar esses dados relacionados ao tempo que seja nativa.

Um banco de dados relacional é uma coleção de dados com relacionamentos predefinidos entre si. Esses itens de dados são organizados na forma de tabelas definidas, com colunas e linhas. Cada linha da tabela representa uma coleção de valores relacionados de um objeto ou entidade. As tabelas são usadas para reter informações sobre os objetos a serem representados no banco de dados. Cada coluna da tabela retém um determinado tipo de dado e um campo armazena o valor em si de um atributo. A linha em cada tabela poderia ser identificada com uma chave única identificando e linha entre várias tabelas pode ser relacionada usando chaves estrangeiras. Esses dados podem ser acessados de formas diferentes, sem reorganizar as tabelas do banco de dados.

Os Bancos de Dados Relacionais foram desenvolvidos para prover acesso facilitado aos dados, possibilitando que os usuários utilizassem uma grande variedade de abordagens no

tratamento das informações. A linguagem padrão dos Bancos de Dados Relacionais é a *Structured Query Language*, ou simplesmente SQL, como é mais conhecida.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e método utilizados no desenvolvimento do aplicativo. Os materiais fazem referência às tecnologias utilizadas para chegar ao objetivo final. O método contém a evolução dos procedimentos utilizados para desenvolver o BDT.

3.1 FERRAMENTAS E TECNOLOGIAS

O Quadro 1 apresenta as ferramentas e as tecnologias utilizadas no desenvolvimento do trabalho.

NOME	VERSÃO	LOCALIZAÇÃO	DESCRICÃO DE USO
Android Studio	3.4.1	https://developer.android.com/studio/index.html	IDE para desenvolvimento Android utilizando linguagem Java.
MySQL	5.7	http://www.mysql.com/	Sistema Gerenciador de Banco de Dados.
DB Designer 4	5.0	http://www.fabforce.net	Utilizado para desenvolver a Modelagem do Banco de Dados.
BrModelo	2.0	http://www.sis4.com/brModelo/	Utilizado para a Criação de modelos conceituais e lógicos de banco de dados.
Oracle	12.1.0.2.0	https://www.oracle.com/br/index.html	Sistema Gerenciador de Banco de Dados Temporais.
MySQL Workbench	6.3	www.mysql.com/products/workbench/	Interface gráfica para administrar e trabalhar com o banco de dados MySQL.
Microsoft Visual Studio	2015	https://www.visualstudio.com/pt-br/	IDE para desenvolvimento do *webservice* utilizando linguagem .net.

Quadro 1 - Ferramentas e tecnologias utilizadas

3.2 PROCEDIMENTOS PARA MODELAGEM E IMPLEMENTAÇÃO

Os procedimentos para definir o banco de dados estão baseados nas fases propostas por Pressman (2005) para o modelo sequencial linear que são análise, projeto, codificação e testes. Outras fases e mesmo denominações foram definidas visando explicitar mais adequadamente os procedimentos adotados para realizar o trabalho. A seguir, as fases definidas para este trabalho:

a) Levantamento de requisitos – o levantamento de requisitos está centrado na definição do problema que o banco de dados a ser desenvolvido visa resolver. Para definir o problema foi realizada análise do aplicativo Calendar Tattoo utilizado no trabalho, conversas

informais com futuros usuários, os quais são tatuadores e detalharam suas necessidades. O levantamento dos requisitos iniciou com conversas com usuário contendo os requisitos que ele considera necessários para o aplicativo, definindo a ideia geral e enfatizando funcionalidades e aspectos de qualidade relevantes.

Desta conversa foram definidos os requisitos funcionais e não funcionais e complementares. Também foi elaborado um esboço inicial dos dados necessários para o banco de dados.

b) Análise – a partir dos requisitos, o problema (o banco de dados) é modelado. Na análise foram utilizados conceitos de bancos de dados temporais e também de bando de dados relacionais para organizar as tabelas e seus dados. Os diagramas de casos de uso com a descrição formalizada dos requisitos foram elaborados.

c) Projeto – nesta fase foram elaborados diagramas de classes, de sequência e de arquitetura. Os componentes e as classes que compõem o sistema serão distribuídos em cada uma das camadas da arquitetura, definindo um diagrama de arquitetura. Também foram definidos os recursos mais relevantes da linguagem para implementar os requisitos principais do sistema, que seria a agenda.

d) Implementação (codificação) do banco de dados – define como o banco de dados será codificado, a forma de uso das linguagens e tecnologias para programar o Banco de Dados Temporário-BDT e o Banco de Dados Relacional-BDR. Esses componentes serão distribuídos pelas respectivas camadas da arquitetura. Procedimentos armazenados (*stored procedures*) no banco de dados serão criados para programar as funções específicas do banco de dados.

e) Realização dos testes – os testes serão realizados com base no uso diário de um estúdio parceiro ao desenvolvimento do aplicativo. Os testes de usuário serão realizados após a finalização da implementação dos bancos. Isso porque o banco de dados será implementado em uma única iteração juntamente com o aplicativo utilizado.

4 RESULTADOS E DISCUSSÃO

Este capítulo apresenta o aplicativo utilizado e também apresenta os bancos de dados que foram desenvolvidos como resultado deste trabalho. Em um primeiro momento está a modelagem dos bancos de dados, após essa seção se encontra a criação dos mesmos. A modelagem apresenta as tabelas desenvolvidas como resultado da análise e do projeto. Por fim, estão exemplos da criação dos bancos de dados.

4.1 APRESENTAÇÃO

O aplicativo utilizado para o desenvolvimento deste trabalho é o Calendar Tatto, que foi desenvolvido para o gerenciamento de agendamentos de estúdios de tatuagens e *body piercings* e é composto por um cadastro do cliente, lista dos estúdios cadastrados, lista de artistas disponíveis por estúdio, galeria de imagens para cada artista, e por fim a agenda do artista selecionado, sendo este último item acessível ao cliente e ao profissional, porém os nomes e informações dos clientes ficam visíveis apenas ao profissional dono da agenda.

Cada artista pode definir como configuração o horário de atendimento, valor por sessão, além de gerenciar suas sessões e divulgar seus trabalhos. Primeiro será apresentando o Banco de Dados Relacional (BDR), onde ficam salvas as informações que são apresentadas aos clientes. Após o BDR será apresentado o BDT, onde ficam salvas as informações que são apresentadas aos funcionários dos estúdios para auxiliarem na tomada de decisões. Por fim será apresentada a forma como é feita a conexão e troca de informações entre aplicativo, *web service* e bancos de dados.

4.2 MODELAGEM E DESCRIÇÃO

Para exemplificar a implementação de um banco de dados relacional foi realizada a modelagem de um banco de dados onde ficam salvas as informações apresentadas.

Esse banco de dados foi desenvolvido em MySQL e possuirá a função de armazenar os dados dos clientes, artistas, estúdios, galeria e a agenda de cada profissional.

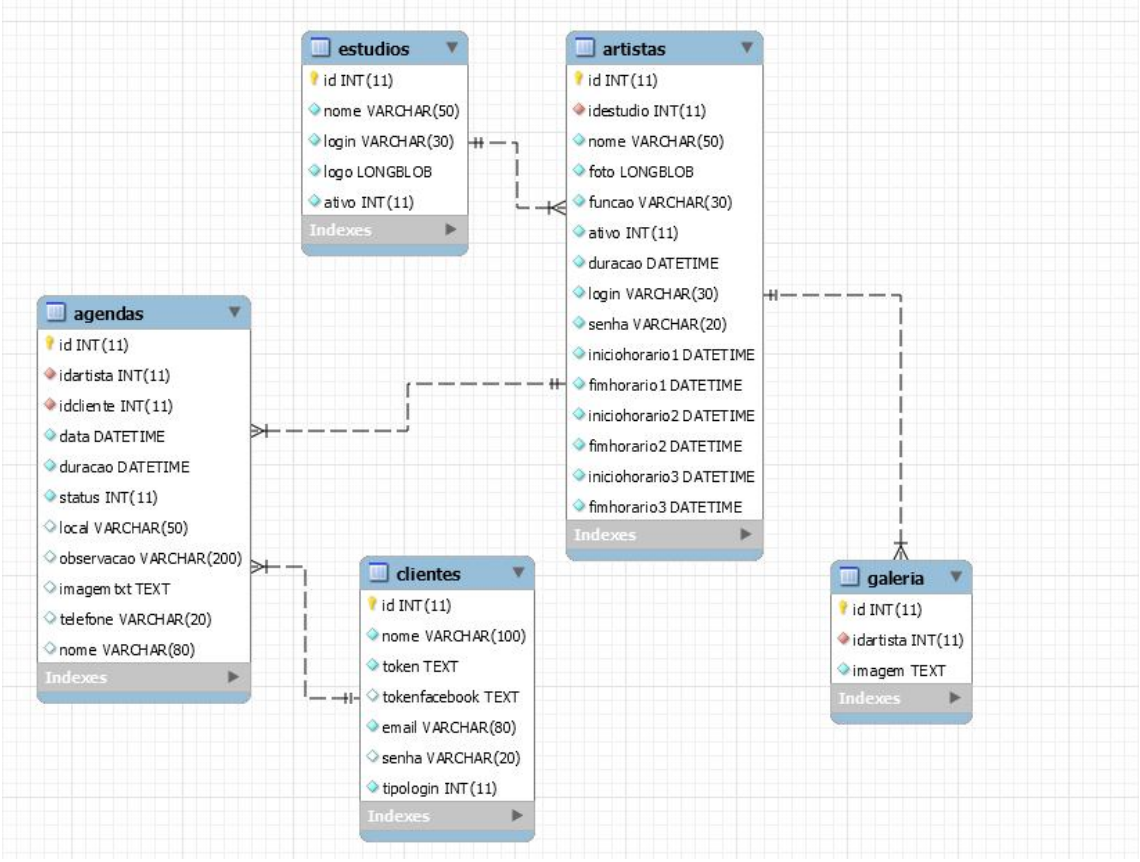

Figura 5 - Modelo proposto para utilização de banco de dados relacional
Fonte: Desenvolvido pelo autor (2017).

Na Figura 5 é apresentado o modelo proposto e desenvolvido para armazenar as informações necessárias para que o aplicativo Calendar Tattoo possa controlar as necessidades de um estúdio de tatuagens.

Para exemplificar a implementação de um banco de dados temporal foi realizada a modelagem de um banco de dados no qual ficam salvas as informações que serão utilizadas para auxilio na tomada de decisões, como verificar a evolução do valor das sessões, por parte dos estúdios de tatuagens.

Os dados salvos nesse banco de dados serão o valor da sessão e a identificação do artista, para que assim seja possível verificar a alteração de valores cobrados pelos artistas ao longo do tempo.

Almeja-se que seja possível calcular o valor da sessão do artista dado um determinado período de tempo, seja ele o mês ou o ano que o usuário desejar. Para que isso seja possível dentro da aplicação o artista pode alterar o valor de suas sessões como configuração. Esse

valor será atrelado a cada sessão que seja agendada e será salva no banco de dados MySQL, após feito esse processo será realizado uma inserção no banco de dados temporal Oracle com os dados relevantes como identificação da sessão, artista e o valor.

Seria necessário armazenar e manipular esses valores nas tabelas do banco de dados. Trabalhando com os conceitos de bancos de dados temporais e os recursos disponíveis atualmente o próprio SGBD toma conta dessa gerência dos dados.

Como é trabalhado com a questão de tempo sempre será armazenada no banco de dados temporal alguma data que seja relevante, será trabalhado com dia, mês e ano.

A Figura 5 apresenta o modelo do banco de dados a ser implementado para o banco de dados temporal.

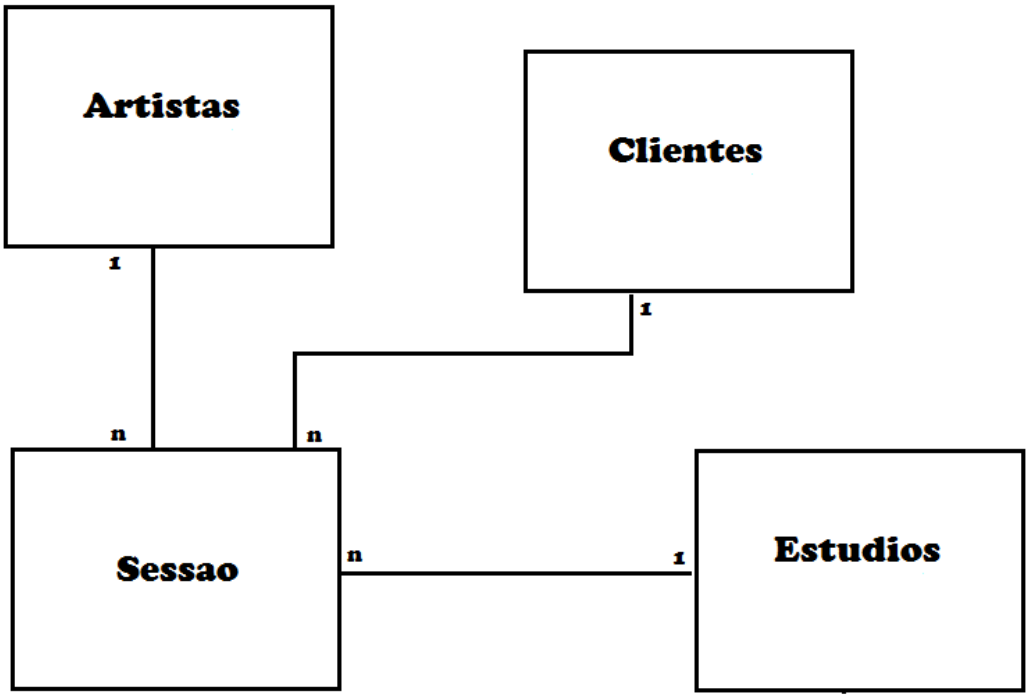

Figura 6 - Modelo proposto para utilização de banco de dados temporal

Fonte: Desenvolvido pelo autor (2017).

Como proposto no estudo será necessário à utilização de quatro tabelas: *ARTISTAS*, *ESTUDIOS*, *CLIENTES e SESSAO*. As tabelas e os seus atributos são apresentados a seguir.

ARTISTAS	
IDARTISTA	INTEGER
NOMEARTISTA	NVARCHAR2(80)

Tabela 1 - Modelo proposto para tabela Artistas

Fonte: Desenvolvido pelo autor (2017).

A Tabela 1 tem IDARTISTA como chave primária e NOMEARTISTA como campos para que seja possível identificar o artista.

ESTUDIOS	
IDESTUDIO	INTEGER
NOMEESTUDIO	NVARCHAR2(100)

Tabela 2 - Modelo proposto para tabela Estudios

Fonte: Desenvolvido pelo autor (2017).

A Tabela 2 tem IDESTUDIO como chave primária e NOMESTUDIO como campos para que seja possível identificar o estúdio**.**

CLIENTES	
IDCLIENTE	INTEGER
NOMECLIENTE	NVARCHAR2(100)

Tabela 3 - Modelo proposto para tabela Clientes

Fonte: Desenvolvido pelo autor (2017).

A Tabela 3 tem IDCLIENTE como chave primária e NOMECLIENTE como campos para que seja possível identificar o cliente**.**

SESSAO	
IDSESSAO	INTEGER
IDESTUDIO	INTEGER
IDARTISTA	INTEGER
IDCLIENTE	INTEGER
VALORSESSAO	DOUBLE

Tabela 4 - Modelo proposto para tabela Sessao

Fonte: Desenvolvido pelo autor (2017).

A Tabela 4 tem IDSESSAO como chave primária. Os campos IDESTUDIO, IDARTISTA e IDCLIENTE como chave estrangeira, para que seja possível recuperar a informação referente a filtros que possam ser realizados. Por fim o campo VALORSESSAO que armazena o valor que foi cobrado pela sessão na data em que foi realizada, para que assim possa ser feito comparativos entre valores de sessão.

4.3 IMPLEMENTAÇÃO

Depois das tabelas criadas no MySQL e no Oracle é necessário acesso as informações dos bancos de dados. Para tornar possível que o aplicativo tenha acesso as informações disponíveis no banco de dados será utilizado um *web service* desenvolvido com a ferramenta Visual Studio 2015 com o protocolo *Simple Object Access Protocol* (SOAP).

SOAP é um protocolo baseado em XML para troca de informações em um ambiente distribuído. É utilizado para troca de mensagens entre aplicativos distribuídos pela rede.

Esses aplicativos, ou *web services*, possuem uma interface de acesso simples e bem definida. Os *web services* são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, o formato XML. O código da função para retornar a galeria do artista é apresentado na Figura 7.

```
<WebMethod(EnableSession:=True)>
Public Function RetGaleria(ByVal sSenha As String, ByVal idArtista As String) As String

    If sSenha = "wstatto!@#060295" Then
        Dim dtImagens As DataTable = ExecutaDataTable("Select id, imagem From Galeria where idartista=" & idArtista, "Tattoo")

        If dtImagens.Rows.Count > 0 Then
            Dim ds = New DataSet()
            ds.Tables.Add(dtImagens)

            Dim swriter As New StringWriter()
            ds.WriteXml(swriter, XmlWriteMode.IgnoreSchema)
            Dim xml As String = swriter.ToString()
            xml = xml.Replace("<NewDataSet>", "")
            xml = xml.Replace("</NewDataSet>", "")

            Return "1|" & xml 'Retorna Galeria
        Else
            Return "2| Não possui imagens cadastradas." 'Não possui imagens cadastradas
        End If
    Else
        Return "0| Senha Incorreta"
    End If
End Function
```

Figura 7 - Função que retorna galeria do artista selecionado
Fonte: Desenvolvido pelo autor (2017).

A cada função é feita uma conexão com o banco de dados que é controlada pela extensão do MySQLClient e assim permite que sejam inseridas, excluídas e recuperas informações no banco de dados (Figura 8).

```
Public Shared Sub ConectarMySql(ByVal BancoDeDadosUs As String)

    If Not mConexao.State = ConnectionState.Open Then

        'Dim strConexao As String = "Data Source=" + ServidorUs + ";user id=" + LoginUs + ";password=" + SenhaUs + "; database=" + BancoDeDadosUs
        Dim strConexao As String = "Data Source=localhost;user id=root;password=r34lt0ur1405!@;Convert Zero Datetime=True;database=" + BancoDeDadosUs

        mConexao = New MySqlConnection()

        mConexao.ConnectionString = strConexao

        mConexao.Open()

    End If

End Sub
```

Figura 8 - Conexão com o banco de dados MySQL
Fonte: Desenvolvido pelo autor (2017).

Como se trata de um aplicativo, essa foi a melhor forma encontrada para que a aplicação não tome um espaço muito grande nos aparelhos dos usuários, assim o banco de dados fica hospedado num servidor na nuvem e fica acessível de qualquer local tendo conexão com a internet. A Figura 9 apresenta a tela principal do aplicativo.

Figura 9 - Tela inicial do aplicativo
Fonte: Desenvolvido pelo autor (2017).

O aplicativo está composto por duas seções, essa seção é definida pelo *login*. Caso seja um cliente acessando é direcionado para seção com as opções de agendar uma sessão ou verificar o histórico de sessões (Figura 10).

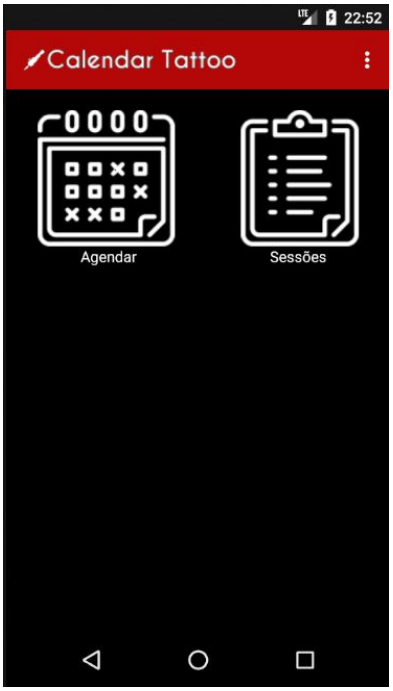

Figura 10 - Tela após login do cliente
Fonte: Desenvolvido pelo autor (2017).

Caso seja o profissional é direcionado para a seção que possui as opções de visualizar sua agenda ou verificar a sua galeria de imagens no aplicativo (Figura 11).

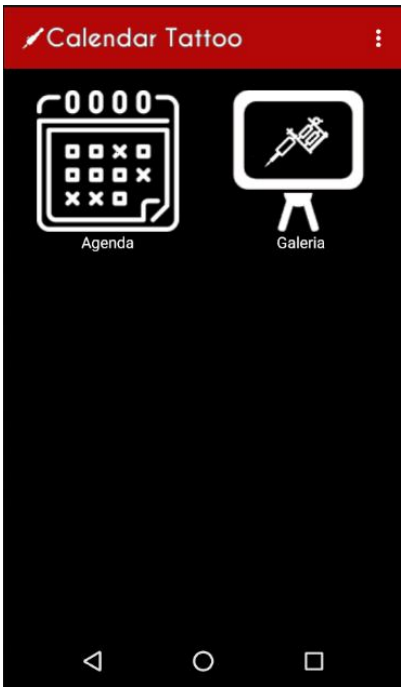

Figura 11 - Tela após login do artista
Fonte: Desenvolvido pelo autor (2017).

A Figura 12 apresenta os estúdios disponíveis para agendar uma sessão de tatuagem ou *body pircieng*. Esses estúdios são cadastrados no banco de dados relacional utilizado no MySQL por mejo de *web services*.

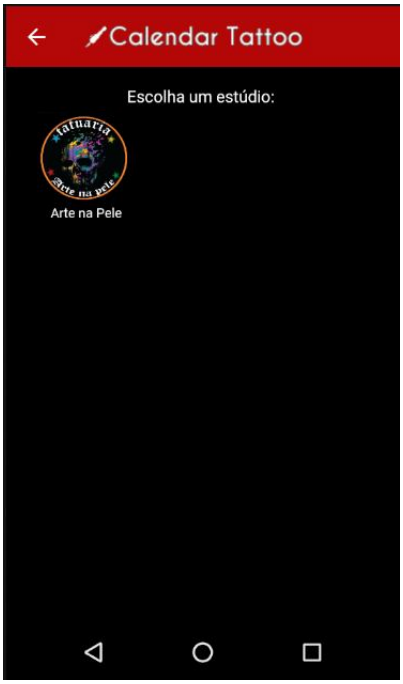

Figura 12 - Tela de seleção de estúdio
Fonte: Desenvolvido pelo autor (2017).

Na Figura 13 é apresentada a lista de artistas disponíveis para o estúdio selecionado na tela apresentada na Figura 12. Esses artistas são cadastrados no banco de dados convencional utilizado no MySQL por meio de *web services*. Nesse cadastrado é incluído as informações do artista e também qual a jornada de trabalho do mesmo.

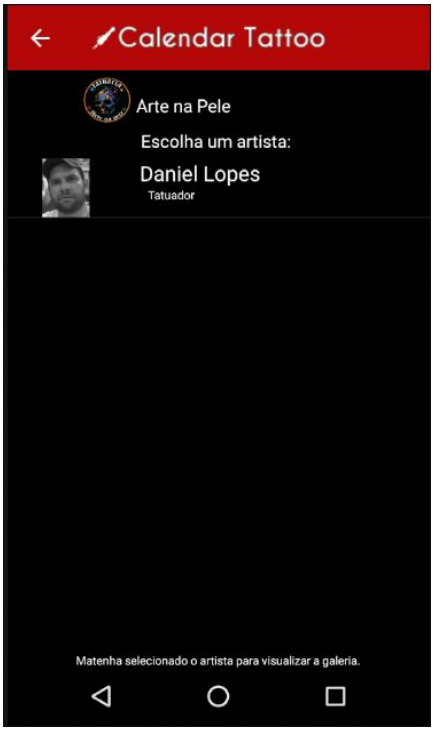

Figura 13 - Tela de seleção de artistas

Fonte: Desenvolvido pelo autor (2017).

Na Figura 14 é apresentada a tela de seleção de horários, ficando a critério do cliente escolher um horário disponível e agendar com o artista desejado.

Figura 14 - Tela de seleção de horários

Fonte: Desenvolvido pelo autor (2017).

Nesse momento em que o cliente realiza o agendamento serão inseridas as informações do cliente, estúdio, artista e sessão no banco de dados relacional e no temporal. Com esse processo será possível verificar essas informações para que seja realizada a comparação dos dados que é o objetivo utilizando o BDT.

Para exemplificar o processo realizado pela informação desde o momento que é inserida no aplicativo até o momento em que é salva no banco de dados será utilizado à tela de configurações por parte do artista (Figura 15).

Figura 15 - Tela de configuração do artistas
Fonte: Desenvolvido pelo autor (2017).

Na tela de configurações são apresentadas todas as informações do artista que acessou o aplicativo, informações como nome, função, *login*, senha, duração das sessões e turnos de trabalho.

Depois que o artista insere essas informações é exibida a mensagem verificando se o mesmo deseja salvar essas informações no banco de dados. Quando armazenada no banco de dados, a informação percorre um caminho, pois o aplicativo está no Android e é necessário que essa informação chegue até o banco de dados MySQL. Para isso é utilizada uma biblioteca chamado kSOAP. kSOAP foi criado originalmente para trabalhar com J2ME e teve um fork criado por Manfred Moser para trabalhar com Android. Primeiro, são passadas as

informações para a classe que utilizará o kSOAP, pois é necessário identificar qual método é necessário para que as informações sejam salvas corretamente.

```
private void salvaInfoArtista() {

    imgArtistaConf.buildDrawingCache();
    Bitmap bitmapToServer = imgArtistaConf.getDrawingCache();
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    bitmapToServer.compress(Bitmap.CompressFormat.PNG, 0, baos);
    byte[] bArray = baos.toByteArray();
    String encodedImage = Base64.encodeToString(bArray, 0);

    data = new String[15];
    data[0] = String.valueOf(idArtista);
    data[1] = txtNomeArtista.getText().toString();
    data[2] = txtLoginArtista.getText().toString();
    data[3] = txtSenhaArtista.getText().toString();
    data[4] = txtFuncaoArtista.getText().toString();
    data[5] = txtInicioTurno1Artista.getText().toString();
    data[6] = txtFimTurno1Artista.getText().toString();
    data[7] = txtInicioTurno2Artista.getText().toString();
    data[8] = txtFimTurno2Artista.getText().toString();
    data[9] = txtInicioTurno3Artista.getText().toString();
    data[10] = txtFimTurno3Artista.getText().toString();
    data[11] = txtDuracaoArtista.getText().toString();
    data[12] = encodedImage;

    if (spnSabado.isChecked()) {
        data[13] = "1";
    } else {
        data[13] = "0";
    }

    if (spnDomingo.isChecked()) {
        data[14] = "1";
    } else {
        data[14] = "0";
    }

    new WSConnect(data, "AtualizaArtista", new WSConnect.OnReturnServicePrimary() {
        @Override
        public void onCompletion(String response) {

            String resposta = response.substring(0, response.indexOf("|"));

            if (resposta.equals("1")) {
                onBackPressed();

                Toast.makeText(getBaseContext(), "Dados salvos!", Toast.LENGTH_SHORT)
.show();
            } else {
                Toast.makeText(getBaseContext(), "Não foi possivel salvar.Verifique os
dados.", Toast.LENGTH_SHORT).show();
            }
        }

        @Override
        public void onError() {
            Toast.makeText(getBaseContext(), "Ocorreu um erro.", Toast.LENGTH_SHORT)
.show();
        }
    }).execute();}
```

Figura 16 - Código da classe que chama a classe WSConnect

Fonte: Desenvolvido pelo autor (2017).

Com isso é chamada a classe de conexão com o *web service*, passando todas as informações inseridas na tela e buscando no *web service* o método que corresponde à ação que o usuário está realizando.

Na classe WSConnect (Figura 17) que será informado caso tenha ocorrido algum erro, como não conseguir conectar ao *web service* ou esteja faltando algum dado necessário na função do *web service*. Em caso de resposta são apresentadas as informações que o *web service* retorna ao fim do método escolhido para essa ação.

```
package com.example.proller.tattoocalendar.ws;

import android.os.AsyncTask;
import android.util.Log;

import org.ksoap2.SoapEnvelope;
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapPrimitive;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransportSE;

public class WSConnect extends AsyncTask<Void, Void, Boolean> {
    private String[] data;
    private String mResponse;
    private String mMethod;
    private OnReturnServicePrimary mListener;

    public WSConnect(String[] data, String mMethod, OnReturnServicePrimary mListener) {
        this.data = data;
        this.mListener = mListener;
        this.mMethod = mMethod;
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected Boolean doInBackground(Void... params) {
        final String METHOD_NAME = mMethod;
        final String NAMESPACE = "http://calendartattoo.com.br/";
        final String SOAP_ACTION = NAMESPACE + METHOD_NAME;
        final String URL =
"http://192.168.0.2/WebServiceRealApps/CalendarTattoo/WSCalendarTattoo.asmx"; //Incubadora

        try {
            SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);

            String sSenha = "wstatto!@#060295";

            if (mMethod.equals("RetEstudios")) {
                request.addProperty("sSenha", sSenha);
            } else if (mMethod.equals("RetArtistas")) {
                request.addProperty("sSenha", sSenha);
                request.addProperty("idEstudio", data[0]);
            } else if (mMethod.equals("RetConfAgenda")) {
                request.addProperty("sSenha", sSenha);
                request.addProperty("idArtista", data[0]);
                request.addProperty("dData", data[1]);
            } else if (mMethod.equals("IncluiAgendamento")) {
                request.addProperty("sSenha", sSenha);
                request.addProperty("idArtista", data[0]);
                request.addProperty("dData", data[1]);
                request.addProperty("idCliente", data[2]);
                request.addProperty("sLocal", data[3]);
                request.addProperty("sObs", data[4]);
                request.addProperty("sImagem", data[5]);
            } else if (mMethod.equals("RetSenhaArtista")) {
                request.addProperty("sSenha", sSenha);
                request.addProperty("sLogin", data[0]);
```

```
            request.addProperty("sSenhaArt", data[1]);
        } else if (mMethod.equals("IncluiCliente")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("sNome", data[0]);
            request.addProperty("sEmail", data[1]);
            request.addProperty("sSenhaUser", data[2]);
            request.addProperty("sTokenFace", data[3]);
            request.addProperty("iTipoLogin", data[4]);
            request.addProperty("sTokenGoogle", data[5]);
            request.addProperty("sFotoGoogle", data[6]);
        } else if (mMethod.equals("RetSessoesCliente")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idCliente", data[0]);
        } else if (mMethod.equals("RetSenhaCliente")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("sEmail", data[0]);
            request.addProperty("sSenhaClie", data[1]);
            request.addProperty("sTokenFace", data[2]);
            request.addProperty("sTokenGoogle", data[3]);
        } else if (mMethod.equals("RetGaleria")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idArtista", data[0]);
        } else if (mMethod.equals("RetConfArtista")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idArtista", data[0]);
        } else if (mMethod.equals("AtualizaArtista")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idArtista", data[0]);
            request.addProperty("NomeArtista", data[1]);
            request.addProperty("Login", data[2]);
            request.addProperty("Senha", data[3]);
            request.addProperty("FuncaoArtista", data[4]);
            request.addProperty("HoraInicio1", data[5]);
            request.addProperty("HoraFim1", data[6]);
            request.addProperty("HoraInicio2", data[7]);
            request.addProperty("HoraFim2", data[8]);
            request.addProperty("HoraInicio3", data[9]);
            request.addProperty("HoraFim3", data[10]);
            request.addProperty("Duracao", data[11]);
            request.addProperty("sFoto", data[12]);
            request.addProperty("sSabado", data[13]);
            request.addProperty("sDomingo", data[14]);
        } else if (mMethod.equals("RetInfoAgendamento")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idAgenda", data[0]);
        } else if (mMethod.equals("AtualizaAgendamento")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idAgenda", data[0]);
            request.addProperty("sDuracao", data[1]);
            request.addProperty("iStatus", data[2]);
            request.addProperty("dValor", data[3]);
        } else if (mMethod.equals("DeletaAgendamento")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idAgenda", data[0]);
        } else if (mMethod.equals("IncluiAgendamentoArtista")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idArtista", data[0]);
            request.addProperty("dData", data[1]);
            request.addProperty("sNome", data[2]);
            request.addProperty("sTelefone", data[3]);
            request.addProperty("sLocal", data[4]);
            request.addProperty("sObs", data[5]);
            request.addProperty("sImagem", data[6]);
        } else if (mMethod.equals("IncluiImagemGaleria")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idArtista", data[0]);
            request.addProperty("sImagem", data[1]);
        } else if (mMethod.equals("DeletaImagemGaleria")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idImagem", data[0]);
        } else if (mMethod.equals("RetConfCliente")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idCliente", data[0]);
        } else if (mMethod.equals("AtualizaCliente")) {
            request.addProperty("sSenha", sSenha);
            request.addProperty("idCliente", data[0]);
            request.addProperty("NomeCliente", data[1]);
```

```
                request.addProperty("Email", data[2]);
                request.addProperty("Senha", data[3]);
                request.addProperty("sFoto", data[4]);
            } else if (mMethod.equals("RetToken")) {
                request.addProperty("sSenha", sSenha);
                request.addProperty("idCliente", data[0]);
                request.addProperty("idArtista", data[1]);
                request.addProperty("sToken", data[2]);
            } else if (mMethod.equals("DeletaToken")) {
                request.addProperty("sSenha", sSenha);
                request.addProperty("idCliente", data[0]);
                request.addProperty("idArtista", data[1]);
                request.addProperty("sToken", data[2]);
            } else if (mMethod.equals("RetFeriados")) {
                request.addProperty("sSenha", sSenha);
                request.addProperty("sData", data[0]);
            } else if (mMethod.equals("RetFds")) {
                request.addProperty("sSenha", sSenha);
                request.addProperty("idArtista", data[0]);
            }

            SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
            envelope.dotNet = true;
            envelope.setOutputSoapObject(request);

            HttpTransportSE httpTransport = new HttpTransportSE(URL);
            httpTransport.call(SOAP_ACTION, envelope);

            SoapPrimitive response = (SoapPrimitive) envelope.getResponse();

            mResponse = response.toString();

            return true;
        } catch (Exception e) {
            e.printStackTrace();
            Log.e("ERRO", e.toString());
        }
        return false;
    }

    @Override
    protected void onPostExecute(Boolean sucess) {
        if (mListener != null) {
            if (sucess) mListener.onCompletion(mResponse);
            else mListener.onError();
        }
    }

    public interface OnReturnServicePrimary {
        public void onCompletion(String response);

        public void onError();
    }

}
```

Figura 17 - Código da classe que conecta com o web service

Fonte: Desenvolvido pelo autor (2017).

Feito esse processo que é realizado no Android Studio e este esteja correto com as informações repassadas, é chamado o *web service* que está vinculado ao Microsoft Visual Studio. Para que não ocorram erros o nome do método e dos atributos tem que ser iguais nas duas partes. A Figura 18 apresenta o código da função que salva as alterações das configurações do artista.

Figura 18 - Código da função que salva as alterações das configurações do artista

Fonte: Desenvolvido pelo autor (2017).

Feito a chamada no *web service* é realizada a verificação se possuem dados no banco de dados para que esses sejam atualizados conforme o artista deseja, esse processo utiliza a conexão demonstrada na Figura 7. Desse modo é trabalhado com o Banco de Dados Relacional. Por fim é utilizado o BDT e para isso ativa-se o versionamento para as tabelas *Sessao, Artistas, Clientes e Estudios*.

Quando o versionamento é habilitado através do procedimento EnableVersioning o workspace manager realiza automaticamente as operações de criar as estruturas de dados.

```
EXECUTE DBMS_WM.EnableVersioning('Sessao', 'VIEW_WO_OVERWRITE', FALSE, TRUE);
```

Listagem 1 – Script de versionamento

Fonte: Desenvolvido pelo autor (2017).

Antes de inserir ou atualizar qualquer dado na tabela versionada faz-se necessário executar um comando que especifica o tempo de validade dos valores que se deseja inserir ou alterar. Para as consultas faz-se necessário o mesmo procedimento, especificando o tempo abrangido pela consulta. Assim podem-se recuperar dados históricos.

Durante a criação das estruturas a tabela versionada será renomeada _LT, e várias colunas são acrescidas a esta tabela para armazenar metadados de versões, essa criação a ser gerenciado pelo próprio Oracle, bem como são disponibilizada algumas *views* para manipulação dos dados. As colunas adicionadas têm nomes que começam com WM_: WM_VERSION, WM_CREATETIME, WM_RETIRETIME, WM_NEXTVER, WM_DELSTATUS e WM_LTLOCK.

Caso alguma tabela que foi versionada possua uma tabela filha, obrigatoriamente deverá ser ativado o recurso para todas as tabelas “filhas” para somente então ativar o recurso de versionamento para as tabelas “pais”.

A Tabela 5 demonstra como ficou a tabela Sessao após ser versionada, passando a ter o nome Sessao_LT.

Sessao	
IDSESSAO	INTEGER
IDESTUDIO	INTEGER
IDARTISTA	INTEGER
IDCLIENTE	INTEGER
VALORSESSSAO	DOUBLE
WM_VALID	WM_PERIOD()
WM_VERSION	NUMBER(38)
WM_CREATETIME	TIMESTAMP(6)
WM_RETIRETIME	TIMESTAMP(6)
WM_NEXTVER	VARCHAR2(500)
WM_DELSTATUS	NUMBER(38)
WM_LTLOCK	VARCHAR2(150)

Tabela 5 – Tabela Sessao_LT

Fonte: Desenvolvido pelo autor (2017).

Assim o campo WM_VALID fica como o tempo de validade e os campos WM_CREATETIME e WM_RETIRETIME como tempo de transação.

Após a estrutura criada será possível iniciar o processo de inserção e de verificação dos dados.

Na Listagem 2 estão representados os *scripts* de inserção das tabelas ARTISTAS e ESTUDIOS observe que a coluna WM_VALID está sendo inserida apenas com data de início e a data final está recebendo o valor NULL por meio do comando DBMS_WM.UNTIL_CHANGED. As referidas tabelas possuem cadastros que, a menos que sofram alterações, estão sempre ativos no que se refere a tempo de validade.

```
INSERT INTO ESTUDIOS(IDESTUDIO, NOMEESTUDIO, WM_VALID)
VALUES (1, 'Arte na Pele', WMSYS.WM_PERIOD(TO DATE('2017-03-01', 'YYYY-MM-DD'), DBMS_WM.UNTIL_CHANGED));
INSERT INTO ESTUDIOS(IDESTUDIO, NOMEESTUDIO, WM_VALID)
VALUES (2, 'Estudio 2', WMSYS.WM_PERIOD(TO_DATE('2017-03-01', 'YYYY-MM-DD'), DBMS_WM.UNTIL_CHANGED));

INSERT INTO ARTISTAS(IDARTISTA, NOMEARTISTA, WM_VALID)
VALUES (1, 'Daniel Lopes', WMSYS.WM_PERIOD(TO_DATE('2017-03-03', 'YYYY-MM-DD'), DBMS_WM.UNTIL_CHANGED));
INSERT INTO ARTISTAS(IDARTISTA, NOMEARTISTA, WM_VALID)
VALUES (2, 'Willian', WMSYS.WM_PERIOD(TO_DATE('2017-03-03', 'YYYY-MM-DD'), DBMS_WM.UNTIL_CHANGED));
```

Listagem 2 – Script de inserção das tabelas ESTUDIOS e ARTISTAS

Fonte: Desenvolvido pelo autor (2017).

Na Listagem 3 está representado o *script* de inserção da tabela CLIENTES. A coluna WM_VALID, nesse caso, está sendo inserida com data de início e data de fim, isso porque existe uma regra de negócio que todo cadastro de cliente não deverá possuir validade superior a um ano, cabendo ao usuário revalidar as informações após este período para manter o cadastro ativo.

```
INSERT INTO CLIENTES(IDCLIENTE, NOMECLIENTE, WM_VALID)
VALUES (1, 'Luís Fernando Dutra', WMSYS.WM_PERIOD(TO_DATE('2017-03-05', 'YYYY-MM-DD'), TO_DATE('2018-03-05', 'YYYY-MM-DD')));
INSERT INTO CLIENTES(IDCLIENTE, NOMECLIENTE, WM_VALID)
VALUES (2, 'Alexandra Picetski', WMSYS.WM_PERIOD(TO_DATE('2017-03-05', 'YYYY-MM-DD'), TO_DATE('2018-03-05', 'YYYY-MM-DD')));
```

Listagem 3 – Script de inserção da tabela CLIENTES

Fonte: Desenvolvido pelo autor (2017).

Na Listagem 4 está o *script* de inserção da tabela SESSAO, nesse caso específico está sendo atribuído um período válido de trabalho para a *workspace*.

```
EXECUTE DBMS_WM.SETVALIDTIME(TO_DATE('2017-03-05 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2017-03-05 23:59:59', 'YYYY-MM-DD HH24:MI:SS'));
```

Listagem 4 – Definindo período de trabalho da workspace

Fonte: Desenvolvido pelo autor (2017).

Com essa definição de período, a *workspace* restringe a inserção apenas aos cadastros que atendam a validação temporal, ou seja, caso tente-se inserir um registro para uma sessão existente no SGBD, mas que, não atenda a validação temporal, o sistema bloqueará a inserção (Listagem 5).

```
INSERT INTO SESSAO(IDSESSAO, IDESTUDIO, IDARTISTA, IDCLIENTE, VALORSESSAO)
VALUES (1, 1, 1, 1, 250.00);

INSERT INTO SESSAO(IDSESSAO, IDESTUDIO, IDARTISTA, IDCLIENTE, VALORSESSAO)
VALUES (1, 1, 2, 2, 120.00);
```

Listagem 5 – Script de inserção da tabela SESSAO

Fonte: Desenvolvido pelo autor (2017).

Depois de efetuada a inserção, foi executada uma pesquisa no SGBD para visualizar como o registro ficou armazenado. Ressalta-se que existem diferenças entre o WM_VALID e o WM_CREATETIME, isso porque todas as tabelas estão versionada de forma bitemporal (Figura 19)

```
SELECT IDCLIENTE, NOMECLIENTE, WM_VALID, WM_CREATETIME, WM_RETIRETIME
FROM CLIENTES_LT
WHERE IDCLIENTE=1
```

IDCLIENTE	NOMECLIENTE	WM_VALID	WM_CREATETIME	WM_RETIRETIME
1	Luís Fernando Dutra	WMSYS.WM_PERIOD('2017-03-05 00:00:00.0', '2017-03-05 23:59:59.0')	05/03/2017 16:20:14,486000000 -02:00	

Figura 19 – Consulta da tabela CLIENTES_LT.

Fonte: Desenvolvido pelo autor (2017).

Com auxilio do operador WM_OVERLAPS com o objetivo de resgatar as informações temporais gravadas, assim verifica se dois períodos se sobrepõem (Figura 20).

```
SELECT * FROM CLIENTES_LT C
WHERE WM_OVERLAPS(C.wm_valid,
    wm_period(TO_DATE('01-01-2017', 'MM-DD-YYYY'),
              TO_DATE('01-01-2018', 'MM-DD-YYYY'))) = 1;
```

ID	NOME
1	Luís Fernando Dutra

Figura 20 – Consulta da tabela CLIENTES_LT utilizando WM_OVERLAPS.

Fonte: Desenvolvido pelo autor (2017).

Depois de realizada a consulta, foi efetuada uma alteração no cadastro de clientes, alterando o nome do cliente de “Luís Fernando Dutra” para “Luís Fernando Giacomini Dutra”. Utilizando as visões MVNO_HIST e PROD_HIST é possível visualizar exatamente cada uma das etapas pelas quais passou o registro, ou seja, toda a história do registro (Figura 21).

```
UPDATE CLIENTES_LT C SET NOMECLIENTE='Luís Fernando Giacomini Dutra'
WHERE IDCLIENTE=1;

SELECT IDCLIENTE, NOMECLIENTE, WM_OPTYPE, WN_CREATETIME
FROM MVNO_HIST WHERE IDCLIENTE=1;
```

IDCLIENTE	NOMECLIENTE	WM_OPTYPE	WM_CREATETIME
1	Luís Fernando Dutra	I	05/03/2017 16:20:14,486000000 -02:00
1	Luís Fernando Giacomini Dutra	U	06/03/2017 13:05:30,105000000 -03:00

Figura 21 – Consulta utilizando views

Fonte: Desenvolvido pelo autor (2017).

5 CONCLUSÃO

A informação tem muita utilidade para quem usufrui da tecnologia, não é necessário apenas guardar a informação, mas sim saber utilizá-la. Os bancos de dados são de grande ajuda no momento de guardar as informações, porém em alguns casos eles acabam guardando apenas o presente momento da informação, o que nem sempre é válido para algumas corporações e usuários, pois precisam destacar dados passados e presentes para traçarem metas e também para verificarem a evolução dos dados.

Para isso o conceito de Banco de Dados Temporal possui vantagem em relação aos outros modelos, pois eles permitem que se armazenem dados históricos. Existem vários tipos de bancos de dados temporais. Cada um pode se encaixar da melhor maneira possível para apresentar os dados ao usuário, mas também é dever de quem faz a análise levar o melhor modelo disponível para a solução apresentada.

Neste trabalho foram apresentados os diferentes tipos de bancos de dados temporais e com base neles foi desenvolvido um pequeno estudo no desenvolvimento do aplicativo Calendar Tattoo juntamente com o banco de dados relacional.

Foi possível verificar que os estúdios de tatuagem necessitam ver a evolução de seus dados e assim com a utilização do Oracle foi possível verificar que é possível mostrar ao usuário dados históricos, ou seja, a sua evolução e validade no tempo.

Na solução proposta no aplicativo, foi utilizado um banco de dados temporal de transação com rótulo temporal baseado em períodos, bitemporal, para que assim sejam mantidas as informações com o momento em que os dados foram inseridos e também o período que esses dados foram válidos.

REFERÊNCIAS

DATE, C. J. **Introdução a sistemas de banco de dados**. 8.ed. Rio de Janeiro: Campus, 1990.

EDELWEISS, N. Banco de Dados Temporais: Teoria e Prática. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, 17; CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 18, 1998. **Anais...** Recife: Sociedade Brasileira de Computação, 1998. p.225-282.

HÜBLER, P. N. **Definição de um gerenciador para o modelo de dados temporal TF-ORM**. 2000. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

PRESSMAN, R, S**. Software Engineering: A practitioner's approach**. 6. Ed. McGrawHill. 2005.

TANSEL, A.U., TIN, E. The expressive power of temporal relational query languages. **IEEE Transaction on Knowledge and Data Engineering**, New York, v.9, n.1, Jan. 1997.