UNIVERSIDADE FEDERAL DO ABC

APLICATIVO PARA MANUTENÇÃO DA *TABLESPACE* USERS EM SISTEMAS RIS

Nathali de Farias Ono

Orientador: Prof. Dr. João Henrique Kleinschmidt

Santo André, 2018

Nathali de Farias Ono

APLICATIVO PARA MANUTENÇÃO DA *TABLESPACE* USERS EM SISTEMAS RIS

Monografia apresentada a Universidade Federal do ABC como requisito para graduação em Engenharia de Informação.

Orientador: Prof. Dr. João Henrique Kleinschmidt

Santo André, 2018

Nathali de Farias Ono

APLICATIVO PARA MANUTENÇÃO DA *TABLESPACE USERS* EM SISTEMAS RIS

Essa monografia foi julgada e aprovada para a obtenção de bacharel em Engenharia de Informação pela Universidade Federal do ABC.

Santo André - SP, 2018.

BANCA EXAMINADORA

_____

Prof. Dr. João Henrique Kleinschmidt

_____ _____

Prof. Dr. Marcelo Perotoni Prof. Dr. Murilo Bellezoni Loiola

RESUMO

O sistema de informação de radiologia (RIS) apresenta grande importância para o gerenciamento de informações em instituições que lidam com diagnósticos por imagem. Os sistemas RIS apresentam grande competitividade comercial fornecendo muitos recursos para a organização das etapas realizadas na área de radiologia, visando melhor atendimento ao paciente. A indisponibilidade do sistema gera fluxo de trabalho manual, sendo que informações importantes, como laudos, imagens e dados dos pacientes, podem ser perdidas ou desviadas, ocasionando muitos prejuízos. Em sistemas RIS que utilizam banco de dados local ORACLE, existe a necessidade de monitoramento do armazenamento lógico de dados (*tablespaces)* para evitar a ocorrência do erro ORA-01653. Este erro pode interromper as atividades dos usuários conectados ao sistema, caso a *tablespace* seja do tipo USERS, impossibilitando a realização das tarefas. Com base nestas informações, este trabalho propõe uma forma de evitar o erro ORA-01653 em um sistema RIS específico, aplicando a modelagem ARIMA para a representação do consumo de dados vinculado ao uso do sistema e, por fim, utilizando as amostras de previsão de consumo, geradas pelo modelo, em um aplicativo desenvolvido para a realização da manutenção preventiva da *tablespace* USERS.

Palavras-chave: RIS; Banco de Dados; *Tablespaces*; ARIMA.

ABSTRACT

The radiology information system (RIS) is important to information management in institutes of diagnostic imaging. RIS systems are strongly competitive providing many resources for radiology workflow, improving the patient care. The downtime system can generate manual workflow and essential information, such as medical reports, images and patients data, can be lost or unlinked, causing losses. In RIS systems that work with ORACLE local database, there is a need for monitoring data logical storage (tablespaces) to avoid the occurrence of ORA-01653 error. This error may interrupt activities that were connected to the system if the tablespace is of USERS type. Based on this information, the objectives of the present work are to propose a way to avoid ORA-01653 error in a specific RIS system applying ARIMA model to represent data usage system and using the forecast samples provided by the model, to develop an application to perform USERS tablespace preventive maintenance.

keywords: RIS; Database; *Tablespaces*; ARIMA.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACR: *American College of Radiology*

AET: *Application Entity Title*

API: *Application Programming Interface*

AR: *Autoregressive*

ARFIMA: *Autoregressive Fractionally Integrated Moving Average*

ARIMA: *Autoregressive Integrated Moving Average*

ARMA: *Autoregressive Moving Average*

BIC: *Bayesian Information Criterion*

DA: *Date*

DBA: *Database Administrator*

DBF: *Database File*

DICOM: *Digital Imaging and Communications in Medicine*

FAC: Função de Autocorrelação

FACP: Função de Autocorrelação Parcial

GB: *Gigabyte*

HL7: *Health Level 7*

IP: *Internet Protocol*

IT: *Information Technology*

JDBC: *Java Database Connectivity*

MA: *Moving Average*

MB: *Megabyte*

MMSE: *Minimum Mean Squared Error*

NEMA: *National Electric Manufactors Association*

PACS: *Picture Archiving and Communication System*

PL: *Procedural Language*

PN: *Patient's Name*

RIS: *Radiology Information System*

SMTP: *Simple Mail Transfer Protocol*

SQL: *Structured Query Language*

TSAF: *Time Series Analysis and Forecast*

VPN: *Virtual Private Network*

VR: *Value Representation*

SUMÁRIO

1. INTRODUÇÃO

A grande evolução da tecnologia nas últimas décadas trouxe muitos benefícios na área de radiologia. Muitos sistemas foram implementados em instituições de saúde e contribuíram para a melhoria no gerenciamento de recursos, facilidade para obter diagnósticos mais precisos e maior agilidade na obtenção de imagens e laudos, aumentando a produtividade e reduzindo custos.

Entre os sistemas que compõem esse cenário, temos o sistema RIS (*Radiology Information System*), que corresponde a uma ferramenta completa para gestão, oferecendo maior cuidado ao paciente e otimizando o fluxo de trabalho. O RIS possui integração com outros sistemas, com a finalidade de centralizar informações sobre o paciente, procedimentos, materiais e atuação profissional.

O sistema RIS utilizado nesse trabalho é distribuído por uma grande empresa que fornece ótimas soluções para a área da saúde e, neste trabalho, o produto será denominado apenas como RIS-x. Atualmente, a empresa conta com mais de 120 instalações do produto em grandes instituições na América Latina e muitas outras distribuídas na Europa e Ásia.

Para manter a satisfação dos clientes, a disponibilidade do sistema RIS-x é essencial para obter o fluxo de trabalho contínuo e oferecer melhor atendimento ao paciente. A interrupção no uso do sistema pode gerar perda ou desvio de informações importantes como laudos, imagens e dados demográficos dos pacientes, dependendo da intervenção manual, retornando inúmeros prejuízos às instituições de saúde e ao próprio paciente.

De forma geral, os sistemas que atuam com banco de dados ORACLE possuem armazenamento de dados em unidades lógicas, denominadas *tablespaces*, e o armazenamento físico dessas unidades pode ser feito por um ou mais *datafiles*, que podem ter tamanho limitado. Para garantir que usuários e aplicações se mantenham conectados, as *tablespaces* precisam ter espaço disponível para a inserção de novos dados no sistema. Caso contrário, é ocasionado o erro “ORA-01653 - *unable to extend table in tablespace*" [1], sucedendo-se assim a falta de operabilidade do sistema, o que torna algo inadmissível em um ambiente em produção.

A solução proposta pela ORACLE para evitar que esse erro ocorra, consiste em utilizar o recurso de *autoextend* [1] para expandir automaticamente o tamanho das *tablespaces*, porém, nem sempre isso pode ser aplicado. No sistema RIS-x devido à arquitetura atual e ao grande crescimento de uma das *tablespaces,* USERS, o recurso de *autoextend* não pode ser ativado e a tentativa de expansão automática da *tablespace* pode causar problemas, caso não haja espaço disponível na partição padrão para gravação de *datafiles*.

Com base nesse problema foi providenciado, pela equipe de desenvolvimento responsável pelo RIS-x, um procedimento manual para a prevenção do erro "ORA-01653". Assim, o procedimento é aplicado semanalmente pela equipe de suporte responsável pelo produto, que atua em incidentes envolvidos em instalações de sistemas na América Latina. O procedimento consiste, basicamente, na conexão ao banco de dados de cada servidor da aplicação RIS, verificação dos espaços disponíveis na *tablespace* e na partição que armazena os *datafiles* no disco rígido do servidor.

Apesar de ser um procedimento simples, pode gerar falhas de operação manual e muito tempo é gasto ao realizar diversas conexões. A maior dificuldade nesse processo, no entanto, é determinar quando se deve adicionar um novo *datafile* com base na ocupação da *tablespace*, pois cada instituição possui um fluxo de uso do sistema. Os indicadores que são fornecidos pelas *views* do ORACLE, mostrando os valores de espaço em uso e espaço livre, não são suficientes para obter essa informação porque não é considerado o consumo envolvido no processo.

Além disso, os recursos de hardware são limitados, devido às exigências pelo baixo custo no projeto, sendo que não há planejamento para dimensionar a capacidade dos discos de acordo com o uso da instituição. Não seria viável analisar a partição do disco ao invés do consumo porque *datafiles* de outras *tablespaces* e arquivos do sistema estão no mesmo local. Devido à arquitetura atual do sistema e integração com outros sistemas de saúde, não é possível realizar o armazenamento *online* do banco de dados.

Com base nessas informações, os objetivos do projeto são identificar um modelo que represente o consumo de dados diário, de acordo com uma série temporal, da *tablespace* USERS no banco de dados de um servidor específico da

aplicação RIS-x e elaborar um aplicativo simples que realize a manutenção automática da *tablespace* de acordo com as amostras geradas pelo modelo escolhido.

A organização da monografia está dividida da seguinte forma. O capítulo 2 apresenta fundamentos sobre o sistema RIS, com uma abordagem geral, fluxo de utilização e módulos do sistema. Em seguida são mostrados os princípios e características das *tablespaces* em um banco de dados ORACLE. E ao final do capítulo, um resumo sobre modelos de previsão de séries temporais.

O capítulo 3 descreve a rotina da manutenção da *tablespace*, que serviu como base para a elaboração do projeto, a coleta e adequação dos dados para utilização na aplicação e a criação de uma tabela externa para facilitar as tarefas executadas pelo aplicativo.

No capítulo 4 são realizadas observações sobre o modelo adotado e sua inserção no aplicativo elaborado.

No capítulo 5 são descritas as etapas executadas pelo aplicativo e os componentes que abrangem sua estrutura. São detalhadas informações sobre os métodos criados.

O capítulo 6 apresenta a conclusão, com as considerações finais e melhorias propostas para trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Sistema de Informação de Radiologia

O Sistema de Informação de Radiologia, RIS, é uma ferramenta utilizada pelo departamento de radiologia, em instituições de saúde, para armazenar, manipular e distribuir dados do paciente e vincular imagens. O sistema possui módulos que gerenciam todas as etapas que um paciente realiza no departamento de radiologia, fornecendo a capacidade de rastreamento e auditoria de muitas informações como laudo e imagens radiológicas.

Os sistemas RIS, de um modo geral, possuem integração com outros sistemas como o HIS (*Hospital Information Systems*), responsável pelos dados demográficos dos pacientes e gerenciamento financeiro [2], e o PACS (*Picture Archiving Communication System*), que estabelece comunicação com equipamentos e armazena as imagens utilizando protocolo DICOM (*Digital Imaging and Communications in Medicine*)[2,3].

O protocolo DICOM foi criado em 1985 pelo ACR (*American College of Radiology*), e NEMA, (*National Electric Manufactors Association*). Estabelece a padronização da busca, do armazenamento e da impressão de imagens médicas em uma arquitetura cliente-servidor [3]. Para estabelecer a comunicação entre equipamentos com a aplicação do protocolo, basicamente, se torna necessária a configuração de três parâmetros: AET (*Application Entity Title*), IP (*Internet Protocol*) e porta definida para canal de comunicação.

A estrutura de dados DICOM, que armazena as informações da imagem, é mostrada no quadro da figura 1. O campo *tag* representa um índice, o campo VR (*Value Representation*) indica o tipo do atibuto, que pode ser PN (Patient's Name) ou DA (*Date*), entre outros [3], e o campo *description* contém a descrição do valor contido na *tag*.

Tag	VR	Description
(0010,0010)	PN	Patient's Name
(0010,0030)	DA	Patient's Birth Date

Figura 1 - Exemplo da estrutura de dados DICOM

A figura 2 mostra uma configuração básica referente a integração entre um sistema RIS genérico e outros sistemas. As imagens arquivadas no PACS e os dados dos pacientes no HIS são enviados para o RIS, assim como os laudos são enviados para o HIS, PACS e Visualizador de laudos WEB, sendo que este último se destina exclusivamente para a utilização do paciente.

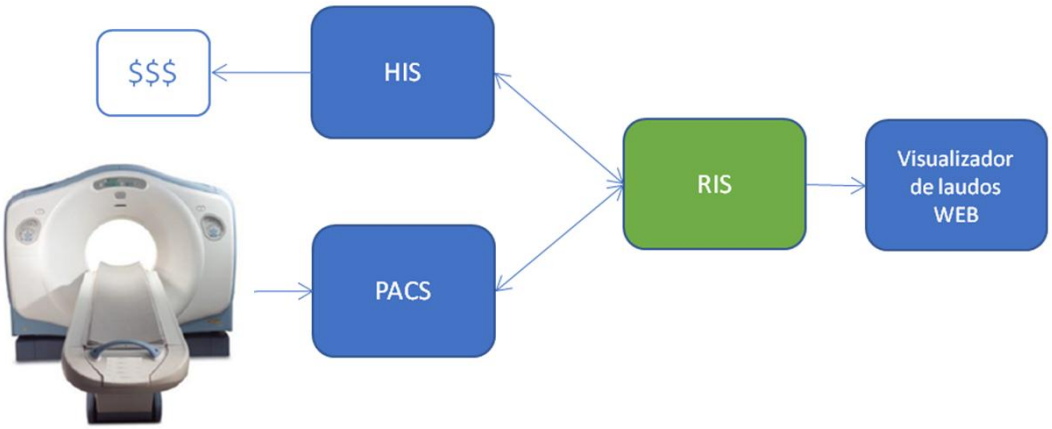


Figura 2 - Esquema básico de integração entre o sistema RIS e outros sistemas

A empresa tratada nesse trabalho fornece tecnologias médicas que atendem a demanda por acesso a serviços de saúde de qualidade. Produz equipamentos de diagnóstico de imagens, além de soluções como o sistema RIS-x, que engloba tecnologias e serviços que visam o gerenciamento eficiente da informação e o cuidado com o paciente.

A figura 3 mostra as etapas de uso do sistema RIS-x conforme o fluxo seguido pelo paciente. O agendamento é o primeiro passo que o paciente realiza, o módulo armazena as informações sobre data, hora e local que o paciente realizará o exame. O

local, neste caso, é definido no sistema como sala de trabalho. Um exame pode ser feito em várias salas de trabalho, mas cada uma delas possui apenas um equipamento para o exame. A informação sobre o local que o paciente realiza o exame é importante, pois faz a alocação do equipamento para o uso do paciente.

Figura 3 - Fluxo de trabalho do departamento de radiologia

O registro é realizado na recepção da instituição de saúde quando o paciente comparece na data agendada. São registradas informações sobre convênio médico, materiais que serão utilizados no procedimento, entre outras informações administrativas.

Seguindo o fluxo, o paciente realiza o exame na sala de trabalho que foi agendada. Como exemplos de exames podem ser citados ressonância magnética, tomografia computadorizada, ultrassonografia, radiografia, entre outros. Após realização do exame, o PACS armazena as imagens produzidas pelo equipamento médico, por meio de alguns eventos de integração, o RIS-i vincula as imagens geradas.

Com os dados do paciente e a disponibilização das imagens, o médico pode realizar o diagnóstico. Nessa etapa pode ser utilizado um recurso com reconhecimento de voz, utilizando *hardware* apropriado e realizando o treino para a adaptação do software de acordo com a pronúncia. Assim, o médico dita o diagnóstico e o laudo é escrito automaticamente. A maioria dos clientes que utilizam o sistema faz o uso da escrita do laudo manualmente, sem a intervenção do recurso de reconhecimento de voz, em consideração aos custos relacionados às licenças.

O sistema também gerencia a lista de procedimentos atuais realizados pelo paciente e o histórico, que podem conter informações de extrema importância para o diagnóstico adequado do paciente.

A etapa de conferência possibilita o estudo de casos e compartilhamento de informações entre as equipes médicas. A distribuição de laudo possibilita ao paciente

de receber as informações diagnósticas por email ou acessar um portal de entrega de laudos via *web*.

Com todas as etapas concluídas, é possível gerar relatórios sobre exames feitos e adequar a diversas categorias como: por médicos, por modalidade, por data, etc. Assim, o sistema RIS-x oferece melhor gerenciamento da informação e agiliza o fluxo de trabalho no departamento de radiologia.

2.2 *Tablespaces* em Banco de Dados Oracle

Em um banco de dados Oracle os objetos, como tabelas e índices, são armazenados em espaços dentro do banco de dados. Esses espaços lógicos são denominados *tablespaces*, os quais estão associados a um ou mais arquivos físicos, chamados *datafiles* que possuem extensão .dbf. Cada *datafile* está associado somente a uma *tablespace*, e pode ser adicionado, removido, movido ou redimensionado de acordo com as necessidades do sistema [4].

Uma *tablespace* é constituída por segmentos lógicos distribuídos em *datafiles*, como mostra a figura 4. O segmento associa os dados de qualquer estrutura que contenha um conjunto de dados, como por exemplo, informações sobre uma tabela ou índice.

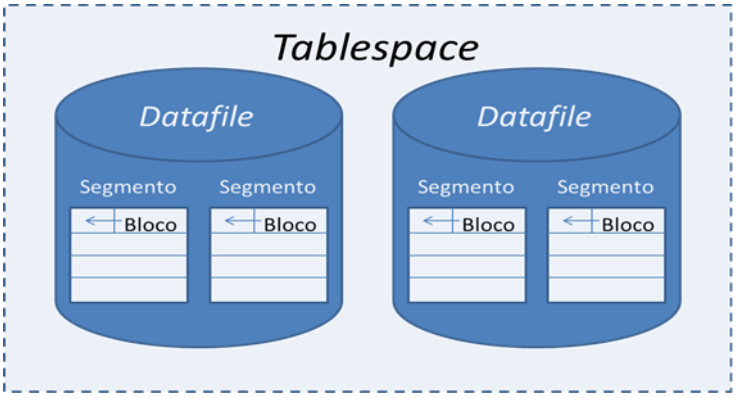


Figura 4 - *Tablespace*, *datafiles*, segmentos e blocos

Um segmento é composto por extensões, que são conjuntos de blocos contíguos dentro de um arquivo de dados. Depois que as extensões existentes não podem mais conter novos dados, o segmento obtém uma nova extensão. Esse processo continuará até que o número máximo interno de extensões por segmento seja atingido. Assim, se todos os segmentos forem ocupados é necessária a geração de um novo *datafile*, a fim de garantir a integridade dos dados e permitir a disponibilidade nos momentos de consulta e a inserção de novos dados no sistema.

A figura 5 apresenta os tipos de *tablespaces*, SYSTEM, SYSAUX, UNDO, TEMP e USERS, que são geradas no processo de criação do banco de dados e suas funções

resumidas e, em seguida, são descritas as funcionalidades de cada uma delas com base em [1,4].

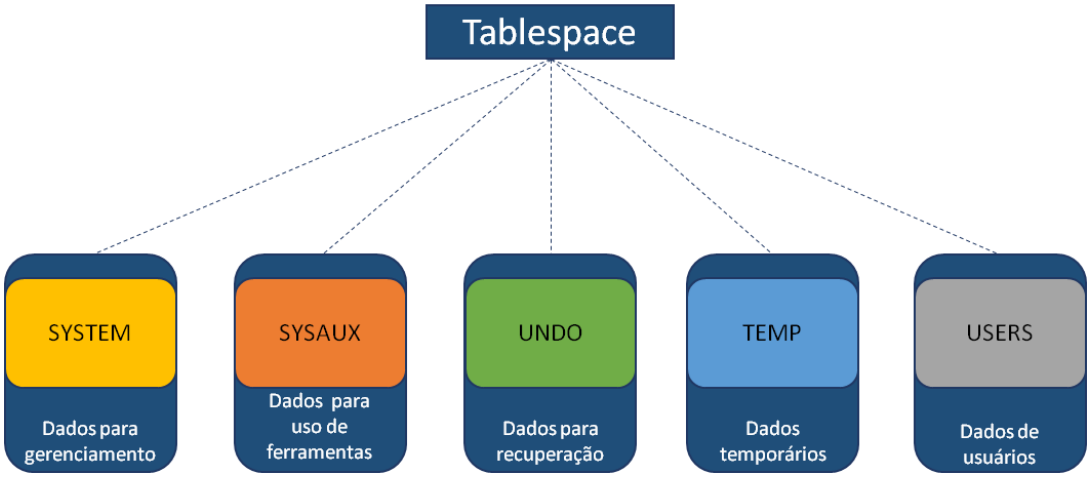


Figura 5 - *Tablespaces* no banco de dados ORACLE

A *tablespace* SYSTEM é gerada automaticamente no momento da criação do banco de dados, onde é armazenado o dicionário de dados do banco. Todas as informações sobre os objetos criados no banco de dados são gravados nessa *tablespace*.

A *tablespace* SYSAUX funciona como componente auxiliar a SYSTEM. Muitos recursos do banco de dados usam a *tablespace* SYSAUX como local padrão para armazenamento de dados. Ela é gerada durante o processo de criação ou de atualização do banco de dados.

A *tablespace* UNDO é utilizada para armazenar informação de recuperação de transações. Não é possível criar qualquer outro tipo de segmento, como tabelas ou índices, na UNDO. No modo de gerenciamento de recuperação automático, cada instância Oracle é destinada a uma *tablespace* UNDO. Dados de recuperação são gerenciados dentro de uma *tablespace* UNDO usando os segmentos, destinados a esta finalidade, que são automaticamente criados e mantidos pela ORACLE.

A *tablespace* TEMP é usada para gerenciar espaço para operações de ordenação e para armazenar dados de tabelas temporárias. Algumas operações SQL (*Structured Query Language*) que requerem ordenação são: SELECT DISTINCT, ORDER BY, GROUP BY, UNION, INTERSECT, MINUS, CREATE INDEX e ANALYZE [1].

A figura 6 mostra uma estrutura SQL simples como exemplo de uma operação de ordenação usando o ORDER BY. É feita a seleção de colunas de uma tabela específica contendo uma condição. Assim, é feita a ordenação com referência a uma das colunas com as cláusulas ASC e DESC, que denotam ordenação ascendente e descendente, respectivamente.

```
SELECT <lista_de_colunas>
FROM <nome_tabela>
WHERE <condição_de_seleção>
ORDER BY {<nome_coluna>|<num_col> [ASC|DESC]}
```

Figura 6 - Estrutura SQL para representação de uma operação de ordenação

A *tablespace* USERS é usada como padrão para os usuários. Sua função desempenha um papel muito importante, pois armazena todos os objetos gerados por usuários no sistema. No caso específico do sistema RIS-x, os objetos são representados pela sessão de usuários, registros dos pacientes, visualização de imagens provenientes dos exames realizados pelos pacientes, laudos com o diagnóstico dos médicos, criação de procedimentos de exames radiológicos, adição de materiais utilizados no exame, etc. A falta de espaço nela pode causar o erro ORA-01653 [1], impossibilitando os usuários a autenticar suas credenciais e operar o sistema.

No ambiente de algumas instituições de saúde, a quantidade de informação a ser produzida diariamente, pode variar e ser analisada como uma série temporal. A previsão do consumo de espaço da *tablespace* USERS e inserção de *datafiles* periodicamente são necessárias, pois a quantidade excessiva de *datafiles* associadas a uma determinada *tablespace* pode gerar aumento do processamento e ocupar espaço em disco desnecessário no servidor.

2.3 Modelos de Previsão de Séries Temporais

Uma série temporal pode ser definida como qualquer conjunto de observações ordenadas no tempo. Os modelos utilizados para representar séries temporais são processos estocásticos, regidos por leis probabilísticas, e uma série particular pode ser descrita por vários modelos diferentes.

Na utilização de modelos para descrever séries temporais, podem ser utilizadas suposições simplificadoras para conduzir a análise de determinadas classes de processos estocásticos [6]. Algumas suposições são divididas em:

a. Processos estacionários ou não estacionários;
b. Processos normais ou não normais, conforme as funções densidade de probabilidade que caracterizam os processos;
c. Processos Markovianos ou não Markovianos, conforme a independência dos valores do processo em instantes diferentes;

Em relação à estacionariedade, os processos podem ser classificados no sentido estrito, amplo ou ergódico. Resumidamente, um processo estritamente estacionário caracteriza-se por propriedades estatísticas invariantes a um deslocamento e um processo no sentido amplo a média é constante e a correlação só depende da diferença de tempo das observações. A referência sobre estacionariedade, neste trabalho, se dirige ao sentido amplo e em termos práticos indica que a série se desenvolve em torno de uma média e variância constantes.

Em relação à normalidade, é necessário que o processo tenha distribuição de probabilidade da variável normal ou gaussiana, que tem como dependência dois parâmetros, a média e o desvio padrão, e sua notação N (μ,σ^2). Um processo estocástico comum utilizado com frequência na literatura e modelos de precisão é o ruído branco. Ele compreende uma sequência de variáveis aleatórias mutuamente independentes e seguem a distribuição normal, N ($0,\sigma^2$), com média zero e variância σ^2.

Os modelos para séries temporais podem ser divididos em duas classes, de acordo com o número de parâmetros. Os modelos paramétricos possuem um número

de parâmetros finitos enquanto os modelos não paramétricos possuem um número infinito de parâmetros.

No grupo de modelos paramétricos, os modelos mais utilizados são: auto-regressivos AR (Autoregressive), médias móveis MA (*Moving Average*), auto-regressivos e de médias móveis ARMA (*Autoregressive Moving Average*), auto-regressivos integrados e de médias móveis ARIMA (*Autoregressive Integrated Moving* Average), modelos de memória longa ARFIMA (*Autoregressive Fractionally Integrated Moving Average*), modelos estruturais e modelos não lineares.

Na análise de modelos paramétricos uma metodologia bem utilizada é conhecida como abordagem de Box e Jenkins [7]. A metodologia se baseia no ajuste de modelos ARIMA a um conjunto de dados, sendo que a construção do modelo é ajustada em um ciclo iterativo. O ciclo iterativo executa as seguintes etapas: analisa uma classe geral de modelos; identifica o modelo com base na análise estatística; identifica os parâmetros do modelo; verifica se o modelo ajustado é adequado e eficiente na previsão. As etapas do ciclo são mostradas na figura 7.

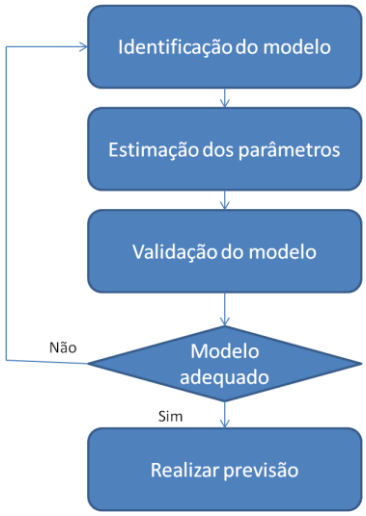


Figura 7 - Ciclo iterativo de Box e Jenkins

Caso o modelo não seja adequado, as etapas são realizadas novamente. Pode ser aplicada também, a identificação de vários de modelos e, assim, pode-se escolher um dos modelos conforme um critério de seleção.

2.3.1 Modelo Auto-regressivos e de Médias Móveis (ARMA)

O modelo ARMA(p,q) é uma combinação de modelos AR(p) e MA(q). O modelo AR(p,q) apresenta o valor de uma variável em um determinado período como uma combinação linear dos valores passados, p, e de um termo aleatório. A equação geral que descreve o modelo é definida como:

$$y_t = c + \sum_{i=1}^{p} \varphi_i \, y_{t-i} + a_t = c + \varphi_1 \, y_{t-1} + \varphi_2 \, y_{t-2} + \cdots + \varphi_p \, y_{t-p} + a_t \tag{2.1}$$

As variáveis, y_t e a_t representam, respectivamente, o valor atual e a variável aleatória, que pode ser considerada como o erro (ou ruído branco), no instante de tempo t, a constante p é conhecida como a ordem do modelo, φ_i $(i = 1,2,3,\ldots,p)$ são os parâmetros do modelo e c é uma constante. Os parâmetros do modelo AR(p) podem ser estimados conforme as equações de Yule-walker [7].

Os modelos MA(q) são aplicados quando há autocorrelação entre os resíduos, ou seja, os erros em períodos anteriores possuem uma relação de dependência [7]. O modelo MA pode ser definido como:

$$y_t = \mu + \sum_{j=1}^{q} \theta_j \, a_{t-j} + a_t = \mu + \theta_1 \, a_{t-1} + \theta_2 \, a_{t-2} + \cdots + \theta_q \, a_{t-q} + a_t \tag{2.2}$$

Os parâmetros do modelo são representados por θ_j $(j = 1,2,3,\ldots,q)$, q é a ordem do modelo e μ a média da série.

Os modelos AR e MA podem ser efetivamente combinados para formar uma classe de modelos muito utilizada conhecida como modelos ARMA. Matematicamente um ARMA(p,q) é representado por:

$$y_t = c + a_t + \sum_{i=1}^{p} \varphi_i \, y_{t-i} + \sum_{j=1}^{q} \theta_j \, a_{t-j} \tag{2.3}$$

Para determinar um modelo apropriado para um conjunto de dados é necessário analisar a FAC (Função de Autocorrelação) e a FACP (Função de Autocorrelação Parcial). Essas medições estatísticas refletem como as observações de uma série estão relacionadas.

Para uma série temporal {x(t), t = 0,1,2,3...} a autocovariância [7] na *lag* k é definida por:

$$\gamma_k = Cov(x_t, x_{t+k}) = E\left[(x_t - \mu)(x_{t+k} - \mu)\right] \tag{2.4}$$

O coeficiente de autocorrelação na *lag* k é definido por:

$$\rho_k = \frac{\gamma_k}{\gamma_0} \tag{2.5}$$

O símbolo E denota o operador esperança e a autocovariância na *lag* zero, γ_0, é a variância da série temporal. A FACP é uma medida de correlação entre as observações passadas, considerando k períodos, e a observação atual. Na primeira *lag* a FACP é igual a FAC.

Para modelos AR as FAC (funções de autocorrelação) podem apresentar comportamento exponencial decrescente, podendo ter componentes senoidais amortecidas. A tabela 1 apresenta os comportamentos comuns das funções de autocorrelação e autocorrelação parcial, FAC e FACP, de alguns modelos comuns ARMA.

Tabela 1 - Comportamento da FAC e FACP para alguns modelos AR e MA

Modelo	FAC	FACP
MA(1)	Pico no lag 1	Decrescimento exponencial
AR(1)	Decrescimento exponencial	Pico no lag 1
MA(2)	Picos no lag 1 e no lag 2	Mistura de exponenciais podendo conter ondas senoides amortecidas
AR(2)	Mistura de exponenciais podendo conter ondas senoides amortecidas	Picos no lag 1 e no lag 2

2.3.4 Modelo Auto-regressivos Integrados e de Médias Móveis (ARIMA)

Os modelos ARMA descritos anteriormente são utilizados para séries temporais estacionárias, mas muitas séries encontradas na prática são não estacionárias. O modelo ARIMA aplicado em uma série não estacionária pode torná-la estacionária utilizando a diferenciação. Considerando um operador de atraso [7], $Ly_t = y_{t-1}$, segue a representação do modelo ARMA:

$$AR(p):\ a_t = \varphi(L)\, y_t$$
$$MA(q):\ y_t = \theta(L) a_t$$
$$ARMA(p,q): \varphi(L) y_t = \theta(L) a_t$$

onde, $\varphi(L) = 1 - \sum_{i=1}^{p} \varphi_i\, L^i$ e $\theta(L) = 1 - \sum_{j=1}^{q} \theta_j\, L_j$

A formulação matemática para um modelo ARIMA utilizando *lags* polinomiais é dada abaixo:

$$\varphi(L)(1-L)^d y_t = \theta(L) a_t \tag{2.6}$$

$$\left(1 - \sum_{i=1}^{p} \varphi_i\, L^i\right)(1-L)^d y_t = \left(1 - \sum_{j=1}^{q} \theta_j\, L_j\right) a_t \tag{2.7}$$

Assim, p e q possuem valores inteiros maiores ou iguais a zero e se referem à ordem das seguintes partes dos modelos AR e MA. O valor inteiro d controla o nível da diferenciação. Em casos práticos comuns, geralmente d=1 é suficiente na maioria dos casos. Quando d=0, o modelo se reduz a um ARMA.

Um caso como ARIMA(p,0,0) é um modelos AR(p), assim como ARIMA(0,0,q) é um modelo MA(q). No caso do ARIMA(0,1,0) é uma especial condição conhecida como *Randon Walk* [8], é muito utilizado para dados não estacionários, como séries econômicas.

3. METODOLOGIA

A metodologia do presente trabalho está dividida em três partes. A primeira parte está relacionada a descrição do procedimento manual de monitoramento e manutenção da *tablespace* USERS*,* aplicado atualmente, o qual motivou a criação de um aplicativo para seguir operação automática. A segunda parte inclui a coleta e condicionamento da série relacionada ao consumo da *tablespace* e a terceira parte mostra a criação de uma tabela externa para obtenção do valor do espaço na partição onde se encontram os *datafiles* no servidor.

De acordo com as políticas de segurança da empresa, não podem ser instalados aplicativos em servidores em produção, voltados aos projetos comerciais, sem passar por uma validação interna. Os projetos devem ser encaminhados para a área responsável em um período específico para homologação. No entanto, para desenvolver o projeto foi utilizado um computador com sistema operacional Windows 7 de uso corporativo, NetBeans IDE 8.0.2 e aplicativos com recursos de conexão remota como ORACLE *SQL Developer* e Cisco VPN 4.6. Os dados foram obtidos de um servidor da aplicação RIS-x, de uma instituição denominada como ‘Hospital A’, com conexão VPN, sistema operacional Windows Server 2012 e banco de dados ORACLE 10g.

3.1 Rotina de Manutenção da *Tablespace* USERS

O banco de dados ORACLE possui diversas *views* destinadas a mostrar informações das *tablespaces* para facilitar o trabalho dos administradores de bancos de dados (DBAs) [8]. Através dessas *views* foram baseadas as etapas da manutenção manual do banco de dados do sistema RIS-x. Será descrito, a seguir, o procedimento realizado pelo departamento responsável por atividades de suporte ao sistema e ao respectivo banco de dados.

Os passos iniciais consistem em acessar o servidor da aplicação RIS-x com o usuário com privilégios de administrador, executar o ORACLE *SQL Developer*, conectar no banco de dados com o usuário que possui privilégios de DBA, clicar com botão direito no ícone da conexão e escolher a opção '*Manage Database*' para processar a *view* DBA_*TABLESPACE*_USAGE_METRICS conforme mostra a figura 8.

TABLESPACE_NAME	PERCENT_USED	PCT_USED	ALLOCATED	USED	FREE	DATAFILES
SYSAUX		96.19	2048	1969.88	78.13	1
USERS		82.77	274042	226815.88	47226.13	28
UNDO		80.2	4096	3285.06	810.94	1
USERS1		74	16384	12123.5	4260.5	4
SYSTEM		54.97	2048	1125.75	922.25	1
TEMP		(null)	(null)	(null)	0	(null)

Figura 8 - Informação sobre as *tablespaces* obtida pela view

O resultado da *view* mostra o nome da *tablespace* em *TABLESPACE*_NAME, o indicador e o valor percentual utilizado por cada *tablespace*, representado pelas colunas PERCENT_USED e PCT_USED. A coluna USED mostra o valor do espaço usado na *tablespace* em MB e a coluna FREE o espaço livre em MB. A coluna *datafiles* mostra a quantidade de *datafiles*, arquivos físicos, associados à *tablespace* correspondente. Os indicadores da ocupação das *tablespaces*, com suas respectivas cores e faixas percentuais, são mostrados na figura 9.

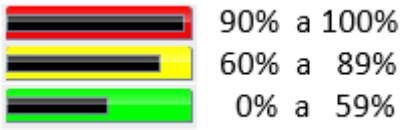


Figura 9 - Porcentagem representada por cada indicador de ocupação da *tablespace*

O monitoramento do crescimento de dados é realizado somente na *tablespace* USERS, sendo que consiste na observação do percentual de espaço usado, sem a análise do consumo de dados. Caso ele esteja apresentando um indicador na cor vermelha, correspondente ao valor acima de 90% de uso, recomenda-se a inserção de um novo *datafile*. Caso contrário, se o indicador mostrar a cor amarela ou verde, não é recomendada nenhuma ação de manutenção.

Para inserir um novo *datafile*, executa-se o *script* da figura 10, para verificar o último *datafile* que foi criado e o diretório em que os mesmos se encontram. A nomeação do arquivo segue o padrão 'USERSRAD_', mostrado pela figura 11, correspondente ao resultado da consulta na *view* DBA_DATA_FILES.

```
SELECT SUBSTR (file_name,0,50),bytes/1024/1024 sizemb
, file_id
, status
, online_status
FROM dba_data_files
WHERE tablespace_name='USERS'
ORDER BY file_id;
```

Figura 10 - *Script* SQL para consulta sobre *datafile*s da *tablespace* USERS

	SUBSTR(FILE_NAME,0,50)	SIZEMB	FILE_ID	STATUS	ONLINE_STATUS
7	E:\ORACLE\ORADATA\RAD\USERSRAD_7.DBF	16383	12	AVAILABLE	ONLINE
8	E:\ORACLE\ORADATA\RAD\USERSRAD_8.DBF	16383	13	AVAILABLE	ONLINE
9	E:\ORACLE\ORADATA\RAD\USERSRAD_9.DBF	16383	14	AVAILABLE	ONLINE
10	E:\ORACLE\ORADATA\RAD\USERSRAD_10.DBF	16383	15	AVAILABLE	ONLINE
11	E:\ORACLE\ORADATA\RAD\USERSRAD_11.DBF	16383	16	AVAILABLE	ONLINE

Figura 11 - Retorno da consulta SQL sobre *datafiles* da *tablespace* USERS

Após obter as informações sobre o diretório, é necessário verificar o espaço na partição de disco onde são gravados os *datafiles*. É recomendado, no mínimo, 20 GB de espaço livre na partição. No exemplo mostrado na figura 12, a partição 'E:' contém

14,2 GB de espaço disponível. Neste caso, a equipe responsável pelas atividades de suporte informa ao administrador da área de Tecnologia da Informação do hospital e o mesmo providencia a adição de disco no servidor.

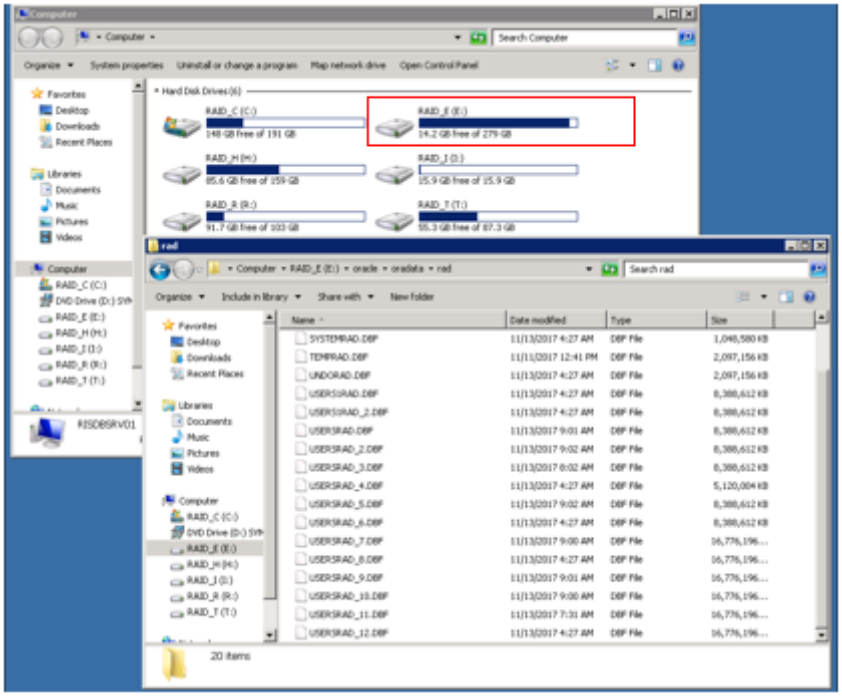


Figura 12 - Partição e diretório para criação dos *datafiles*

Caso haja espaço disponível na partição, pode ser adicionado o *datafile* de 16383 MB, utilizando o *script* apresentado na figura 13.

```
ALTER tablespace users
ADD datafile 'E:\ORACLE\ORADATA\RAD\USERSRAD_12.DBF'
SIZE 16383M;
```

Figura 13 - *Script* SQL para expandir a *tablespace* USERS

3.2 Coleta e Condicionamento dos Dados para Aplicação

As amostras relacionadas ao consumo diário da *tablespace* USERS, foram obtidas por meio de um *script* SQL, adaptado de [9], aplicado no banco de dados ORACLE que relaciona quatro *views*: DBA_HIST_*TABLESPACE*_STAT, DBA_HIST_TBSPC_SPACE_USAGE, DBA_HIST_SNAPSHOT e DBA_*TABLESPACES*, pode ser visto na figura 14. As documentações sobre as *views* declaradas podem ser obtidas em [10].

```
SELECT TO_CHAR (sp.begin_interval_time,'DD-MM-YYYY') data
, ts.tsname as tablespace
, max(round((tsu.tablespace_size* dt.block_size )/(1024*1024),2) )
, max(round((tsu.tablespace_usedsize* dt.block_size )/(1024*1024),2))
FROM DBA_HIST_TBSPC_SPACE_USAGE tsu
, DBA_HIST_TABLESPACE_STAT ts
, DBA_HIST_SNAPSHOT sp
, DBA_TABLESPACES dt
WHERE tsu.tablespace_id= ts.ts#
AND tsu.snap_id = sp.snap_id
AND ts.tsname = dt.tablespace_name
AND ts.tsname IN ('USERS')
GROUP BY TO_CHAR (sp.begin_interval_time,'DD-MM-YYYY'), ts.tsname
ORDER BY ts.tsname, data;
```

Figura 14 - *Script* SQL para coletar amostras do consumo diário da *tablespace.*

O *script* resulta nos valores máximos de espaço total e espaço usado convertidos para MB, com aproximação de duas casas decimais pela função *round*. Essa conversão em MB relaciona o tamanho da *tablespace* multiplicado por cada bloco contido e ocupado por ela, depois dividido por 2^{20} (1048576 bytes). Essa divisão ajusta os dados de bytes para MB. O agrupamento realizado na penúltima linha do *script* retorna o valor máximo por data e a última linha faz a ordenação das datas.

Assim, foram obtidas 140 amostras de consumos diários da *tablespace* para o hospital A, subtraindo os valores do espaço livre e espaço usado retornados pelo script. O gráfico com as amostras é mostrado na figura 15. Observa-se a variação da série em torno de uma média constante, valor mínimo em torno de 60 MB e valor máximo em torno de 200 MB.

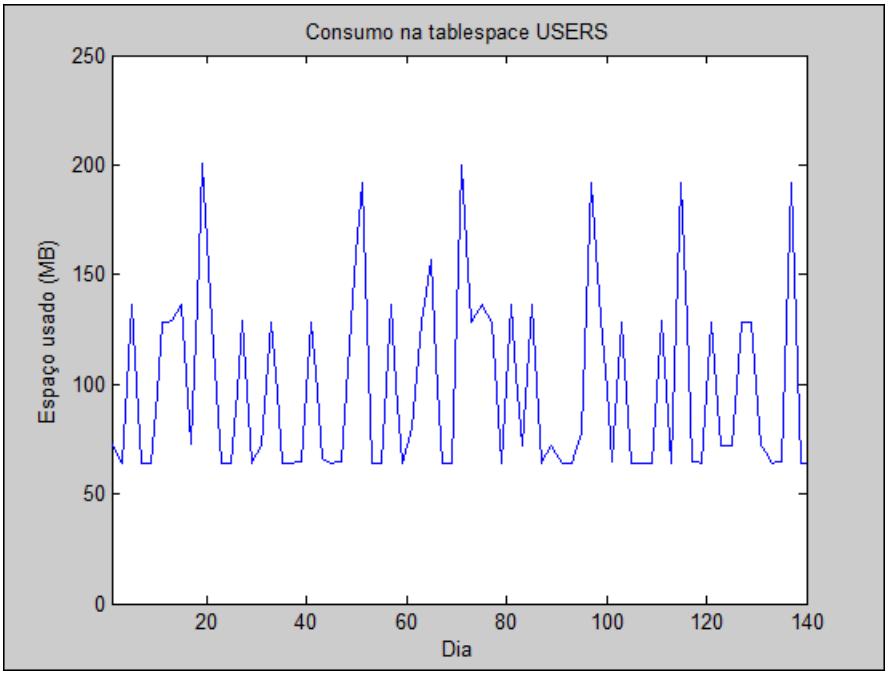


Figura 15 - Consumo da *tablespace* USERS - Hospital A

A análise das séries foi feita com a *toolbox* do MATLAB *Time Series Analysis and Forecast* (TSAF) [12]. A ferramenta possui interface gráfica intuitiva, como pode ser vista na figura 16, e possui seções para tratamento de séries temporais, incluindo recursos para tratamento de sazonalidade e filtragem usando diferenciação.

A ferramenta permite o ajuste de um modelo utilizando o BIC (*Bayesian Information Criterion*) [12]. O critério associa uma penalidade de acordo com o número de parâmetros do modelo**.** O modelo que alcança o menor valor BIC é escolhido como melhor modelo a ser adotado.

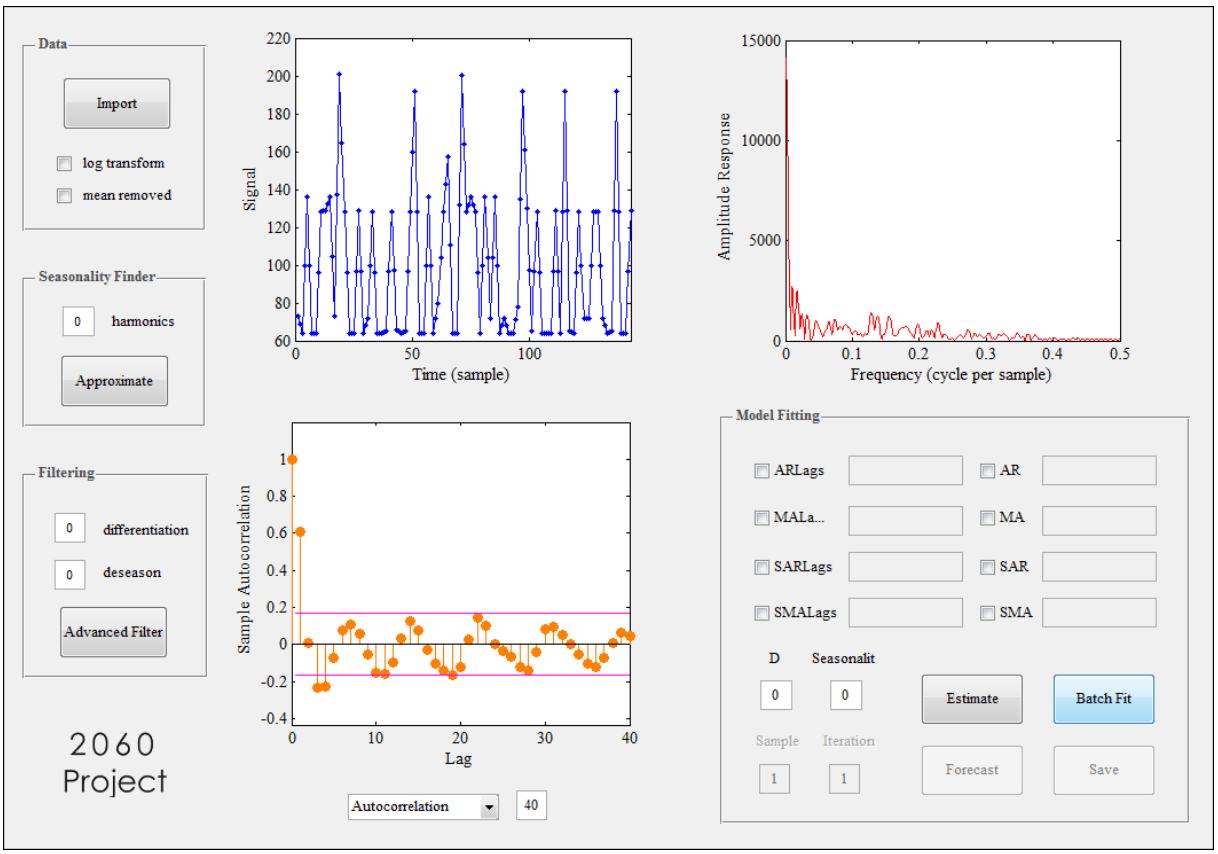


Figura 16 - Interface da toolbox Time Series Analysis and Forecast [11]

O procedimento padrão para utilização do recurso 'Batch Fit', com o propósito de identificar um modelo, na ferramenta é mostrado nos itens abaixo.

1. Importação da série pelo botão '*Import*' em '*Data*';
2. Observar se o espectro em frequências mostra uma componente DC (frequência igual a zero) e em caso positivo, clicar em '*mean removed*';
3. Verificar se há fator de sazonalidade na série inserindo as harmônicas significativas na seção de '*Seasonality Finder*';
4. Verificar se há necessidade de realizar a diferenciação da série, se apresentar tendência;
5. Clicar no botão '*Batch Fit*' para ajustar um modelo para a série selecionada na seção '*Model Fitting*', indicando os parâmetros ARIMA (p,d,q) que serão ajustados no teste;
6. Salvar o melhor modelo, seguindo o critério BIC, obtido na etapa anterior;
7. Configurar a quantidade de amostras para previsão em '*Samples*';

8. Clicar no botão '*Forecast*';
9. As amostras são geradas e armazenadas, por padrão, em um arquivo '*Forecast*.xls'.

Em relação à série obtida e inserida na interface, observou-se que a componente DC do espectro em frequências era dominante. Ao clicar em '*mean removed*' a componente DC é removida e alguns picos se tornam visíveis no espectro. Tomando os três picos mais relevantes do espectro, foi inserida a quantidade de harmônicas significantes na sessão '*Seasonality Finder*' e feita a aproximação. O resultado mostrado na figura 17 indica que a sazonalidade é aproximadamente observada a cada '0' amostras, sendo que não é necessário fazer ajuste. Caso fosse necessário remover a sazonalidade, insere-se o valor apresentado no campo '*deseason*' na seção '*Filtering*'. Se a série apresentar tendência, pode-se utilizar o campo '*differentiation*' para diferenciar a série e torná-la estacionária.

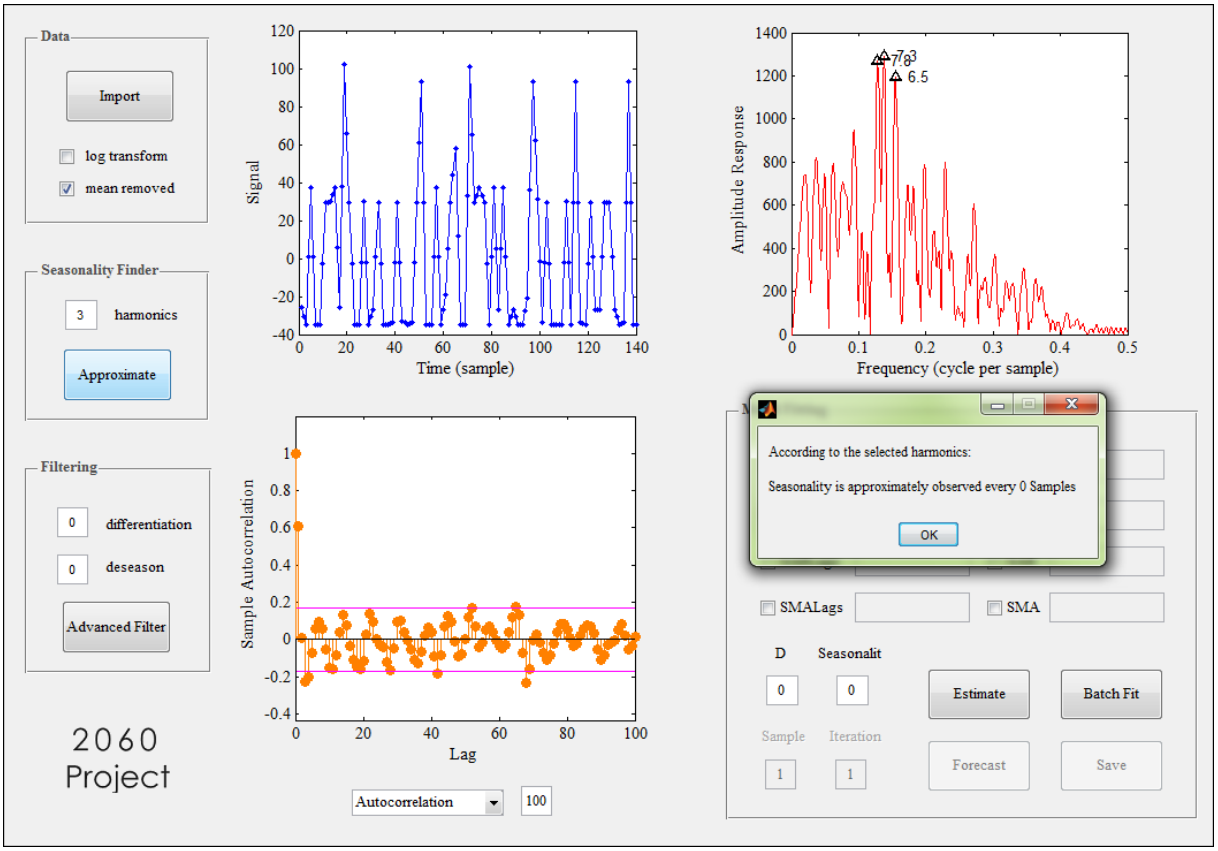


Figura 17 - Verificação de sazonalidade na interface TSAF

Ao clicar em '*Batch Fit*' é feito o ajuste do modelo conforme o menor valor do BIC, como mostra a figura 18. Foram inseridos os valores de amostras para serem geradas na previsão no campo '*Sample*' e selecionada a opção '*Forecast*'.

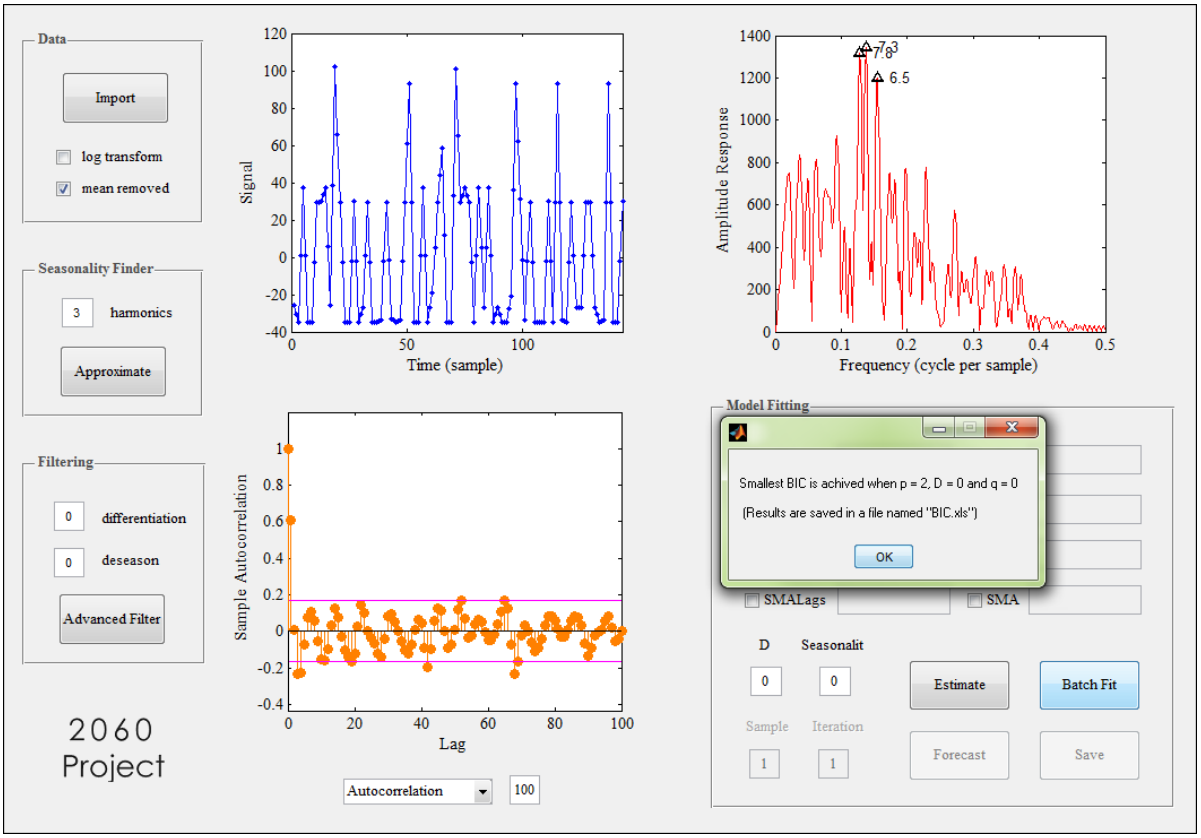


Figura 18 - Aproximação do modelo pela interface TSAF

3.2. Criação da Tabela Externa no banco de dados

Com o propósito de unificar as informações no banco de dados e para facilitar a comunicação com as ferramentas de conectividade oferecidas pela empresa, foi criada uma tabela externa no banco de dados ORACLE que suporta a aplicação RIS-x. A tabela externa é muito relevante para a manutenção automática funcionar de maneira adequada, pois ela armazena o valor do espaço livre da partição onde se encontram os *datafiles*.

Inicialmente foram criadas duas pastas TEMP e BIN, na raiz C:\. Na pasta BIN foi inserido um *script*, arquivo DISK_FREE.BAT, com o objetivo de retornar o valor do espaço livre na partição E:\, local padrão para a gravação de *tablespaces,* como mostra a figura 19.

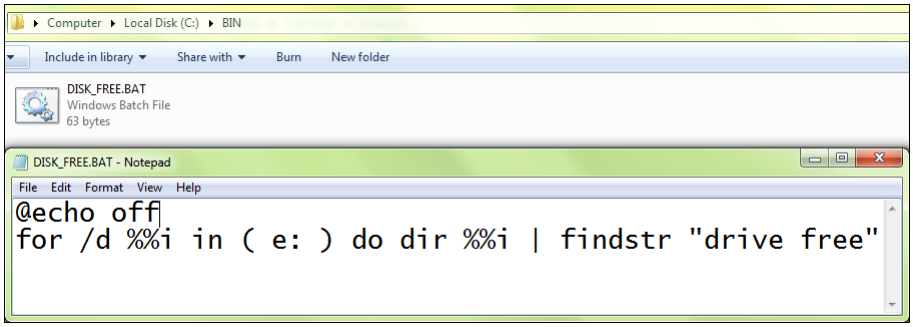


Figura 19 - *Script* para identificar espaço livre na partição

Foi feito o vínculo dos diretórios de *log* às pastas criadas, conforme mostra o *script* SQL na figura 20.

```
CREATE directory log_dir as 'c:\TEMP';
CREATE directory bin_dir as 'c:\BIN';
```

Figura 20 - *Script* para vínculo dos diretórios de *logs* às pastas criadas

Após isso, criou-se um usuário no banco de dados e foram concedidas a ele as permissões de escrita e leitura no diretório log_dir e permissão de execução no diretório bin_dir, de acordo com o *script* SQL da figura 21. A tabela externa foi criada seguindo o *script* PL/SQL da figura 22.

```
GRANT READ, WRITE on directory log_dir to GEHCIT;
GRANT EXECUTE on directory bin_dir to GEHCIT;
```

Figura 21 - *Script* para atribuir permissão ao usuário

```
create table GEHCIT.windows_disk_free
(line varchar2(100)
)
organization external
(
    type oracle_loader
    default directory log_dir
    access parameters
    (
        records delimited by newline
        preprocessor bin_dir: 'DISK_FREE.BAT'
        fields terminated by "~"
    )
    location (log_dir:'DUMMY.DAT')
)
reject limit unlimited;
```

Figura 22 - *Script* para criação da tabela externa

Após a criação da tabela externa, o valor do espaço livre na partição é armazenado, unificando as informações necessárias e, desta forma, facilitando a comunicação estabelecida pelo uso do aplicativo de manutenção da *tablespace* USERS.

4. ANÁLISE DA SÉRIE E DO MODELO OBTIDO

Aplicando o TSAF na série obtida para o hospital A obteve-se o modelo ARIMA(2,0,0) como retorno do menor valor do BIC entre os modelos testados. Como o modelo apresenta estacionariedade, seu fator integrador corresponde a zero, não há parâmetros em MA indicando que não há dependência entre os erros. Pode-se observar então que o modelo é um AR(2). A figura 23 mostra as características do modelo como média, parâmetros AR e variância.

ARIMA(2,0,0) Model:

Conditional Probability Distribution: Gaussian

Parameter	Value	Standard Error	t Statistic
Constant	61.0599	7.07801	8.62672
AR{1}	0.967578	0.0608428	15.9029
AR{2}	-0.583613	0.0716221	-8.14851
Variance	533.608	78.7134	6.77913

Figura 23 - Modelo ARIMA(2,0,0) - Hospital A

Os gráficos da FAC e FACP da série apresentam proximidade nos comportamentos esperados a um modelo AR(2), conforme figuras 24 e 25. A FAC apresenta a mistura de exponenciais com ondas senoidais amortecidas e a FACP mostra as lags 1 e 2 como as mais significativas.

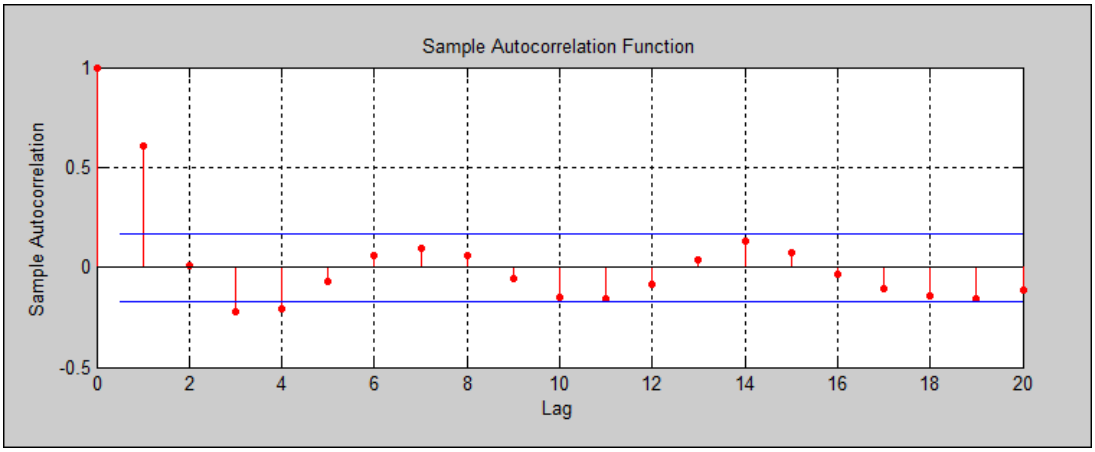


Figura 24 - Gráfico da FAC da série obtida

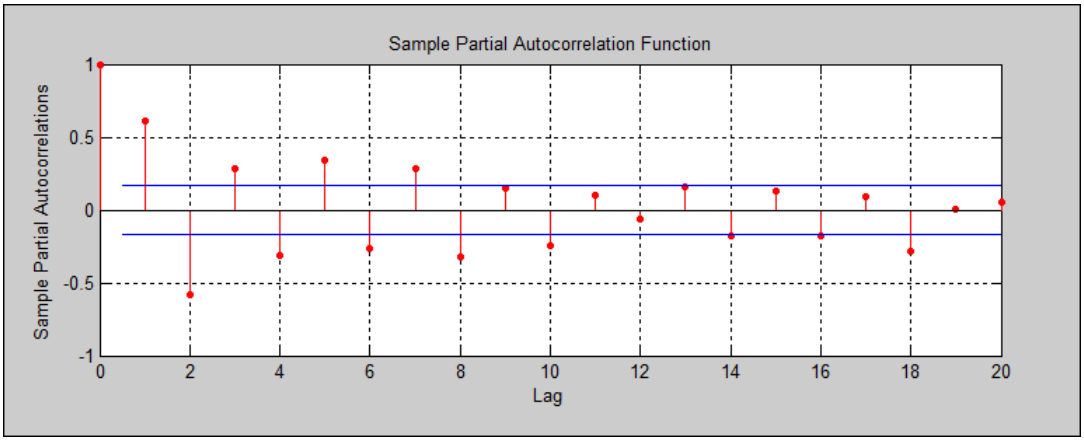


Figura 25 - Gráfico da FACP da série obtida

A simulação com o modelo obtido, usando as funções *arima* e *simulate* do MATLAB, foi realizada e o gráfico correspondente é mostrado na Figura 26. Observa-se o acompanhamento e a variação da série obtida pelo modelo simulado em torno de uma média próxima da real.

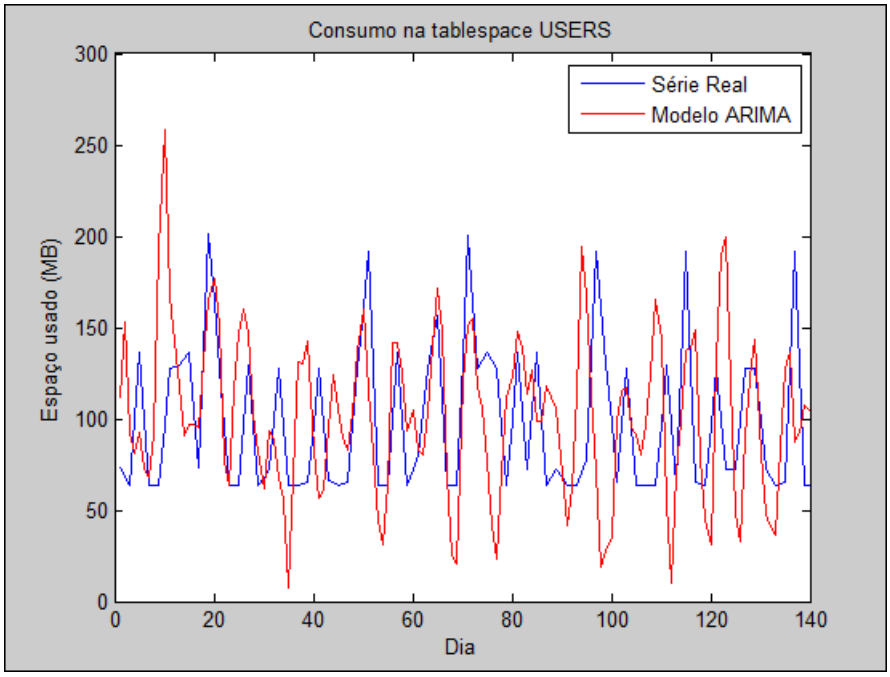


Figura 26 - Simulação consumo da *tablespace* USERS

Os gráficos correspondentes aos resíduos e sua distribuição são mostrados nas Figuras 27 e 28. Observa-se o gráfico com valores variando em torno de uma média próxima de zero, sendo que o histograma apresenta uma aproximação à distribuição normal. Sendo assim, o modelo pode ser considerado para representar a série.

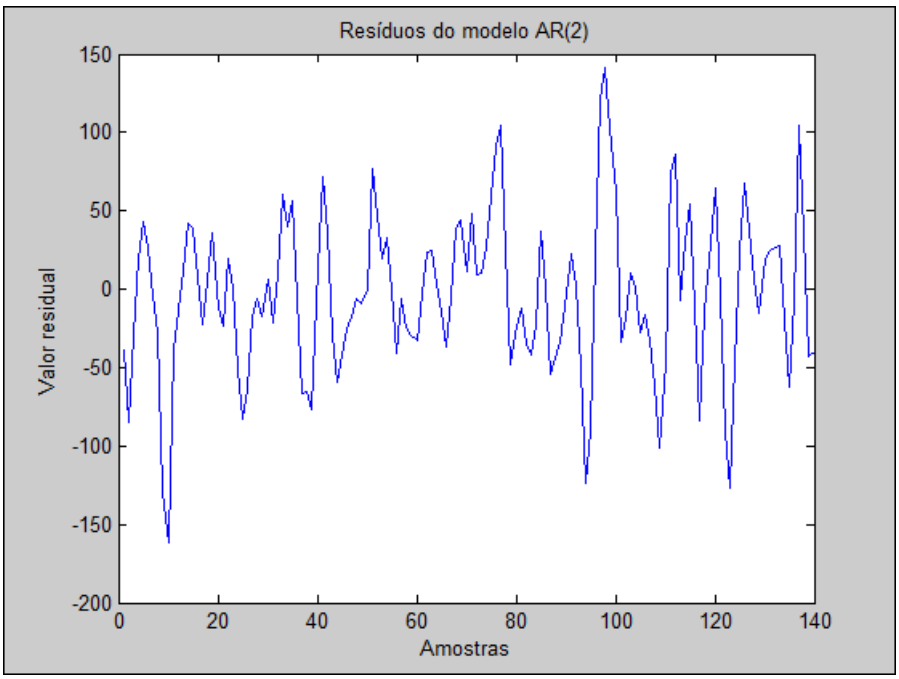


Figura 27 - Gráfico dos resíduos do modelo AR(2)

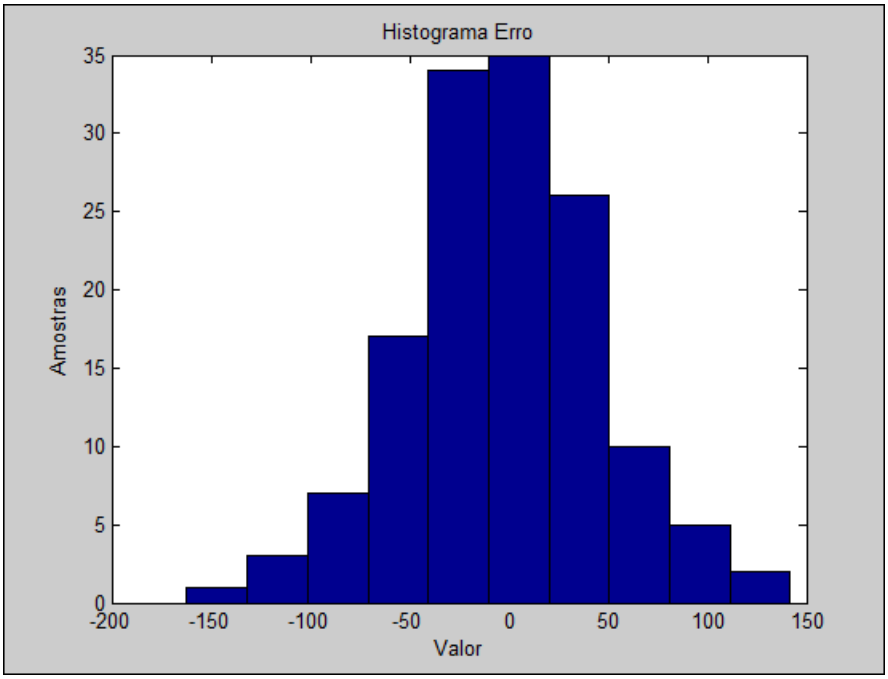


Figura 28 - Distribuição dos resíduos

5. APLICATIVO DE MANUTENÇÃO DA *TABLESPACE USERS* NO SISTEMA RIS-x

Foi desenvolvido um aplicativo para manutenção da *tablespace* do banco de dados do sistema RIS-x em linguagem JAVA utilizando a plataforma *Netbeans*. O propósito geral da aplicação é realizar a verificação diariamente, sendo acionada por um *script* com agendamento programado no sistema operacional (*Task Scheduler*). Esses passos são mostrados no diagrama de blocos da figura 29 e, quando estiver na data prevista, efetua a manutenção. Com o intuito de realizar testes, o aplicativo foi acionado de forma manual diariamente, para acompanhar as rotinas de verificação e manutenção elaboradas.

O objetivo do aplicativo é realizar o monitoramento e a manutenção em *background*, ou seja, sem que haja a percepção do usuário e sem prejudicar o desempenho do sistema ou de sua conectividade. O acesso ao servidor do sistema RIS-x é restrito, o cliente não possui conexão direta ao banco de dados e permissão às tabelas e *views* destinadas para uso do DBA. Assim, para efeitos de demonstração, foi elaborada uma interface simples para retornar as informações geradas pelas rotinas realizadas pelo aplicativo.

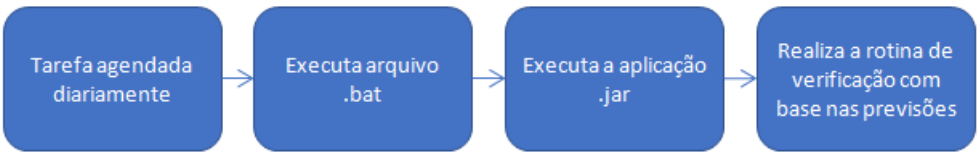


Figura 29 - Etapas iniciais para acionamento da aplicação

A primeira vez que a aplicação é executada, os valores de previsão gerados são carregados e arquivos auxiliares armazenam informações como a data prevista para manutenção, o espaço livre na *tablespace* USERS, que é atualizado após manutenção, e o índice de referência da última amostra de previsão que foi lida. Esse índice de referência é usado quando o aplicativo precisa gerar uma nova data para manutenção, coletando como base a amostra posterior ao índice armazenado. Esses arquivos

auxiliares podem ficar alocados no disco local do computador com conexão remota ou no disco local do servidor.

Nas demais execuções, se a data atual corresponde à data prevista para manutenção, é realizada a comunicação com o banco de dados e verifica-se a disponibilidade de espaço na partição, de acordo com o diretório padrão para gravação dos *datafiles*. Se houver espaço disponível para o recebimento do *datafile*, é feita a manutenção da *tablespace*. A rotina da aplicação consiste, resumidamente, nas etapas mostradas pelo fluxograma na figura 30.

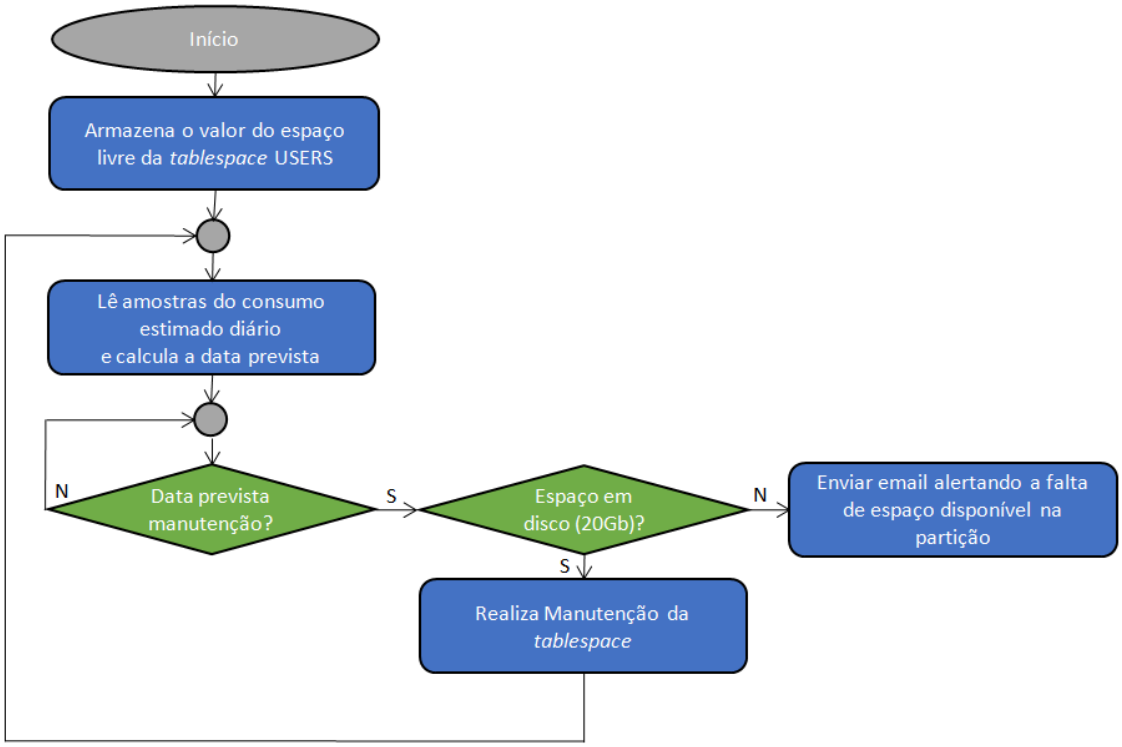


Figura 30 - Etapas realizadas pela aplicação

As etapas de manutenção são apresentadas na figura 31. O aplicativo realiza a comunicação com o banco de dados pelo JDBC, conector desenvolvido para o envio de instruções SQL para um banco de dados relacional. Caso não tenha espaço disponível para a adição de um *datafile*, é enviado um *email* para a equipe responsável, com a finalidade de comunicar ao cliente para providenciar a adição de mais discos.

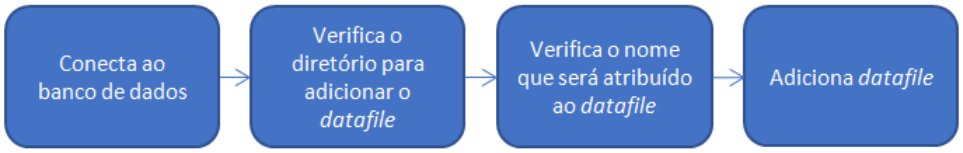


Figura 31 - Etapas da manutenção

A figura 32 mostra a imagem da tela do aplicativo quando está realizando a verificação diária. A tela mostra informações sobre o modo de monitoramento, a data atual, a data prevista para adicionar o *datafile* e o status com o banco de dados. Os indicadores da *tablespace* são mostrados também na imagem. Neste caso, observa-se que o percentual usado já ultrapassou 90%, porém, devido ao baixo consumo ainda não há necessidade de adicionar um *datafile*.

	TABLESPACE_NAME	PERCENT_USED	PCT_USED	ALLOCATED	USED	FREE	DATAFILES
1	USERS		93.15	17312	16126.44	1185.56	7
2	UNDO		78.36	1024	802.38	221.63	1
3	SYSAUX		53.13	1024	544	480	1
4	SYSTEM		52	1024	532.44	491.56	1
5	USERS1		33.55	8096	2716.31	5379.69	3
6	TEMP		(null)	(null)	(null)	0	(null)

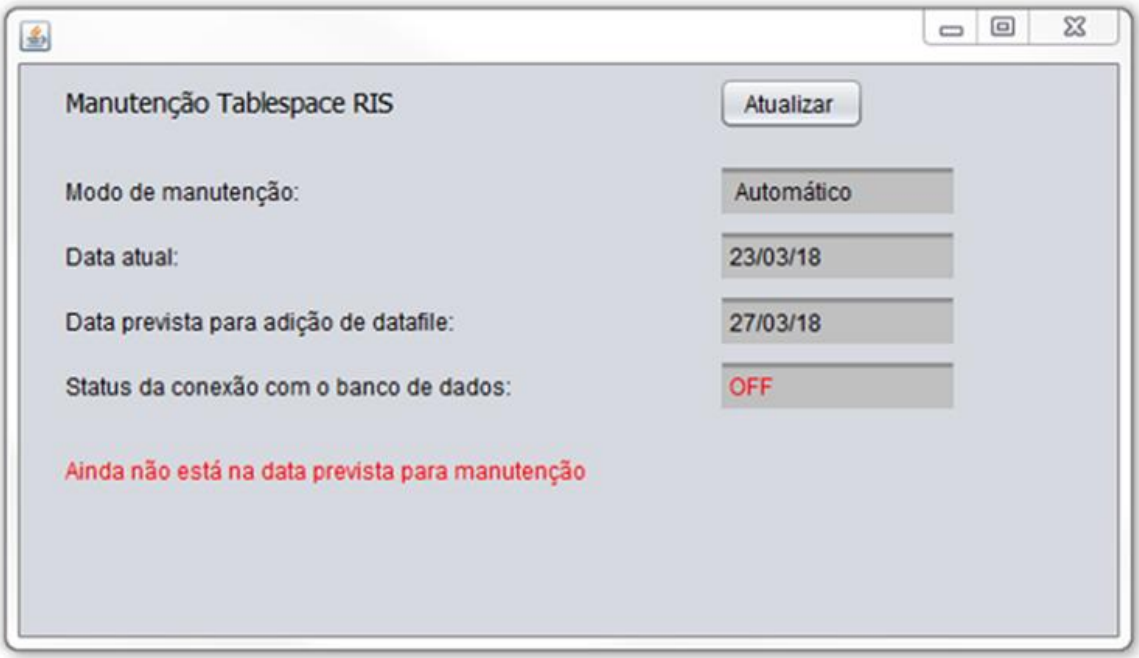
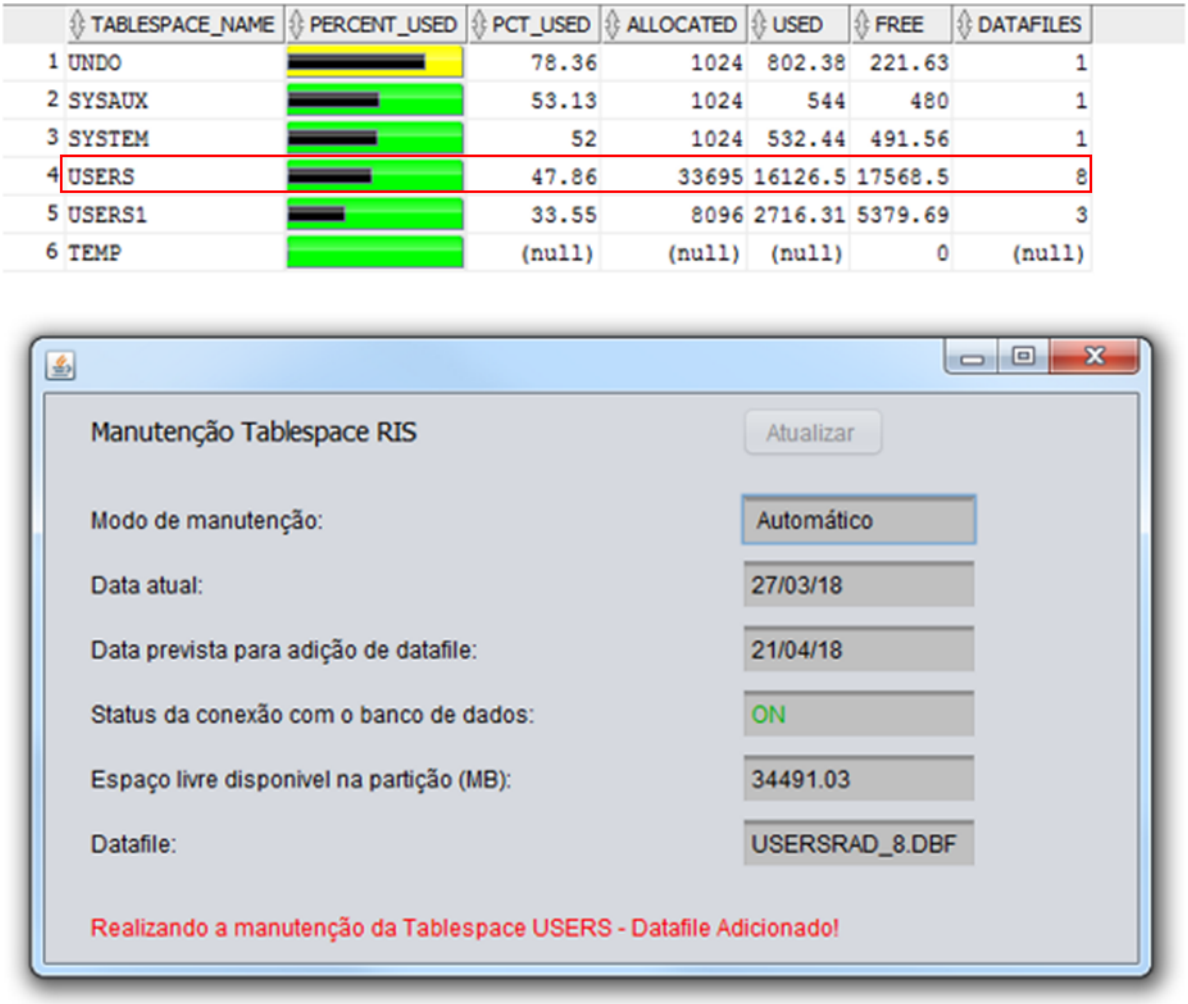


Figura 32 - Verificação da data prevista para manutenção da *tablespace* USERS

A figura 33 mostra informações sobre o modo de manutenção realizado na data prevista e a atualização direta na *view* que mostra as *tablespaces* do banco de dados. São mostrados os dados que aparecem na tela de verificação, assim como o espaço livre na partição e o nome do *datafile* que foi adicionado.



	TABLESPACE_NAME	PERCENT_USED	PCT_USED	ALLOCATED	USED	FREE	DATAFILES
1	UNDO		78.36	1024	802.38	221.63	1
2	SYSAUX		53.13	1024	544	480	1
3	SYSTEM		52	1024	532.44	491.56	1
4	USERS		47.86	33695	16126.5	17568.5	8
5	USERS1		33.55	8096	2716.31	5379.69	3
6	TEMP		(null)	(null)	(null)	0	(null)

Figura 33 - Manutenção da *tablespace* USERS

As tabelas abaixo mostram as descrições sobre os métodos adotados no projeto e suas respectivas classes. A tabela 2 descreve os métodos da classe *File* que se relaciona com a leitura e escrita de arquivos. A tabela 3 mostra os métodos da classe *Database* que envolve a manutenção. Por último, a tabela 4 apresenta a classe *Check* que é responsável pelo ajuste das datas na rotina de verificação.

Tabela 2 - Descrição dos métodos da classe *File*

Método	Descrição
ReadFileForecast()	Lê arquivo com informações de previsão
ReadFileTSFreeSpace()	Lê arquivo com o valor do espaço disponível na TS em MB
PrintReadFileForecast()	Retorna o valor de previsão lido
PrintReadFileForecastIndex()	Retorna o índice de previsão lido
PrintReadFileTSFreeSpace()	Retorna o valor do espaço livre na *tablespace*
WriteFileTSFreeSpace()	Armazena o valor do espaço livre na *tablespace*
WriteFileForecastIndex()	Armazena índice de previsão lido
WriteFileForecastDate()	Armazena a data de previsão para manutenção

Tabela 3 - Descrição dos métodos da classe *Database*

Método	Descrição
ConnectionDB()	Realiza conexão com DB e autenticação de usuário
SelectFreeSpaceE()	Verifica espaço livre na partição que armazena os *datafiles*
SelectFreeSpaceTS()	Verifica espaço livre na *tablespace* USERS
Select*Datafile*Dir()	Verifica diretório do *datafile*
Get*Datafile*()	Mostra *datafile* atual
Alter*Datafile*Dir()	Altera a *tablespace* USERS adicionando um novo *datafile* no diretório especificado

Tabela 4 - Descrição dos métodos da classe *Check*

Método	Descrição
CurrentDate()	Verifica data atual
FutureDate()	Verifica a data prevista para manutenção
PrintDate()	Retorna a data futura

Os métodos mencionados nas tabelas são utilizados na classe principal *JFrame*. A relação entre as classes pode ser vista no diagrama de classes (apêndice A). A classe *JavaMailApp* foi adaptada da API *JavaMail* [13] e é responsável pelo envio de emails utilizando o protocolo SMTP.

6. CONCLUSÃO

A realização da rotina de manutenção automática com base no modelo obtido e nas condições descritas, para o hospital A, tem mostrado resultados satisfatórios e de acordo com o esperado. Foi feito o acompanhamento da adição de um *datafile*, no modo de manutenção automático, e o mesmo foi adicionado na data prevista, prevenindo a ocupação total da *tablespace* USERS e a ocorrência do erro ORA-01653.

Pretende-se utilizar o aplicativo desenvolvido em todas as instalações do sistema RIS-x da América Latina após aprovação interna. Os erros causados pela manutenção manual geram impacto ao fluxo de trabalho nas instituições de saúde, causando prejuízos e reclamações dos clientes. Entretanto, além de contribuir para melhorias relacionadas à disponibilidade do sistema e a economia de tempo na realização das tarefas de manutenção da *tablespace*, a aplicação pode retornar maior satisfação dos clientes sobre produtos e serviços oferecidos pela empresa.

Considerando uma futura melhoria, seria interessante a implementação de um algoritmo que realizasse a adequação contínua do modelo com base em amostras novas que seriam coletadas diariamente pelo aplicativo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Deshpande, P.S. “SQL & PL/SQL for Oracle 10g Black Book”, Dreamtech Press, 2006.

[2] Huang, H.K. “PACS and Imaging Informatics: Basic Principles and Applications”, Wiley, 2010.

[3] Bidgood, D.W e Horii, S.C. “Introduction to ACR-NEMA DICOM standard”. Radiographics, 1992.

[4] Puga, S., França, E. e Goya, M. “Banco de Dados”, Pearson, 2014.

[5] Priestley, M. B. “*Discussion of the paper by Professor Manridakis and Dr. Hibon*”, Journal of the Royal Statistical Society, 1979.

[6] Moretin, P. A. e Toloi C. M. C. “Análise de Séries Temporais”, Edgard Blücher, 2006.

[7] Box, G. E. P., Jenkins, G. M. e Reinsel, G. C. “Time Series Analysis”, Willey, 2013.

[8] Kumar, R. A. “Easy Oracle Automation: Oracle 10g Automatic Storage, Memory and diagnostic Features”, Rampant TechPress, 2004.

[9] ORACLE: DBA’s Day Out, 2009 – Estimate *Tablespace* Growth. Disponível em: <shaharear.blogspot.com.br/2009/09/estimate-*tablespace*-growth.html>, acesso em 12/09/2018.

[10] Ault, M. Liu, D. e Tumma, M. “Oracle Database 10g New Features: Oracle 10g Reference for Advanced Tuning & Administration”, Rampant TechPress, 2003.

[11] MATLAB and Time Series Analysis and Forecast Toolbox Release 2013b, The MathWorks, Inc., Natick, Massachusetts, United States.

[12] Schwarz, Gideon E. "Estimating the dimension of a model". 1978.

[13] JavaMail API – Oracle, 2017. Disponível em: <http://www.oracle.com/technetwork/java/javamail/index.html>, acesso em 15/02/2018.

APÊNDICE A

A1. Diagrama de Classes

Figura 34 - Diagrama de classes do aplicativo

A2. Métodos implementados na classe *File*

```
public void ReadFileForecast() throws IOException {
    try {
        FileReader file = new FileReader(dirRead);
        FileReader index = new FileReader(dirRead_index);
        BufferedReader rd = new BufferedReader(file);
        BufferedReader rd_index = new BufferedReader(index);
        index_c = rd_index.readLine();
        c = Integer.parseInt(index_c);
        for (int j = 0; j < c; j++) {
            lt = rd.readLine();
        }
    } catch (NullPointerException ex) {
        System.out.printf("Último valor do arquivo. Carregar arquivo");
        return;
    }
    line = lt;
    c = c + 1;
}

public Object PrintReadFileForecast() {
    return (line);
}

public int PrintReadFileForecastIndex() {
    return (c);
}

public void WriteFileForecastIndex() throws IOException {
    FileWriter fileWrite_c = new FileWriter(dirWrite_c);
```

```
    BufferedWriter wrc = new BufferedWriter(fileWrite_c);
    wrc.write(String.valueOf(c));
    wrc.close();
}

public void WriteFileForecastDate(String dataF) throws IOException {
    FileWriter fileWrite_d = new FileWriter(dirWrite_d);
    BufferedWriter wrd = new BufferedWriter(fileWrite_d);
    wrd.write(dataF);
    wrd.close();
}

public void ReadFileTSFreeSpace() throws FileNotFoundException, IOException {
    FileReader file = new FileReader(dirRead_TS);
    BufferedReader rdts = new BufferedReader(file);
    tsfs = rdts.readLine();
}

public void WriteFileTSFreeSpace(String freeSpace) throws IOException {
    FileWriter fileWrite_TS = new FileWriter(dirWrite_TS);
    try (BufferedWriter wrts = new BufferedWriter(fileWrite_TS)) {
        wrts.write(freeSpace);
    }
}

public Object PrintReadFileTSFreeSpace() {
    return (tsfs);
}
```

A3. Métodos implementados na classe Database

```
public void ConnectionDB() {
    try {
        Class.forName("oracle.jdbc.OracleDriver");
        conn = DriverManager.getConnection(DB_URL, USER, PASS);
        stmt = conn.createStatement();
        System.out.println("\nConexao DB Realizada com Sucesso!");
    } catch (SQLException | ClassNotFoundException | NullPointerException se) {
        System.out.println("Falha na Conexão");
    }
  }

  public void Select*Datafile*Dir() {
    try {
        sql = "select * from dba_data_files where *tablespace*_name='USERS'order
by 1 desc";
        rs = stmt.executeQuery(sql);
        while (rs.next()) {
            String diret = rs.getString("file_name");
            int id = rs.getInt("file_id");
            if (id > idmax) {
                idmax = id;
                nameidmax = diret;
            }
            id_count = id_count + 1;
        }
    } catch (SQLException | NullPointerException ee) {
        System.out.println("Falha ao realizar a busca  ");
    }
  }
```

```
public double SelectFreeSpaceE() throws SQLException {
    try {
        sql = "SELECT * FROM GEHCIT.windows_disk_free";
        rs1 = stmt.executeQuery(sql);
        while (rs1.next()) {
            freespace = rs1.getString("line");
        }
        freespace = freespace.replaceAll(" ", "[^,.]");
        freespace = freespace.replaceAll("[^0-9]", "");
        freespace = freespace.substring(1, freespace.length());
        fs = Double.parseDouble(freespace);
    } catch (SQLException | NullPointerException ee) {
        System.out.println("Falha ao realizar a busca - DISKSPACE");
    }
    fs = Math.round(fs) / Math.pow(10, 6);
    fs = Double.valueOf(df.format(fs));
    return (fs);
}

public void SelectFreeSpaceTS() {
    try {
        sql = "select (max(TABLESPACE_SIZE/256)-
max(TABLESPACE_USEDSIZE/256)) AS FREESPACE_TS from
DBA_HIST_TBSPC_SPACE_USAGE WHERE TABLESPACE_ID='4' group by
TABLESPACE_ID";
        rs2 = stmt.executeQuery(sql);
        while (rs2.next()) {
            freespace = rs2.getString("FREESPACE_TS");
        }
    } catch (SQLException | NullPointerException ee) {
```

```
            System.out.println("Falha ao realizar a busca - FREESPACE_TS");
        }
    }

    public void Alter Datafile Dir() throws SQLException {
        try {
            String dir = "E:\\ORACLE\\ORADATA\\RAD\\USERSRAD_" + id_count +
".DBF";
            sql1 = "alter tablespace users add datafile '" + dir + "' size 16383M";
            System.out.println("Tentando adicionar o datafile - ID_DIR - " + sql1);
            alt = stmt.executeUpdate(sql1);
            System.out.println("Datafile adicionado!");

        } catch (SQLException | NullPointerException ee) {
            System.out.println("Falha ao adicionar o datafile");
        }
    }

    public String Get Datafile() {
        datafile_name = "USERSRAD_" + id_count + ".DBF";
        return (datafile_name);
    }
```

A4. Métodos implementados na classe *Check*

```
public String FutureDate(int ndays) {
    dateFormat = new SimpleDateFormat("dd/MM/yy");
    date = new Date();
    calend = new GregorianCalendar();
    data = dateFormat.format(date);
    calend.add(Calendar.DAY_OF_MONTH, ndays);
    dataF = dateFormat.format(calend.getTime());
    return (dataF);
  }

  public String CurrentDate() {
    dateFormat = new SimpleDateFormat("dd/MM/yy");
    date = new Date();
    calend = new GregorianCalendar();
    data = dateFormat.format(date);
    calend.get(1);
    dataC = dateFormat.format(calend.getTime());
    return (dataC);
  }

  public String PrintDate(String dataF) {
    String dateF = dataF;
    return (dateF);
  }
```