UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

TARCISIO TROJAN COELHO

APLICATIVO PARA RESOLUÇÃO DE PROBLEMAS DE MÁQUINAS INDUSTRIAIS

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2017

TARCISIO TROJAN COELHO

APLICATIVO PARA RESOLUÇÃO DE PROBLEMAS DE MÁQUINAS INDUSTRIAIS

Monografia de Especialização apresentada ao Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de Especialista em Desenvolvimento para Dispositivos Móveis.

Orientador: Robson Ribeiro Linhares

CURITIBA

2017

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Coordenação do Curso de Especialização em Desenvolvimento para Dispositivos Móveis

TERMO DE APROVAÇÃO

“Aplicativo para Resolução de Problemas de Máquinas Industriais”

por

“Tarcísio Trojan Coelho”

Este Trabalho de Conclusão de Curso foi apresentado às 19:38 do dia 18 de dezembro de 2017 na sala B201 como requisito parcial à obtenção do grau de Especialista em Desenvolvimento para Dispositivos Móveis na Universidade Tecnológica Federal do Paraná - UTFPR - Campus Curitiba. O(a) aluno(a) foi arguido pela Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho aprovado.

_____
Prof. Robson Ribeiro Linhares
(Presidente/Orientador - UTFPR/Curitiba)

_____
Profa. Maria Cláudia Figueiredo Pereira Emer
(Avaliador 1 – UTFPR/Curitiba)

_____
Prof. Marco Aurélio Wehrmeister
(Avaliador 2 – UTFPR/Curitiba)

“A Ata de Aprovação assinada encontra-se na Coordenação do Curso.”

Resumo

Coelho, Tarcisio T., Aplicativo para Resolução de Problemas de Máquinas Industriais. 2017. 59f. Trabalho de Conclusão de Curso de Especialização (Desenvolvimento para Dispositivos Móveis) - Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

A manutenção e o ajuste de máquinas nas indústrias podem enfrentar problemas recorrentes, alarmes de máquinas muitas vezes têm soluções conhecidas, mas depender da memória de quem achou a solução pode ser uma atitude arriscada. Desenvolver um aplicativo para dispositivos móveis que seja integrado com banco de dados e que sincronize com outros celulares da mesma empresa, que seja organizado de forma que quem usar, saiba chegar até a solução de forma rápida e direta, pode ser uma solução eficaz contra alarmes e defeitos reincidentes, assim como pode ajudar nos ajustes das máquinas. O aplicativo do dispositivo móvel foi desenvolvido para a plataforma Android e o programa do servidor usado para sincronizar com outros celulares foi desenvolvido em Java. Integração com banco de dados SQL foi fundamental para atingir a organização esperada. Diagramas UML baseados no padrão MVC (Model-View-Controller) serviram de base para o desenvolvimento do aplicativo. O aplicativo foi muito bem elogiado quanto ao que promete facilitar na questão de manutenção e ajuste de máquinas. Os usuários que testaram o aplicativo afirmaram que é um aplicativo simples e direto, bem estruturado e funcional e que certamente terá muito apoio para que seja alimentado com as informações já existentes de para ajudar nas manutenções futuras. Conclui-se que, apesar de ser um aplicativo que entrega o esperado, necessita de alguém para alimentá-lo com informações e pode ser melhorado de forma que tenha integração com planilha de manutenções já existente ou possibilidade de carregar fotos no banco de dados.

Palavras chave: manutenção, banco de dados, ajuste de máquinas, android.

Abstract

Coelho, Tarcisio T., Industrial machines troubleshooting App. 2017. 59f. Trabalho de Conclusão de Curso de Especialização (Desenvolvimento para Dispositivos Móveis) - Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Industrial machines adjustment and maintenance can face repeated problems, machine alarms often have known solutions, but counting on men's ability to remember can be risky. Developing an mobile app integrated to database and being able to sync with other users at the same enterprise and being well organized, can be a great solution for those who are facing repeated problems at complex machines and for those who need a "step-by-step" guide easy to use. The app was developed to the Android mobile platform and the server program used to help sync with other smartphones using the same app was developed using Java technology. SQL database helped a lot to implement the organization required. MVC based UML diagrams were the fundamentals to the app development. Because of the promise of helping at maintenance and machine setup, good praises have been said to the application. Those who tested the app said it is a simple app, well structured and surely they want to have somebody in charge of feeding the app to help solving future problems. In conclusion, although it is an app that delivers the expected, it needs to be fed by someone with informations and can be improved with integration to spreadsheets or possibility of loading photos in the database.

Keywords: maintenance, database, machines setup, Android.

Sumário

1 INTRODUÇÃO

Neste capítulo será apresentada a proposta do aplicativo, em quê contexto ele se insere, o objetivo para o qual ele foi criado, assim como sua justificativa e o escopo de criação e a metodologia usada.

1.1 CONTEXTO

Para aumentar a produção, o ser humano criou máquinas, que, com as revoluções industriais, se tornaram cada vez mais automatizadas. Com a automação e a flexibilidade de cada máquina, ficou muito mais fácil produzir vários modelos do mesmo produto praticamente na mesma máquina. Para produzir um modelo diferente, algumas características da máquina têm que ser alteradas. (Automação Produção, 2018).

No meio industrial, essa mudança na máquina para produzir um modelo diferente é chamado de *setup.* Quando a máquina é instalada, uma das formas de garantir um bom *setup* é que algum especialista no processo realize essa alteração e ensine os operadores da máquina. Após algum tempo, quem executa essa alteração na máquina geralmente é a pessoa que opera a máquina, utilizando os conhecimentos passados pelo especialista. Então grande parte do conhecimento sobre a máquina é passado no dia a dia com a convivência com a máquina. Sendo assim, o conhecimento passa de um especialista para alguém que está trabalhando na máquina há pouco tempo, e dessa para outra pessoa que também não tem experiência na máquina. (Setup de máquina, 2018).

Se o número de informações que uma pessoa tem que passar para a outra é grande, a possibilidade de algumas informações não serem passadas aumenta. Sendo assim, a possibilidade de o novo operador não ter o conhecimento necessário a respeito de todas as partes da máquina também aumenta.

Quanto maior e mais complexa a máquina, mais detalhes existem para serem ajustados e maior também o número de itens a serem verificados. Com isso aumenta a dificuldade das pessoas lembrarem de todos os detalhes, de todas as formas de resolver os problemas.

Uma das formas utilizadas para passar o conhecimento sobre a manutenção de certa máquina para outras pessoas é através de documentos gerados nas manutenções passadas. Em cada manutenção nova, é criado um documento para registrar o que foi feito. Uma das importâncias de documentar um projeto é manter um registro fiel das atividades. Assim, caso seja necessário passar essas informações para outra pessoa, ela poderá se inteirar de todos os acontecimentos e de todas as informações até agora necessitadas e já cadastradas. (Importância da Documentação, 2018).

Procurar um documento para se informar sobre o que deve ser feito pode demorar mais do que o esperado para realizar certos procedimentos. É comum na indústria cada máquina ter uma interface homem-máquina (IHM), porém esta é utilizada para interação com a máquina de forma automatizada. Quanto maior a complexidade de um produto, mais difícil é a identificação de defeitos ou erros de processo através de sensores, pois seriam necessários muitos sensores para analisar tantos detalhes.

A organização de documentos através de banco de dados facilita a consulta de um documento ou procedimento para realizar um procedimento específico, porém em certas ocasiões o uso de dispositivos móveis pode tornar a manutenção mais fácil pelo fácil acesso a informações que antes estavam em um computador que se encontra longe da máquina.

A empresa XYZ situada em Curitiba, no Paraná, possui máquinas automatizadas que, dependendo da demanda, precisam realizar setups mais de dez vezes por dia. A fim de auxiliar os operadores e o time de manutenção, foi definido um projeto para criação de plataformas móveis de auxílio na resolução de problemas.

1.2 OBJETIVO

1.2.1 Objetivo Geral

Criar um aplicativo que auxilie o operador de determinada máquina a resolver problemas a partir do histórico de manutenções.

1.2.2 Objetivos específicos

- Obter os dados das manutenções realizadas para disponibilizar na aplicação móvel;
- Desenvolver um protótipo com integração a um banco de dados para armazenar e organizar os dados inseridos pelos usuários;
- Desenvolver um WebService para realizar a comunicação entre os dispositivos móveis e o servidor do banco de dados*;*
- Avaliar com o usuário final se o aplicativo atende a necessidade para acesso aos dados.

1.3 JUSTIFICATIVA

Se existe uma grande quantidade de informações úteis a serem armazenadas e posteriormente requisitadas pelo operador da máquina na hora de resolver um problema, considerando que o uso de um dispositivo móvel pode facilitar o acesso a soluções utilizadas por manutenções realizadas anteriormente, não é interessante para a empresa continuar usando cadernos e formulários para registro de informações e consulta de dados. O uso de dispositivos móveis pode tornar a manutenção mais rápida, permitir que cada vez mais soluções sejam adicionadas no mesmo sistema e tornar o registro mais fácil.

Com o acesso móvel a dados sobre manutenção e com o armazenamento de resultados em banco de dados, o operador poderá realizar uma consulta no exato lugar em que estiver na máquina, sem precisar se deslocar dezenas de metros até ter acesso a um computador. Como o registro dos dados no aplicativo será em banco de dados, a informação não precisará ser passada pessoalmente para o operador do turno seguinte, pois assim que este precisar realizar uma consulta para resolver certo problema, a informação já estará disponível para ele.

As vantagens do uso de banco de dados no lugar de uma planilha são:

- A estrutura de organização: diferente de uma planilha, que o usuário pode inserir qualquer dado em qualquer célula, o banco de dados é mais estruturado, sendo impossível o usuário inserir um tipo de texto diferente do formatado pela tabela.

- A robustez de ferramentas de análise e processamento de dados: bancos de dados conseguem analisar e processar dados com ferramentas mais robustas, enquanto planilhas não são muito eficientes em trabalhar com grande quantidade de dados ao mesmo tempo.
- A visualização de dados e a facilidade de acesso: enquanto uma planilha, assim que ela é carregada, apresenta ao usuário todas as informações que nela estão, o banco de dados apresenta somente as informações que são pertinentes ao que o usuário procura saber.

1.4 ESCOPO

Como o aplicativo funcionará:

- O aplicativo possibilitará procurar algum dado de manutenção por meio do nome da máquina e da parte específica da máquina em questão;
- Ao selecionar a informação para auxiliar no *setup* da máquina, o usuário poderá adicionar algum comentário ou dado importante que esteja faltando naquelas informações, assim como classificar as informações como útil ou não;
- O sistema será compatível somente com a plataforma Android;
- A adição ou remoção de itens na lista (exceto na parte de comentários) será realizada pelo administrador do sistema, que será responsável por manter as informações organizadas.

O sistema não possibilitará:

- Chat com outros usuários especialistas;
- Carregar gravações de áudio ou vídeo para o banco de dados;
- Adicionar ou remover itens nas listas.

1.5 METODOLOGIA

A metodologia utilizada para o projeto envolve as seguintes ações:

- Revisão dos fundamentos teóricos com ênfase em livros ou material online relacionado a, programação de aplicativos em *Android,* banco de dados SQL para *Android* e programação em Java;

- Implementação do aplicativo utilizando ferramentas gratuitas para programação de dispositivos móveis.
- Testes em aparelho próprio para avaliação de primeira instância, entrevista com o público-alvo para coleta de sugestões de melhoria do aplicativo para dispositivo móvel e do aplicativo para computador.

1.6 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em quatro capítulos. Sendo o primeiro capítulo a introdução, o segundo capítulo possui a fundamentação teórica. Na fundamentação serão expostas as ferramentas usadas para desenvolvimento do aplicativo, assim como a metodologia de desenvolvimento.

O capítulo três se encarrega de explicar como o aplicativo foi organizado, quais foram os requisitos do sistema, como o sistema foi modelado pelo UML, envolvendo casos de uso e diagramas de classe, de comunicação e de atividades, terminando com a implementação do aplicativo em si. Os códigos mais importantes para o desenvolvimento do aplicativo e o resultado da avaliação com usuários também estão nesse capítulo.

O último capítulo, composta pela conclusão, expõe os resultados obtidos, assim como as possíveis melhorias ou ideias para futuros aplicativos e as dificuldades encontradas no desenvolvimento do aplicativo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentada a fundamentação teórica, baseada em monitoramento de máquinas industriais e ferramentas usadas para a programação do protótipo, como banco de dados, UML e WebServices.

2.1 MONITORAMENTO DE MÁQUINAS INDUSTRIAIS

Em uma máquina industrial, quando ocorre alguma falha, é gerado um alarme na IHM (interface homem-máquina). Na IHM devem constar informações importantes para a produção, como o estado atual da máquina, estado atual de atuadores (como cilindros e motores, assim como sensores, somo sensores fim de curso, sensores de peça, etc.), dados da produção e alarmes. (DA ROCHA, 2018).

No caso de uma subestação, cada alarme ou alteração de estado que é indicada na IHM (Figura 1), o usuário é informado em tempo real por meio de uma lista de alarmes online, contendo data e hora, equipamento e mensagem explicando o problema ocorrido. (Elipse, 2017).

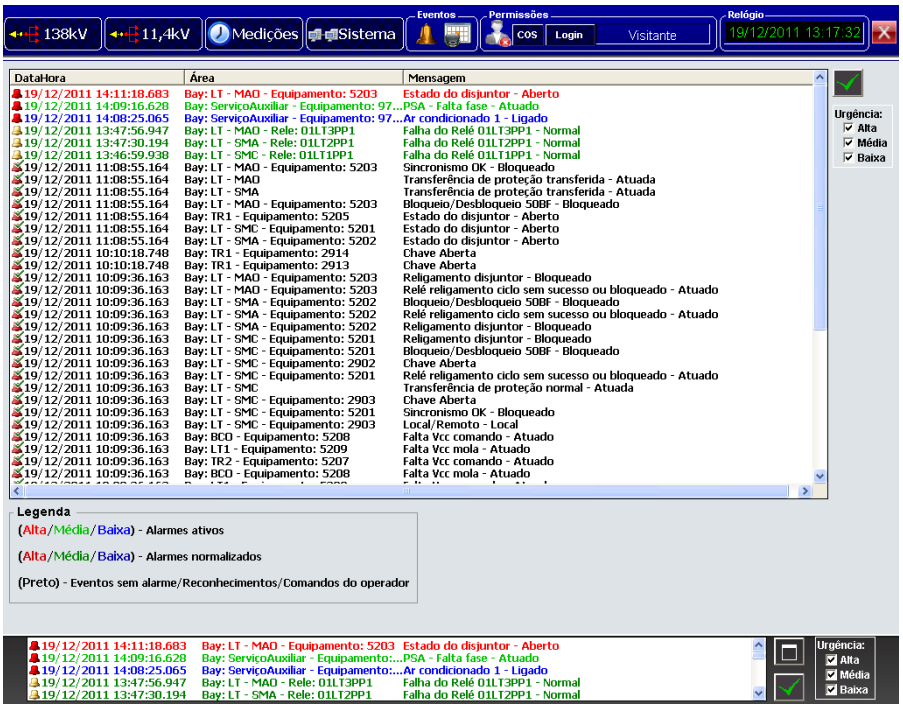

Figura 1 – Tela de alarmes da subestação. (Elipse, 2017).

Geralmente as máquinas não deixam de funcionar de uma hora para outra por causa de danos em todas as partes da máquina, mas sim em poucas partes ou somente uma parte da máquina, que seja vital para seu funcionamento. (Simetriza, 2017).

Com base no alarme que apareceu na IHM, a pessoa responsável pela máquina ou até mesmo quem realizará a manutenção da máquina, seja a troca de alguma peça ou o ajuste do equipamento, já vai saber onde ocorreu o erro.

Informar a solução referente ao erro ocorrido não faz parte do papel da IHM, portanto as pessoas responsáveis por manter a máquina em produção precisam ter o conhecimento sobre a máquina para poder realizar a tarefa de possibilitar a máquina a voltar a produzir caso aconteça algum erro. Existem erros de processo e falhas elétricas ou mecânicas, portanto para cada tipo de falha deve haver uma pessoa certa para resolvê-la. Ou seja, para resolver falhas mecânicas, é indicado um técnico da manutenção mecânica, assim como é indicado um técnico da manutenção elétrica para resolver falhas elétricas. As falhas de processo, quem resolve é quem conhece do processo, que pode ser um especialista ou até mesmo um operador treinado. Os ajustes da máquina realizados pelo operador podem ser auxiliados pelo especialista da máquina.

2.2 COMPONENTES DE INTERAÇÃO COM O USUÁRIO NO APLICATIVO MÓVEL

Os componentes mais importantes usados na aplicação móvel são dois: as ListViews e as Intents. Com estes dois componentes principais foi possível realizar a organização necessária para o funcionamento esperado para o aplicativo, apresentado no Capítulo 3.

2.2.1 ListViews

Como forma de organização no aplicativo para possibilitar o usuário de selecionar facilmente a parte da máquina que deseja, foram utilizadas listas, ou *ListViews*. As listas são utilizadas em muitos aplicativos conhecidos mundialmente, como, por exemplo, no aplicativo da Figura 2, denominado WhatsApp.

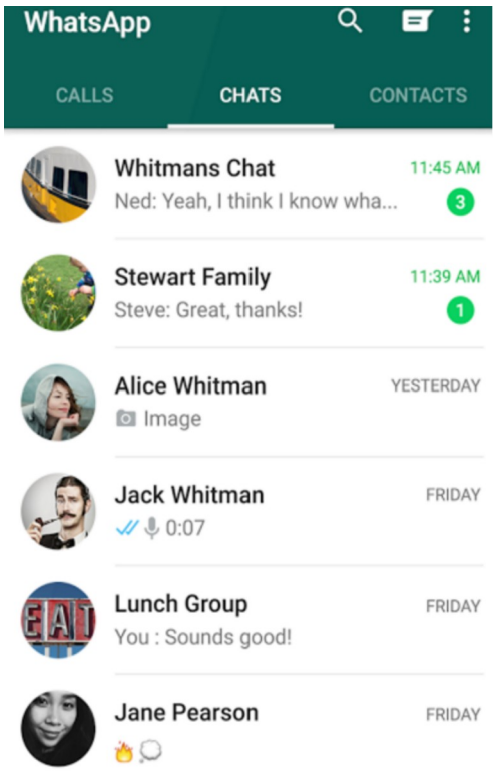

Figura 2 – Exemplo de lista utilizada pelo aplicativo Whatsapp (GooglePlay, 2017).

Assim como o os contatos e as conversas existentes no aplicativo do WhatsApp, o conteúdo do aplicativo a ser desenvolvido também será exibido em listas.

As listas são formas de organizar informações, porém é o programador que decide quais informações ele quer exibir na lista. Conforme o site de desenvolvedores do Android, pode-se exibir uma lista de itens roláveis por meio de uma ListView. Por meio de um *Adapter* são inseridos automaticamente itens da lista. O *Adapter* obtém conteúdo de uma origem como uma matriz ou consulta de banco de dados e converte em uma exibição cada resultado de item. Essa exibição é colocada na lista. (Android Developer, 2017).

2.2.2 Intents

Quando o usuário clicar em algum item da lista, selecionando uma parte da máquina, o aplicativo exibe outra tela com uma lista de alarmes pertencentes àquela parte da máquina. A Intent é responsável por passar informações entre telas do aplicativo, assim como navegar entre uma tela e outra como mostra a Figura 3.

Por meio de uma *Intent* é possível interagir com componentes do mesmo aplicativo ou de outras aplicações, como por exemplo iniciar uma Activity externa para tirar uma foto. (AndroidPro Intents, 2018).

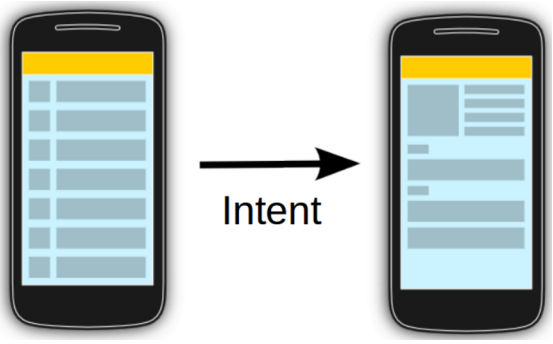

Figura 3 – Função de uma *Intent.* Fonte: Vogella, 2017.

Através de uma *Intent* é possível fornecer dados de uma tela para outra, de forma que a tela que foi carregada saiba o que ela deve exibir. A sequência das telas é organizada na seguinte forma:

- Primeiro o usuário seleciona a parte da máquina em questão, a *Intent* passa a informação da parte selecionada.
- Após a parte da máquina selecionada, o usuário seleciona um alarme ou problema, em relação à parte da máquina selecionada. A *Intent* fornece o problema ou alarme selecionado.
- Após a seleção do alarme ou problema, as possíveis soluções para o problema selecionado serão exibidas, com possibilidade de inserção de informações adicionais e votação para registrar qual das soluções foi responsável por resolver o problema.

2.3 BANCO DE DADOS

O banco de dados a ser utilizado será o SQLite por ser uma ferramenta gratuita e suficiente para a aplicação em questão. Ele se torna suficiente pelos seguintes aspectos:

- É uma ferramenta que possui as características completas dos bancos de dados SQL,
- Funciona como banco de dados em um único arquivo,

- Suporta mais de 100 terabytes de informação,
- Não precisa ser instalado nem configurado.

Para gerenciar os bancos de dados no aplicativo serão usados comandos SQL no próprio aplicativo, como a seguir:

- *CREATE TABLE* – Para criar a tabela,
- *put* – para definir valores de cada campo da tabela,
- *insert* – para inserir uma nova linha na tabela contendo os dados inseridos pelo comando *put.*
- *Query* – para ler informações do banco de dados (SQLite, 2017).

Para isso, no aplicativo do dispositivo *Android*, a classe que vai lidar com o banco de dados herdará a classe *SQLiteOpenHelper,* que é uma classe que gerencia criação de banco de dados para *Android*.

2.4 FERRAMENTAS DE DESENVOLVIMENTO

As ferramentas de desenvolvimento utilizadas envolvem o Android Studio (para desenvolvimento do aplicativo do dispositivo móvel), o Eclipse (para desenvolvimento do programa do servidor), a UML (para modelagem do aplicativo) e as ferramentas de comunicação, que envolvem webservices, Apache TomCat e RESTFul.

2.4.1 *Android Studio*

O programa que oferece as ferramentas mais rápidas para criação de aplicativos em plataforma Android é o Android Studio. Nele, existem recursos que permitem que o programador crie aplicativos de alta qualidade. Alguns dos recursos são citados abaixo:

- Edição de código de nível global
- Depuração
- Ferramentas de desempenho
- Sistema flexível de compilação

- Criação/implantação instantâneas (Android Developer, 2017).

Para desenvolver o aplicativo para o dispositivo móvel foi utilizada a versão 2.3.3 do *Android Studio.*

2.4.2 *Eclipse*

O desenvolvimento do programa para computador será realizado com a ferramenta Eclipse. A linguagem utilizada será Java e para desenvolvimento da interface gráfica do utilizador serão utilizados os recursos do *Java Swing*.

Os componentes do *Java Swing* são suficientes para a criação da interface gráfica necessária para o projeto, por conter botões, listas, menus, escolhedor de arquivos, tabelas, e outros.

2.4.3 UML

Conforme o site Lucidchart (2017), a UML, que significa Linguagem de Modelagem Unificada, tem aplicações tanto em fluxos do processo quanto em desenvolvimento de programas. A UML é uma ferramenta de modelagem visual rica tanto de forma semântica quanto sintática, e pode ser aplicada para criação de programas complexos, seja estruturalmente e/ou para comportamentos.

Para a modelagem do aplicativo serão utilizados:

- Diagrama de caso de uso, com o objetivo de auxiliar a comunicação entre os analistas e o cliente, descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário (SAMPAIO, Diagramas de casos de uso, 2017).
- Diagrama de classes, com o objetivo de descrever os vários tipos de objetos no sistema e o relacionamento entre eles (SAMPAIO, Diagramas de classes, 2017).
- Diagrama de comunicação, com o objetivo de mostrar as interações entre os objetos ou funções associados com linhas de vida e mensagens transmitidas entre linhas de vida. São um tipo de diagrama de interação que pode-se usar para explorar o comportamento dinâmico de um sistema ou aplicativo (IBM, 2017).

- Diagrama de atividades, para representar tanto a estrutura quanto os comportamentos mais importantes do sistema e com base neles construir e programar o modelo executável, que é o sistema materializado. Ilustra graficamente como será o funcionamento do software, como será a execução de alguma de suas partes, como será a atuação do sistema na realidade de negócio na qual está inserido (Ateomomento, 2017).

2.4.4 WebServices

Para realizar comunicação entre aplicações diferentes e para integrar sistemas, é utilizada uma solução chamada de WebService. O formato universal para dados estruturados na Web, o formato XML, é usado para realizar a comunicação entre sistemas novos e sistemas que já existem em plataformas diferentes, por meio da internet. O uso do WebService tem como vantagens:

- Agilidade e eficiência na comunicação
- Segurança (não necessita intervenção humana)
- Disponibilidade de recursos para qualquer aplicação cliente (Soawebservices, 2017)

Para o aplicativo funcionar corretamente, deve existir comunicação entre o servidor e os dispositivos móveis. Para o aplicativo desenvolvido foi usado Webservice para atualizar o banco de dados. Como existem duas vias de comunicação, ou seja, do dispositivo mobile para o servidor, e do servidor para o dispositivo mobile, serão usados dois programas diferentes. No servidor, será usado um servidor de Webservice, o Apache Tomcat. Já no dispositivo mobile será usado um consumidor de Webservice, o RESTful.

2.4.5 *Apache Tomcat*

Apache Tomcat é um programa usado por muitas organizações e indústrias (Tomcat, 2017). Ele executa, como um servidor de aplicações, pequenas aplicações que rodam em um servidor *Java, ou Java Servlets (Theserverside, 2017)*. O que torna o uso do Tomcat a melhor opção é que ele usa menos recursos e

significantemente menos complexidade. Existem vários subsistemas que trabalham com Tomcat (Davis, 2017).

No sistema, o Tomcat será usado no servidor como um servidor de Webservice, aguardando pedido de algum dispositivo móvel para atualização de banco de dados.

2.4.6 *RESTful*

REST se tornou uma das mais importantes tecnologias para aplicações web, mais de uma década após sua introdução. As principais linguagens de desenvolvimento agora incluem *frameworks* para criar *webservices RESTful*. Um serviço baseado no REST é chamado de serviço RESTful. Enquanto REST não depende de nenhum protocolo, quase todos os serviços RESTful usam HTTP como seu protocolo subjacente (Drdobbs, 2017).

Utilizar REST no android se torna mais fácil que outras tecnologias por ela ser simplesmente um serviço disponibilizado na internet por meio do protocolo http. Com os parâmetros enviados pelo aplicativo, são recebidas requisições que devolvem dados formatados, por meio de um serviço contido no aplicativo (Milfont, 2017).

No dispositivo móvel, o RESTful será usado como consumidor de Webservice, enviando pedidos ao servidor para atualizar o banco de dados. Os pedidos consistem em receber o banco de dados atualizado e enviar textos complementares ao banco de dados.

3 APLICATIVO PARA RESOLUÇÃO DE PROBLEMAS DE MÁQUINAS INDUSTRIAIS

O sistema desenvolvido consiste em duas partes: o programa no servidor e o aplicativo para dispositivo móvel (Figura 4). Para atender a todas as especificações do cliente, foram levantados requisitos tanto do programa que vai ser executado em um computador no qual um gerente de manutenção junto a um líder de produção poderá gerenciar o banco de dados alimentando-o com informações úteis para o operador da máquina a respeito de manutenções e informações de *setup*.

O sistema é composto por 5 níveis. O nível 1 contém o nome das máquinas, o nível 2 contém o nome das partes de cada máquinas, o nível 3 é composto pela descrição dos problemas de cada parte de máquina, o nível 4 contém a descrição das soluções de cada problema e o último nível, o nível 5, é composto pelos comentários de cada solução.

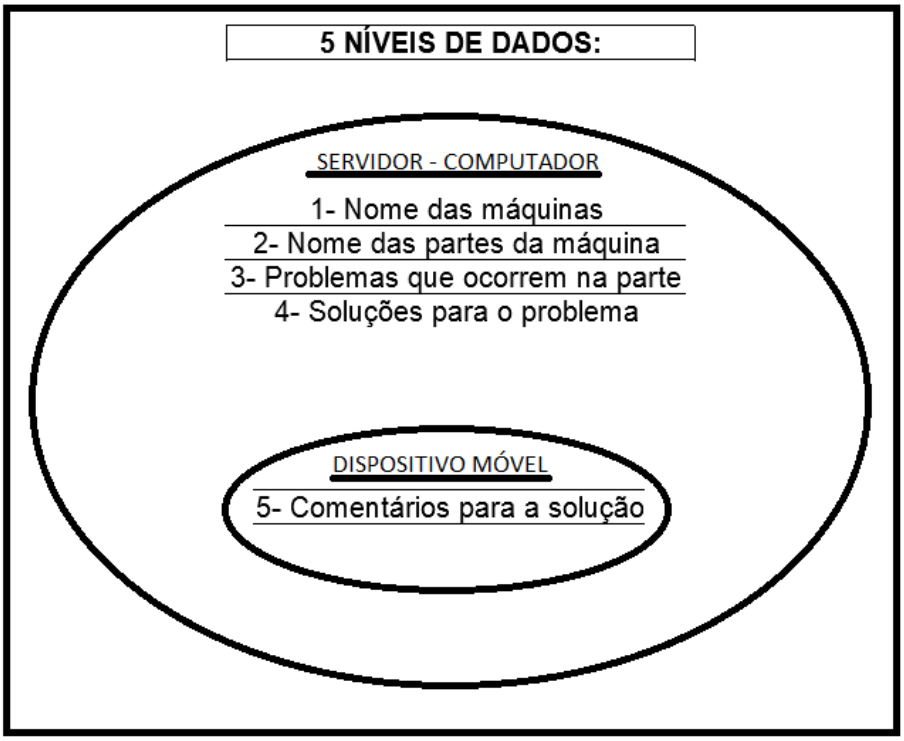

Figura 4 – Apresentação do sistema – Alimentação por meio dos níveis de dados.

A Figura 5 explica a arquitetura física, de forma que ambos dispositivos se comunicam por meio de nuvem (Webservice) e mantém banco de dados no dispositivo para não ser obrigatória a conexão com a internet para uma consulta ao banco de dados.

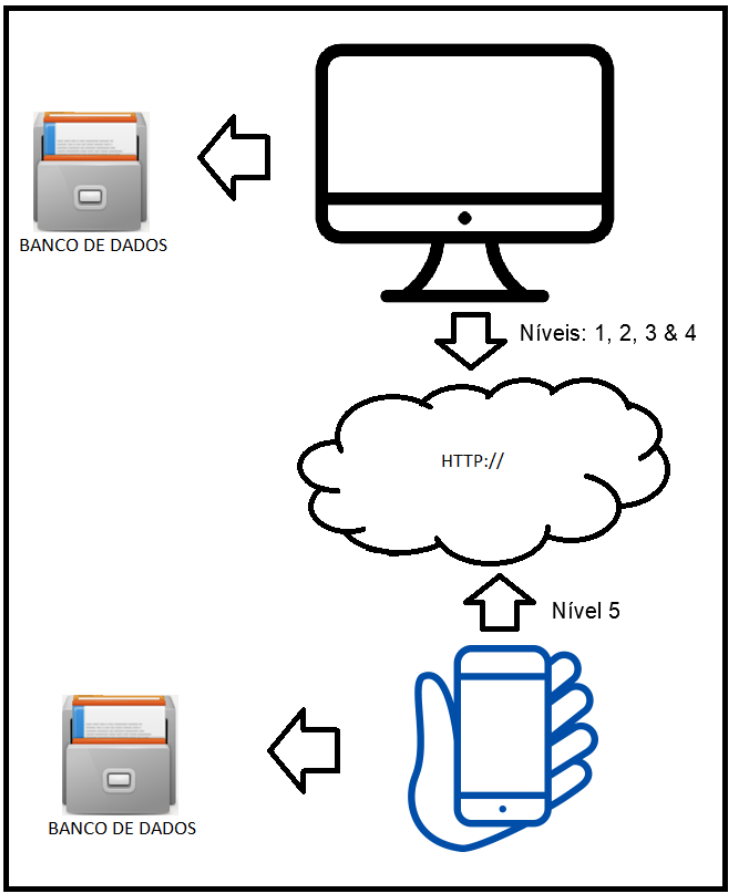

Figura 5 – Arquitetura física do sistema.

Como o sistema é composto tanto da parte do dispositivo móvel quanto da parte do servidor, a Figura 5 ilustra em quais níveis cada parte vai poder alimentar o banco de dados e a nuvem. Através do servidor, será possível alimentar o banco de dados com informações do nível 1 ao 4, restando somente ao aplicativo do dispositivo móvel alimentar o banco de dados com informações do nível 5, ou seja, os comentários das soluções.

3.1 REQUISITOS DO SISTEMA

Os requisitos do sistema foram divididos em requisitos funcionais e requisitos não funcionais.

3.1.1 Requisitos funcionais

Os requisitos funcionais descrevem as funcionalidades do sistema e foram divididos em requisitos funcionais do dispositivo móvel e da interface do computador.

3.1.1.1 Da interface do computador

- [RF001] O sistema deve possibilitar o cadastro e gerenciamento (adição, edição e exclusão) de máquinas, partes de máquinas, problemas ou procedimentos e soluções dos problemas ou passos dos procedimentos;
- [RF002] O sistema deve permitir o gerenciamento (edição e exclusão) dos comentários enviados pelos usuários do aplicativo nos dispositivos móveis;
- [RF003] O sistema deve possibilitar a procura de texto existente nos cadastros;

3.1.1.2 Do dispositivo móvel

- [RF004] O sistema deve permitir o cadastro de comentários
- [RF005] O sistema deve possibilitar a procura de texto existente nos cadastros;
- [RF006] Os itens de escolha devem ser exibidos em listas
- [RF007] O sistema deve informar ao usuário qual item foi escolhido;
- [RF008] O aplicativo, ao ser aberto, não deverá exibir introdução.

3.1.2 Requisitos não-funcionais

Assim como os requisitos funcionais, os não-funcionais foram divididos em requisitos do aplicativo do dispositivo móvel e requisitos da interface do computador.

3.1.2.1 Da interface do computador

- [RNF001] O sistema deve ser implementado em Java

- [RNF002] O sistema deve ter resposta ao clique do usuário de no máximo 3 segundos
- [RNF003] O sistema deve ser executado somente em sistema operacional Windows versão 7 ou superior;

3.1.2.2 Do dispositivo móvel

- [RNF004] O sistema deve ser compatível somente com Android 7.1 ou superior;
- [RNF005] O sistema deve ter resposta ao clique do usuário de no máximo 3 segundos;
- [RNF006] O sistema deve ser atualizado com os dados do servidor sempre que o aplicativo for aberto ou quando o usuário inserir algum comentário, se existir conexão com o servidor.

3.2 MODELAGEM DO SISTEMA

A modelagem do sistema é composto por diagramas de casos de uso, diagramas de classes, diagramas de comunicação e diagramas de atividades.

3.2.1 Diagramas de casos de uso

Os casos de uso foram separados em casos de uso do aplicativo do dispositivo móvel e casos de uso do programa no servidor. A descrição, os eventos, os atores, as pré-condições, as pós-condições, assim como os fluxos de cada caso de uso do aplicativo do dispositivo móvel (Figura 6) foram definidas da seguinte forma:

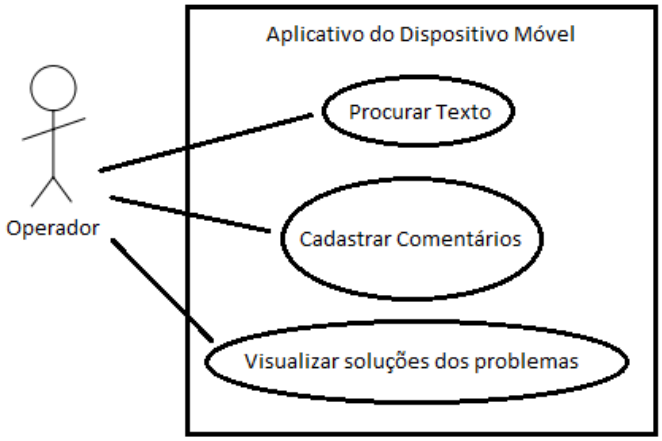

Figura 6 – Diagrama de casos de uso do aplicativo do dispositivo móvel.

UC001 – Procurar texto

Descrição: permite que o usuário procure um texto no banco de dados.

Eventos:

– Sistema exibe campo de pesquisa de texto.

– Usuário informa texto a ser procurado.

– Sistema exibe lista de opções com textos que contém o texto digitado pelo usuário.

– Usuário seleciona opção na lista.

– Sistema exibe a lista que contém a opção selecionada pelo usuário.

Atores:

- Usuário (operador da máquina)

Pré-condições:

- Aplicativo deve exibir o campo de pesquisa de texto.

Pós-condições:

1. Conclusões com sucesso:

- Sistema exibe lista selecionada pelo usuário.

2. Conclusões sem sucesso:

- Usuário não encontra o texto procurado e abandona a busca.

Fluxo básico:

1. Usuário fornece o texto a ser pesquisado

2. Sistema exibe lista contendo o resultado da pesquisa de textos que contém o texto informado pelo usuário

3. Usuário seleciona opção de texto desejada (A1)

4. Sistema exibe lista de origem do texto da opção selecionada.

5. Fim do caso de uso

Fluxos alternativos:

A1: em “Usuário seleciona opção de texto desejada”, caso o usuário não deseja selecionar nenhuma das opções de texto exibidas pelo resultado de busca do sistema

A1.1. Retornar o fluxo básico para o item “Usuário fornece o texto a ser pesquisado”

UC002 – Cadastrar comentários

Descrição: permite que o usuário cadastre um comentário como informação adicional a uma solução.

Eventos:

– Usuário seleciona solução a ser comentada.

– Sistema exibe campo de inserção de comentário.

– Usuário insere texto a ser adicionado como comentário.

– Sistema cadastra o comentário na lista.

Atores:

- Usuário (operador da máquina)

Pré-condições:

- Aplicativo aberto na lista de soluções.

Pós-condições:

1. Conclusões com sucesso:

- Sistema cadastra o comentário na lista.

Fluxo básico:

1. Usuário seleciona a solução a ser comentada

2. Sistema exibe campo de inserção do texto do comentário

3. Usuário insere o comentário

4. Sistema cadastra o comentário

5. Fim do caso de uso

UC003 – Visualizar soluções dos problemas

Descrição: permite que o usuário visualize soluções para um problema cadastrado no sistema.

Eventos:

– Sistema exibe lista de máquinas.

– Usuário seleciona máquina.

– Sistema exibe lista de partes da máquina selecionada.

– Usuário seleciona parte da máquina.

– Sistema exibe lista de problemas da parte selecionada.

– Usuário seleciona problema da parte da máquina.

– Sistema exibe lista de soluções do problema selecionado.

Atores:

– Usuário (operador da máquina)

Pré-condições:

– Aplicativo deve estar aberto na tela inicial.

Pós-condições:

1. Conclusões com sucesso:

– Sistema exibe lista de soluções selecionada pelo usuário.

2. Conclusões sem sucesso:

– Usuário não encontra máquina cadastrada no sistema.

– Usuário não encontra parte da máquina cadastrada no sistema.

– Usuário não encontra problema cadastrado no sistema.

Fluxo básico:

1. Sistema exibe lista de máquinas
2. Usuário seleciona máquina (A1)
3. Sistema exibe partes cadastradas da máquina selecionada
4. Usuário seleciona parte (A2)
5. Sistema exibe problemas cadastrados da parte selecionada
6. Usuário seleciona problema (A3)
7. Sistema exibe soluções cadastradas do problema selecionado
8. Fim do caso de uso

Fluxos alternativos:

A1: em “Usuário seleciona máquina”, caso a máquina desejada pelo usuário não esteja cadastrada

A1.1. Avançar o fluxo básico para “Fim do caso de uso”

A2: em “Usuário seleciona parte”, caso a parte da máquina desejada pelo usuário não esteja cadastrada

A2.1. Avançar o fluxo básico para “Fim do caso de uso”

A3: em “Usuário seleciona problema”, caso o problema desejado pelo usuário não esteja cadastrada

A3.1. Avançar o fluxo básico para “Fim do caso de uso”

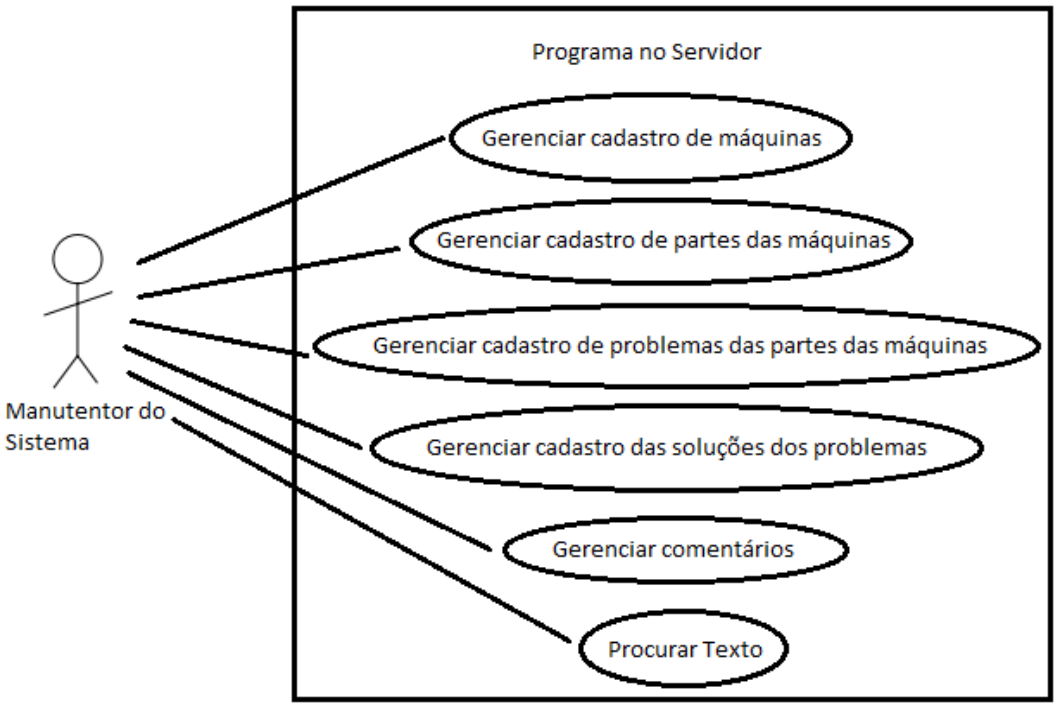

Figura 7 – Diagrama de casos de uso do programa no Servidor.

A descrição, os eventos, os atores, as pré-condições, as pós-condições, assim como os fluxos de cada caso de uso do programa utilizado no servidor (Figura 7) foram definidas como a seguir:

UC004 – Gerenciar cadastro de máquinas

Descrição: permite que o manutentor do sistema adicione, edite ou exclua máquinas no banco de dados do sistema.

Eventos:

– Sistema exibe lista de máquinas.

– Manutentor gerencia lista de máquinas

– Sistema exclui, edita ou adiciona máquina na lista.

Atores:

– Manutentor do sistema

Pré-condições:

– Aplicativo deve estar aberto na tela inicial.

Pós-condições:

Conclusões com sucesso:

– Sistema adiciona máquina na lista.

– Sistema edita máquina na lista.

– Sistema exclui máquina da lista.

Fluxo básico:

1. Sistema exibe lista de máquinas
2. Manutentor seleciona opção para adicionar uma máquina (A1)(A2)
3. Sistema exibe campo para inserção do nome da máquina
4. Manutentor insere nome da máquina
5. Sistema adiciona máquina no banco de dados
6. Fim do caso de uso

Fluxos alternativos:

A1: em “Manutentor seleciona opção para adicionar uma máquina”, caso o manutentor opte por editar o nome de uma máquina

A1.1. Sistema exibe campo para edição do nome da máquina

A1.2. Manutentor insere o novo nome da máquina

A1.3. Sistema edita o nome da máquina selecionada

A1.4. Avançar o fluxo básico para “Fim do caso de uso”

A2: em “Manutentor seleciona opção para adicionar uma máquina”, caso o manutentor opte por remover uma máquina

A2.1. Sistema exclui a máquina selecionada

A2.2. Avançar o fluxo básico para “Fim do caso de uso”

UC005 – Gerenciar cadastro de partes das máquinas

Descrição: permite que o manutentor do sistema adicione, edite ou exclua partes das máquinas no banco de dados do sistema.

Eventos:

– Sistema exibe lista de partes das máquinas.

– Manutentor gerencia lista de partes das máquinas

– Sistema exclui, edita ou adiciona partes da máquina na lista.

Atores:

– Manutentor do sistema

Pré-condições:

– Aplicativo deve estar aberto na tela inicial.

Pós-condições:

Conclusões com sucesso:

– Sistema adiciona parte da máquina na lista.

– Sistema edita parte da máquina na lista.

– Sistema exclui parte da máquina da lista.

Fluxo básico:

1. Sistema exibe lista de partes das máquinas
2. Manutentor seleciona opção para adicionar uma parte de máquina (A1)(A2)
3. Sistema exibe campo para inserção do nome da parte da máquina
4. Manutentor insere nome da parte da máquina
5. Sistema adiciona parte da máquina no banco de dados
6. Fim do caso de uso

Fluxos alternativos:

A1: em “Manutentor seleciona opção para adicionar uma parte de máquina”, caso o manutentor opte por editar o nome de uma parte da máquina

A1.1. Sistema exibe campo para edição do nome da parte da máquina

A1.2. Manutentor insere o novo nome da parte da máquina

A1.3. Sistema edita o nome da parte da máquina selecionada

A1.4. Avançar o fluxo básico para “Fim do caso de uso”

A2: em “Manutentor seleciona opção para adicionar uma parte de máquina”, caso o manutentor opte por remover uma parte de máquina

A2.1. Sistema exclui a parte da máquina selecionada

A2.2. Avançar o fluxo básico para “Fim do caso de uso”

UC006 – Gerenciar cadastro de problemas das partes das máquinas

Descrição: permite que o manutentor do sistema adicione, edite ou exclua problemas das partes das máquinas no banco de dados do sistema.

Eventos:

– Sistema exibe lista de problemas das partes das máquinas.

– Manutentor gerencia lista de problemas das partes das máquinas

– Sistema exclui, edita ou adiciona problema da parte da máquina na lista.

Atores:

– Manutentor do sistema

Pré-condições:

– Aplicativo deve estar aberto na tela inicial.

Pós-condições:

Conclusões com sucesso:

– Sistema adiciona problema da parte da máquina na lista.

– Sistema edita problema da parte da máquina na lista.

– Sistema exclui problema da parte da máquina da lista.

Fluxo básico:

1. Sistema exibe lista de problemas das partes das máquinas
2. Manutentor seleciona opção para adicionar um problema da parte da máquina (A1)(A2)
3. Sistema exibe campo para inserção do problema da parte da máquina
4. Manutentor insere nome do problema da parte da máquina
5. Sistema adiciona problema da parte da máquina no banco de dados
6. Fim do caso de uso

Fluxos alternativos:

A1: em “Manutentor seleciona opção para adicionar um problema da parte da máquina”, caso o manutentor opte por editar o nome de um problema da parte da máquina

A1.1. Sistema exibe campo para edição do problema da parte da máquina

A1.2. Manutentor insere o novo problema da parte da máquina

A1.3. Sistema edita o problema da parte da máquina selecionada

A1.4. Avançar o fluxo básico para “Fim do caso de uso”

A2: em “Manutentor seleciona opção para adicionar um problema da parte da máquina”, caso o manutentor opte por remover um problema da parte da máquina

A2.1. Sistema exclui o problema da parte da máquina selecionada

A2.2. Avançar o fluxo básico para “Fim do caso de uso”

UC007 – Gerenciar cadastro das soluções dos problemas

Descrição: permite que o manutentor do sistema adicione, edite ou exclua soluções dos problemas no banco de dados do sistema.

Eventos:

– Sistema exibe lista de soluções dos problemas.

– Manutentor gerencia lista de soluções dos problemas

– Sistema exclui, edita ou adiciona soluções dos problemas na lista.

Atores:

– Manutentor do sistema

Pré-condições:

– Aplicativo deve estar aberto na tela inicial.

Pós-condições:

Conclusões com sucesso:

– Sistema adiciona solução do problema na lista.

– Sistema edita solução do problema na lista.

– Sistema exclui solução do problema da lista.

Fluxo básico:

1. Sistema exibe lista de soluções dos problemas

2. Manutentor seleciona opção para adicionar uma solução do problema (A1)(A2)
3. Sistema exibe campo para inserção da solução do problema
4. Manutentor insere solução do problema
5. Sistema adiciona solução do problema no banco de dados
6. Fim do caso de uso

Fluxos alternativos:

A1: em “Manutentor seleciona opção para adicionar uma solução do problema”, caso o manutentor opte por editar uma solução do problema
A1.1. Sistema exibe campo para edição da solução do problema
A1.2. Manutentor insere a nova solução do problema
A1.3. Sistema edita a solução do problema
A1.4. Avançar o fluxo básico para “Fim do caso de uso”

A2: em “Manutentor seleciona opção para adicionar uma solução do problema”, caso o manutentor opte por remover uma solução do problema
A2.1. Sistema exclui a solução do problema selecionada
A2.2. Avançar o fluxo básico para “Fim do caso de uso”

UC008 – Gerenciar comentários

Descrição: permite que o manutentor do sistema edite ou exclua comentários no banco de dados do sistema.

Eventos:

– Sistema exibe lista de comentários.
– Manutentor gerencia lista de comentários
– Sistema exclui ou edita comentário na lista.

Atores:

– Manutentor do sistema

Pré-condições:

– Aplicativo deve estar aberto na tela inicial.

Pós-condições:

Conclusões com sucesso:

– Sistema edita comentário na lista.

– Sistema exclui comentário da lista.

Fluxo básico:

1. Sistema exibe lista de comentários
2. Manutentor seleciona opção para editar um comentário (A1)
3. Sistema exibe campo para edição do comentário
4. Manutentor insere o comentário editado
5. Sistema edita o comentário no banco de dados
6. Fim do caso de uso

Fluxos alternativos:

A1: em “Manutentor seleciona opção para editar um comentário”, caso o manutentor opte por excluir um comentário

A1.1. Sistema exclui o comentário selecionado

A1.2. Avançar o fluxo básico para “Fim do caso de uso”

UC009 – Procurar texto

Descrição: permite que o manutentor procure um texto no banco de dados.

Eventos:

– Sistema exibe campo de pesquisa de texto.

– Manutentor informa texto a ser procurado.

– Sistema exibe lista de opções com textos que contém o texto digitado pelo usuário.

– Manutentor seleciona opção na lista.

– Sistema exibe a lista que contém a opção selecionada pelo usuário.

Atores:

– Manutentor do sistema

Pré-condições:

- Aplicativo deve exibir o campo de pesquisa de texto.

Pós-condições:

1. Conclusões com sucesso:

- Sistema exibe lista selecionada pelo manutentor.

2. Conclusões sem sucesso:

- Manutentor não encontra o texto procurado e abandona a busca.

Fluxo básico:

1. Manutentor fornece o texto a ser pesquisado
2. Sistema exibe lista contendo o resultado da pesquisa de textos que contém o texto informado pelo manutentor
3. Manutentor seleciona opção de texto desejada (A1)
4. Sistema exibe lista de origem do texto da opção selecionada.
5. Fim do caso de uso

Fluxos alternativos:

A1: em “Manutentor seleciona opção de texto desejada”, caso o manutentor não deseja selecionar nenhuma das opções de texto exibidas pelo resultado de busca do sistema

A1.1. Retornar o fluxo básico para o item “Usuário fornece o texto a ser pesquisado”

3.2.2 Diagramas de Classes

Para a modelagem do sistema usando diagrama de classes foi usado padrão MVC (*Model-View-Controller*).

Como a visualização inicial do programa, a classe “*ListarMaquinasView*”(Figura 8) além de se comunicar com a classe que a controla, se comunica também com a *view* que exibe as partes da máquina selecionada, chamada “*ListarPartesMaquinaView*”(Figura 9).

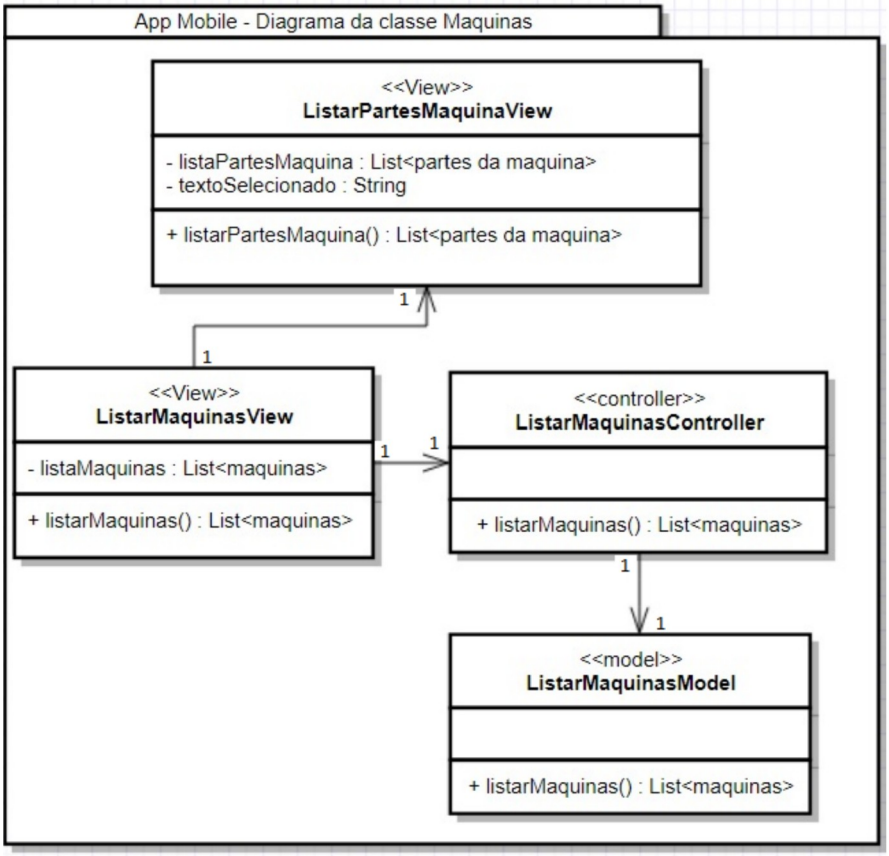

Figura 8 – Diagrama da classe *ListarMaquinasView*.

A classe *ListarMaquinasView* é responsável por apresentar ao usuário a lista de máquinas cadastradas no banco de dados. Assim que o usuário seleciona a máquina, a classe *ListarPartesMaquinaView* realiza um papel semelhante à classe *ListarMaquinasView*, porém ao invés de apresentar as máquinas, ela é responsável por apresentar ao usuário a lista de partes da máquina selecionada.

A classe controladora, a *ListarMaquinasController* é responsável por enviar a ação do usuário à classe *ListarMaquinasModel*. Esta, por sua vez, se encarrega de armazenar a escolha do usuário e executar a *intent* que faz o aplicativo deixar de exibir as máquinas e passar a exibir as partes da máquina selecionada.

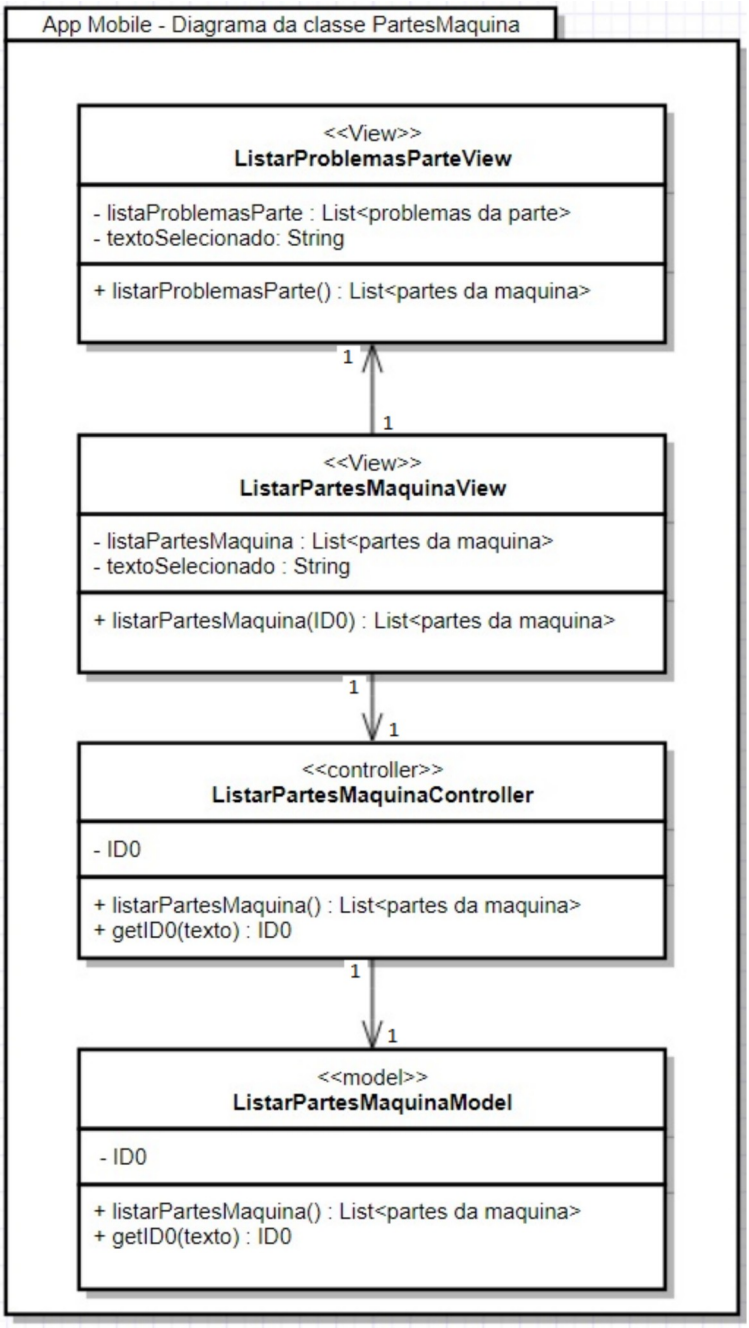

Figura 9 – Diagrama da classe *ListarPartesMaquinaView.*

A classe *ListarPartesMaquinaView*(Figura 9) exibe o nome da máquina selecionada e a lista de partes da máquina, comunica-se com a classe *ListarProblemasParteView* e com sua classe controladora, a *ListarPartesMaquinaController*. Esta classe controladora carrega o texto da máquina selecionada da classe *ListarMaquinasView* para sua classe modelo, a *ListarPartesMaquinaModel*, para esta recuperar a *ID* da máquina selecionada. Cada

vez que o usuário seleciona uma opção, as classes modelos são responsáveis por passar adiante a informação do texto e da *ID* da opção selecionada.

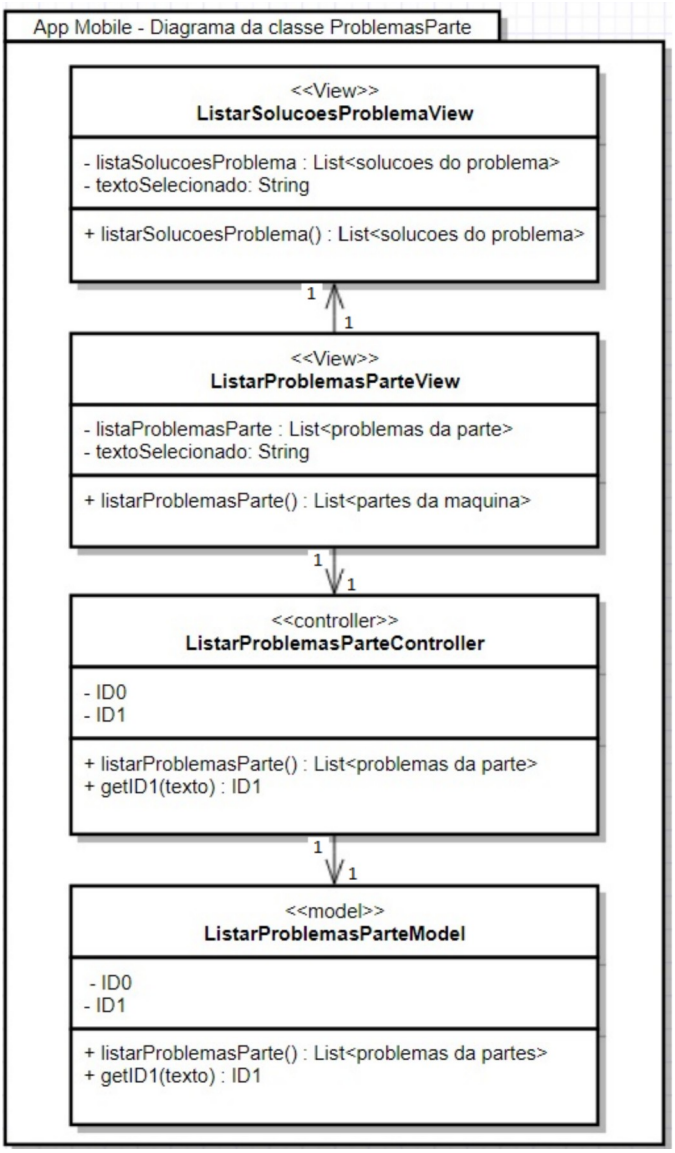

Figura 10 – Diagrama da classe *ListarProblemasParteView*.

A classe *ListarProblemasParteView* (Figura 10) é responsável por exibir a lista dos problemas cadastrados da parte da máquina selecionada, além de exibir quais foram a máquina e a parte da máquina selecionada anteriormente. A classe controladora, *ListarProblemasParteController*, informa à classe modelo a opção selecionada. A classe *ListarProblemasParteModel* tem a função de resgatar o texto da *ID* da parte da máquina selecionada para alimentar a classe *ListarProblemasParteView* com os itens que esta deverá exibir para o usuário.

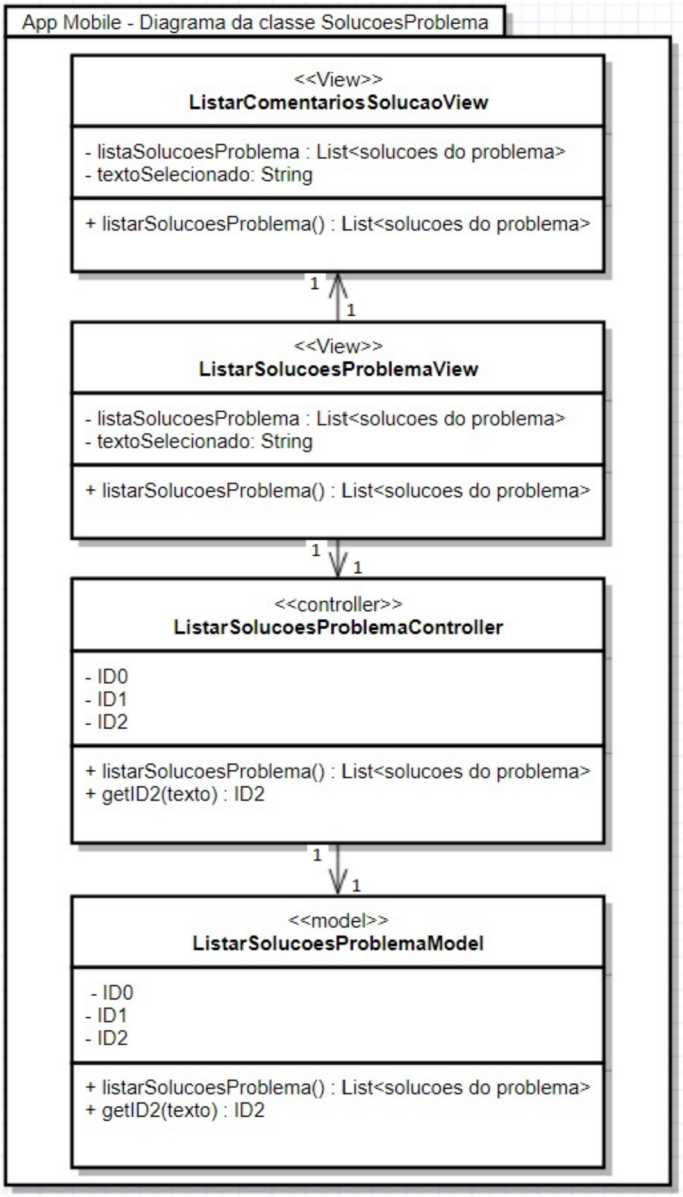

Figura 11 – Diagrama da classe *ListarSolucoesProblemaView*.

Na Figura 11, a classe *ListarSolucoesProblemaModel* resgata o texto da opção selecionada na lista dos problemas da parte da máquina, além de carregar a *ID* de cada seleção anterior e fornecer à classe *ListarSolucoesProblemaView* a lista das soluções cadastradas para serem exibidas para o usuário. A classe controladora *ListarSolucoesProblemaController* passa a informação do item selecionado da classe *ListarSolucoesProblemaView* para a classe *ListarSolucoesProblemaModel*.

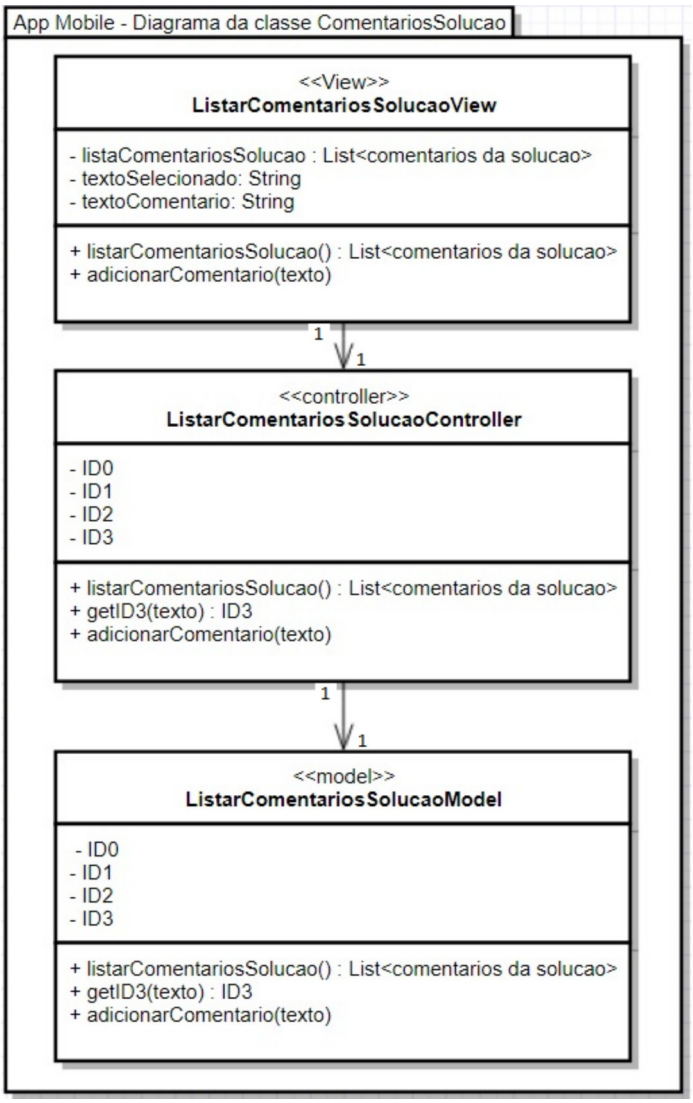

Figura 12 – Diagrama da classe *ListarComentariosSolucaoView*.

Na figura 12, como não será carregada outra *view* ao ser selecionado um item, a classe *ListarComentariosSolucaoView* exibe ao usuário a lista de comentários cadastrados para a solução selecionada na *view* anterior e um espaço para o usuário inserir texto para ser cadastrado como novo comentário. A classe controladora *ListarComentariosSolucaoController* é responsável por informar para a classe modelo o texto inserido. A classe modelo, *ListarComentariosSolucaoModel*, além de usar todas as *IDs* das escolhas anteriores para resgatar somente os comentários cadastrados para serem exibidos na classe *view,* também resgata o texto da *ID* da solução selecionada na *view* anterior.

A identificação de cada escolha feita pelo usuário é passada por meio das classes para que cada controlador possa selecionar o conteúdo a ser mostrado, assim, para acessar a *view* usada para comentar alguma solução, as 4 identificações tem que ser passadas, desde a *ID* da máquina até a *ID* da solução.

3.2.3 Diagramas de Comunicação

Como complemento dos diagramas de classe, os diagramas de comunicação mostram mais claramente como funciona a interação entre as classes a respeito de cada ação do usuário. Para cadastrar um comentário, como mostra a Figura 13, somente é preciso o usuário, que é o operador da máquina, inserir o texto do comentário.

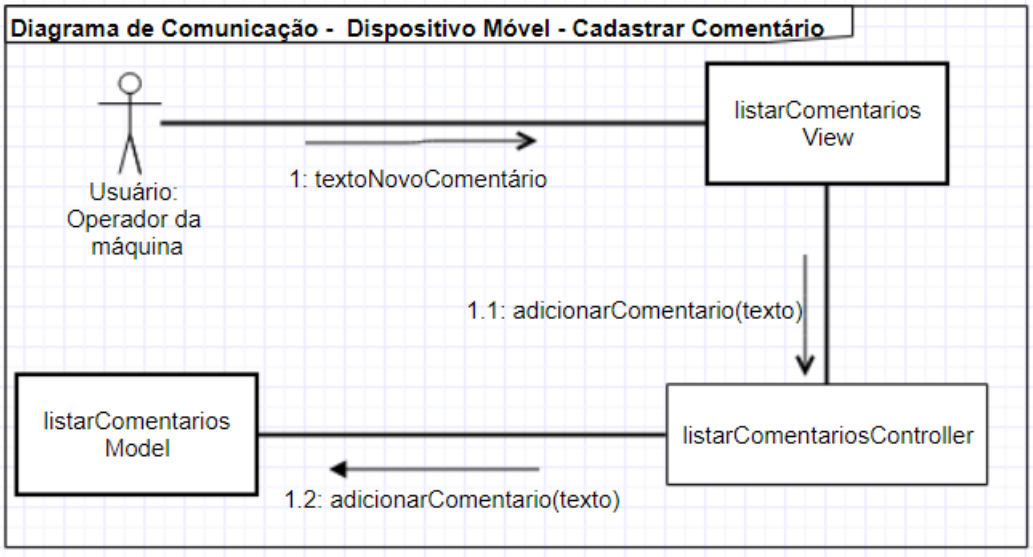

Figura 13 – Diagrama de comunicação – Cadastrar comentário.

Já no servidor, como o ator é o manutentor do sistema, ele pode adicionar, editar ou excluir nomes de máquinas, partes das máquinas, problemas das partes e soluções dos problemas. Já na parte de comentários, o manutentor do sistema somente poderá excluir.

A Figura 14 mostra como deve funcionar o gerenciamento do nome das máquinas. Como gerenciamento, se encontram as funções de adicionar, excluir ou editar algum nome.

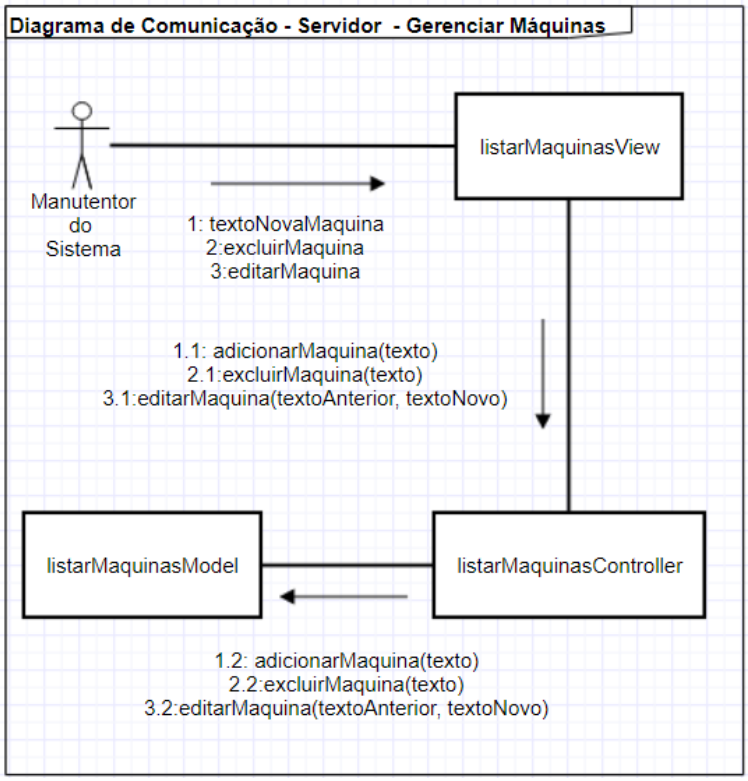

Figura 14 – Diagrama de comunicação – Gerenciar máquinas.

A Figura 15 tem o comportamento semelhante ao diagrama da Figura 14, porém ele atua no nível das partes das máquinas ao invés do nível do nome das máquinas.

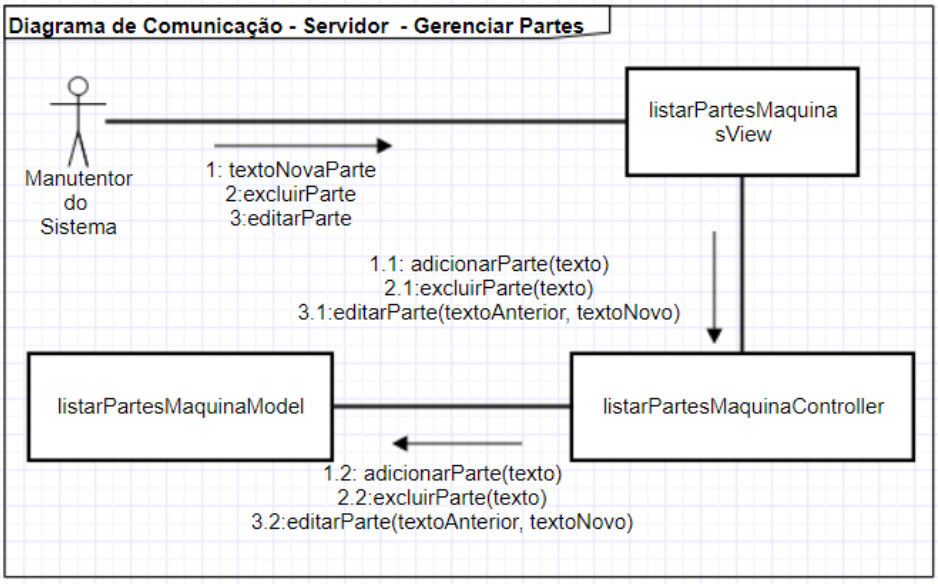

Figura 15 – Diagrama de comunicação – Gerenciar partes das máquinas.

Na Figura 16 é ilustrado o cadastro de um problema de uma respectiva parte de uma máquina. Enquanto no aplicativo do dispositivo móvel, o problema somente pode ser selecionado, no programa do servidor o problema pode ser inserido como novo, pode ser apagado ou editado.

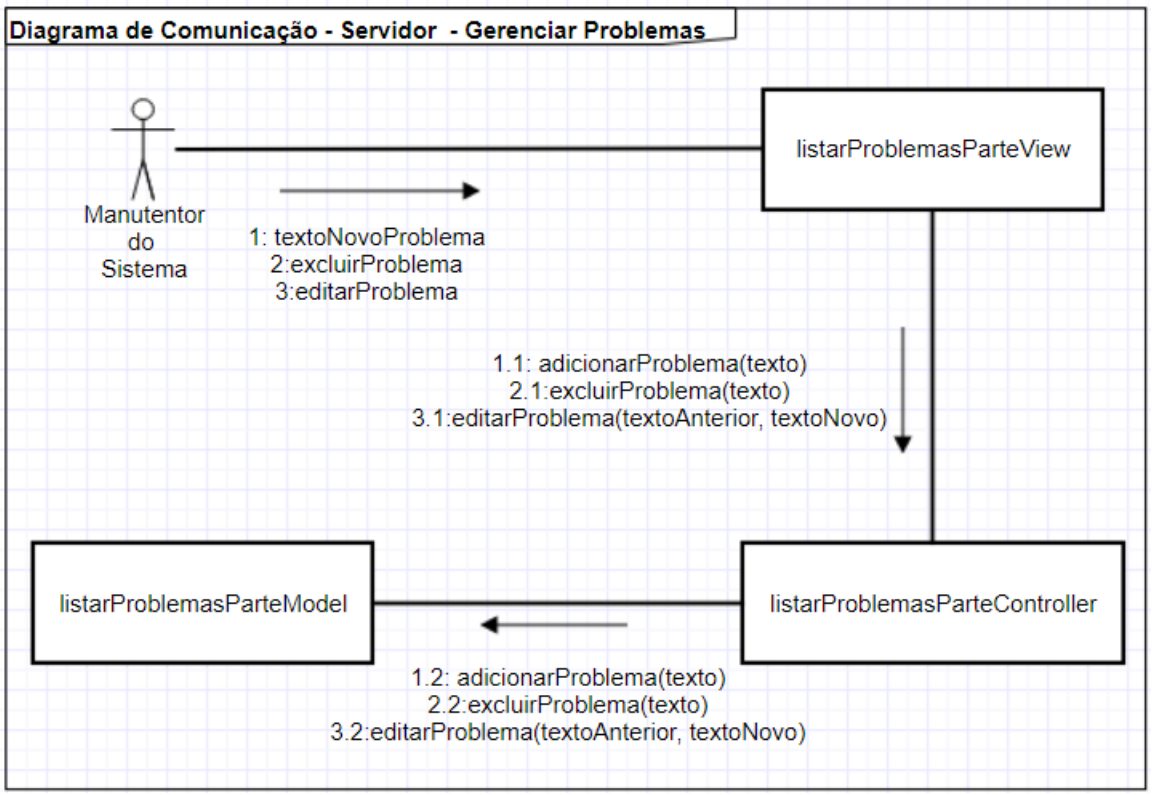

Figura 16 – Diagrama de comunicação – Gerenciar problemas.

A Figura 17, semelhante à Figura 16, também modela a adição, edição ou exclusão de informações, entretanto se trata do nível posterior ao da Figura 16, ou seja, no aplicativo do dispositivo móvel, assim que o usuário selecionar algum problema, as soluções para esse problema serão exibidas.

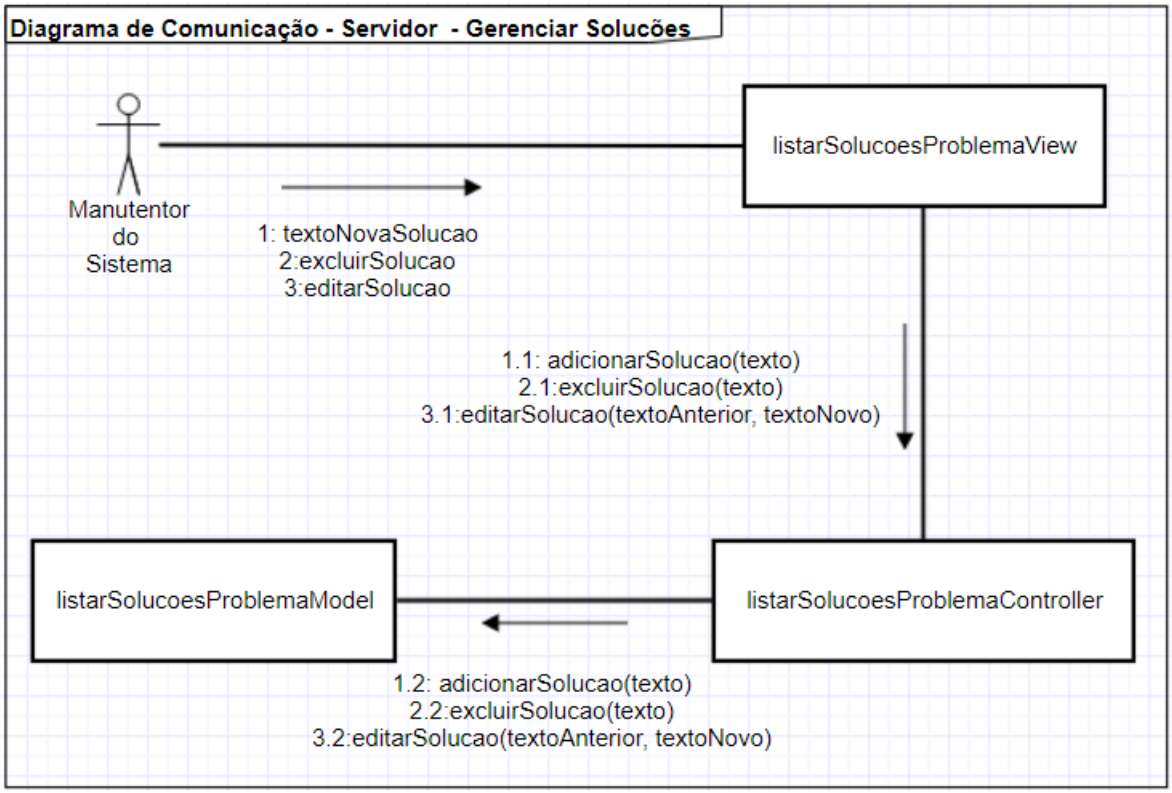

Figura 17 – Diagrama de comunicação – Gerenciar soluções.

Com relação aos diagramas de comunicação do servidor, a Figura 18 mostra o diagrama que diferencia dos demais diagramas do servidor por não incluir a adição de comentários, mas somente a edição ou exclusão.

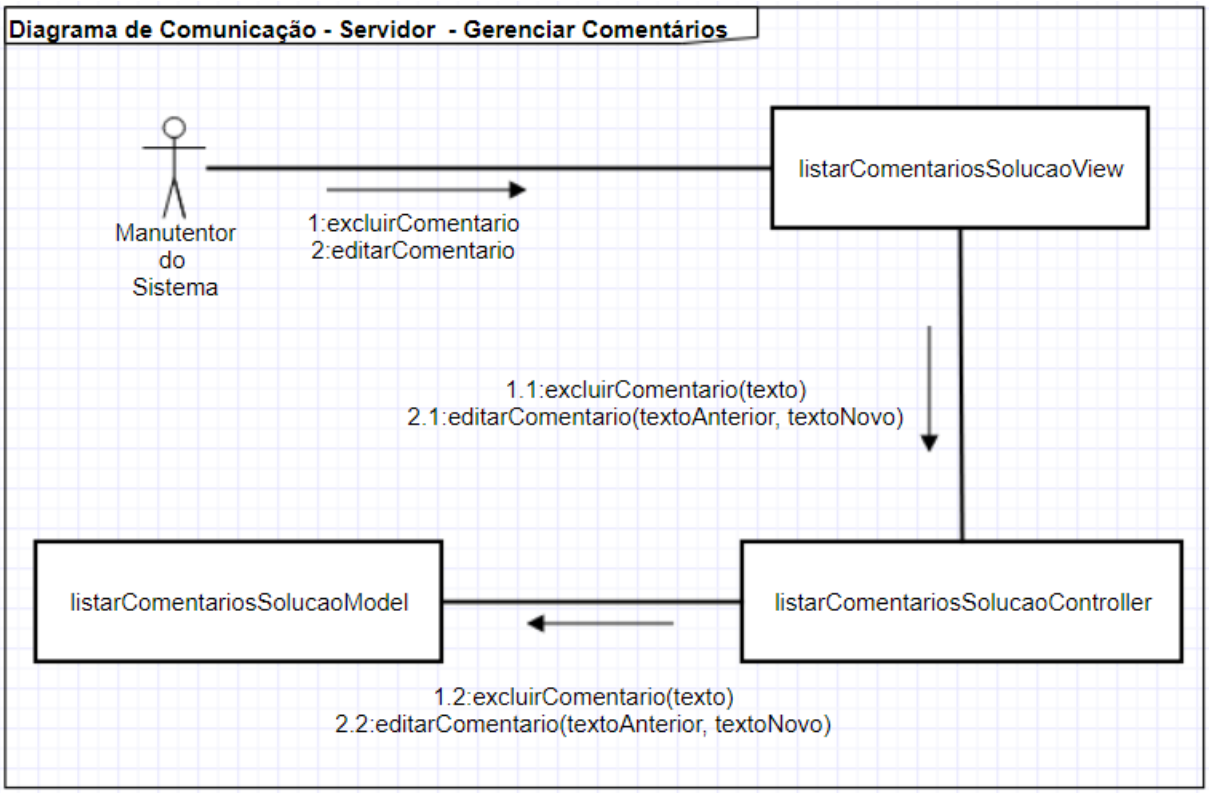

Figura 18 – Diagrama de comunicação – Gerenciar comentários.

3.2.4 Diagramas de Atividades

Como diagramas de atividades, os principais são o de adicionar comentário (Figura 19), editar parte da máquina (Figura 20), e excluir máquina (Figura 21).

O diagrama da Figura 19 mostra como funciona também a adição de informações também em outras partes do programa, porém no servidor, pois no dispositivo móvel não é possível adicionar nome de máquinas, por exemplo.

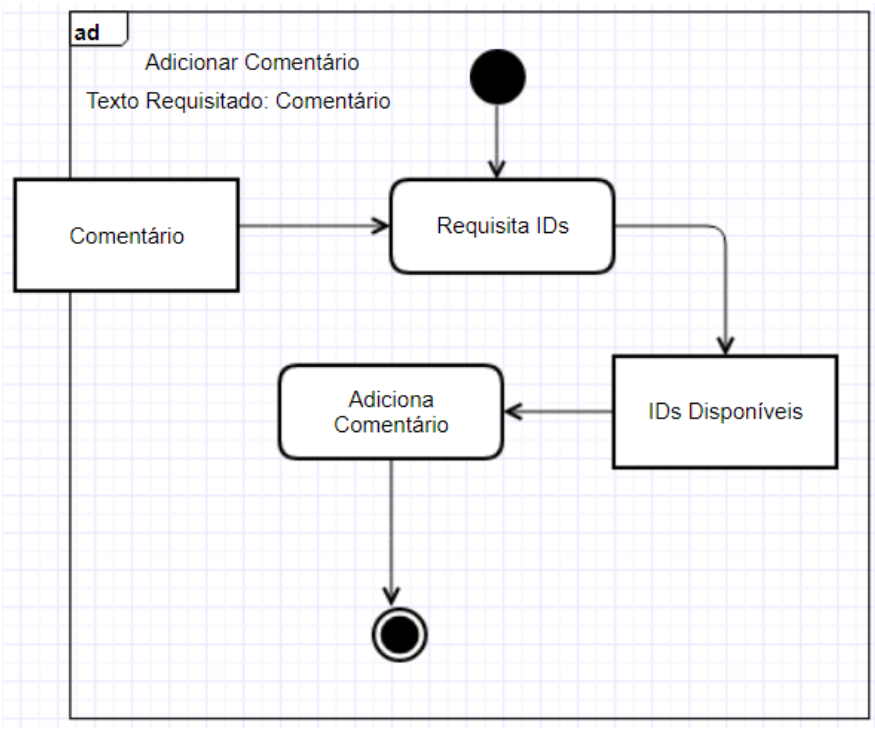

Figura 19 – Diagrama de Atividades – Adicionar comentário.

O diagrama da Figura 20 mostra o funcionamento da função de editar o nome da parte da máquina, disponível somente no programa do servidor.

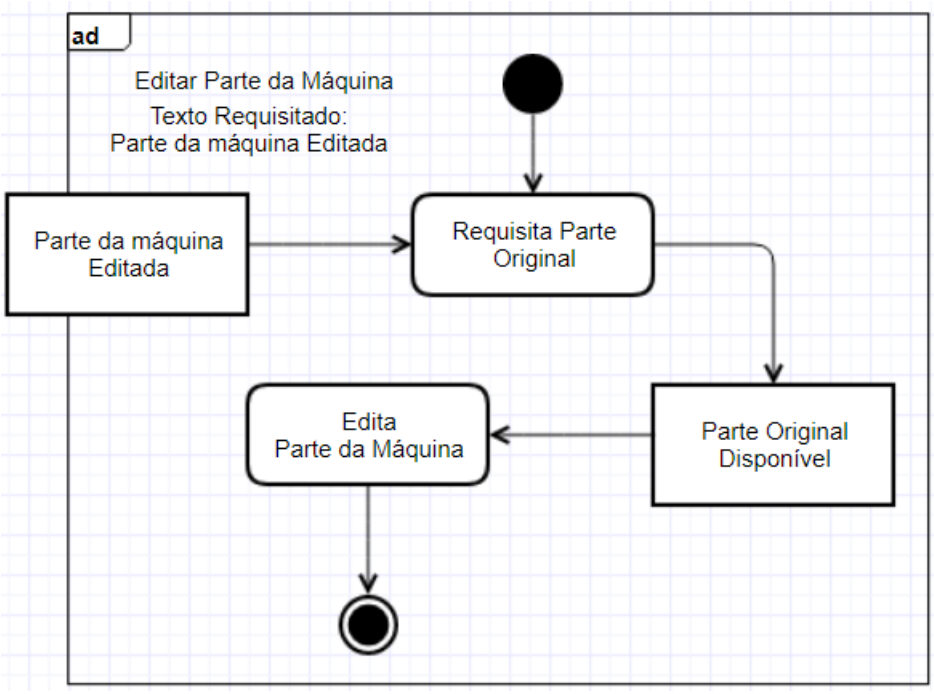

Figura 20 – Diagrama de Atividades – Editar parte da máquina.

Na Figura 21, como exemplo da função de excluir informações, o diagrama de excluir máquina faz parte da modelagem de funções que somente funcionam no programa do servidor.

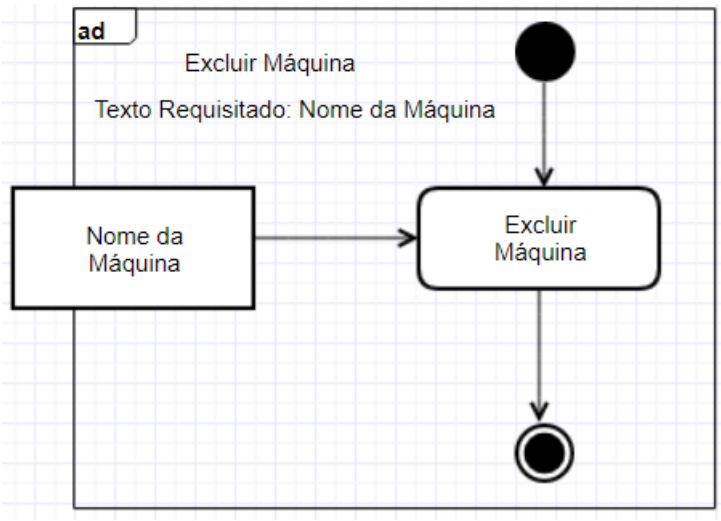

Figura 21 – Diagrama de Atividades – Excluir máquina.

Na Figura 20 é explicado o comportamento do sistema no que se trata de editar alguma informação. No caso do servidor, é permitida a edição de todos os campos, assim como a exclusão, demonstrada na Figura 21.

3.3 IMPLEMENTAÇÃO DO APLICATIVO

Para fins de facilitar o acesso a informações de soluções de problemas ou de passos de procedimentos de máquinas industriais grandes, de forma digital, as soluções foram disponibilizadas em um aplicativo desenvolvido para plataformas móveis, a princípio somente para Android, com o auxílio da ferramenta Android Studio, instalada em equipamento pessoal para fins de testes iniciais. No ambiente em que será usado o aplicativo não é permitido o uso de aparelhos celulares, portanto os aparelhos serão selecionados, então não será necessário desenvolver o aplicativo para versões anteriores do Android.

3.3.1 App Android

Para apresentar o aplicativo desenvolvido, este capítulo é dividido em visualização (com ênfase na interação com o usuário) e códigos importantes (para

dar ênfase aos pontos mais importantes na programação com relação ao comportamento do aplicativo).

3.3.1.1 Visualização

Ao abrir o aplicativo, o nome das máquinas é exibido em uma lista, como mostra a Figura 22.

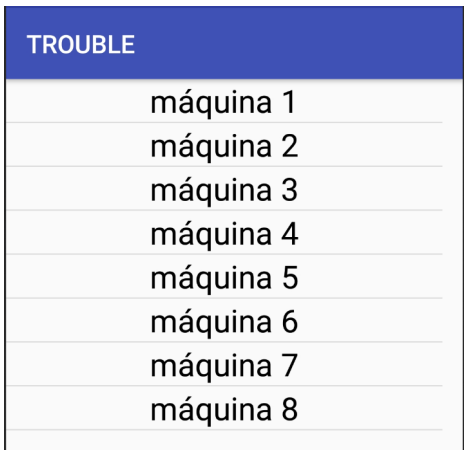

Figura 22 – Opções de nome das máquinas.

Assim que uma máquina é selecionada, outra lista com o nome das partes da máquina selecionada é exibida (Figura 23). Com a exibição das partes das máquinas, também é exibida a escolha anterior, ou seja, o nome da máquina selecionada.

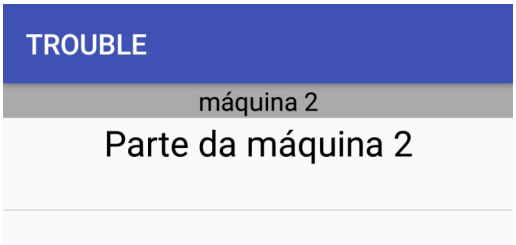

Figura 23 – Nome das partes das máquinas.

Selecionando uma parte da máquina, são exibidos os problemas ou os procedimentos pertencentes à parte selecionada (Figura 24). O texto que exibia a máquina selecionada, agora exibe também a opção selecionada da parte da máquina.

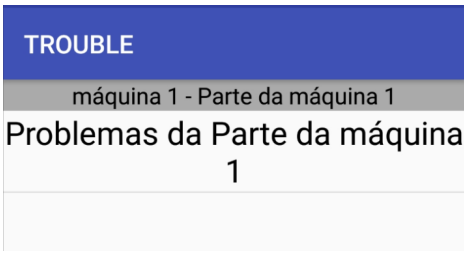

Figura 24 – Descrição dos problemas da parte selecionada.

Assim que o usuário seleciona algum problema na lista, são exibidas as possíveis soluções para o problema selecionado (Figura 25). Para seguir o padrão das listas anteriores, o texto de *feedback* (destacado com o fundo acinzentado) de seleção agora contém também a descrição do problema selecionado.

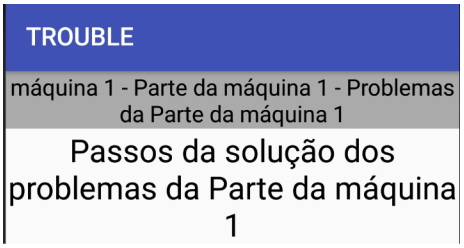

Figura 25 – Descrição das soluções do problema selecionado.

Ao ser selecionada alguma solução, é possível inserir comentários relacionados a cada etapa ou solução (Figura 26), de forma a melhorar a quantidade e qualidade de informações relacionadas ao problema ou procedimento em questão.

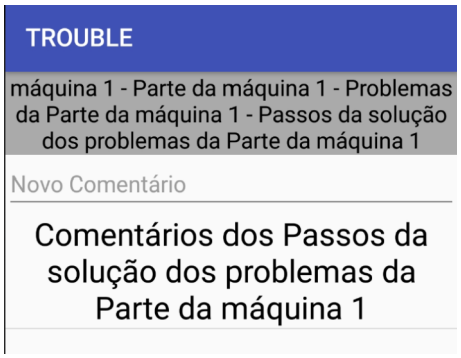

Figura 26 – Comentários da solução selecionada.

Com os comentários adicionados pelos usuários (Operadores das máquinas), o manutentor do sistema poderá editar as informações contidas nos níveis anteriores

incorporando as informações presentes nos comentários. Assim, na hora da consulta, as informações mais úteis terão acesso mais fácil.

3.3.1.2 Códigos Importantes

A partir das listas criadas respeitando os níveis de pesquisa desde o nome da máquina até a lista de comentários feitos a uma certa solução de um problema, foi necessário realizar a comunicação entre os níveis para as classes buscarem as informações corretas no banco de dados, para, então, essas informações serem exibidas nas listas conforme a seleção feita pelo usuário.

A começar pela primeira lista, a lista de nome das máquinas, o programa foi feito de forma a incrementar somente o *ID* responsável pelo primeiro nível de informações, o *ID0* (id zero). Assim, o nome das máquinas possuem *ID0* com valores maiores que zero, porém o *ID* usado nos demais níveis não são alterados, sendo cadastrados todos com valor zero.

Ao pesquisar nomes das máquinas no banco de dados os *IDs* pertencentes aos demais níveis (*ID1* ao *ID4*) permanecem em zero, pois somete os itens do primeiro nível possuem essa característica. A Figura 27 mostra o método utilizado para pesquisar o conteúdo a ser exibido na lista.

```
//LEVEL 0 - ArrayList para o nome das Máquinas
public ArrayList<Troubles> listaMaquinas()
{
    SQLiteDatabase db = getReadableDatabase();
    String colunas[] = new String[]{"ID0", "ID1", "ID2", "ID3", "ID4", "Texto"};
    Cursor cursor = db.query( table: "Troubles", colunas,
            selection: "ID1 = 0  AND ID2 = 0  AND ID3 = 0 AND ID4 = 0 ",
            selectionArgs: null,  groupBy: null,  having: null,  orderBy: "Texto");
    ArrayList<Troubles> retornoL0 = new ArrayList<~>();
    if (cursor.moveToFirst())
    {
        do
        {
            retornoL0.add(new Troubles(cursor.getInt( columnIndex: 0),
                    (cursor.getInt(  columnIndex: 1)),(cursor.getInt( columnIndex: 2)),
                    (cursor.getInt( columnIndex: 3)),(cursor.getInt( columnIndex: 4)),
                    (cursor.getString( columnIndex: 5))));
        } while (cursor.moveToNext());
    }
    return retornoL0;
}
```

Figura 27 – Pesquisa de cadastros do primeiro nível.

Assim que o usuário seleciona uma máquina, o texto exibido na opção selecionada é passado para a *activity* que exibe a lista de partes da máquina selecionada, pertencente ao nível dois.

Como o banco de dados é organizado por *ID*, foi preciso primeiramente resgatar o *ID0* da máquina selecionada. No método ilustrado pela Figura 28 é realizada a consulta do *ID0*.

```
//LEVEL 1 - Int para o ID0 do Texto
public int getID0 (String text)
{
    SQLiteDatabase db = getReadableDatabase();
    String colunas[] = new String[]{"ID0", "ID1", "ID2", "ID3", "ID4", "Texto"};
    Cursor cursor = db.query( table: "Troubles", colunas,
            selection: "ID1 = 0 AND Texto LIKE '%" + text + "%'" ,
            selectionArgs: null,  groupBy: null,  having: null,  orderBy: null,  limit: null);
    cursor.moveToFirst();
    return cursor.getInt( columnIndex: 0);
}
```

Figura 28 – Consulta do *ID0* pertencente ao nome da máquina selecionada.

Com a informação do *ID0*, é possível mostrar os cadastros do segundo nível, representado pelas partes da máquina selecionada. Enquanto o texto do nome da máquina é exibido no início da *activity*, a *ID* pertencente ao nome da máquina é usado para pesquisar as partes cadastradas no banco de dados.

Se a máquina que contém o *ID0* igual a um, por exemplo, então no banco de dados serão pesquisados os cadastros que tem o *ID0* igual a um, o *ID1* maior que zero, e os demais *IDs* iguais a zero, como mostra a Figura 29, com um método que pede somente a *ID0* como argumento.

```
//LEVEL 1 - ArrayList para o nome das Partes das Máquinas
public ArrayList<Troubles> listaParteMaquinas(int ID0)
{
    SQLiteDatabase db = getReadableDatabase();
    String colunas[] = new String[]{"ID0", "ID1", "ID2", "ID3", "ID4", "Texto"};
    Cursor cursor = db.query( table: "Troubles", colunas, selection: "ID0 = " + ID0 +
            " AND ID1 > 0 AND ID2 = 0 AND ID3 = 0 AND ID4 = 0 ",
            selectionArgs: null, groupBy: null, having: null, orderBy: "Texto");
    ArrayList<Troubles> retornoL1 = new ArrayList<~>();
    if (cursor.moveToFirst())
    {
        do
        {
            retornoL1.add(new Troubles(cursor.getInt( columnIndex: 0),
                    (cursor.getInt( columnIndex: 1)),(cursor.getInt( columnIndex: 2)),
                    (cursor.getInt( columnIndex: 3)),(cursor.getInt( columnIndex: 4)),
                    (cursor.getString( columnIndex: 5))));
        } while (cursor.moveToNext());
    }
    return retornoL1;
}
```

Figura 29 – Pesquisa de cadastros do segundo nível.

As informações a respeito da escolha do usuário no primeiro nível são somente duas, ou seja, o nome da máquina e o *ID0*. Estas duas informações serão passadas à *activity* do terceiro nível (problemas da parte selecionada) junto ao texto da opção selecionada no segundo nível. A Figura 30 mostra o método responsável por resgatar o texto da opção selecionada e enviar as três informações para o terceiro nível de escolha.

```
myListView1.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    {
        String machinePartSelected = parent.getItemAtPosition(position).toString();
        Intent i = new Intent( packageContext: Main2Activity.this, Main3Activity.class);
        i.putExtra( name: "machineNameTo3", machineSelected);
        i.putExtra( name: "machinePartNameTo3", machinePartSelected);
        i.putExtra( name: "machineIDto3", machineID);
        startActivity(i);
    }
});
```

Figura 30 – Resgate do texto da opção selecionada e envio de informações para o terceiro nível.

Representando o nível 3 estão os problemas da parte selecionada. Para exibir a lista de problemas cadastrados, primeiro é preciso saber o *ID* da parte da máquina selecionada. A Figura 31 mostra o código do método usado para resgatar o *ID*. Como argumento do método são passados o *ID* da máquina e o texto da opção de parte da máquina selecionada, tendo como retorno o *ID* esperado.

```
//LEVEL 2 - Int para o ID1 do Texto
public int getID1 (int idL0, String text)
{
    SQLiteDatabase db = getReadableDatabase();
    String colunas[] = new String[]{"ID0", "ID1", "ID2", "ID3", "ID4", "Texto"};
    Cursor cursor = db.query( table: "Troubles", colunas, selection: "ID0 = " + idL0 +
            " AND ID2 = 0 AND Texto LIKE '%" + text + "%'" , selectionArgs: null,
            groupBy: null, having: null, orderBy: null, limit: null);
    cursor.moveToFirst();
    return cursor.getInt( columnIndex: 1);
}
```

Figura 31 – Consulta do *ID1* pertencente à parte selecionada.

Com os *IDs* já disponíveis, é possível exibir a lista de problemas cadastrados para a parte selecionada. A Figura 32 contém o método usado para exibir a lista dos problemas. A diferença desse método para o método da Figura 29 é a adição da *ID1*. Da mesma forma foram feitos os métodos para exibir os outros dois níveis, ou seja, o método para exibir as soluções dos problemas e o método para exibir os comentários realizados para a solução selecionada.

```
//LEVEL 2 - ArrayList para os problemas das Partes
public ArrayList<Troubles> listaProblemasParte(int ID0, int ID1)
{
    SQLiteDatabase db = getReadableDatabase();
    String colunas[] = new String[]{"ID0", "ID1", "ID2", "ID3", "ID4", "Texto"};
    Cursor cursor = db.query( table: "Troubles", colunas, selection: "ID0 = " + ID0 +
            " AND ID1 = " + ID1 + " AND ID2 > 0 AND ID3 = 0 AND ID4 = 0 ",
            selectionArgs: null, groupBy: null, having: null, orderBy: "Texto");
    ArrayList<Troubles> retornoL2 = new ArrayList<~>();
    if (cursor.moveToFirst())
    {
        do
        {
            retornoL2.add(new Troubles(cursor.getInt( columnIndex: 0),
                    (cursor.getInt( columnIndex: 1)),(cursor.getInt( columnIndex: 2)),
                    (cursor.getInt( columnIndex: 3)),(cursor.getInt( columnIndex: 4)),
                    (cursor.getString( columnIndex: 5))));
        } while (cursor.moveToNext());
    }
    return retornoL2;
}
```

Figura 32 – Pesquisa de cadastros do terceiro nível.

Além de navegar entre os cadastros, procurando soluções de problemas, é possível também adicionar comentários para as soluções. Para isso basta o usuário selecionar alguma solução na lista representada pela Figura 25 e inserir o comentário no campo disponível acima da lista de comentários, mostrada pela Figura 26. A Figura 33 mostra o método usado para adicionar um item na lista de comentários da solução selecionada.

```
//LEVEL 4 - Adiciona novo comentário
public boolean adicionaComentario(int ID0, int ID1, int ID2, int ID3, String comentario)
{
    SQLiteDatabase db = getReadableDatabase();
    ContentValues pr = new ContentValues();
    pr.put("ID0", ID0);
    pr.put("ID1", ID1);
    pr.put("ID2", ID2);
    pr.put("ID3", ID3);
    pr.put("ID4", 1);
    pr.put("Texto", comentario);
    db.insert( table: "Troubles", nullColumnHack: null, pr);
    return true;
}
```

Figura 33 – Adição de comentário.

Os comentários sempre são inseridos com o *ID4* com valor igual a um com o objetivo de adicionar no banco de dados local. Ao se comunicar com o servidor, os comentários adicionados são organizados pelo servidor e distribuídos com o id sequencial para os demais dispositivos móveis.

A forma de organização na adição de itens pelo servidor é a mesma usada no aplicativo do dispositivo móvel, cada item do segundo nível, por exemplo, carrega o ID do item selecionado no primeiro nível, possibilitando, então, que o aplicativo funcione como o esperado.

3.4 AVALIAÇÃO DO APLICATIVO COM O USUÁRIO

Através da avaliação do aplicativo com cinco futuros usuários foi possível ter algumas ideias para melhoria do aplicativo. Quanto à usabilidade, o aplicativo foi classificado somente com pontos positivos por apresentar usabilidade ótima sob a ótica dos usuários. Além da ótima usabilidade, a organização dos dados, a simplicidade e a objetividade também foram elogiadas.

Quanto à funcionalidade, os pontos positivos foram com relação ao quanto poderá ajudar se for implementado e alimentado com informações regularmente. Os pontos negativos da funcionalidade envolvem a falta de integração com a planilha existente e a falta da função de carregar fotos.

4 Conclusão

A manutenção de máquinas industriais pode ser auxiliada pela tecnologia de dispositivos móveis. O desenvolvimento do presente aplicativo possibilitou o acesso fácil de informações importantes para a resolução de problemas das máquinas industriais pelo fato de ter uma organização de informações que facilite a navegação.

Como anteriormente o acesso a tais informações era difícil, ou seja, precisava conseguir acesso a um computador, procurar a pasta onde ficam os relatórios, para então conseguir alguma informação, sentiu-se a necessidade de desenvolver um método mais fácil para acesso a essas informações.

Com o aplicativo desenvolvido, as informações existentes estão no dispositivo móvel que pode ser carregado facilmente até a máquina, tornando desnecessário, então, imprimir ou anotar alguma informação em algum papel para levar até a máquina, pois a informação estará completa ali na mão do usuário do aplicativo.

As dificuldades no desenvolvimento estão relacionadas ao design do aplicativo para mostrar ao usuário somente as informações necessárias para cada etapa do processo de procura de informações e à comunicação entre o aplicativo do dispositivo móvel e o programa no servidor para sincronizar os dados inseridos.

Melhorias no aplicativo estão relacionadas à facilidade de inserção e procura de informações. Caso o usuário pudesse inserir fotos, carregar informações a partir de uma planilha ou procurar alguma informação a partir de uma caixa de texto, sendo exibidas informações de todo o banco de dados que contém as palavras inseridas para pesquisa, o aplicativo estaria mais completo, do ponto de vista dos usuários que o testaram.

O resultado obtido foi um aplicativo que pode mostrar informações pertinentes especificamente ao que o usuário procurar, de forma organizada e direta, assim como dar feedback a respeito do processo de procura de informações (por exemplo, quando o usuário seleciona uma máquina, o nome dessa máquina aparece no topo do aplicativo para lembrar ao usuário sobre sua escolha e também para impedir confusão a respeito das informações).

Referências

ANDROID DEVELOPER: Android Studio, o IDE oficial do Android. Disponível em <https://developer.android.com/studio/index.html>. Acesso em: 12 out. 2017.

ANDROID DEVELOPER: Layouts, visualização em lista. Disponível em <https://developer.android.com/guide/topics/ui/layout/listview.html>. Acesso em: 24 set. 2017.

ANDROIDPRO INTENTS. Disponível em <https://www.androidpro.com.br/intents/> Acesso em: 14 out. 2017

ATEOMOMENTO: Diagrama de atividades, Disponível em <http://www.ateomomento.com.br/uml-diagrama-de-atividades>. Acesso em: 14 out. 2017.

AUTOMAÇÃO PRODUÇÃO: Automação da produção industrial. Disponível em <http://mundoeducacao.bol.uol.com.br/geografia/automacao-producao-industrial.htm>. Acesso em: 14 out. 2017

da Rocha, Jordão Silva: **O QUE É IHM**. Disponível em <http://www.ajautomacao.com/o-que-e-ihm/>. Acesso em: 14 out. 2017

Davis, Matthew. **JBOSS VS TOMCAT:** choosing a java application server, Disponível em <https://www.futurehosting.com/blog/jboss-vs-tomcat-choosing-a-java-application-server/>. Acesso em: 18 out. 2017.

DRDOBBS: Tutorial de Webservices, Disponível em <http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069>. Acesso em: 18 out. 2017.

ELIPSE: Sistema de automação de uma subestação. Disponível em <https://www.elipse.com.br/case/elipse-power-automatiza-subestacao-padre-fialho-da-energisa/>. Acesso em: 14 out. 2017.

GOOGLEPLAY: Aplicativo Whatsapp. Disponível em <https://play.google.com/store/apps/details?id=com.whatsapp&hl=en>. Acesso em: 24 set. 2017.

IBM: Diagramas de comunicação, Disponível em <https://www.ibm.com/support/knowledgecenter/pt-br/SS5JSH_9.1.0/com.ibm.xtools.sequence.doc/topics/ccommndiag.html>. Acesso em: 14 out. 2017.

LUCIDCHART: O que é UML. Disponível em <https://www.lucidchart.com/pages/pt/o-que-%C3%A9-uml>. Acesso em: 17 out. 2017.

Milfont, Luis. **DESMISTIFICANDOANDOID**: Como consumir Webservice, Disponível em <http://desmistificandoandroid.blogspot.com.br/2015/05/como-consumir-webservice-rest-no.html>. Acesso em: 18 out. 2017.

SAMPAIO, Marcus C.: **Diagramas de casos de uso**, Disponível em <http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/usecases.htm>. Acesso em: 14 out. 2017.

SAMPAIO, Marcus C.: **Diagramas de classes**, Disponível em <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/uml/diagramas/classes/classes1.htm>. Acesso em: 14 out. 2017.

SETUP DE MÁQUINA: Tempo perdido ou tempo investido. Disponível em <http://www.kitemes.com.br/2014/08/11/setup-de-maquina-tempo-perdido-ou-tempo-investido/>. Acesso em: 14 out. 2017

SIMETRIZA: Principais causas de desgaste em equipamentos industriais. Disponível em <http://www.simetriza.com.br/conheca-as-principais-causas-de-desgaste-em-equipamentos-industriais/>. Acesso em: 14 out. 2017.

SOAWEBSERVICES: Como funcionam os Webservices, Disponível em <http://www.soawebservices.com.br/como-funciona.aspx> Acesso em: 21 nov. 2017.

SQLITE: Índice de palavras. Disponível em <https://www.sqlite.org/keyword_index.html>. Acesso em: 10 out. 2017.

THESERVERSIDE: Apache TomCat, Disponível em <http://www.theserverside.com/definition/Tomcat>. Acesso em: 18 out. 2017.

TOMCAT: Apache Tomcat, Disponível em <http://tomcat.apache.org/>. Acesso em: 18 out. 2017.

VOGELLA: Android Intents – Tutorial. Disponível em <http://www.vogella.com/tutorials/AndroidIntent/article.html>. Acesso em: 24 set. 2017.