Desenvolvimento de Software para Auxiliar na Inspeção de Pavimentos em Sítios Aeroportuários - InspecPav

Leandro E. S. de Oliveira, André C. da Silva

[1]Grupo de Pesquisa Mobilidade e Novas Tecnologias de Interação
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de Ciência e Tecnologia de São Paulo (IFSP)
Avenida Thereza Ana Cecon Breda, N.º 1896,
Vila São Pedro – 13183-250 – Hortolândia – SP – Brasil

leandro.e@aluno.ifsp.edu.br, andre.constantino@ifsp.edu.br

Abstract. *Due to the adversities encountered in the situational inspection process of pavements at Viracopos Airport, the present work aims to present part of the development process of an application to assist in inspections. Addressing the choices made as well as the technologies used. As a result, it is expected that the study and development of this project will provide a prototype that can collaborate with the Viracopos airport inspection process, thereby alleviating the problems of disorganization, inaccurate reports and wasted time in the processes.*

Resumo. *Devido às adversidades encontradas no processo de inspeção situacional de pavimentos do Aeroporto Viracopos, o presente trabalho tem como objetivo apresentar parte do processo de desenvolvimento de um aplicativo para auxiliar nas inspeções. Abordando as escolhas tomadas, assim como as tecnologias empregadas. Como resultado, espera-se que o estudo e o desenvolvimento desse projeto forneçam um protótipo que consiga colaborar com o processo de inspeção do aeroporto Viracopos, com isso amenizando os problemas de desorganização, relatórios imprecisos e desperdícios de tempo nos processos.*

1. Introdução

A Engenharia Civil, mais especificamente em pavimentação aeroportuária, possui órgãos responsáveis por garantir a qualidade dos elementos que compõem a segurança da aviação civil, como a Agência Nacional de Aviação Civil (ANAC). Uma de suas documentações de padrões a serem seguidos é a SGPA (Sistema de Gerenciamento de Pavimentos Aeroportuários) [INFRAERO 2017]. A ANAC, por meio do RBAC nº 153 (Regulamentos Brasileiros da Aviação Civil), define que o Operador de Aeródromo deve estabelecer e documentar requisitos e procedimentos de monitoramento e de avaliação do estado do pavimento baseados em metodologia de sistema de gerenciamento de pavimentos.

Dentro da RBAC nº 153 são listados diversos requisitos que devem ser atendidos pelos aeroportos; abrangem desde treinamentos de profissionais até condições estruturais de pavimentos. Dentro da subseção de manutenção aeroportuária, existe o índice 153.203 - áreas pavimentadas - generalidades, onde é possível encontrar a seção 2, listagem de diversos tópicos que definem as condições que o pavimento deve atender. Na busca pela

garantia desses itens, a inspeção com *check-list* é uma ferramenta poderosa para identificação de defeitos, e para posteriormente que as correções sejam programadas.

Atualmente para realização desse procedimento, um inspetor deve ir para o setor que passará pela inspeção portando diversas folhas impressas para apontamento dos defeitos bem como o dispositivo móvel para registro fotográfico de patologias. Após a finalização da inspeção, deve retornar para sua base de trabalho, descarregar as fotos do processo no seu computador, associar cada defeito à sua respectiva fotografia e então elaborar um relatório para o engenheiro responsável. Como todo processo é realizado de forma "manual", existem aberturas para erros. Por exemplo, não vincular a foto correta ao defeito, ou apontar de maneira errada horário e duração da inspeção, isso tudo colabora de maneira negativa com o prazo e a qualidade para realizar a manutenção. Existe ainda desperdício de tempo entre descarregar as fotos, fazer associação de qual foto é referente a qual defeito, elaborar o relatório e por fim enviar.

Como dito por Ohno (1997), "desperdícios se referem a todos os elementos da produção que só aumentam os custos sem agregar valor". Visando minimizar os desperdícios identificados nesse processo, o objetivo desde trabalho é o desenvolvimento de uma aplicação para o processo de inspeção de pavimentos com *check-list*, seguindo a SGPA. Esta aplicação possibilitará ao usuário elaborar *check-lists* evidenciando as irregularidades do pavimento, vincular de forma automática a patologia com suas respectivas fotos e também a localização exata, gerar os relatórios de cada setor específico do pavimento contando com posicionamento via GPS, apontamento automático de horário de início e término da atividade e duração da atividade.

Este artigo está estruturado da seguinte maneira: a Seção 2 descreve quais são os elementos utilizados no referencial teórico, a Seção 3 aborda o aplicativo utilizado como trabalho correlato chamado INPAV. Seção 4 evidencia a metodologia utilizada para o desenvolvimento do trabalho, e a Seção 5 descreve o desenvolvimento do trabalho levando em consideração a relação dos casos de uso com os incrementos entregues. As conclusões são apresentadas na Seção 6.

2. Referencial teórico

2.1. Manual de Sistema de Gerenciamento de Pavimentos Aeroportuários (SGPA)

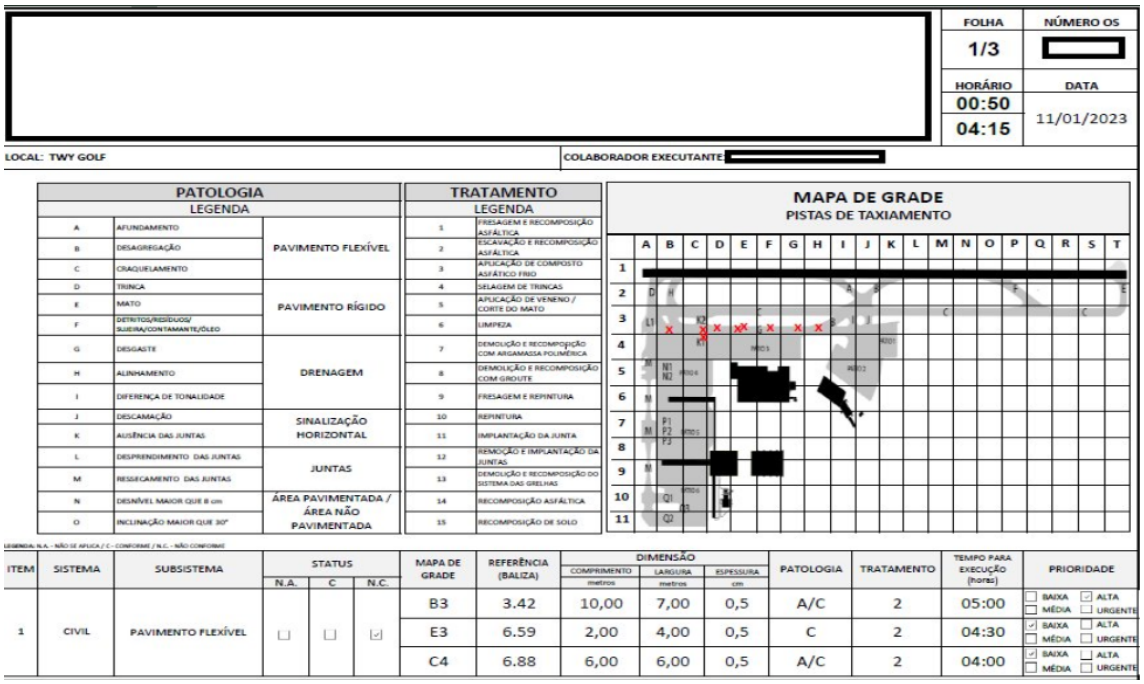
ANAC é o órgão responsável pelos aeroportos no Brasil. Foi criada em 2005, com objetivo de regular e fiscalizar as atividades de aviação civil e de infra-estrutura aeronáutica e aeroportuária [Brasil 2005]. Dentre uma das suas atribuições, cabe à ANAC certificar a integridade do pavimento de pistas de pouso e decolagem, táxis, pátios e outras áreas de circulação e manobras.

Para garantir a segurança, estes aeroportos são submetidos à regulamentações específicas e obrigatórias que detalham e instituem processos que devem ser seguidos. A documentação que trata especificamente sobre o tema desse trabalho é o REGULAMENTO BRASILEIRO DA AVIAÇÃO CIVIL nº 153 EMENDA nº 06 [ANAC 2012]. Dentre outras documentações, esse regulamento faz com que os operadores aeroportuários devam estabelecer e documentar requisitos e procedimentos de monitoramento e de avaliação do estado do pavimento baseados em metodologia de sistema de gerenciamento de pavimentos.

Sabendo disso, o procedimento de avaliação contínua do estado de conservação de pavimentos é adotado. Essa avaliação é realizada a cada oito horas pelos profissionais conhecidos como "Fiscais de pátio", e mensalmente pelo inspetor técnico do setor de engenharia e manutenção. Ambos possuem objetivos similares: encontrar patologias que impeçam a operação. Porém, o fiscal de pátio faz a inspeção buscando os defeitos e, caso identificado, pode acionar a equipe de manutenção de plantão para um reparo paliativo, mas se a escala do defeito for grande, poderá isolar a região, visto que um reparo de maior escala necessita de equipe especializada para reparo. A avaliação do técnico é mais profunda e precisa. Conta com, além da bagagem técnica do inspetor, também com uma ferramenta de auxílio, conhecida como *check-list*.

Posteriormente, baseado nesse *check-list*, é que as manutenções serão programadas pelo engenheiro responsável. Esta ferramenta possui variações de acordo com o setor que é realizada a inspeção. Por exemplo, quando a inspeção está sendo realizada em uma posição de aeronave, podem ser identificadas patologias tanto em pavimentos flexíveis, rígidos, sinalizações horizontais (pinturas) e sinalizações verticais (placas). Já quando a área inspecionada é uma *taxiway*(pista de movimentação de aeronaves), as patologias que podem ser encontradas são em pavimento flexível e de sinalização horizontal.

Devido a essa variação de patologias que podem ser encontradas de acordo com o local, o modelo de *check-list* é construído de forma "genérica", possuindo todos os possíveis defeitos e, assim sendo, existem campos que não são preenchidos durante uma determinada inspeção.



FOLHA	NÚMERO OS
1/3	
HORÁRIO	DATA
00:50 / 04:15	11/01/2023

LOCAL: TWY GOLF — COLABORADOR EXECUTANTE:

PATOLOGIA

LEGENDA		
A	AFUNDAMENTO	PAVIMENTO FLEXÍVEL
B	DESAGREGAÇÃO	
C	CRAQUELAMENTO	
D	TRINCA	PAVIMENTO RÍGIDO
E	MATO	
F	DETRITOS/RESÍDUOS/ SUJEIRA/CONTAMANTE/ÓLEO	
G	DESGASTE	DRENAGEM
H	ALINHAMENTO	
I	DIFERENÇA DE TONALIDADE	
J	DESCAMAÇÃO	SINALIZAÇÃO HORIZONTAL
K	AUSÊNCIA DAS JUNTAS	
L	DESPRENDIMENTO DAS JUNTAS	JUNTAS
M	RESSECAMENTO DAS JUNTAS	
N	DESNÍVEL MAIOR QUE 8 cm	ÁREA PAVIMENTADA / ÁREA NÃO PAVIMENTADA
O	INCLINAÇÃO MAIOR QUE 30°	

TRATAMENTO

LEGENDA	
1	FRESAGEM E RECOMPOSIÇÃO ASFÁLTICA
2	ESCAVAÇÃO E RECOMPOSIÇÃO ASFÁLTICA
3	APLICAÇÃO DE COMPOSTO ASFÁTICO FRIO
4	SELAGEM DE TRINCAS
5	APLICAÇÃO DE VENENO / CORTE DO MATO
6	LIMPEZA
7	DEMOLIÇÃO E RECOMPOSIÇÃO COM ARGAMASSA POLIMÉRICA
8	DEMOLIÇÃO E RECOMPOSIÇÃO COM GROUTE
9	FRESAGEM E REPINTURA
10	REPINTURA
11	IMPLANTAÇÃO DA JUNTA
12	REMOÇÃO E IMPLANTAÇÃO DA JUNTAS
13	DEMOLIÇÃO E RECOMPOSIÇÃO DO SISTEMA DAS GRELHAS
14	RECOMPOSIÇÃO ASFÁLTICA
15	RECOMPOSIÇÃO DE SOLO

MAPA DE GRADE
PISTAS DE TAXIAMENTO

ITEM	SISTEMA	SUBSISTEMA	STATUS N.A.	STATUS C	STATUS N.C.	MAPA DE GRADE	REFERÊNCIA (BALIZA)	DIMENSÃO COMPRIMENTO (metros)	DIMENSÃO LARGURA (metros)	DIMENSÃO ESPESSURA (cm)	PATOLOGIA	TRATAMENTO	TEMPO PARA EXECUÇÃO (horas)	PRIORIDADE
1	CIVIL	PAVIMENTO FLEXÍVEL	☐	☐	☑	B3	3.42	10,00	7,00	0,5	A/C	2	05:00	☐ BAIXA ☑ ALTA ☐ MÉDIA ☐ URGENTE
						E3	6.59	2,00	4,00	0,5	C	2	04:30	☑ BAIXA ☐ ALTA ☐ MÉDIA ☐ URGENTE
						C4	6.88	6,00	6,00	0,5	A/C	2	04:00	☑ BAIXA ☐ ALTA ☐ MÉDIA ☐ URGENTE

Figura 1. Modelo atual utilizado para registro das patologias encontradas durante inspeção

Atualmente, a inspeção é realizada por um colaborador que percorre a área preenchendo um modelo impresso como pode ser observado (Figura 1), onde são apontadas as patologias encontradas. Dentro desse apontamento, são imprescindíveis informações

como: local, estado de conformidade, referência, mapa de grade, defeito, dimensões e prioridade.

2.2. Desenvolvimento Web

A fim de atender as novas demandas que surgiram com o advento da Web, fez-se necessário a criação de uma nova repartição da Engenharia de Software, essa chamada Engenharia Web. Construída sobre a Engenharia de Software onde, na busca constante da qualidade, utilizam-se de metodologias, ferramentas e técnicas de desenvolvimento, a Engenharia Web surge para atender as novas características que a diferencia do desenvolvimento de aplicações locais, tais como sensibilidade ao conteúdo, carga imprevisível, acesso simultâneo dentre outras [PRESSMAN 2011].

Como abordado por [Beder 2012], com passar do tempo, houveram mudanças significativas na utilização e função das aplicações Web em comparação com o começo da Internet. Atualmente aplicações Web carregam muito mais responsabilidades e funcionalidades, cada vez mais integradas com a estratégia de negócios, visando mais do que nunca agregar valor ao usuário final.

2.3. Modelo Incremental

O modelo adotado no desenvolvimento deste trabalho é o Incremental. Consiste em ciclos de trabalho que possuem, dentre outras etapas, especificação, projeto, testes de unidade, integração com sistema e coleta de *feedback*. Quando diversos ciclos de incrementos são entregues e unificados, eles compõem o projeto final [PRESSMAN 2011].

3. Trabalhos correlatos

Buscando um melhor entendimento sobre o assunto e o ambiente para qual este software será desenvolvido, foram realizadas buscas de sistemas com propostas similares.

Durante a busca, o aplicativo encontrado e levado em consideração para este artigo foi o INPAV, software de inspeção em pavimento flexível voltado para as rodovias de Portugal. O sistema começou a ser desenvolvido pelo Engenheiro Nuno Verdelho Trindade em 2006, prototipado em 2009 e amplamente testado em campo e apresentado em congressos [EngenhariaCivil.com].

Neste aplicativo, seria possível realizar a inspeção de pavimento, adicionar ou retirar determinada patologia, ajustar posição real do defeito em relação sinal de GPS, dividir a tarefa de inspeção em mais de um inspetor.

O aplicativo integra também uma interface Web onde é possível monitorar e acompanhar o processo de inspeção sendo realizado e a relação de dados que estão sendo apontados durante a inspeção.

4. Materiais e métodos

Primeiramente, será realizada a pesquisa bibliográfica relacionada ao tema do sistema, como desenvolvimento Web, desenvolvimento para dispositivos móveis, persistência de dados, dentre outros.

Será realizada pesquisa por aplicações que buscam atender a mesma necessidade. A partir dessa busca, será possível identificar quais foram os pontos atendidos, quais são os aspectos que ainda não foram solucionados e possíveis melhorias.

A partir de reuniões com os técnicos responsáveis pelas inspeções e os engenheiros responsáveis pela programação das atividades, será gerado um *backlog*, provendo os requisitos iniciais para aplicação. *Backlog* é um termo utilizado para descrever um conjunto de tarefas, atividades ou itens pendentes que precisam ser realizados ou concluídos. O desenvolvimento desse *product backlog*, conjunto de *backlog* que compõem o produto final (aplicativo) ,se dá a partir da análise e refinamento para identificação das principais "dores"do cliente. Com essas necessidades identificadas, é então iniciado o processo de definir quais são as funcionalidades que irão atender a demanda do cliente. Nesse momento, não precisa ser necessariamente identificada cada uma das especificações do *backlog*, visto que esse processo de refinamento será realizado em cada novo ciclo de trabalho curto com prazo definido, onde o desenvolvedor entrega um novo conjunto de funcionalidades ou incremento, também conhecido como *"Sprint"*.

A metodologia adotada será incremental, ocorrendo quatro *Sprints* de desenvolvimento baseadas no *backlog*. Ao início de uma nova *Sprint* será, então, realizado o refinamento dos requisitos para cada item do *backlog*, o planejamento da funcionalidade, modelagem de acordo com os requisitos que foram refinados, o desenvolvimento, testes e a implementação do novo incremento no *software*.

5. Desenvolvimento do trabalho

5.1. Fase de conceituação

Foram realizadas duas reuniões inicialmente. Uma com os técnicos responsáveis pelas inspeções e outra com o engenheiro responsável pelos técnicos. A partir dessas reuniões foi possível elaborar uma primeira versão de requisitos do sistema. Dentre as dificuldades relatas pelos técnicos, o fato de ser um processo que gera muitos documentos acaba por atrapalhar a organização e emissão do relatório final, ocorrendo até mesmo casos em que documentos foram perdidos e não foram listados para realização da manutenção. Por parte do engenheiro, o principal ponto ressaltado foi a necessidade do sistema realizar a contabilização do tempo da atividade de forma automática, com o principal objetivo de evitar fraudes de duração de uma inspeção.

Quanto aos requisitos não-funcionais é ressaltada a importância de que seja de fácil aprendizado, ou seja, usabilidade. Facilidade de aprender a utilizar a ferramenta e também que seja de fácil manuseio. Após a coleta de *feedback* com o orientador do projeto a partir do protótipo, foi constado o excesso de iterações para realizar determinadas tarefas, em outras palavras, haviam muitos toques na tela para realizar uma ação, ponto esse listado e que será corrigido ao longo do desenvolvimento.

A aplicação também deve ser confiável, visto que tem como objetivo o auxiliar no processo de garantia de qualidade e operabilidade do aeroporto. Deverá estar disponível para realizar um serviço que foi requisitado por um usuário. A taxa de falhas deve ser mínima, fazendo com que a taxa de falhas seja um requisito não funcional de grande relevância.

Durante o desenvolvimento, também serão adotados meios para garantir a segurança da aplicação, visto que se tratam de dados sensíveis. Então, serão empregados, por exemplo: controle de usuários a partir de autenticação, permissões de acesso sendo manipuladas somente pelos administradores do sistema.

De posse dos dados colhidos durante as entrevistas, foi realizada uma análise para extrair requisitos para este sistema. Após a identificação das possíveis funcionalidades que o sistema deve possuir, foi elaborado a primeira versão de um caso de uso (Figura 2).

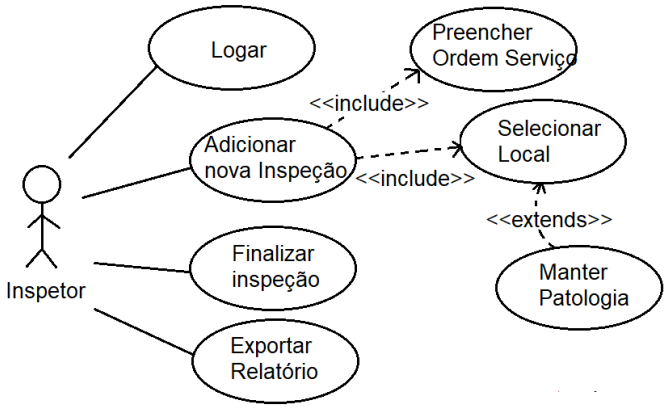


Figura 2. Diagrama de casos de uso para o contexto da aplicação para auxiliar na inspeção de pavimentos em sítios aeroportuários

Como pode ser observado, foi identificado um ator, no caso o inspetor responsável por realizar a inspeção do local. Buscando garantir a integridade e segurança da inspeção, foi identificada a funcionalidade que proporciona ao ator, no caso inspetor, realizar o login da aplicação. Esse ator é capaz de adicionar uma nova inspeção adicionando um novo número de ordem de serviço e vinculando esse número à um local onde essa inspeção será realizada. Deve também ser capaz de listar quais são as patologias encontradas e em quais locais elas foram identificadas e apontadas. Ao final, a inspeção pode ser encerrada e, então, pode-se gerar um relatório em PDF.

Após a elicitação dos requisitos, foi possível elaborar protótipos para facilitar o entendimento e validar com o inspetor se o projeto está caminhando alinhado com as suas expectativas e demandas (Figura 3). Baseado em uma pesquisa, o Canva foi o escolhido para elaboração de um protótipo de baixa fidelidade, a fim de transmitir a ordenação e fluxo de ações demandadas pelo usuário.

O Canva é uma plataforma de design gráfico que permite criar facilmente uma variedade de materiais visuais, desde apresentações e panfletos até posts para redes sociais, usando uma interface intuitiva e modelos pré-fabricados [Canva 2023].

Na Figura 3, é possível identificar: A) Tela inicial, para onde o usuário será direcionado após efetuar o login. B) Tela de preenchimento da ordem de serviço, momento em que o inspetor preenche os campos com os respectivos detalhes da ordem de serviço. C) Tela de localização, na qual o inspetor irá selecionar qual quadrante do mapa de grade ocorrerá a inspeção. D) Tela de apontamento de nova patologia, inspetor poderá adicionar uma nova patologia ao local que está sendo vistoriado, visto que, nesse momento, o sistema já terá a informação de qual é a ordem de serviço, quem é o inspetor responsável e qual setor está sendo realizada a inspeção. E) Tela de identificação de patologia, onde o inspetor poderá identificar o tipo de pavimento que contém aquela determinada patologia. F) Tela de especificação de patologia, onde poderá ser apontado a qual subsistema de defeito pertence aquela patologia, qual sua dimensão, inserir uma referência de localização

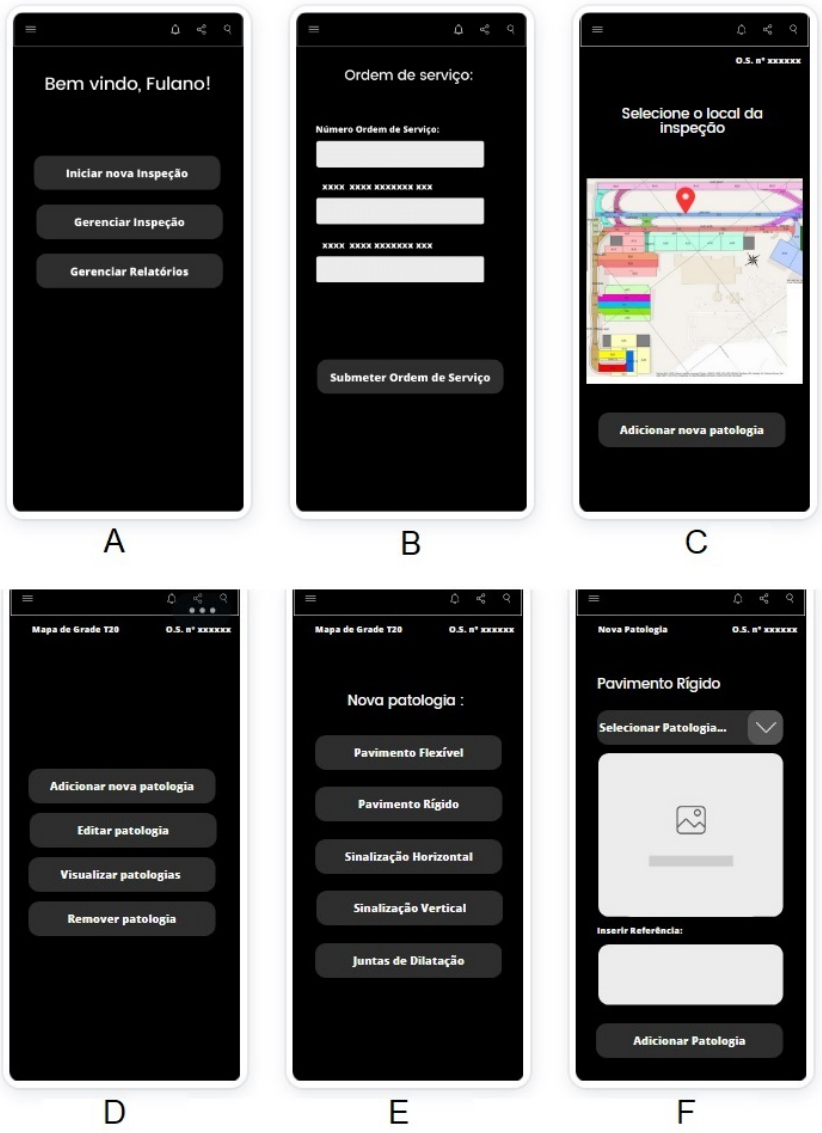


Figura 3. Protótipos de telas para o aplicativo em desenvolvimento

e uma registro fotográfico do defeito.

Essa primeira versão de protótipos de telas foi desenvolvido para realizar a validação com a engenheria responável pela equipe de manutenções de pavimento do aeroporto está que representa a figura de *Podruct Owner* do projeto, este que é o responsável por fornecer as informações necessárias para tomada de decisões quanto aos rumos do software. Foram sugeridas alterações de *layout* para o funcionamento da tela de listagem das patologias. Para o desenvolvimento, foi adotado um modelo de exibição similar à navegação de aplicativos de músicas, como *Spotfy*. O protótipo leva o cenário de cadastramento de uma nova patologia em pavimento rígido, partindo do principio que o usuário já está previamente logado como um inspetor. Com base no *feedback* do orientador, foi constatado que para realização, há um número elevado de iterações por parte do usuário. A redução de interações será adotada como principio para todo o desenvolvimento do trabalho, visando a melhor experiência de usuário e facilidade nos aspectos de usabilidade e

facilidade de aprendizado.

5.2. Seleção de Tecnologias para Desenvolvimento do Protótipo

Após pesquisas e ponderamentos quanto à escolha da tecnologia, foi então decidido pelo desenvolvimento em *React Native*. A escolha leva em conta principalmente o aspecto do momento de mercado de trabalho e as possíveis portas que seriam abertas tendo em vista a bagagem que o desenvolvimento poderá agregar.

Para o desenvolvimento do *back-end* da aplicação, optou-se então pela utlização de Node.JS, buscando o reaproveitamento do conhecimento em JavaScript.

5.3. Desenvolvimento das Telas Iniciais, Componentes e Revisões nos Casos de Uso

O primeiro aspecto do desenvolvimento foi um componente em JavaScript para disponibilizar acesso à câmera do aparelho de forma nativa, sendo este disponibilizado para ser consumido posteriormente. O foco inicial foi começar pelas tarefas que se mostravam mais complexas. Assim sendo, o desenvolvimento se iniciou a partir da tela inicial (Figura 3a), que é exibida ao usuário após o *login*. O fluxo original direcionaria o usuário para uma tela para cadastro de ordem de serviço (Figura 3b), e posteriormente para tela de identificação da localização onde seria vinculada essa ordem (Figura 3c) e a inspeção seria iniciada. Porém, a fim de diminuir a quantidade de interações do usuário, o fluxo foi otimizado. Essa otimização resultou em alterações no diagrama de casos de uso, como pode ser observado na comparação entre a Figura 2 e Figura 4.

Com o início do desenvolvimento da aplicação, foi constatada a necessidade de reorganizar as funcionalidades identificadas no diagrama de casos de uso da Figura 2 para otimizar o fluxo de telas e o acesso e alteração de dados. Iniciando desenvolvimento da aplicação pelas funcionalidades mais relevantes para o usuário, focando então em "Cadastrar e vincular Ordem de serviço à localização", "Gerenciar Ordens de Serviço Cadastradas (CRUD)", "Exportador de Relatórios". Foi dividido o desenvolvimento da aplicação inicialmente em 4 incrementos, seriam eles: Cadastro Ordem de Serviço, Gerenciador Ordem de Serviço, Exportador de Relatórios e *back-end* da aplicação.

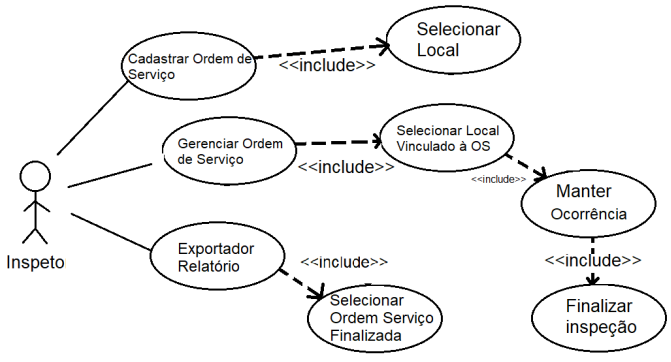


Figura 4. Diagrama de casos de uso atualizado

5.4. Desenvolvimetno do *Back-end* Aplicação

O desenvolvimento do *back-end* da aplicação foi realizado como terceiro incremento, implementado entre as entregas de "Gerenciador Ordem de serviço"(incremento 2) e "Exportador de Relatório"(incremento 4). Será abordado o *back-end* agora visando contextualizar sobre o desenvolvimento desta parte e agrupar os assuntos sobre fluxo de telas.

5.4.1. Modelo Lógico do Banco de Dados

Após reunião com orientador do projeto, foi decida qual seria a estrutura utilizada para persistência dos dados, como pode ser observado no modelo lógico (Figura 5). Buscando aproveitar o conhecimento adquirido durante o curso de Análise e Desenvolvimento de Sistemas, a ferramenta escolhida foi o PostgreSQL, pois foi o sistema gerenciador de banco de dados empregado nas disciplinas cursadas.

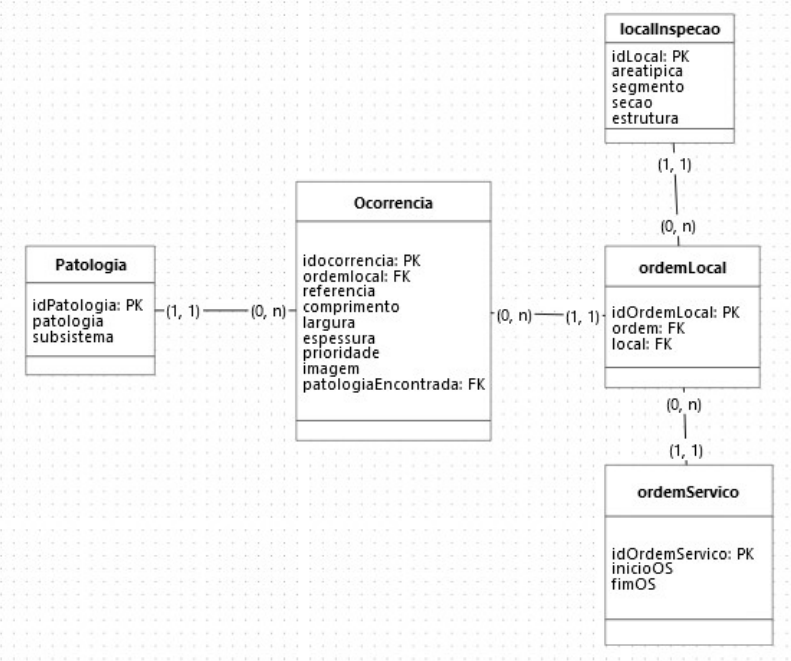


Figura 5. Modelo Lógico Banco de Dados

Para desenvolvimento, o ponto focal foi a tabela ocorrência. Nesta tabela é possível persistir os dados de ocorrências que são encontradas no decorrer da inspeção, contendo todos os dados necessários para o relatório. A estrutura adotada possibilita também trabalhar no aspecto organizacional do funcionamento da aplicação, trazendo robustez ao projeto. Desta forma, por exemplo, ao final de todas as inspeções que foram realizadas em determinada ordem de serviço, é possível buscar a relação existente entre ordem de serviço e o local que está sendo realizada a inspeção na tabela ordemlocal, e de posse desta informação, selecionar dentre as ocorrências quais compõem aquela inspeção.

5.4.2. Gerenciamento de Dados: Conexão, Manipulação e Persistência

Buscando o aproveitamento do conhecimento da linguagem JavaScript, foi selecionado o Node.JS para o desenvolvimento da parte de *back-end* da aplicação. Para conexão, manipulação e persistência dos dados, foi utilizada uma biblioteca específica do Node.JS chamada "Knex", que é responsável por fazer interações do Node com banco de dados SQL de maneira programática. O código empregando a biblioteca Knex é mostrado na Figura 6.

```
1  /**
2   * @param { import("knex").Knex } knex
3   * @returns { Promise<void> }
4   */
5  exports.up = function(knex) {
6    return knex.schema.createTable('ocorrencia', table => {
7      table.increments('idocorrencia').primary()
8      table.integer('ordemlocal').notNullable().references('idordemlocal').inTable('ordemlocal')
9      table.string('referencia')
10     table.integer('patologiaencontrada').notNullable().references('idpatologia').inTable('patologia')
11     table.float('comprimento').notNullable()
12     table.float('largura').notNullable()
13     table.float('espessura').notNullable()
14     table.string('prioridade').notNullable()
15     table.text('imagem').notNullable()
16
17   })
18 };
19
20 /**
21  * @param { import("knex").Knex } knex
22  * @returns { Promise<void> }
23  */
24 exports.down = function(knex) {
25   return knex.schema.dropTable('ocorrencia')
26 };
27
```

Figura 6. Exemplo da criação de tabelas de maneira programática utilizando a biblioteca "Knex"

A linha 5 da Figura 6 utiliza o *exports.up* para mapear qual é a ação de criação do banco de dados, e precisa obrigatoriamente ter uma *export.down* para mostrar como é desfeita essa alteração, como pode ser obvervado na linha 24. dentro da função anônima da linha 5, temos o retorno do knex utilizando o método para criação de tabela, chamado *createTable* que recebe como parâmetro o nome da tabela a ser criada e também uma função que contém a estrutura dessa tabela.

Na linha 7 da Figura 6, é possível observar o objeto *table* sendo manipulado e um método chamado *increments* sendo acessado recebendo o nome da coluna como parâmetro. Além disso, pode ser observado a chamado também do método *primary()* que garante que essa coluna seja caracterizada como chave primária da tabela.

Na linha 8 da Figura 6, é possível visualizar pela primeira vez como criar uma coluna do tipo inteiro utilizando o método *integer* com características de ser não nula, e chave estrangeira referenciando uma outra tabela. Isso ocorre através dos métodos *notNullable*, *references* que recebe como parâmetro o nome da coluna que é referenciada, e o *inTable* que recebe como parâmetro qual é a tabela que contém a coluna referenciada.

Pode ser visto na Figura 6, linha 9 e 14 a criação da uma coluna do tipo varchar, utilizando o método *string*. A diferença entre elas é que na linha 9, não é necessário cria-lá como *notNullable*, pois a regra de negócio permite que o inspetor não insira uma referência. A linha 10 cria uma coluna similar à da linha 8, porém chamada "patologia-encontrada".

As linhas 11, 12 e 13 da Figura 6 são responsáveis por armazenar valores decimais, por isso são do tipo *float*. A criação da coluna responsável pelo armazanamento da imagem convertida em *base64*, é criada através da linha 15, utilizando o método *text*, pois

é um campo que armazena sequências de muitos caracteres.

Para que o knex tenha controle de como retroceder no estado da aplicação após a criação da tabela, dentro do *exports.down* é passado como parâmetro o nome da tabela criada para o método *dropTable*, para que quando necessário, seja excluída essa tabela.

Explorando um pouco mais o assunto e as possibilidades do knex, foi empregado também o conceito de *migrations*, para que as alterações do banco de dados sejam feitas de maneira ordenada e consistente. O conceito passa por um sistema de versionamento do banco de dados, trazendo uma abordagem sistemática que armazena toda e qualquer alteração, sendo versionada e persistida em colunas diferentes do restante do banco de dados, proporcionando a rastreabilidade de todas as alterações.

Cada *migration* criada passa por uma estrutura de *export.up* e *export.down*. O objetivo dessa separação é estrutura-lá de maneira a saber exatamente o que deve ser feito para criar (*up*) e também para excluir (*down*) (Figura 6), possibilitando o versionamento e restaurações de estado anteriores do banco de dados.

5.5. Desenvolvimento *Front-End* da Aplicação

Como pode ser observado no diagrama de casos de uso da Figura 4, o fluxo das telas passa então a seguir o diagrama de casos de uso atualizado e dividido em três incrementos principais, como mostrado na Figura 7. São eles:

a) Cadastrar de Ordem de Serviço, e Selecionar Local (vínculo de Ordem de serviço com local da inspeção);

b) Gerenciar Ordem de Serviço, Selecionar Local Vinculada à OS (localizações daquela ordem de serviço), e Manter Ocorrências;

c) Exportar Relatório, e Selecionar Ordem de Serviço Finalizadas.

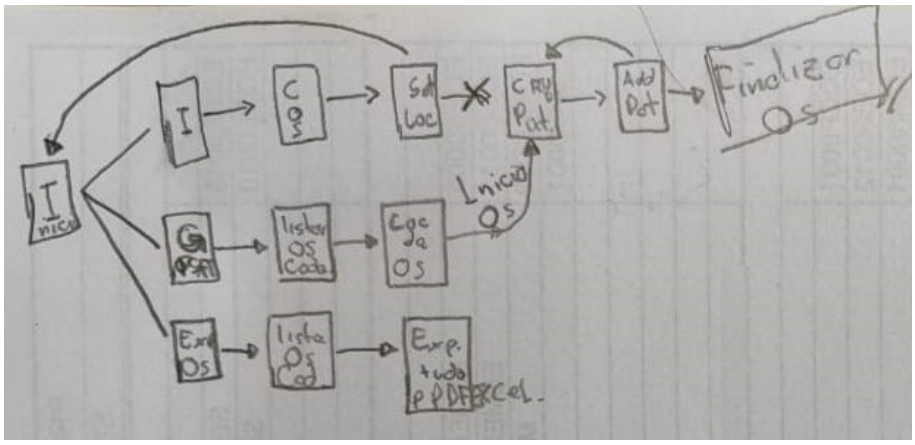


Figura 7. Esboço do Diagrama de Fluxo de Telas

Além destes incrementos, existe ainda o incremento responsável pelo *back-end* da aplicação, que por ser responsável por disponibilizar as *API's* necessárias para aplicação pode ser desenvolvido em paralelo ao *front-end*. Este incremento foi o terceiro a ser desenvolvido, pois surgiram demandas específicas de manipulação de dados principalmente entre os componentes da aplicação.

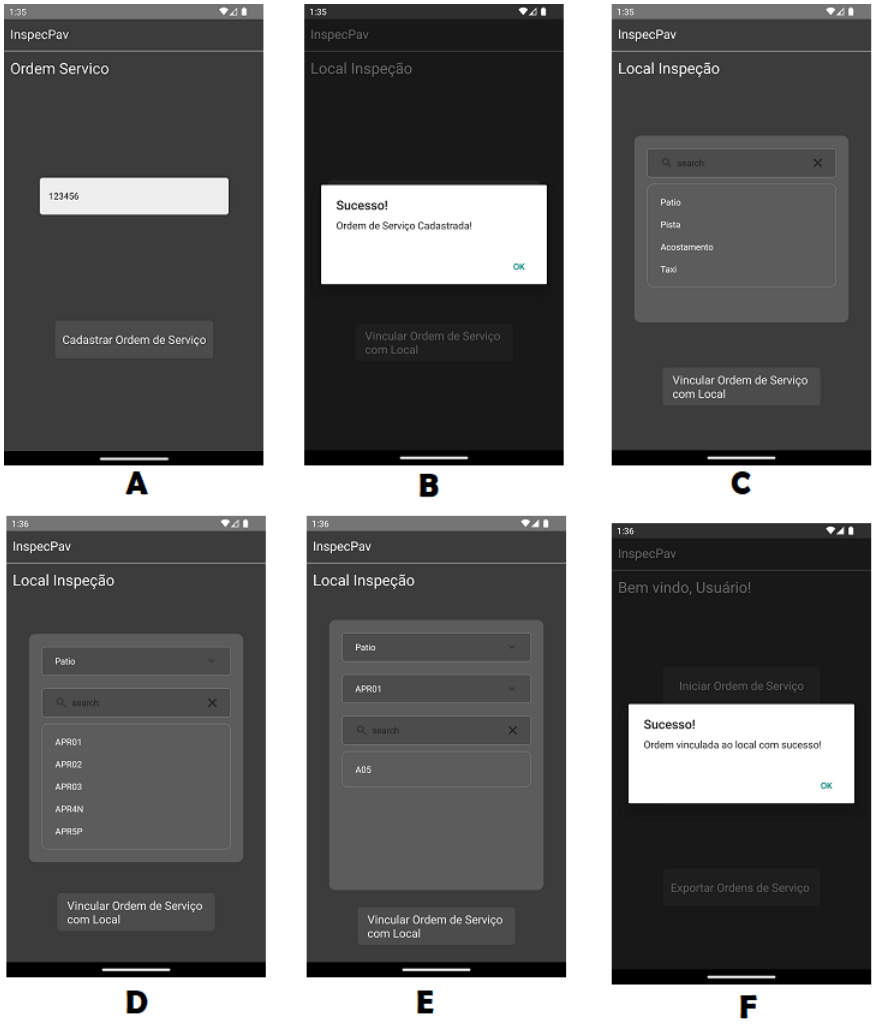


Figura 8. Capturas de Tela Fluxo de Telas A - Cadastrar Ordem de serviço e Vincular à Localização

5.5.1. Fluxo de Telas A

O Fluxo A é composto por duas telas principais: Tela para cadastro ordem de serviço (Figura 8a) e tela para vínculo de ordem de serviço ao local necessário (Figura 8c). Na tela de cadastro de ordem de serviço, a comunicação é feita com o *back-end* através do *framework* "Axios", responsável por lidar com requisições HTTP. É inserida então no banco de dados, na tabela "ordemservico"(Figura 5) o valor inserido pelo usuário e dois campos com valor nulo, data e hora do inicio e fim da inspeção. Na tela de vínculo da ordem de serviço ao local da inspeção, é então apresentado ao usuário três campos selecionáveis que são preenchidos de forma dinâmica de acordo com a seleção do usuário (Figura 8e).

Partindo de um agrupamento macro para o específico. O primeiro *combo box* que é apresentado ao usuário é a de área típica, separada em "Acostamento, Pátio, Pista e Taxí"(Figura 8e). Ao selecionar um valor, é realizada a filtragem dos dados que irão alimentar o segundo *combo box* com os valores de segmento. Ao selecionar segmento

desejado, é então alimentado o último *combo box* com os valores específicos de cada localização que pode receber uma inspeção.

Acostamento é constituído por um único segmento e 20 seções. Pátio é constituído por seis segmentos e 24 seções. Pista é constituído por um único segmento e 15 seções. Taxí é constituído por 25 segmentos e 54 seções. Ao final da seleção por parte o usuário, através da *API*, é possível realizar a consulta na tabela "localinspecao"(Figura 5) em busca da combinação dos dados coletados com o valor previamente armazenado e selecionar o identificador da correspondência.

De posse do identificador da localização exata, é realizada então a inserção dos valores na tabela "ordemlocalizacao"(Figura 5), traçando vínculo entre local da inspeção e a ordem de serviço, finalizando o fluxo de telas A (Figura 8f).

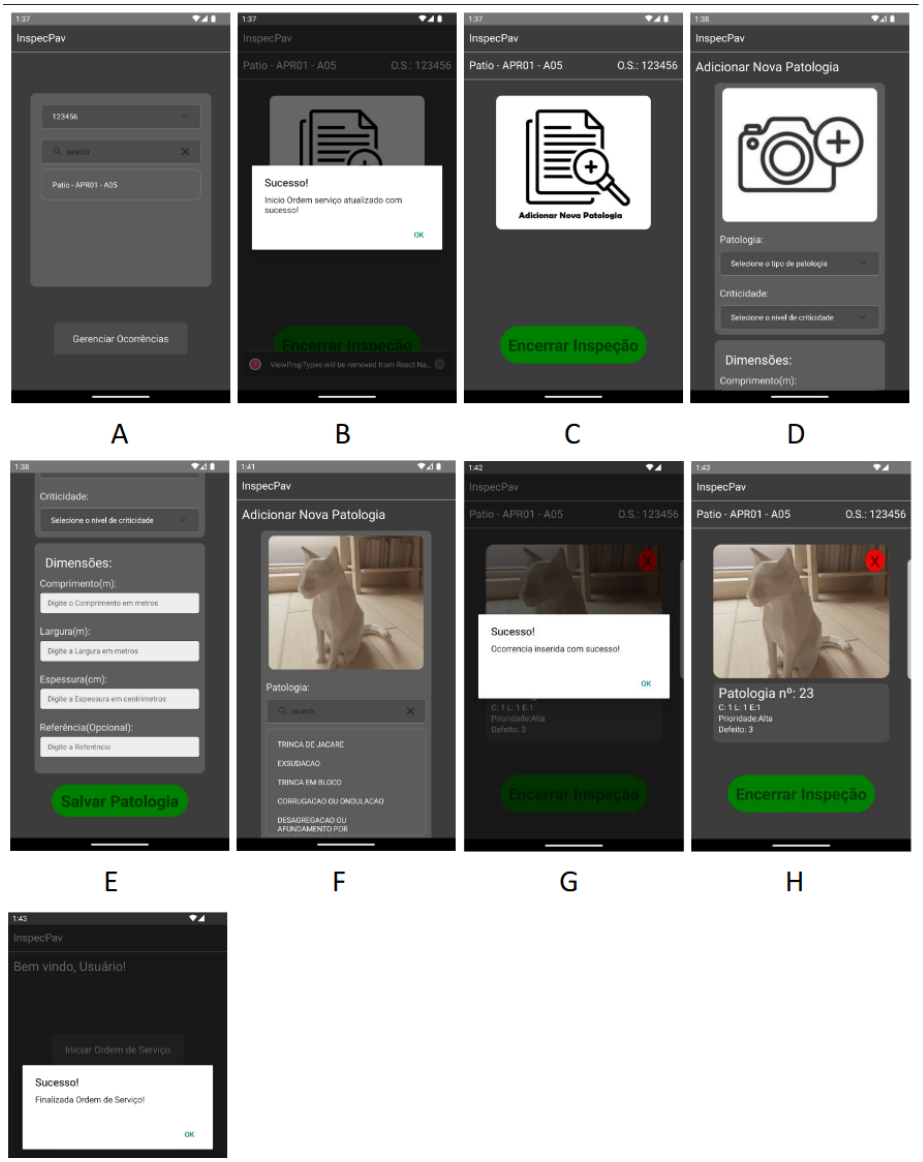


Figura 9. Telas do fluxo B - Gerenciador de Ordem de Serviço e Localização e Gerenciador de Patologias

5.5.2. Fluxo de Telas B

O fluxo B (Figura 9) é definido pelo gerenciador de ordem de serviço e localizações daquela ordem de serviço, bem como o gerenciador de ocorrências.

Ao acessar a tela de Gerenciador de ordem de serviço (Figura 9a), o usuário é direcionado à uma tela com um único *combo box*, esse que por sua vez é alimentado com dados que são coletados através da *API* seguindo a lógica de selecionar no banco de dados apenas os valores de ordem de serviço que são compostos por início e fim da ordem iguais a nulo. Esse processamento tem por objetivo trazer para como opção para seleção do usuário apenas as ordens de serviço que estão vinculadas à localização, mas ainda não foram iniciadas inspeções.

Finalizada a seleção de qual é a ordem de serviço que o usuário pretende trabalhar, é feita requisição para *API* que, com base na ordem de serviço selecionada, buscar na tabela "ordemlocal"quais são as localizações vinculadas à aquela determinada ordem de serviço. É então renderizado na tela um segundo *combo box*, esse que é alimentado com dados de localizações que estão vinculados à ordem de serviço retornados da *API*.

Ao selecionar então qual é a localização que deseja começar a realizar a inspeção, usuário é redirecionado para tela de gerenciamento de ocorrências (Figura 9b), e em *background* é realizada outra requisição para a *API*, porém dessa vez a requisição tem o comportamento de realizar um *update* no estado da ordem de serviço em questão. Essa atualização no estado tem por objetivo atualizar o valor de início da ordem para o valor atual daquele momento, coletado através do método *"moment"* fornecido pelo JavaScript.

Dentro da tela de gerenciamento de ocorrências, o usuário pode ter acesso visual e rápido as informações superiores onde é exibido qual a localização que está atuando e referente a qual ordem de serviço é aquela inspeção (Figura 9c). Abaixo, é exibido um componente do tipo *Caroucel* onde serão exibidas as ocorrências registradas por ele.

É possível inserir novas ocorrências de acordo com a necessidade do usuário, e todas são listadas e exibidas no componente *Caroucel*. Cada item do *Caroucel* possui: uma imagem da patologia registrada que é clicável através do componente *"touchableOpaccity"* que tem por objetivo o comportamento de, ao clicar na imagem, ser redirecionado para tela para poder editar as características daquela ocorrência, como tamanho, referência etc.; um título contendo o número de identificação daquela ocorrência e uma breve descrição com as dimensões daquele defeito.

Caso o usuário queira cadastrar uma nova ocorrência, é redirecionado para tela de Cadastro de Ocorrência (Figura 9d). Ao ser renderizado esse componente, é realizada requisição para alimentar os dados que compõem quais são as possíveis patologias que podem ser identificadas naquela localização. A requisição é feita com base na identificação do local, alimentando os dados de acordo com a estrutura ali presente.

Após essa solicitação, a tela é exibida. Nela, o primeiro elemento é um componente que é responsável por acessar de maneira nativa a câmera do dispositivo. Esse acesso é realizado através de uma biblioteca específica do JavaScript chamada *image-picker*. Após a captura da imagem, é realizada uma conversão para um tipo de dado chamado *base64*, responsável por armazenar a imagem codificada em formato de texto.

O conceito da tela de adicionar ocorrência é montar um objeto e prepará-lo para o armazenamento no banco de dados, ele é composto por: idordemlocal, valor recuperado através da *API* selecionando através da ordem de serviço e local da inspeção; defeito, informação recebida do usuário através do *combo box* alimentado com as possíveis patologias daquele lugar; dimensões, recebida do usuário através de três *inputs* (Figura 9e), largura, comprimento e espessura; referência, única informação recebida de maneira opcional através de *input* e por último a imagem que é armazenada em formato textual utilizando a tecnologia de *base64* (Figura 9f).

Ao final da adição, o usuário pode salvar a patologia (Figura 9g). Assim, é realizada uma requisição do tipo *POST* à *API*. Ao ser inserida no banco de dados, esse objeto de ocorrência recebe de forma automática (*AUTOINCREMENT*) uma chave identificadora.

O usuário é informado que a patologia foi corretamente inserida (Figura 9g) e é redirecionado para tela de gerenciamento de ocorrências esta que, por sua vez, é atualizada com a nova ocorrência cadastrada (Figura 9h).

O usuário pode repetir o processo quantas vezes conforme necessário. Ao finalizar a inspeção, ele pode encerrar aquela ordem de serviço utilizando o botão "Encerrar Inspeção"posicionado ao fim da tela. Ao finalizar a inspeção, uma nova requisição é feita à *API*, com objetivo de atualizar agora o campo "FimOS"da ordem de serviço, visto que a vistoria chegou ao fim (Figura 9i).

5.5.3. Fluxo de Telas C

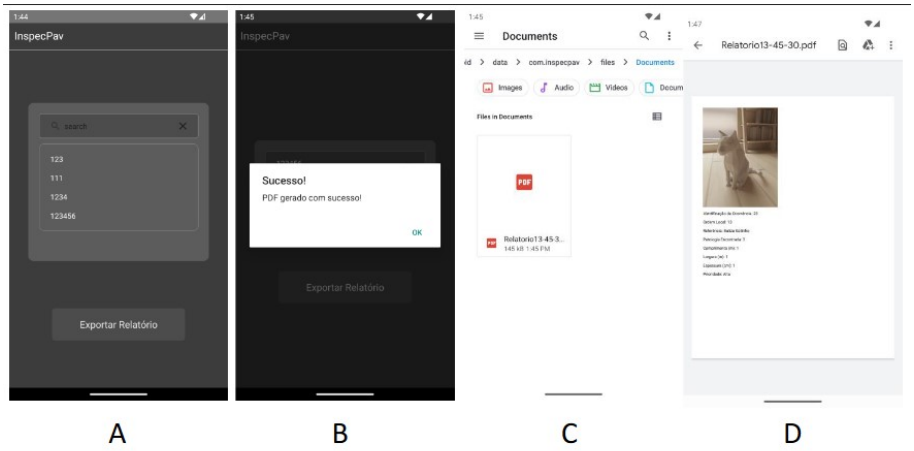


Figura 10. Capturas de Tela Fluxo de Telas C - Exportador de Ordens de Serviço e Listagem Ordens de Serviço Finalizadas

Quanto ao terceiro fluxo de telas, C (Figura 10), possui Exportador de Ordens de Serviço. Ao ser renderizado o componente na tela do usuário é realizada um requisição à *API* similar ao processo de vínculo de ordem de serviço com localização, porém, neste momento, a seleção é feita quando "InicioOS"e "FimOS"possuem valores diferentes de nulo, ou seja, são ordens de serviço que já passaram pelo processo de inspeção e possuem

valores de início e fim definidos, para que possam ser tratados no futuro e contabilizado qual foi o tempo desprendido naquela atividade.

Esses dados retornados, possuem a listagem de quais são as ordens de serviço finalizadas e prontas para serem exportadas. Com base nesses dados é renderizado o *combo box* na tela com as ordens (Figura 10a).

Ao selecionar a ordem de serviço que deseja exportar, o usuário dispara mais uma requisição, esta que é responsável por: realizar a consulta na tabela "ordemlocal"com base na ordem selecionada; recuperar todos os "ID's"de ordemlocal que estão vinculados, realizar a consulta na tabela "ocorrencia"buscando por todas as correspondencias que possuem a chave estrangeira de "idordemlocal".

5.6. Preparação para Exportação de Dados

```
73  const buildHTML = (results: any[]) => {
74    return `
75      <html>
76      <head>
77
78      </head>
79      <body>
80        ${results.map((result, index) => `
81          ${result.ocorrencias.map((ocorrencia:any) => `
82            <div>
83              <img src="${ocorrencia.imagem}" alt="Imagem" style="width: 300px; height: auto;"/>
84              <p>Identificação da Ocorrência: ${ocorrencia.idocorrencia}</p>
85              <p>Ordem Local: ${ocorrencia.ordemlocal}</p>
86              <p>Referência: ${ocorrencia.referencia}</p>
87              <p>Patologia Encontrada: ${ocorrencia.patologiaencontrada}</p>
88              <p>Comprimento (m): ${ocorrencia.comprimento}</p>
89              <p>Largura (m): ${ocorrencia.largura}</p>
90              <p>Espessura (cm): ${ocorrencia.espessura}</p>
91              <p>Prioridade: ${ocorrencia.prioridade}</p>
92            </div><br/>
93          `).join('')}
94        `).join('')}
95      </body>
96    </html>
97      `;
98  };
```

Figura 11. Trecho código responsável pela estruturação HTML para exportar para PDF

Com essa massa de dados, é o momento de realizar a organização desses dados dentro de uma estrutura de HTML, que será renderizado respeitando os parâmetros necessários para exibição de cada um dos atributos de cada objeto que foi recuperado (Figura 11).

Pode ser observado na Figura 11 a estrutura necessária para funcionalidade de exportar para PDF.

Dentro da constante declarada como *BuildHTML* linha 73 à 96, é então definida a estrutura HTML do documento. Para realizar o mapeamento das localizações vinculadas à ordem de serviço, é então utilizado o método *.map*(linha 80) no conjunto de dados chamado *"result"* fazendo com que percorra todos os componentes daquele conjunto de dados. Para que seja listada todas as ocorrências que estão armazenadas dentro das localizações, é utilizado novamente o método.*map* (linha 81) para que seja percorrida e exibida cada uma das ocorrências listadas. A exibição dos valores é feito através do acesso ao objeto utilizando "ocorrencias."e o nome do campo que deseja ser exibido, como pode ser

visto nas linhas 83 à 91. Após a renderização desse HTML, é possível então exportar esse documento para PDF utilizando a biblioteca de conversão chamada "RNHTMLtoPDF", ferramenta utilizada em projetos *React Native* para gerar arquivos PDF a partir de conteúdo HTML.

Ao final, o relatório é exportado (Figura 10b). O diretório definido por padrão é: android/data/com.inspecpav/files/documents (Figura 10c). A Figura 10d exemplifica como fica estruturado o PDF após a conversão.

6. Conclusão

O propósito deste artigo foi apresentar o desenvolvimento de um protótipo para auxiliar no processo de inspeção em pavimentos de sítio aeroportuário de Viracopos. Para alcançar esse objetivo, foi desenvolvido protótipo funcional considerando os casos de uso da Figura 4, contemplando todos os fluxos de: Cadastrar Ordem de Serviço, Gerenciar Ordem de Serviço e Exportar Relatório.

Uma das principais dificuldades enfrentadas durante o desenvolvimento deste projeto foi a inesperada descontinuação de uma biblioteca essencial que estava sendo utilizada. Este obstáculo se tornou evidente quando a biblioteca começou a exibir um *pop-up* na aplicação, recomendando a migração para uma alternativa, devido à falta de suporte futuro. Essa situação exigiu uma reavaliação rápida das decisões de design e uma investigação cuidadosa de soluções alternativas.

Embora desafiador, esse episódio destacou a importância de monitorar ativamente o ecossistema de desenvolvimento e estar preparado para tomar medidas rápidas e eficazes diante de mudanças inesperadas ou descontinuações de ferramentas essenciais. Através dessa experiência, foi possível aprender valiosas lições sobre a importância da flexibilidade e adaptabilidade em projetos de desenvolvimento de software.

Como oportunidades de melhorias para trabalhos futuros, é essencial que o inspetor seja identificado, e que essa informação seja vinculada às ocorrências identificadas. Sabendo disso, pode ser implementado um sistema de autenticação de usuário, onde será necessário realizar o *login* na aplicação, trazendo robustez aos lançamentos de patologias e também segurança ao aplicativo.

Visando evitar que inspeções sejam realizadas em locais incorretos, é possível implementar uma verificação usando dados do GPS do aparelho. Quanto ao formato de exportação do documentos gerados, seria útil aos responsáveis pelo cronograma de manutenção, ter também o acesso aos dados através de planilha eletrônica para manipulação.

Fazem parte do desenvolvimento deste artigo os conhecimentos adquiridos e articulados de disciplinas como Algoritmos e Programação, Banco de Dados, Análise Orientada a Objetos, Engenharia de *Software*, Arquitetura de *Software*, Estruturas de Dados e Desenvolvimento para Dispositivos Móveis.

Referências

ANAC (2012). *Aeródromos - Operação, Manutenção e Resposta À Emergência*. ANAC, 6th edition.

Beder, D. M. (2012). *Engenharia Web: uma abordagem sistemática para o desenvolvimento de aplicações web*. EdUFSCar, 1th edition.

Brasil (2005). Lei nº 11.182, de 27 de setembro de 2005. cria a agência nacional de aviação civil - anac, e dá outras providências. *Diário Oficial da República Federativa do Brasil*. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2004-2006/2005/Lei/L11182.htm . Acessado em: 12/04/2023.

Canva (2023). Sobre o Canva. Disponível em: https://www.canva.com/pt_br/about/ . Acessado em: 27/11/2023.

EngenhariaCivil.com. *Sistema Inovador de Inspeção e Gestão de Estradas Desenvolvido em Portugal*. Engenharia Civil na Internet. Disponível em: https://www.engenhariacivil.com/inpav-sistema-inovador-inspecao-gestao-estradas . Acesso em: 12/04/2023.

INFRAERO (2017). *Manual de Sistema de Gerenciamento de Pavimentos Aeroportuários – SGPA*. INFRAERO, 1th edition. Disponível em: https://www.gov.br/anac/pt-br/centrais-de-conteudo/aeroportos-e-aerodromos/manuais-e-cartilhas/manual-para-sgpa-v3.pdf/view . Acessado em: 12/04/2023.

PRESSMAN, R. (2011). *Engenharia de software: uma abordagem profissional*. Porto Alegre: McGraw-Hill, 7th edition.

Documento Digitalizado Público

TCC - Anexo I

Assunto: TCC - Anexo I
Assinado por: Andre Constantino
Tipo do Documento: Relatório
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Andre Constantino da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/03/2024 15:30:38.

Este documento foi armazenado no SUAP em 10/03/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1605985
Código de Autenticação: e6609e1344