INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DE PERNAMBUCO
Campus Recife
Curso Superior Tecnológico em Análise e Desenvolvimento de Sistemas

LOURIVALDO JOSÉ FLAVIO COUTINHO VASCONCELOS

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA FISCALIZAÇÃO DO USO DOS RECURSOS HÍDRICOS NO ESTADO DE PERNAMBUCO (SIGFIS)

Recife
2021

LOURIVALDO JOSÉ FLAVIO COUTINHO VASCONCELOS

ljfcv@a.recife.ifpe.edu.br

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA FISCALIZAÇÃO DO USO DOS RECURSOS HÍDRICOS NO ESTADO DE PERNAMBUCO (SIGFIS)

Projeto desenvolvido na Disciplina de Trabalho de Conclusão de Curso II como requisito parcial para a obtenção do Título Superior de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Marco Antonio de Oliveira Domingues

Recife

2021

FICHA CATALOGRÁFICA

V331d Vasconcelos, Lourivaldo José Flávio Coutinho.

Desenvolvimento de um aplicativo móvel para fiscalização do uso dos recursos hídricos no estado de Pernambuco (SIGFIS) / Lourivaldo José Flávio Coutinho Vasconcelos; Orientador Prof. Dr. Marco Antonio de Oliveira Domingues - Recife, 2021.

76f.; il.

Trabalho de Conclusão de Curso (Tecnológico em Análise e Desenvolvimento de Sistemas) – IFPE – campus Recife.

. Inclui Referências.

1. Aplicativo móvel 2. Desenvolvimento de recursos hídricos. 3. Android (recurso eletrônico) 4. Análise de sistemas. I. Vasconcelos, Lourivaldo José Flávio Coutinho. II. IFPE. III. Título.

CDD 004.21

Catalogação na fonte elaborada pelo Bibliotecário Thiago Leite Amaro da Silva - CRB4/ 2.021

Trabalho de Conclusão de Curso apresentado por **LOURIVALDO JOSÉ FLAVIO COUTINHO VASCONCELOS** à coordenação de Análise e Desenvolvimento de Sistemas, do Instituto Federal de Pernambuco, sob o título de "**DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA FISCALIZAÇÃO DO USO DOS RECURSOS HÍDRICOS NO ESTADO DE PERNAMBUCO (SIGFIS)**", orientado pelo Prof. **Marco Antonio de Oliveira Domingues** e aprovada pela banca examinadora formada pelos professores:

Recife, 13 de maio de 2021

_____

Prof. Ph.D. Marco Antônio de Oliveira Domingues

CSIN/DASE/IFPE

_____

Prof. M.Sc. Anderson Luiz Souza Moreira

CSIN/DASE/IFPE

_____

Prof. M. Sc. Hélvio Alessandro de Lima Ferreira

APAC

AGRADECIMENTOS

Em primeiro lugar, a Deus, que fez com que meus objetivos fossem alcançados, durante todos os meus anos de estudos.

Aos meus pais que sempre estiveram do meu lado, a minha irmã Maria Gislainy Flávia e a minha esposa Alesandra Isla pela força, compreensão, companheirismo e cumplicidade.

A equipe de fiscais e informática da Agência Pernambucana de Águas e Clima pelas orientações.

Ao Instituto Federal de Educação Ciência e Tecnologia de Pernambuco (IFPE) por toda sua estrutura disponibilizada essencial no meu processo de formação profissional, pela dedicação, e por tudo o que aprendi ao longo dos anos do curso.

Ao professor Dr. Marco Antônio, por ter sido meu orientador e ter desempenhado tal função com dedicação e amizade.

E a todos que participaram, direta ou indiretamente do desenvolvimento deste trabalho, enriquecendo o meu processo de aprendizado.

“*O sucesso é a soma de pequenos esforços repetidos dia após dia.*”

Robert Collier

RESUMO

A gestão dos recursos hídricos vem passando por transformações tornando urgente o desenvolvimento de ferramentas de apoio à gestão e fiscalização, com caráter educativo e regulador. Este trabalho tem como objetivo apresentar o desenvolvimento de um aplicativo que auxilie na atividade de fiscalização dos recursos hídricos utilizando sistemas de informações geográficas. Intitulado SIGFIS, utilizando o sistema operacional Android com linguagem de programação Kotlin, com dados em formatos georreferenciados e planos de informações, o aplicativo visa auxiliar os agentes fiscais em suas visitas a campo, acurácia nas autuações e agilidade nos serviços, visto que reúne informações relevantes e imprescindíveis ao seu trabalho.

Palavras-chave: Aplicativo móvel. Recurso hídrico. Android.

ABSTRACT

The management of water resources has been undergoing transformations, making it urgent to develop tools to support management and inspection, with an educational and regulatory character. This work aims to present the development of an application that helps in the inspection of water resources using geographic information systems. Entitled SIGFIS, using the Android operating system with Kotlin programming language, with data in georeferenced formats and information plans, the application aims to assist tax agents in their field visits, accuracy in assessments and agility in services, since it gathers relevant information and essential to your work.

Keywords: Mobile application. Water resource. Android.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

ANA - Agência Nacional de Águas e Saneamento Básico
APAC - Agência Pernambucana de Águas e Clima
API - Application Programming Interface
AWS - Amazon Web Services
CBHs - Comitês de Bacias Hidrográficas
CGIS - Canada Geographic Information System
CNRH - Conselho Nacional de Recursos Hídricos
CONSU - Conselhos Gestores de Reservatórios
DNPM - Departamento Nacional de Pesquisa Mineral
FGV - Fundação Getúlio Vargas
FGVcia - Fundação Getúlio Vargas Centro de Tecnologia de Informação
GIS - Geographic Information System
GPS - Sistema de Posicionamento Global
IDE - Integrated Development Environment
JSON - JavaScript Object Notation
ONU - Organização das Nações Unidas
PERH - Política Estadual de Recursos Hídricos
PNRH - Política Nacional de Recursos Hídricos
PWA - Progressive Web Apps
SBL - Serviços Baseados em Localização
SDK - Software Development Kit
SEINFRA - Secretaria de Infraestrutura e Recursos Hídricos
SIG - Sistema de Informações Geográficas
SIGHPE - Sistema de Geoinformação Hidrometeorológico de Pernambuco
SIGRH - Sistema Integrado de Gestão de Recursos Hídricos
SINGREH - Sistema Nacional de Gerenciamento de Recursos Hídricos
SIRH - Sistema Integrado de Recursos Hídricos
SRHE - Secretaria de Recursos Hídricos e Energéticos
UNESCO - Organização das Nações Unidas para Educação, Ciência e Cultura
XML - Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Atualmente a temática ambiental vem sendo palco para diversos debates mundiais. Devido, principalmente, aos impactos causados pela industrialização e crescimento populacional a redução dos recursos naturais é um tema de preocupação global. Neste sentido a água é o bem essencial para a vida e para o processo de desenvolvimento das nações. Portanto, emerge a necessidade de promover um desenvolvimento sustentável aliado a uma gestão com ferramentas capazes de contribuir com procedimentos de fiscalização, controle e uso consciente.

Ao longo do tempo a gestão dos recursos hídricos vem passando por constantes transformações que vão desde disputas em relação ao uso até modelos de administração da infraestrutura hídrica. Conforme afirma Santos (2013, p. 4), “é necessário fazer uma gestão mais intensa dos recursos hídricos a fim de garantir a disponibilidade de água em seus diferentes tipos de uso”.

Uma das ferramentas de gestão promovidas pelas políticas de recursos hídricos é a fiscalização do uso, com caráter educativo e regulador. Entretanto tais instrumentos necessitam de aperfeiçoamento, Miranda et al. (2021). Um dos fortes aliados da gestão pública neste segmento é o sistema de informações geográficas.

Desta forma, este trabalho tem como objetivo apresentar o desenvolvimento de um aplicativo voltado para um sistema de informações geográficas que auxilie na atividade de fiscalização do uso dos recursos hídricos, intitulado como SIGFIS[1].

Com esta intenção, será adotada uma metodologia com etapas voltadas para a coleta e criação de um banco de dados, seguidos por uma padronização e compatibilização destes dados. Aliado a isto, a revisão da legislação auxilia na geração de mapas temáticos com programa de geoprocessamento.

Como resultado espera-se que o SIGFIS possa contribuir com a obtenção de dados com maior precisão cartográfica, com melhorias nos procedimentos de fiscalização, facilitando os trabalhos de campo dos agentes afim de contribuir de forma efetiva com as tomadas de decisão, promovendo maior segurança jurídica aos processos de qualquer agência ou órgãos reguladores das águas.

[1] A prática de trabalho executada pela Apac – Agência Pernambucana de Águas e Clima foi utilizada modelo para o SIGFIS, podendo ser replicada para outros órgãos executores da política de recursos hídricos de acordo com suas especificidades.

2 TRABALHOS RELACIONADOS

Durante a pesquisa para realização deste trabalho destacaram-se alguns trabalhos relacionados às práticas de gestão dos recursos naturais, que se utilizaram de sistemas de informação para alcançar seus resultados, no entanto, nenhum deles faz uso de mapas em um aplicativo com intuito de auxiliar as vistorias em campo.

Alves et al. (2018a) no artigo “Um método para gerenciamento do processo de fiscalização dos recursos hídricos” propuseram um aplicativo para dispositivos móveis com foco no cadastro das informações, capaz de auxiliar na gestão dos recursos hídricos. Assim como no artigo “O uso de tecnologia da informação na fiscalização e denúncias do uso de recursos hídricos” dos mesmos autores que teve a mesma intenção.

Almeida et al. (2019) na pesquisa sobre o “Uso de Business Intelligence na Gestão de Recursos Hídricos: o caso da Fiscalização do Uso da Água” desenvolveram um dashboard para dispositivos moveis, com uma arquitetura semelhante ao presente trabalho, com enfoque para métricas e gráficos capazes de monitorar a inspeção dos recursos hídricos, onde o agente pode emitir um auto de constatação ou um termo de compromisso no momento da abordagem em campo.

Oliveira e Zeilhofer (2017) no trabalho “Sistema de Suporte à Decisão baseado em Lógica Fuzzy para Outorga de Recursos Hídricos Superficiais” abordaram o uso em lógica fuzzy para o auxílio na tomada de decisão na gestão de recursos hídricos com base em dados já coletados previamente em campo.

3 FUNDAMENTAÇÃO TEÓRICA

Em função da importância da água e de uma gestão integrada e participativa entre as nações, estados e municípios emerge a necessidade de ferramentas que promovam uma evolução na gestão dos recursos hídricos. Em alguns estados brasileiros como Pernambuco pode-se considerar que sua implantação ainda esteja em seus estágios iniciais. Desta forma, um sistema ou um aplicativo móvel capaz de fornecer informações geográficas seria um grande aliado para o desenvolvimento da gestão dos recursos hídricos.

3.1 Gestão e políticas dos recursos hídricos

Segundo dados da ONU a escassez de água atinge milhões de pessoas mundialmente, além da quantidade a qualidade da água também é um fato preocupante, tanto que o número de pessoas que não tem acesso à água salubre passa de bilhões (UNESCO, 2009).

Na América Latina, que contém 30% da água doce do mundo, o panorama não é diferente. Por exemplo, 40% da população tem acesso a apenas 10% da disposição hídrica. Um dos pontos mais críticos é a bacia do Plata, que supre, aproximadamente, 50% da população da Argentina e vizinhos como Bolívia, Brasil, Paraguai e Uruguai, resultado de uma gestão hídrica deficiente (Naidoo; Davidson-Harden, 2015).

No Brasil, a importância com os recursos hídricos ainda é falha, mas está ganhando força como destaca Freitas (2008, p. 30) “o Brasil, nos últimos anos, vem tomando consciência do problema. Afinal, um povo que possui os maiores rios do mundo tem dificuldade em imaginar que pode ficar sem água.”

O debate sobre a gestão dos recursos hídricos tem ganhado destaque nos setores público e privado nos últimos anos. Problemas com escassez na região Nordeste nos últimos 06 anos, eventos no Sudeste nos anos de 2014/2015 são fortes indicadores de como fatores climáticos somados a má gestão podem potencializar um problema tão recorrente no país. Embora que a problemática se inicie na escassez de chuvas, a falta de planejamento sobre os recursos afeta a produtividade das empresas, racionamento e rodízio de água para a população e altas na conta de energia.

Os primeiros passos em relação a gestão hídrica no Brasil iniciou-se 1933 com a criação da Diretoria de Águas ou Serviço de Águas, no Ministério da Agricultura. Em 1934, ficou a cargo do Departamento Nacional de Pesquisa Mineral (DNPM). Nessa época, foi editado o Código de Águas (BRASIL, 1934), o qual permanece em vigor atualmente, estabelecido pelo Decreto nº 24.643, de 10 de julho de 1934 (BORSOI e TORRES, 1997). Já em 1930 a gestão dos recursos hídricos no Brasil entra em um modelo burocrático com o objetivo de cumprir e fazer cumprir os dispositivos legais sobre águas, com extensa legislação a ser obedecida.

Mas foi com a Constituição Federal de 1988, onde extinguiu o domínio privado e estabeleceu que todos os corpos de água, a partir de outubro de 1988, passariam a ser de domínio público, que os recursos hídricos passaram a ser reconhecidos dentro da estrutura global do meio ambiente. (BRASIL, 1988) Foi criada a Lei nº 9.433 em janeiro de 1997, também chamada como lei das águas, que estabelece a Política Nacional de Recursos Hídricos (PNRH) e prevê como instrumento de controle e gestão das águas a cobrança pelo seu uso (BRASIL, 1997).

Os fundamentos da Política Nacional dos Recursos Hídricos estão no artigo 1º da Lei n. 9.433/97:

> I – reconhecer a água como bem econômico e dar ao usuário uma indicação de seu real valor;
> II – incentivar a racionalização do uso da água;
> III – obter recursos financeiros para o financiamento dos programas e intervenções previstos nos planos de recursos hídricos. (BRASIL 1997, art. 1º)

Com a intenção de desenvolver mecanismos capazes de oferecer suporte administrativo e jurídico à gestão dos recursos, a Lei Federal nº 9.433/97 instituiu o Sistema Nacional de Gerenciamento de Recursos Hídricos (SINGREH) que tem como objetivo coordenar o gerenciamento e implantação da PNRH visando preservar e recuperar os recursos hídricos. O sistema é composto por órgãos reguladores: o Conselho Nacional de Recursos Hídricos (CNRH); os Conselhos de Recursos Hídricos dos Estados e do Distrito Federal, os órgãos dos poderes públicos federal, estaduais, do Distrito Federal e municipais cujas competências se relacionem com a gestão de recursos hídricos; os Comitês de Bacias Hidrográficas (CBHs);

Criada com a Lei nº 9.984 de 17 de julho de 2000 a partir da reformulação do Decreto nº 3.692 de 19 de dezembro de 2000 a Agência Nacional de Águas e Saneamento Básico (ANA), é uma autarquia vinculada ao Ministério do

Desenvolvimento Regional com competência para promover Política Nacional de Recursos Hídricos. Além de responsável pela execução da Política Nacional de Recursos Hídricos, a ANA disciplina o uso dos recursos hídricos no Brasil. (BRASIL, 2000)

De acordo com Tucci (2001, p. 90), a Lei nº 9.433/97 e suas disposições foi resultado de vários estudos e acompanhamento de experiências na gestão de recursos hídricos com a intenção de promover melhorias em todo país, conforme destaca:

> o sistema criado se sobrepõe, mas não se opõe, à estrutura administrativa existente. A Lei mantém as competências dos organismos existentes e potencializa sua atuação. Cria somente os organismos necessários à execução das novas atividades, as quais, por terem base territorial diversa da divisão político-administrativa do país, não poderiam ser exercidas pelos organismos existentes, que têm base municipal, estadual ou federal. As Agências de Água têm como área de atuação uma ou mais bacias hidrográficas e suas competências primordiais são o planejamento dos recursos hídricos da bacia e a cobrança pelo uso da água.

A Política Nacional de Recursos Hídricos estabelece também instrumentos institucionais como os Planos de Recursos Hídricos desenvolvidos por bacias hidrográficas, prevê sobre direitos e deveres no uso da água com ligação com o Sistema de Informação de Recursos Hídricos interligando estados e municípios com as ações e políticas públicas referentes à água (SIRVINSKAS, 2015).

Como afirma Souza Júnior (2004, p. 152), o uso de um modelo democrático de “gestão de recursos hídricos, no aspecto restrito à participação social, representa um avanço, conquanto outros países desenvolvidos possuem estruturas bastante centralizadas de gestão”.

Em Pernambuco com a Lei Estadual n° 11.426 17 de janeiro de 1997 o estado editou sua Política Estadual de Recursos Hídricos (PERH) revogada e substituída pela Lei n° 12.984 de 30 de dezembro de 2005 e teve como finalidade: o planejamento do uso dos recursos hídricos, garantindo a sua qualidade, disponibilidade, conservação e aproveitamento de forma racional, em benefício das gerações atuais e futuras, ensejando o desenvolvimento sustentável. (PERNAMBUCO, 2005).

Na gestão de recursos hídricos em Pernambuco atuam também a Secretaria de Infraestrutura e Recursos Hídricos (Seinfra), criada pela Lei nº 16.520, de 27 de dezembro de 2018, com a finalidade de implementar a Política Estadual de Recursos Hídricos e de Saneamento, com a ajuda do Conselho Estadual de Recursos Hídricos

(CRH) dos Comitês Estaduais de Bacias Hidrográficas e de Conselhos Gestores de Reservatórios (Consu), conforme demonstrado na figura 1. (PERNAMBUCO, 2005).

Figura 1 – Unidades estaduais de gestão de recursos hídricos

Fonte: AGÊNCIA NACIONAL DE ÁGUAS, 2021.

Adaptado de: https://progestao.ana.gov.br/portal/progestao/panorama-dos-estados/pe

Atua ainda na gestão de recursos hídricos a Agência Pernambucana de Águas e Clima (Apac), criada pela Lei nº 14.028, de 26 de março de 2010, com a função de executar a Política Estadual de Recursos Hídricos, regular os usos múltiplos da água em âmbito estadual, realizar monitoramento hidrometeorológico e previsões de tempo e clima no estado, bem como operar e alimentar o Sistema Integrado de Gestão de Recursos Hídricos (SIGRH) (PERNAMBUCO, 2010)

Através do Decreto Estadual n° 38.752 de 22 de outubro de 2012 a fiscalização do uso dos recursos hídricos ficou a cargo dos agentes fiscais da Apac, onde ficaram como responsáveis pelo acompanhamento e verificação de ocorrências de infrações às normas referentes aos recursos hídricos, a emissão de relatórios das fiscalizações realizadas e o ato de lavrar instrumentos de fiscalização (PERNAMBUCO, 2012).

Entretanto para que o acompanhamento e fiscalização sejam eficazes, os órgãos regulamentadores necessitam de organização e ferramentas que forneçam base de dados para consultas às informações e condutas para fiscalização e monitoramentos de forma prática e atualizada, para garantir o bom cumprimento e

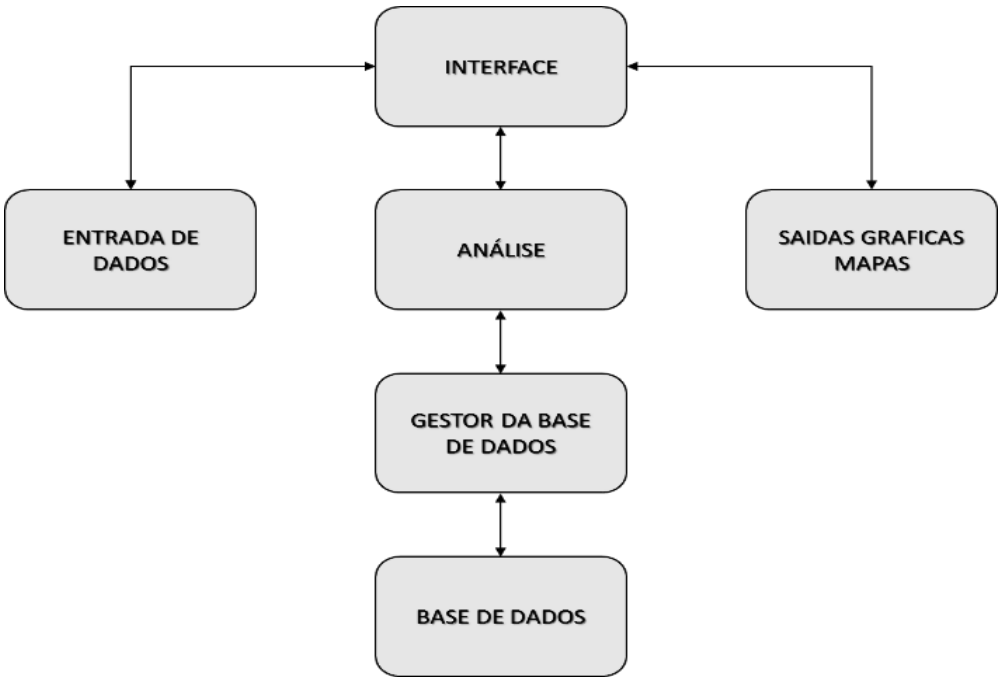
acertadas tomadas de decisão. Porém os instrumentos disponíveis capazes de promover esta melhoria em sua maioria não estão regulamentados e/ou implantados.

3.2 Sistema de Informações Geográficas

Ao longo dos anos o interesse pela informação geográfica tem sido uma constante. Segundo Cavalcante (2015) esse interesse é justificado pelo fato de ampliar e dar suporte a áreas tão distintas como as geociências, economia e gestão, sociologia e saúde, engenharias, planejamento e monitoramento espacial, entre outras.

Utilizada em diversas áreas como engenharia, economia, saúde, geociências e monitoramento espacial, seu uso permite ao usuário a correlação de variáveis distintas proporcionando análise, simulação e diversos cenários. Para tal, foram criadas ferramentas específicas de geoprocessamento chamadas de Sistema de Informações Geográficas (SIG), conhecido também por sua sigla em inglês GIS - *Geographic Information System*, dotado de funcionalidades como: cartografia digital, GPS, sensoriamento remoto, aerofotogrametria, processamento digital de imagens, entre outros. Grande parte das aplicações SIG apresentam uma estrutura geral com interface de comunicação com usuário, uma base de dados e sua gestão, e funcionalidades para entrada e edição de dados, sua análise, produção e impressão dos mapas (Figura 2).

Figura 2 – Estrutura geral de uma aplicação SIG



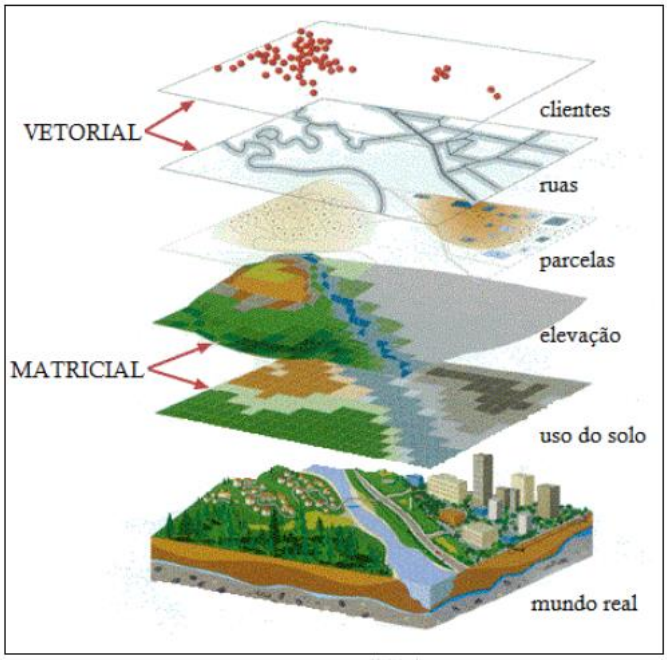
Fonte: elaborado pelo autor, 2021.
Adaptado de: http://www.faef.revista.inf.br/imagens_arquivos/

Segundo Silva (2003, p. 12), geoprocessamento representa qualquer tipo de processamento de dados georreferenciados. Segundo Melo (2006, p. 47) um SIG pode ser considerado como:

> A combinação de hardware, software, dados, metodologias e recursos humanos envolvidos que operam de forma coerente para analisar e produzir novas informações geográficas. Parte dos recursos humanos é formada pelos usuários do SIG; em geral, são especialistas que coletam, manuseiam, armazenam, recuperam, examinam e geram novas informações georreferenciadas em um ambiente computacional para solucionar problemas de planejamento e gerenciamento espacial.

Em um SIG os dados são representados graficamente de três formas: por área, pontos e linhas. Cada objeto carrega consigo a informação da localização espacial daquele dado, a forma como ele está representado e seu tamanho (RIGAUX; SCHOOL; VOISARD, 2002). Os formatos podem ser através de linhas, pontos e polígonos, classificando-se como vetorial, ou através de uma estrutura de grade de células de tamanho fixo, pixels, classificada de matricial(*raster*), conforme representado na figura 3. De modo geral cada dado está relacionado a uma tabela de atributos onde ficam armazenadas as informações relacionadas ao objeto.

Figura 3 – Representação de planos de informação de um SIG



Fonte: TERC, 2017.

3.2.1 Histórico

A representação geográfica existe há vários séculos, porém, a história do SIG é mais recente. Segundo Meneses (2003, p. 51) a evolução do SIG pode ser dividida em três fases: a manipulação e visualização de banco de dados, operações analíticas de dados não gráficos e estrutura organizacionais e análise espacial. O primeiro SIG é datado da década de 60 no Canadá e registrado como CGIS - *Canada Geographic Information System*. Teve como objetivo o mapeamento das áreas de recursos naturais existentes.

Entretanto existia muita dificuldade na utilização. Os monitores não possuíam a resolução necessária, os computadores eram extremamente caros e a capacidade de armazenamento e a velocidade de processamento eram muito baixas e poucos eram os especialistas capazes de manusear, o que naquele momento dificultou o desenvolvimento para outras áreas do mercado.

O avanço da tecnologia e o aumento de novos e mais acessíveis recursos de hardware facilitaram o desenvolvimento do SIG. No Brasil o início do geoprocessamento se deu na década de 80, principalmente devido a tecnologia de sistemas de informação geográfica entrar em um estágio de crescimento acelerado causada pelos avanços da microinformática e do estabelecimento de centros de estudos sobre o assunto (ASSAD; SANO, 1998). Em 2000, a indústria de SIG tinha ultrapassado a barreira dos U$7 bilhões (LONGLEY et al., 2011).

Segundo Câmara e Medeiros (2003, p. 415) esse novo paradigma é motivado pelo “aguçar da nossa percepção relativa aos problemas ecológicos, urbanos e ambientais, em entender, de forma cada vez mais detalhada, processos de mudança local e global e pela necessidade de compartilhar dados entre instituições e com a sociedade”.

3.2.2 Principais aplicações

De acordo com Câmara e Medeiros (1998, p. 48) as características dos dados geográficos permitem sua utilização como ferramentas para produção de mapas, suporte para análise espacial de fenômenos, ou como um banco de dados geográficos, com funções de armazenamento e recuperação de informação espacial. Nesse contexto e levando em consideração a dimensão física e a localização

espacial o SIG é capaz de, através da relação de um Sistema de coordenadas e seus atributos, estabelecer relações topológicas existentes para armazenar e visualizar a topologia de um mapa considerando vizinhança, proximidade e pertinência entre objetos geográficos.

Conforme apontado por Câmara e Ortiz (1998, p. 66) as funções de um SIG se concentram em fornecer informações sobre determinada condição, localização, tendência, rota, padrão ou modelo, além daquelas de processamento gráfico e de imagens (quadro 1).

Quadro 1 - Funções de processamento gráfico e de imagens de um SIG

Funções	Descrição
Análise Geográfica	"Álgebra de mapas" (reclassificação, intersecção, operações boleanas e matemáticas entre mapas, e consulta ao banco de dados).
Processamento de imagens	Realce por modificação de histograma, filtragem espacial, classificação estatística por máxima verossimilhança, rotação espectral (componentes principais), transformação IHS-RGB, e registro.
Modelagem de terreno	Determinação do modelo (grade regular ou triangular) a partir de pontos esparsos ou linhas, geração de mapas de contorno (isolinhas), geração de mapas de declividade e de aspecto, visualização 3D (com imagens e temas), cálculo de volumes, e análise de perfis.
Geodésia e fotogrametria	Realização, por software, de procedimentos de restituição e ortoretificação digital antes executados por equipamentos analógicos. Fundamental para uso em aplicações de cartografia automatizada e atualização de mapeamentos.
Modelagem de redes	Cálculo do caminho ótimo e crítico.
Produção cartográfica	Permite, na área de plotagem, colocar legendas, textos explicativos e notas de crédito. Possuir uma biblioteca de símbolos é também atributo fundamental.

Fonte: CÂMARA e ORTIZ, 1998.

Pode-se destacar a forma de visualização em um SIG que pode ser realizada em forma de mapas, de gráficos ou de relatórios, estando os dois últimos associados, na maioria das vezes, a mapas.

De acordo com Graça (2009, p. 22) as aplicações de SIG têm apresentado excelentes resultados na gestão dos recursos naturais em especial: gestão de florestas; análise de habitats naturais e planejamento de vias de migração; preservação de rios; gestão de recursos para lazer; gestão de aquíferos; gestão de cheias; preservação de áreas úmidas; gestão de terras agrícolas; modelação de

aquíferos e dispersão de poluentes; análises de impacto ambiental; e análise de visibilidade.

No estado de Pernambuco a Secretaria de Infraestrutura e Recursos Hídricos e a Apac utilizam alguns sistemas como complementares à gestão são eles: o SIRH (Sistema Integrado de Recursos Hídricos) e o SIGHPE Sistema de Geoinformação Hidrometeorológico de Pernambuco.

3.3 Dispositivos Móveis

Os dispositivos móveis incluem qualquer tecnologia portátil e conectada, como telefones celulares (*smartphones*), leitores de livros digitais como (*e-readers*), *tablets*, consoles manuais de videogames e aparelhos portáteis de áudio. Segundo a UNESCO (2014), os mais utilizados pelos educadores e estudantes sem dúvidas são os *smartphones* e *tablets* e isso se deve pela facilidade de acessar informações e compartilhá-las.

Em relação a história dos dispositivos móveis, Cirilo (2020) declara que:

> Em 2007 surgia o primeiro smartphone realmente inteligente, segundo Taboada (2015), e que ele criou uma ruptura no padrão de acesso móvel à Internet; afinal, em 2007, era disponibilizado o primeiro iPhone da Apple. A maneira como as pessoas se relacionariam na Internet após seu surgimento e o padrão de aparelhos para se conectar, navegar e trocar mensagens sofreu uma guinada de 180 graus, que nunca mais parou de ser aprimorada.

Os dispositivos móveis estão cada vez mais ganhando espaço e popularidade, visto que estão presentes no cotidiano das pessoas, podendo ser utilizados tanto para atividades pessoais como profissionais. De acordo com levantamento feito pela 31ª Pesquisa Anual do FGVcia em junho de 2020, o Brasil já conta com mais de um smartphone por habitante, ao todo são 234 milhões de dispositivos ativos no país. Ao somar-se a quantidade de notebooks e tablets, são 342 milhões de dispositivos portáteis, ou seja, 1,6 dispositivo portátil por habitante (FGV, 2020).

Segundo a consultoria Gartner (2019) “Entre as empresas que desenvolveram e implantaram pelo menos três tipos diferentes de aplicativos, os mais comuns são aplicativos móveis (91%)”. Diante disso, percebe-se que é devido a popularidade dos dispositivos móveis, pois a comunicação se tornou mais acessível já que pode ser acessada de qualquer lugar, além de outras necessidades.

Os dispositivos móveis estão em constante evolução, se tornaram capazes de combinar funções de voz, texto, internet, aplicativos, pesquisa, redes sociais e “serviços pervasivos baseados em localização” segundo Dery, Kolb e Macornick (2014, p. 559). De acordo com Rosa (2015) os dispositivos móveis atuais contam com os mais diversos tipos de sensores, como GPS, acelerômetro, microfone, câmera e giroscópio, tais sensores possibilitam a criação dos mais diversos tipos de SBLs (Serviços Baseados em Localização). A aplicação descrita neste trabalho faz uso de alguns destes sensores como GPS, microfone e câmera.

De forma complementar percebe-se que alguns serviços móveis podem ser acessados a qualquer hora e em qualquer lugar; permitem a disseminação de informações e serviços a um grande número de pessoas de forma rápida; e, ampliam o canal de comunicação entre Estado e Sociedade principalmente, para a reportar os problemas para os cidadãos (AGUIAR, 2010).

3.3.1 Aplicativos para dispositivos móveis

O rápido aumento no número de smartphones com recursos multimídias cada vez mais avançados, possibilitou aos usuários a utilização nas mais diversas tarefas. Grande parte das funcionalidades presentes nos aparelhos são possíveis graças aos aplicativos, chamados também de apps, softwares voltados especificamente para dispositivos móveis.

Conforme destaca Lima (2017, p. 23):

> Por meio desses aplicativos, o telefone se transforma em um grande pacote de ferramentas que permitem, por exemplo, acessar redes sociais, conteúdos educacionais, entretenimento, jogos eletrônicos, edição de fotos, localização geográfica, acesso a bancos e outros serviços. Dessa forma, os aplicativos customizam os dispositivos de acordo com interesses e necessidades dos usuários.

Conforme afirma DÂMASO (Techtudo, 2020), aplicativo “É um programa de software presente em dispositivos móveis, como celulares e tablets, ou no computador e em smart TVs. Eles podem ser executados offline ou online, além de apresentarem versões pagas ou gratuitas, obtidas em lojas de aplicativos”.

O crescimento da aquisição de smartphones em cada vez mais parcelas da sociedade, atrelado ao aumento de usuários com a acesso à internet tornou o desenvolvimento de aplicações móveis uma grande oportunidade de negócio.

Também vem crescendo a cada dia a quantidade de aplicativos para diversas finalidades como organização de finanças, transporte, entretenimento, fitness, viagens, estudos entre outros.

Segundo destaca PICOLI (2021) é essencial o desenvolvimento de aplicativos móveis capazes de otimizar tarefas, facilitar a comunicação entre departamentos, digitalizar documentos, automatizar processos, proporcionar segurança às transações, dentre outras funcionalidades.

3.3.2 Desenvolvimento mobile

O avanço tecnológico dos dispositivos móveis elevou as funcionalidades dos aparelhos celulares há muito além de chamadas telefônicas ou mensagens de texto. Estes recursos foram permitidos graças ao desenvolvimento constante nos sistemas operacionais para criação de aplicativos melhores e com cada vez mais recursos e serviços para o usuário.

Para DA SILVA (2014) “O sistema operacional é responsável por gerenciar diversos recursos do aparelho celular, as linguagens de programação são utilizadas na programação do aplicativo e o IDE, também conhecido como ambiente de desenvolvimento integrado, fornece ferramentas que auxiliam na criação do aplicativo”.

No mercado, há diversas plataformas para aparelhos celulares, como: Android (Google), IOS (Apple Inc), Windows Phone (Microsoft), etc. E para realização do desenvolvimento dos aplicativos, existem três formas distintas que é o desenvolvimento de aplicativos nativos, Progressive Web App e o desenvolvimento de aplicativos híbridos, cuja decisão de qual aplicativo utilizar irá depender dos recursos que pretende utilizar e o que pretende alcançar. O aplicativo que será apresentado como resultado deste trabalho foi desenvolvido no formato nativo, pois fornece melhor acesso aos recursos de hardware, melhor desempenho e integração facilitada com a ferramenta de visualização de mapas utilizada, além de existir possibilidade de utilizar o aparelho sem conexão à internet.

3.3.3 Aplicações Nativas

Segundo Prezotto e Boniati (2014) citado por MATOS (2017) o desenvolvimento nativo:

> [...] é aquele no qual um aplicativo é projetado e construído especificamente para uma plataforma. Todas as funcionalidades da plataforma estão disponíveis sem restrição e existem padrões de interface gráfica e experiência de usuário específicos, que ajudam o usuário a entender como aquele aplicativo funciona, já que todos os outros aplicativos daquela plataforma seguem os mesmos padrões.

REIS (2019) afirma que “O desenvolvimento de aplicações móveis nativas teve início juntamente com a popularização dos smartphones na segunda metade da década passada e se transformando numa tendência para década seguinte e até o presente momento”.

Para SILVA (2019) “[...] este estilo de programação é possível alcançar um ótimo grau de usabilidade, tendo todos os recursos que o dispositivo oferece para utilização em aplicações”. Observa-se que os aplicativos nativos são desenvolvidos para utilização em uma plataforma específica, como por exemplo o IOS que usa o Swift ou Android que utiliza o Kotlin, vale ressaltar que podem ser utilizadas diferentes linguagens de programação em uma mesma plataforma. Assim, o aplicativo nativo conseguirá acessar todo o potencial do dispositivo móvel mediante a própria arquitetura do sistema operacional, como câmera, calendário, lista de contatos, álbum de fotos, GPS, etc.

De acordo com GARBADE (2018) pode-se perceber algumas vantagens na escolha do desenvolvimento nativo, que são:

Quadro 2 – Vantagens desenvolvimento nativo

Alta performance	uma vez que o código nativo tem acesso direto ao sistema operacional e às funcionalidades do host, por exemplo a câmera do dispositivo móvel, o desempenho geral do aplicativo melhora. Especialmente ao renderizar gráficos ou conteúdo multimídia. Com isso, reduz os riscos de tempo de inatividade devido a travamentos ou congelamentos.
Melhor interface do usuário	a interface é agradável aos usuários, devido os aplicativos nativos serem integrados perfeitamente ao sistema operacional do dispositivo móvel. Dessa forma, os usuários que estão familiarizados com a interface do sistema, reconhecerão um novo aplicativo como algo comum e positivo.
Melhor posicionamento nas lojas de aplicativos	se um aplicativo tiver uma percepção alta e positiva da usabilidade que o usuário teve do aplicativo, consequentemente este aparelho ficará melhor posicionado nas lojas de aplicativos e assim obterá uma maior visibilidade.

Fonte: desenvolvido pelo autor (2021)

Em relação as desvantagens de um aplicativo nativo segundo SAMBASIVAN et al (2011) citado por Silva (2014) “A principal desvantagem de um aplicativo nativo está no fato de ser executado apenas na plataforma para a qual foi desenvolvido, aumentando o tempo, custo e o esforço para disponibilizar um mesmo aplicativo para mais de uma plataforma”. Em vista disso, caso uma empresa escolha desenvolver uma quantidade de aplicativos para mais de um sistema operacional usando o desenvolvimento nativo, gastará mais tempo e recursos financeiros, pois o desenvolvimento se tornará mais caro, porque exige mais soluções, testes, recursos, em consequência disso, a empresa terá que contratar programadores especializados nos diferentes sistemas operacionais, uma vez que para desenvolver uma aplicação nativa, cada dispositivo tem seu próprio processo de desenvolvimento exclusivo. Levando isto em consideração, para o aplicativo desenvolvido neste trabalho não se fez necessário dar suporte em múltiplas plataformas, uma vez que é voltado para um público restrito.

Portanto, como o desenvolvimento nativo é feito de forma personalizada para cada plataforma, torna-se uma ferramenta mais rápida e confiável, além de oferecer uma experiência melhor ao usuário, pois permite a construção de designers mais sofisticados e interface mais adaptada à plataforma, além de existir possibilidade de utilizar o aparelho sem conexão à internet.

3.3.4 Aplicações PWA

Progressive Web App – web app (PWA) é uma página web que possui interface e algumas funcionalidades que se assemelham a uma aplicação móvel, porém não é necessário instalar do aplicativo Play Store, pois o seu acesso é através do navegador, como o Chrome.

De acordo com Toonen (2020) o PWA foi desenvolvido a partir de tecnologias da web como HTML, CSS e JavaScript, porém com funcionalidades diferentes do aplicativo nativo.

LePage (2020) afirma que “Os *Progressive Web Apps* (PWA) são desenvolvidos e aprimorados com APIs modernas para fornecer recursos aprimorados, confiabilidade e capacidade de instalação, ao mesmo tempo que

alcançam qualquer pessoa, em qualquer lugar, em qualquer dispositivo com uma única base de código".

Na tabela abaixo são ilustradas as principais vantagens e desvantagens conforme o blog GeekHunter (2020):

Quadro 3 – Vantagens e desvantagens - Aplicações PWA

VANTAGENS	DESVANTAGENS
Utilização independente do navegador ou dispositivo; Funcionam offline graças ao service worker; Permitem o envio de push notifications; Permite adicionar um ícone na tela inicial; Atualização automática; Experiência similar a um aplicativo nativo.	Dificuldade ao requisitar: NFC; Bluetooth; Suporte cross-browser (cada navegador é um universo diferente; Ausência em lojas de aplicativos dos sistemas operacionais.

Fonte: Geekhunter (2020)

Portanto, a proposta do *Progressive Web App* é trazer acessibilidade, ou seja, fazer com que o conteúdo web seja disponível em todos os dispositivos e assim trazer uma melhor experiência para o usuário.

3.3.5 Aplicações Híbridas

O termo *Cross-Platform Development* (Desenvolvimento Multi Plataformas), também conhecido como Desenvolvimento Híbrido ou Desenvolvimento de Plataforma Cruzada, de acordo com Asper Brothers (2020) "[..] consiste em construir um único aplicativo que pode ser executado em vários sistemas operacionais, em vez de desenvolver diferentes versões de aplicativos para cada plataforma". Isto é, com a sua capacidade de funcionar em diversas plataformas móveis, permitirá que a empresa economize custos e diminua o tempo de desenvolvimento.

De acordo com SILVA (2017) o desenvolvimento híbrido "[...] é um aplicativo implementado com linguagens de programação para web e que são executados similarmente a aplicativos móveis nativos, assim como web". Assim, o aplicativo híbrido é construído na linguagem HTML5, CSS e JavaScript, de modo que o código seja integrado para as funcionalidades do dispositivo móvel. E sobre o seu desenvolvimento, SILVA (2019) afirma que:

> "[...]traz a combinação de aplicações WEB e nativas, embutindo um renderizador HTML dentro de um container de aplicativos nativos. Este padrão permite que as aplicações tenham seu desenvolvimento realizado

> apenas uma vez, distribuindo-o para múltiplos aplicativos de forma genérica, trazendo benefícios como tempo reduzido para desenvolvimento cross platform e compatibilidade com praticamente todos os dispositivos que suportem navegadores WEB no mercado, mas perdendo em performance e possíveis incompatibilidades com componentes de aplicativos nativos"

Em relação ao desenvolvimento, segundo VENTEU (2018) "[...] é iniciado com uma única programação, essa programação é testada e compilada de forma única, e atribuí simultaneamente às plataformas, caso o desenvolvedor queira acrescentar e alterar algum detalhe ou requisito do aplicativo, apenas modifica-se todos os códigos fonte, de uma só vez, não precisando alterar todos, um a um. [...] A manutenção também se torna mais barata uma vez que a mão de obra é mais genérica e fácil de ser encontrada no mercado". De acordo com SuccessiveTech (2019) pode-se citar alguns benefícios do desenvolvimento híbrido, que são:

Quadro 4 - Benefícios do desenvolvimento híbrido

Redução de custos e tempo de desenvolvimento	ao invés de construir vários aplicativos para cada plataforma, o que levaria muito tempo, há a possibilidade de desenvolver um único aplicativo que possa funcionar em todas as plataformas. Isso reduz o tempo de desenvolvimento e o custo;
Manutenção mais fácil com custos reduzidos	é possível sincronizar atualizações em todas as plataformas, economizando tempo e dinheiro. Além de também atualizar os aplicativos automaticamente desde que tenha internet. Significa que os usuários poderão ter a versão mais recente do aplicativo o tempo todo.
Capacidade de atingir efetivamente o público-alvo	permite que com apenas um único aplicativo funcione de forma harmoniosa e eficiente em todas plataformas.
Código reutilizável	o código pode ser reutilizável, assim evita desenvolver códigos novos e exclusivos para cada plataforma. O desenvolvedor pode usar um único código, repetidamente, economizando tempo e recursos.
Integração de nuvem simples	os aplicativos podem ser facilmente integrados à nuvem para serviços de hospedagem. Isso significa que o aplicativo é mais funcional e escalável, pois a única fonte de código pode ser coordenada com diferentes extensões e plug-ins

Fonte: SuccessiveTech (2019)

Segundo Venteu (2018) “O desenvolvimento híbrido é uma ótima opção para situações onde não há necessidade de alta performance do aplicativo, pois não funcionam tão rápido quanto um aplicativo nativo”. Percebe-se que há diversas vantagens na utilização do desenvolvimento híbrido, no entanto possui algumas desvantagens, de acordo com Colussi (2020) são:

Quadro 5 – Desvantagens desenvolvimento híbrido

Baixa Performance	ao desenvolver um aplicativo híbrido, se tem camadas intermediárias entre o código e a renderização do aplicativo em si, removendo um pouco de performance;
Falta de integração com alguns recursos do sistema:	se utiliza plugins desenvolvidos pela comunidade para conseguir acesso aos recursos nativos do sistema. Isso pode significar falta de suporte a novos recursos, dificuldade em atualizações ou mesmo a falta completa de algum recurso;
UX potencialmente ruim	a UX (User Experience – experiência do usuário) pode sofrer um pouco em plataformas híbridas. Porém, : conforme a tecnologia avança e com maiores recursos de responsividade disponíveis, esse é um item que não chega a atrapalhar tanto nos dias atuais.

Fonte: Venteu (2018)

Ambos desenvolvimentos tanto o híbrido como o nativo, possuem defeitos e qualidades, que devem ser analisados antes de escolher o desenvolvimento padrão para a aplicação. De acordo com Silva (2019) “No lado nativo, é possível visualizar os benefícios de interações avançados com a UI e melhor performance dos apps, mas com a desvantagem da disponibilidade em única plataforma.”

Já em relação ao lado híbrido Silva (2019) afirma que “[...] as múltiplas plataformas são o melhor benefício evidente, podendo ter como efeito colateral uma pior performance e perda de algumas capacidades nativas dos dispositivos”. Dessa forma, o desenvolvimento híbrido é um meio termo entre a aplicação nativa e o Progressive Web App.

Uma vez explicado o formato escolhido de desenvolvimento nativo, e suas vantagens em relação aos demais tipos, se faz necessário definir a plataforma que

será utilizada, que no caso deste trabalho é o sistema operacional Android, que será abordado no próximo capítulo.

3.3.6 Android

Android é uma das plataformas mais utilizadas no mundo, sendo um sistema operacional para dispositivos como aparelhos celulares, tabletes dentre outros, o mesmo possui um amplo número de tarefas como por exemplo disponibilizar uma API afim de o usuário consiga tirar fotos ou reproduzir músicas, além de possibilitar instalar programas no dispositivo.

Para Nocera (2018) “Apesar de o sistema Android ser propriedade da google, a mesma foi desenvolvida pela empresa Android Inc e comprada pela Google em julho de 2005”.

De acordo com Maximiliano (2020) “O Android surgiu em 2003, na cidade de Palo Alto na Califórnia e foi desenvolvido por Andy Rubin, Rich Miner, Nick Sears e Chris White, empresários já iniciados no ramo da tecnologia, que fundaram a Android Inc.” De acordo com o site oficial do Android (2021) “O principal objetivo do Android é criar uma plataforma de software aberta disponível para operadoras, OEMs (*Original Equipament Manufacturer*) e desenvolvedores para transformar as ideias inovadoras deles em realidade [...]”.

De fato, a plataforma Android é completa para tecnologia móvel, visto que envolve um pacote com programas para celulares, já com um sistema operacional, middleware, aplicativos e interface do usuário. Conforme Lecheta (2010), o Android foi criado com objetivo de atender a necessidade dos usuários que pediam celulares com cada vez mais “[...] recursos como câmeras, músicas, *bluetooth*, ótima interface visual, jogos, GPS, acesso à internet e e-mails, e agora ainda temos a Tv digital”.

Conforme Nogueira (2012) “O Android foi criado baseado no Kernel do Linux, e utiliza uma máquina virtual personalizada a Dalvik VM que foi criada para aperfeiçoar os recursos de memória e hardware em um ambiente móvel”. Segundo Borghezan (2018) a plataforma do Android “[...] não necessita de um hardware específico de celular para rodar seu sistema, apenas que o aparelho consiga rodar alguma versão do Android”. Dessa forma, é uma plataforma flexível, pois permite que o seu software funcione em uma grande diversidade de modelos de dispositivos móveis.

Garcia (2013) ratifica as principais características do Android:

- Tem seu código totalmente aberto (open source);
- Roda em diferentes hardwares de diferentes fabricantes;
- Desenvolvido pela Google;
- Os aplicativos são desenvolvidos em Java;
- Oferece uma grande variedade de aplicativos, tanto gratuitos como pagos;
- Permite que qualquer interessado crie e distribua jogos e aplicativos gratuitamente ou não.

O Android possui uma arquitetura flexível que por possuir uma característica de código aberto (open source), qualquer desenvolvedor pode integrar aplicações nativas com sua aplicação desenvolvida, isto é, permite que todos contribuam para melhorar a plataforma, por exemplo o desenvolvedor pode substituir a interface nativa pela sua interface desenvolvida.

Outro aspecto relevante são as bibliotecas do Android que além de estarem em constante evolução, o usuário pode contar com os artefatos históricos das versões anteriores e também com os pacotes mais recentes como a Biblioteca de Suporte denominada de AndroidX, além de que se assemelham às APIS da estrutura do Android. Essas bibliotecas estão acessíveis aos desenvolvedores através do SDK do Android, que a documentação do Developer Android (2020) destaca maneiras de usar as bibliotecas de suporte como:

Quadro 6 - Uso das bibliotecas de suporte

Compatibilidade retroativa para APIs mais recentes	Fornece suporte com versões anteriores para classes e métodos de estrutura mais recentes.
Conveniência e classes auxiliares	Fornecimento de várias classes auxiliares, especialmente para o desenvolvimento da interface com o usuário.
Depuração e utilitários	Há vários recursos que oferecem utilidade além do código que é incorporado ao app.

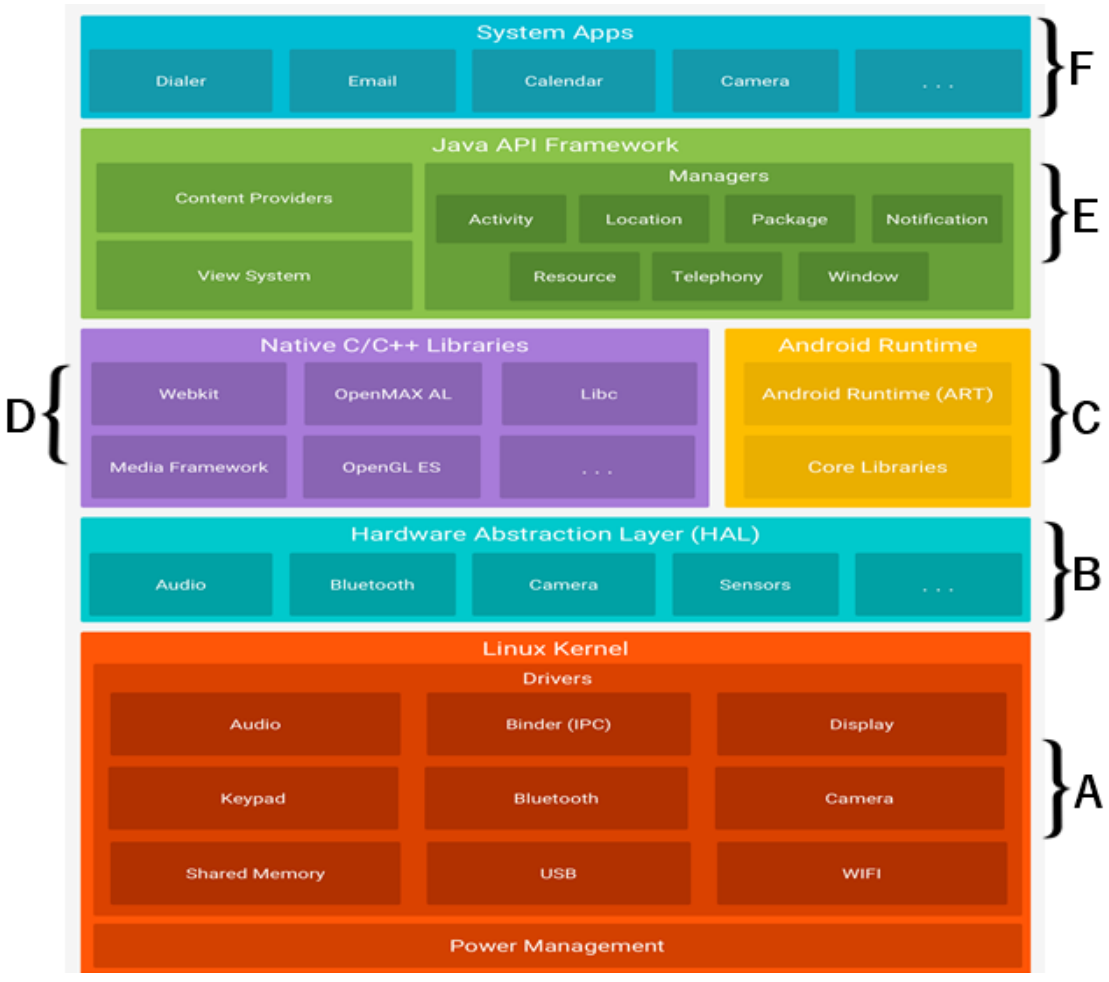
Fonte: Developer Android (2020)

Ainda sobre as bibliotecas, Pimenta (2012) afirma que:

> O Android possui um conjunto de bibliotecas (Runtime), que oferecem a maioria das funcionalidades das principais bibliotecas da linguagem Java. As aplicações Android rodam em seus próprios processos, com sua própria instância da máquina virtual (VM) Dalvik. Esta por sua vez, foi desenvolvida para executar várias máquinas virtuais de forma eficiente, utilizando arquivos.dex, que são otimizados para consumir pouca memória. A VM do Android roda classes compiladas na linguagem Java que são transformadas em arquivos.dex, usando a ferramenta “dx” do SDK.

O Android funciona como uma pilha de softwares baseada no sistema Linux, sendo composto por camadas, sendo elas, o Kernel Linux, Camada de abstração de hardware, Android Runtime, Bibliotecas, estrutura da Java API e aplicativos do sistema, conforme demonstrado na figura 4:

Figura 4 - A pilha de software do Android.



Fonte: Developer (2020)

https://developer.android.com/guide/platform?hl=pt-br

A arquitetura da plataforma Android é composta pelo Linux Kernel versão original, como observado na Figura 4 parte A, sendo a camada mais baixa da

arquitetura, no entanto a partir de 2014 foi utilizada a versão do Kernel Linux 3.4 ou superior, dependendo do modelo do aparelho utilizado. Ela é responsável pelos principais serviços do sistema, como serviços de segurança, gerenciamento de memória e processos, serviços de rede e drivers. De acordo como artigo do Developer Android (2020) “Usar um kernel do Linux permite que o Android aproveite os recursos de segurança principais e que os fabricantes dos dispositivos desenvolvam drivers de hardware para um kernel conhecido”.

A segunda camada denominada de Hardware Abstraction Layer (HAL), conforme observado na figura 4 parte B, possui a função de expor as capacidades do hardware do dispositivo para a estrutura da JAVA API de maior nível. Conforme artigo do Developer Android (2020) “A HAL consiste em módulos de biblioteca, que implementam uma interface para um tipo específico de componente de hardware, como o módulo de câmera ou Bluetooth”. Percebe-se por exemplo que quando o desenvolvedor faz uma solicitação ao hardware para receber um arquivo via *bluetooth*, o sistema Android carrega o módulo da biblioteca para esse hardware, e em seguida o hardware implementa uma interface para este tipo específico de componente.

Para dispositivos com Android versão 5.0 (API nível 21) ou mais recente, cada aplicativo executa o próprio processo com uma instância própria do Android Runtime, que consta na figura 4 Parte C. Segundo o site Developer “O ART é projetado para executar várias máquinas virtuais em dispositivos de baixa memória executando arquivos DEX, um formato de *bytecode* projetado especialmente para Android, otimizado para oferecer consumo mínimo de memória”. Antes do Android versão 5.0, o Dalvik era o tempo de execução do Android.

Além do Android possuir um sistema operacional e máquina virtual, também é formado por um conjunto de bibliotecas C/C++, que através delas é utilizado vários componentes do sistema com objetivo de fornecer áudio, vídeo, funções para navegadores Web/ gráficos, etc. Segundo Pereira (2009) “Bibliotecas de multimídia, visualização de camadas 2D e 3D, funções para navegadores web, funções para gráficos, funções de aceleração de hardware, renderização 3D, fontes bitmap e vetorizadas e funções de acesso ao banco SQLite.” Conforme apresentado na Figura 4 parte D.

Na terceira camada, conforme ilustrado na Figura 4 parte E, é chamada de Java API Framework, isto é, a camada do framework de aplicação é composta por programas que visam gerenciar as funções básicas do dispositivo móvel, além de

possibilitar a formação de blocos de programação afim de simplificar a reutilização de componentes e serviços de sistema modulares. Dentre os principais elementos desta camada destacam-se alguns que de acordo com Pereira e Silva (2009) citado por Correia (2012) são:

Quadro 7 – Elementos da camada JAVA API Framework

Activity manager	É responsável por gerenciar o ciclo de vida de todas as atividades, desde o momento em que uma atividade é iniciada até o seu encerramento.
Package manager	É responsável por estabelecer comunicação com o restante do sistema e indicar quais os pacotes que estão sendo utilizados no dispositivo, bem como a capacidade de cada um desses pacotes.
Windows manager	É responsável por gerenciar as apresentações de janelas, indicando quais estarão ativas e quais não estarão.
Content providers	É responsável por estabelecer a troca de informações entre aplicativos por meio do compartilhamento de dados possibilitando entre os aparelhos.
View system	É responsável por disponibilizar todo o tratamento gráfico para a aplicação sendo eles botões, layouts e frames.

Fonte: Fabrício Correia (2012)

Por último, no nível mais alto, conforme ilustra a figura 4 parte F, encontra-se o System Apps que engloba um conjunto de aplicativos principais para e-mail, envio de SMS, calendários, navegador de internet, contatos, chamadas telefônicas, etc. Que segundo Correia (2012) “[...] que envolvem tanto aplicações originais, desenvolvidas pelo Google, quanto desenvolvidas por terceiros as quais estão instaladas no sistema. [...] é responsável por possibilitar a iteração do usuário comum com as interfaces dos aplicativos”.

Dentre muitas características que o Android possui, Nicolai (2015) ressalta que “[...] o Android possui nativamente um framework de aplicações, uma máquina virtual Dalvik (Java) otimizada para dispositivos móveis, um navegador web baseado na engine de código aberto Webkit, suporte a arquivos de mídia de áudio e vídeo e um ambiente de desenvolvimento rico em ferramentas”. Em relação aos aplicativos do Android, os desenvolvedores contam com uma plataforma disponibilizada pelo Google, denominada “Google Play” que é uma loja virtual que oferece um extenso catálogo de aplicativos gratuitos ou pagos. Afanaci Junior (2012) afirma que o processo para se cadastrar é simples “Basta ter um cadastro com perfil de

desenvolvedor, concordar com os termos de uso estipulados e pagar uma taxa de registro". Com o cadastro efetivado, o desenvolvedor terá acesso a envio de aplicações, configurações e monitoração dos dados relacionados aos aplicativos enviados.

3.3.7 Kotlin

Kotlin é uma linguagem de programação orientada a objeto de código aberto e gratuito, desenvolvido pela empresa JetBrains, a mesma que criou a plataforma Android Studio. A Kotlin veio com objetivo de facilitar a programação em linguagem Java com a redução de codificação. De acordo com Sartori (2018) "A linguagem trabalha da seguinte forma, após o código Kotlin ser escrito, este é compilado para a linguagem java, permitindo o seu total suporte para comunicar códigos Java com Kotlin e vice-versa, após isso então será tudo compilado como um código Java normal".

"O Kotlin é uma linguagem de programação moderna e estaticamente tipada usada por mais de 60% dos desenvolvedores Android profissionais. Seu uso ajuda a estimular a produtividade, a satisfação do desenvolvedor e a segurança do código." (Developer Android, 2021)

De acordo com a pesquisa Stack Overflow citado pelo site geeksforgeeks (2020) a linguagem Kotlin "[...] ocupa a 4° lugar entre as linguagens de programação mais amadas. Além disso, o número de usuários do Kotlin na comunidade Github está aumentando significativamente".

A linguagem Kotlin além de ser compatível com o Java também permite ser usada para desenvolvimento web e desenvolvimento de aplicativos. Para o blog Movile (2020) o código Kotlin pode ser inserido em um projeto já existente, chamando a partir de um código Java. Em relação ao seu código, Movile (2020) afirma que "[...] além de possuir maior legibilidade, é também muito menos verboso. [...] é possível criar um código muito mais limpo, conciso e simples, utilizando funções de alta ordem do que em Java."

O Kotlin possui alguns recursos que garantem vantagem comparando com a linguagem Java que Souza (2018) cita alguns exemplos que são "proteção contra nulo (*Null Safety*), funções estendidas, lambdas, classes de dados (Data Classes),

imutabilidade e co-rotinas. Tornando a linguagem Kotlin ainda mais visada e valorizada".

O Kotlin possui algumas características que a diferem de Java, como ilustra a tabela 2.

Quadro 8 – Vantagens da linguagem Kotlin

Expressivo e conciso	Os recursos modernos da linguagem Kotlin permitem que você se concentre em expressar suas ideias e escrever menos código *boilerplate*[i2].
Código mais seguro	O Kotlin tem muitos recursos de linguagem para ajudar a evitar erros comuns de programação, como exceções de ponteiro nulo. Os apps para Android que contêm código Kotlin são 20% menos propensos à falha.
Interoperabilidade	O Kotlin é completamente interoperável com a linguagem de programação Java. Portanto, é possível ter o quanto de Kotlin que quiser no seu projeto.
Simultaneidade estruturada	As corrotinas Kotlin tornam o código assíncrono tão fácil de trabalhar quanto o código de bloqueio. As corrotinas simplificam drasticamente o gerenciamento de tarefas em segundo plano para tudo, desde chamadas de rede até o acesso a dados locais.

Fonte – Developer Android (2021)

Devido às vantagens citadas anteriormente o aplicativo resultado deste trabalho foi desenvolvido de forma nativa utilizando a linguagem Kotlin e o sistema operacional Android. Outros pontos que favoreceram a escolha da linguagem Kotlin foi a existência de exemplos no site oficial do Android e na documentação da ferramenta de exibição de mapas e conversão de código JAVA para Kotlin feita de forma automática pela IDE Android Studio.

[2] Boilerplate – Em programação de computadores, código boilerplate se refere a seções de código que devem ser incluídas em muitos lugares com pouca ou nenhuma alteração.

4 METODOLOGIA PROPOSTA

Para o desenvolvimento do SIGFIS, foram adotadas as etapas descritas a seguir:

a) Inicialmente foi realizada uma revisão bibliográfica e documental a respeito da evolução da gestão dos recursos hídricos e políticas adotadas em âmbito nacional.

b) Para resultar em um aplicativo capaz de atender as expectativas propostas, foram coletados dados e informações através de reuniões realizadas com o departamento de informática da agência estadual de águas em conjunto com fiscais responsáveis pela gestão de recursos hídricos na região da pesquisa.

c) Com a coleta dos dados de arquivos em formatos georreferenciados, foi decidido que o aplicativo armazenaria em seu banco informações espaciais com formatos compatíveis com aplicações de geoprocessamento. Para o aplicativo do SIGFIS foi utilizado o programa de geoprocessamento de software livre QGIS (2020). Com o intuito de padronizar os dados, foi feita uma revisão cartográfica e optou-se por trabalhar com o sistema de coordenadas de projeção geográfica (datum SIRGAS2000). Para esta etapa, os dados georreferenciados obtidos junto à equipe de fiscalização, possuíam diversas extensões, sendo todos convertidos para o “.shp” (referente a shapefile), por meio do QGIS.

d) Para a montagem dos planos de informações, que conceitualmente reúnem informações que se referem aos aspectos de uma região (GEODEN, 2020), foi realizado um estudo através de pesquisas bibliográficas e com as reuniões mencionadas anteriormente. Como consequência, foram definidas diretrizes de pesquisa, uma com base de dados cadastrais e temáticos: (municípios, bacias hidrográficas e hidrografia de rios e reservatórios); e outra com dados territoriais legalmente relevantes (processos de fiscalização, unidades de conservação, área de proteção dos mananciais, zoneamento de aquíferos, hidrogeologia, municípios em estiagem e águas de domínio da União).

4.1 Modelagem do banco de dados

O ponto inicial na modelagem do aplicativo trata da importação dos arquivos com os planos de informações para um banco de dados que possa ser reconhecido pelo aplicativo SIGFIS. Para tal, foi usado o banco de dados PostgreSQL com a extensão espacial PostGIS, e QGIS para processar os arquivos com informação geográfica .shp e realizar a conversão e importação no PostgreSQL.

Em seguida foi construído um sistema com a responsabilidade de realizar consulta nesse banco de dados e transformar o resultado para o formato “GeoJSON”. Baseado em Json, este formato aberto permite representar informações geográficas através de formas como: pontos, linhas e polígonos com coordenadas geográficas, juntamente com seus atributos não-espaciais (EMTU, 2019). Assim, foi possível visualizar no SIGFIS todos os planos de informações elaborados e realizar atualizações a cada novo processo de fiscalização criado.

4.2 Desenvolvimento do aplicativo

Para que o aplicativo tenha acesso aos recursos do dispositivo, como conexão com a Internet, GPS e sistema de arquivos é necessário adicionar algumas permissões no *AndroidManifest.xml* do projeto:

```
1 <uses-permission android:name="android.permission.INTERNET" />
2 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
3 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
4 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
5 <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
6 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
7 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

O código na linha 1 informa que a aplicação necessita ter acesso à Internet e na linha 2, se existe conexão disponível. Nas linhas 3 e 4 são permissões necessárias para utilizar os serviços de localização. Na linha 5 permite executar operações em segundo plano, como atualizar os arquivos com planos de informações. Nas linhas 6 e 7 são as permissões para ler e escrever arquivos, como os arquivos de

plano de informação e relatórios exportados. Essas permissões serão solicitadas ao usuário à medida que forem necessárias, através de uma caixa de diálogo, caso sejam negadas o aplicativo não funcionará corretamente.

Para que seja possível utilizar o serviço de mapas Mapbox é necessário configurar a credencial de acesso na linha 4:

```
1 override fun onCreate(savedInstanceState: Bundle?) {
2  super.onCreate(savedInstanceState)
3  // Mapbox Access token
4  Mapbox.getInstance(applicationContext, getString(R.string.mapbox_access_token))
5 }
```

Para exibir o mapa em tela é necessário adicionar no arquivo de layout, o xml como segue:

```
1 <com.mapbox.mapboxsdk.maps.MapView
2    android:id="@+id/mapView"
3    android:layout_width="match_parent"
4    android:layout_height="match_parent"
5    mapbox:mapbox_cameraZoom="12" />
```

Na Atividade (*Activity* [3]) que contém o mapa, é registrado o evento que será executado quando o mapa está pronto para ser manipulado, linha 1. Na linha 2 é adicionado o evento que será executado quando o usuário mover o mapa, esse evento é usado para recarregar as informações da área em que o usuário se encontra, como rios próximos. Na linha 3 é definida a camada inicial como satélite, na qual as demais camadas serão adicionadas. Na linha 4 será requisitado o acesso aos serviços de localização. Na linha 5 será chamado o método *loadLayers()* que irá carregar no mapa as camadas com planos de informações, que posteriormente serão exibidas no mapa pelo método *setupLayers()*.

```
1 override fun onMapReady(mapboxMap: MapboxMap) {
2  mapboxMap.addOnMoveListener(this)
```

[3] Uma Activity é um módulo único e independente que normalmente está relacionada diretamente com uma tela de interface de usuário e suas funcionalidades correspondentes. Disponível em: <http://www.androidpro.com.br/activity-intro/> acesso em 25 de abril de 2021.

```
3    mapboxMap.setStyle(Style.SATELLITE) {
4      enableLocationComponent(it)
5      loadLayers(it)
6      setupLayers(it)
7    }
8  }
9
10 private fun loadLayers(loadedMapStyle: Style) {
11   for (s in sourcesMap) {
12     val source = GeoJsonSource(s.key, URI("http://10.0.0.185:5000/layers/" + s.value +".geojson"))
13     loadedMapStyle.addSource(source)
14   }
15 }
16
17 private fun setupLayers(loadedMapStyle: Style) {
18     val layer = LineLayer("geojson-fiscalizacao", "geojson-fiscalizacao")
19     layer.setSourceLayer("contour")
20     layer.setProperties(
21         PropertyFactory.lineJoin(Property.LINE_JOIN_ROUND),
22         PropertyFactory.lineCap(Property.LINE_CAP_ROUND),
23         PropertyFactory.lineColor(Color.parseColor("#ff69b4")),
24         PropertyFactory.lineWidth(1f)
25     )
26     loadedMapStyle.addLayer(layer)
27     //...código omitido (carregamento das demais camadas)
28 }
```

4.3 Desenvolvimento da Interface de Programação de Aplicações (API) de Integração

Para que o aplicativo possa carregar os dados dos planos de informações, é feita na API de integração uma consulta no banco de dados PostgreSQL, como mostrado na função *getFiscalizacoes()* (linha 2). Como resultado da consulta será retornado um texto no formato geojson contendo um campo com um conjunto de metadados (linhas 18 a 23) associados a um outro campo com as informações geométricas (linha 17), que no trecho de código seguinte é o ponto com latitude e

longitude de uma vistoria realizada, que usa a função *ST_AsGeoJSON* do PostGis para conversão do campo em geojson.

```
export class LayersRepository {
 async getFiscalizacoes(): Promise<string> {
   const entityManager = getManager();
   const fileQuery = entityManager.query(`
SELECT json_build_object(
   'type', 'FeatureCollection',
    'crs',  json_build_object(
      'type',     'name',
      'properties', json_build_object(
         'name', 'EPSG:4674'
       )
),
   'features', json_agg(
      json_build_object(
         'type',      'Feature',
         'id',        "id",
         'geometry',   ST_AsGeoJSON(geom)::json,
         'properties', json_build_object(
            'type', 'fiscalizacao',
            'title', 'Fiscalizações',
            'pf', "proc.fisc.",
            'dt', "dt_entrada",
            'pa', "proc_outor"
         )
      )
   )
)
   FROM fiscalizacoes`, []);
   //...código omitido (retorno do resultado)
}}
```

4.4 Objetivos do aplicativo móvel SIGFIS

Além de proporcionar apoio à gestão e fiscalização dos recursos hídricos por meio de um SIG, o aplicativo SIGFIS tem como meta auxiliar os agentes fiscais em suas visitas a campo, no sentido de proporcionar economia e agilidade nos serviços, visto que reúne informações relevantes e imprescindíveis ao seu trabalho. Sendo assim, os requisitos propostos para o aplicativo adotam os seguintes pontos:

a) Leitura do sensor GPS com o reconhecimento em tempo real do posicionamento do aparelho móvel;

b) Leitura dos planos de informações definidos no SIGFIS considerando o posicionamento do aparelho móvel;

c) Bufferização de áreas previstas para fiscalização em campo como forma de melhorar rendimento e economia de espaço de armazenamento do aparelho móvel;

d) Cadastro de usos de recursos hídricos identificados em campo como forma de registro inicial e auxiliar aos relatórios de fiscalização;

e) Alerta aos agentes fiscais no caso de fiscalização realizada em posicionamento inserido em algum território legalmente relevante.

4.5 Tecnologias utilizadas

A partir da definição dos objetivos a serem alcançados pelo aplicativo foram adotados recursos tecnológicos capazes de cumprir todas as etapas de criação e desenvolvimento, conforme segue:

a) Diagramação dos Recursos;

b) Servidor WEB;

c) Sistema operacional e Linguagem de Programação;

d) Interface de Programação de Aplicações (API) de Integração;

e) Sistema de Gerenciador de Banco de Dados;

f) API de Geolocalização.

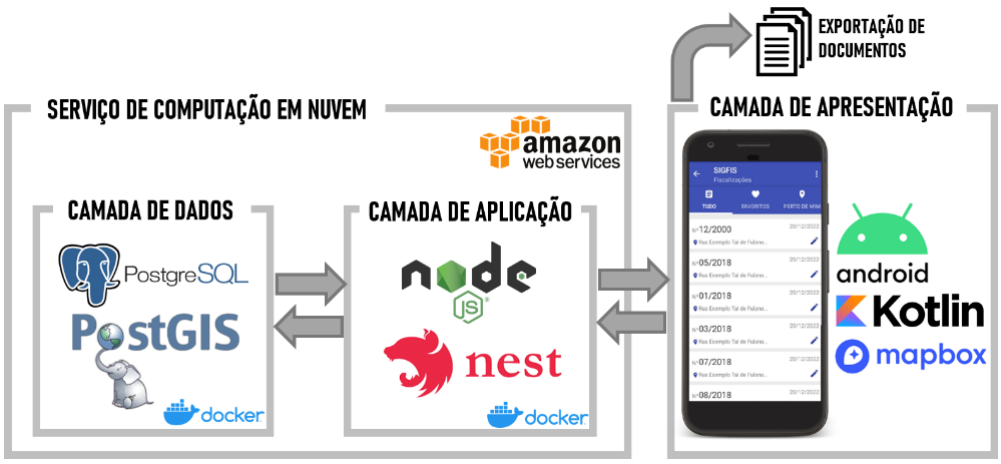
Os usos das tecnologias citadas acima serão expostos no resultado obtido com o desenvolvimento do aplicativo.

5 ANÁLISE E IMPLEMENTAÇÃO

Com as bases de dados e os planos de informações definidos, traçadas a modelagem e as tecnologias utilizadas, tem-se como resultado a arquitetura do aplicativo e suas camadas.

Conforme a Figura 5 apresenta, o banco de dados utilizado foi o PostgreSQL com a API de geolocalização MapBox. Vale ressaltar que o PostgreSQL foi o repositório escolhido por ser um banco de dados relacional, ser utilizado por órgãos de fiscalização e poder ser aplicado facilmente em soluções geográficas.

Figura 5 - Arquitetura de desenvolvimento do aplicativo móvel SIGFIS



Fonte: elaborada pelo autor (2021)

A arquitetura do aplicativo é dividida em três camadas:

a) Camada de Apresentação: responsável por interagir diretamente com o fiscal. Foi utilizado o sistema operacional Android e a linguagem de programação de código fonte aberto Kotlin, escolhida por se tratar de uma linguagem concisa e segura estaticamente tipada. A API de geolocalização utilizada foi o Mapbox que permite a interação de mapas quando utilizados por Android SDK em aplicativos.

b) Camada de Aplicação: responsável por intermediar a comunicação entre a camada de apresentação e de dados. Foi utilizado um servidor em nuvem na Amazon Web Services (AWS) em conjunto com a plataforma Docker e uma API na linguagem de programação NodeJS como a interface de comunicação. A opção

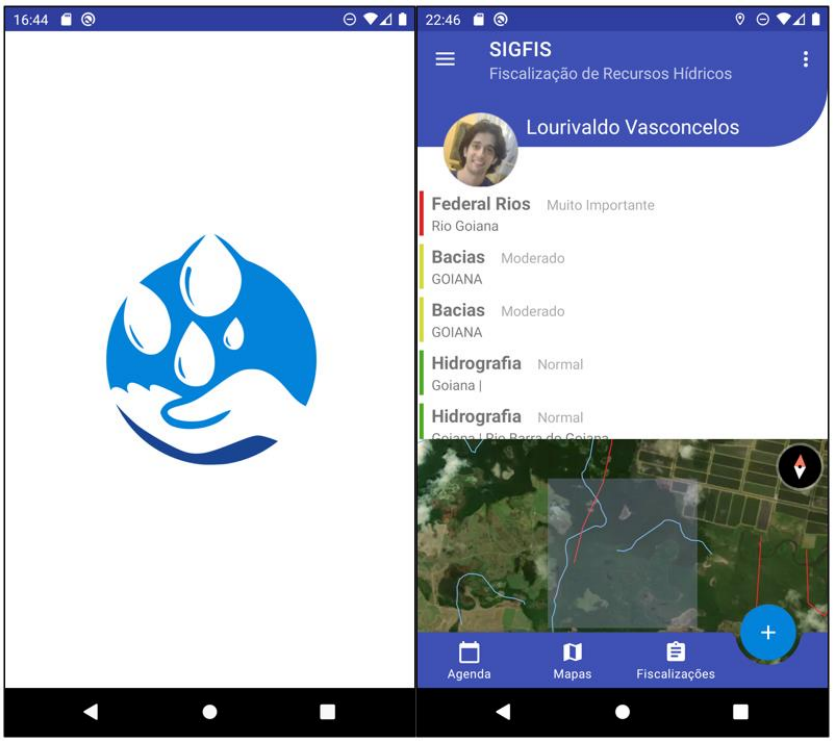
pelo servidor e a plataforma proporcionam ao SIGFIS agilidade e rapidez nas aplicações e mobilidade quanto a ajustes necessários. Na API NodeJS foi configurada a segurança empregando o protocolo HTTPS, utilizada a instância com Ubuntu Server 18.04 LTS, que fornece 1GB de Memória de processamento, 8GB de Memória de armazenamento e 1 CPU, recursos suficientes para o protótipo do aplicativo.

c) Camada de Dados: responsável por armazenar os planos de informação e os dados das fiscalizações, através do servidor AWS. Foi adotado para o SIGFIS o sistema gerenciador de banco de dados relacional PostgreSQL versão 11.4-R1, optado por fornecer 1 GB de Memória de processamento, 20 GB de Memória de armazenamento e 1 CPU, recursos suficientes para o protótipo do aplicativo.

5.1 Implementação do protótipo – SIGFIS

Na Figura 6, ao entrar no aplicativo é possível visualizar a tela de carregamento e logo em seguida a tela inicial onde são exibidas as informações geográficas presentes na localização atual do fiscal. Nessa tela há uma listagem com as informações contidas nos planos de informações que foram importadas para o banco de dados, como existência de rios federais, bacias hidrográficas, unidades de conservação, municípios em situação de emergência e processos de fiscalização. Essa listagem é elencada pelo nível de relevância na realização de uma vistoria nesse local, de forma a auxiliar o fiscal na identificação de pontos como: processos de fiscalização próximos, de forma a facilitar o preenchimento dos dados durante uma nova vistoria ou na identificação de possíveis reincidências; em casos de rios federais é possível evitar autuações equivocadas, visto que o órgão estadual não possui competência para a ação da fiscalização em corpo d’água de domínio da União; quando situado em uma Unidade de Conservação existe a possibilidade de agravamento da multa caso identificada uma infração. Essas informações estão disponíveis sem necessidade de haver conexão com internet, desde que tenham sido carregadas anteriormente.

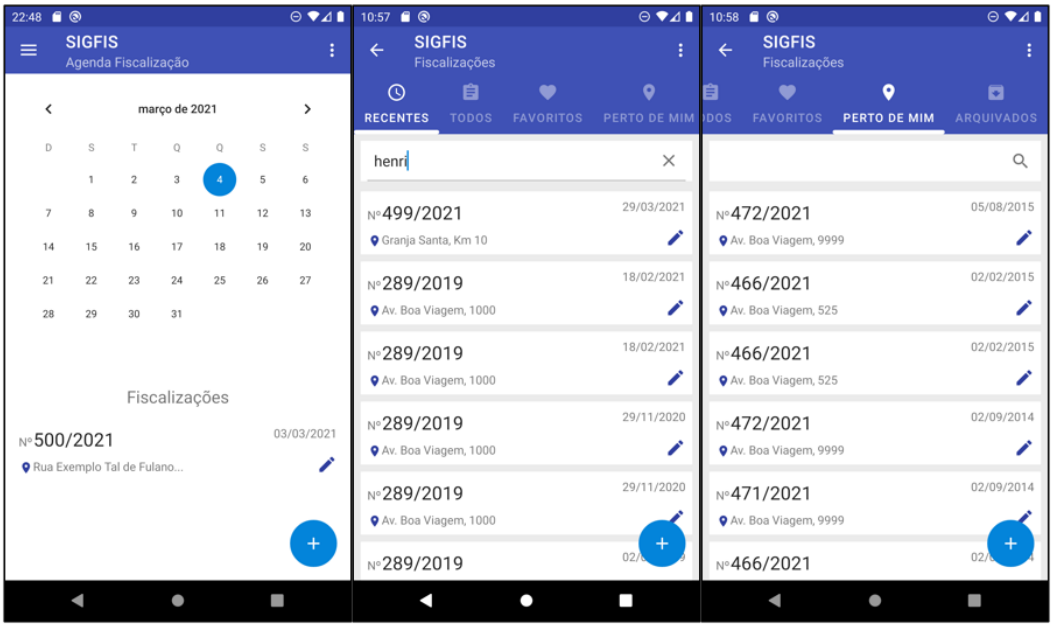
Figura 6 - Tela de carregamento e tela inicial



Fonte: elaborada pelo autor (2021)

A partir da tela inicial é possível acessar a agenda Figura 7, onde é possível encontrar fiscalizações realizadas e agendar vistorias. A listagem de fiscalizações, Figura 7, os itens são organizados nas categorias recentes, todos, minhas, perto de mim e arquivados. Na categoria recentes ficam as criadas ou editadas recentemente. Na categoria todos se encontra uma listagem completa das fiscalizações criadas. Na categoria minhas ficam todos itens criados pelo próprio fiscal. Na categoria perto de mim ficam as fiscalizações próximas a localização atual do fiscal. Na categoria arquivados ficam agendamentos expirados e processos arquivados. Em todas categorias é possível filtrar por informações como número do processo, endereço, nome do usuário da água, entre outros.

Figura 7 - Fiscalizações e Agendamento

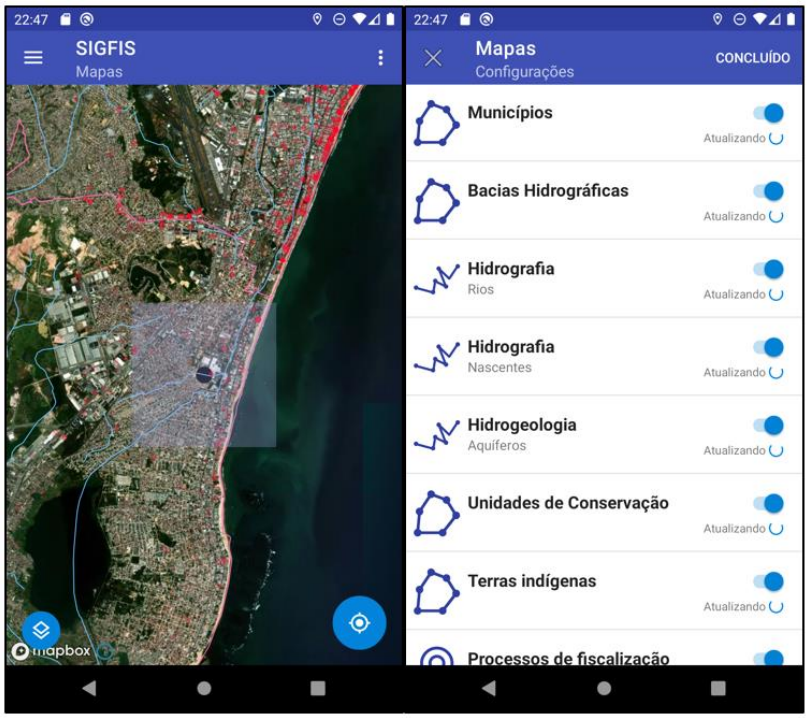


Fonte: elaborada pelo autor (2021)

Na Figura 8, observa-se a listagem dos planos de informações disponíveis para visualização no mapa, com suas informações de data e hora de atualização, status de atualização, e opções de habilitar/desabilitar exibição.

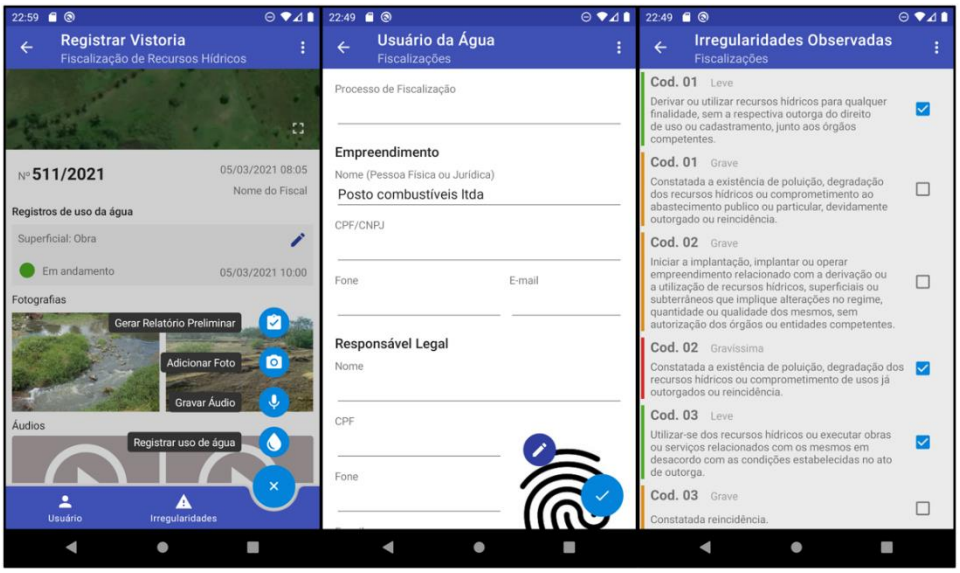
Foi usado o mapa de satélite como padrão, podendo ser adicionados os planos de informações de hidrografia, municípios, processos de fiscalização, e outros à medida que forem necessários, de forma a não sobrecarregar a visualização.

Figura 8 - Planos de informações



Fonte: elaborada pelo autor (2021)

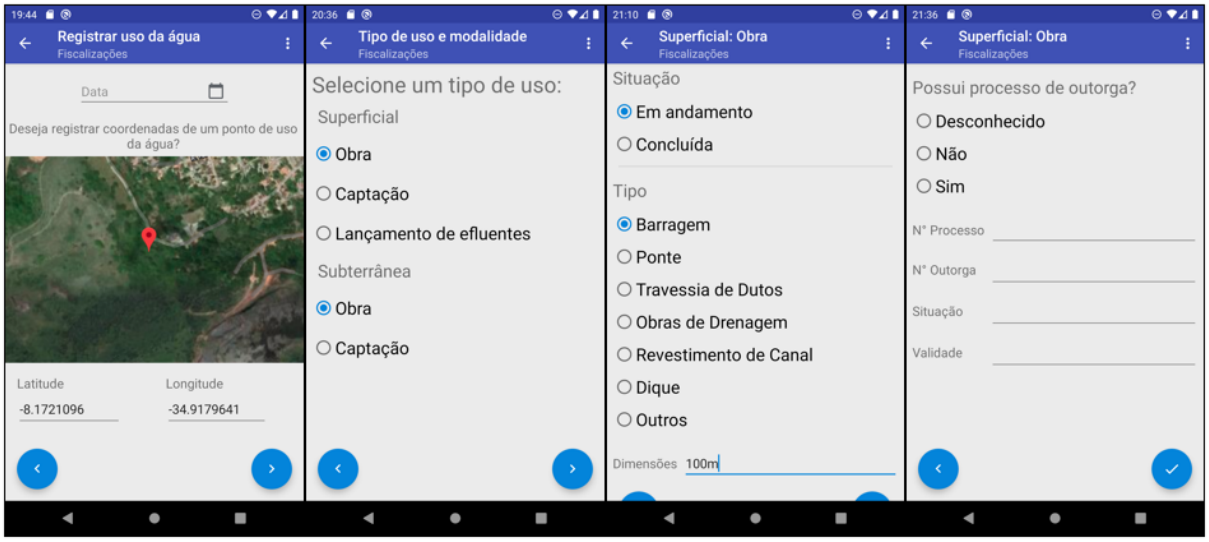
Figura 9 - Registro de usuário e infrações



Fonte: elaborada pelo autor (2021)

Na Figura 9, visualiza-se a tela de registro de nova vistoria, com atalho para visualização do mapa da localização atual do fiscal. Além das opções de cadastrar um uso da água, captura de fotos e áudios, cadastro de informações do usuário da água e cadastro de irregularidades observadas durante vistoria, geração do relatório de auditoria.

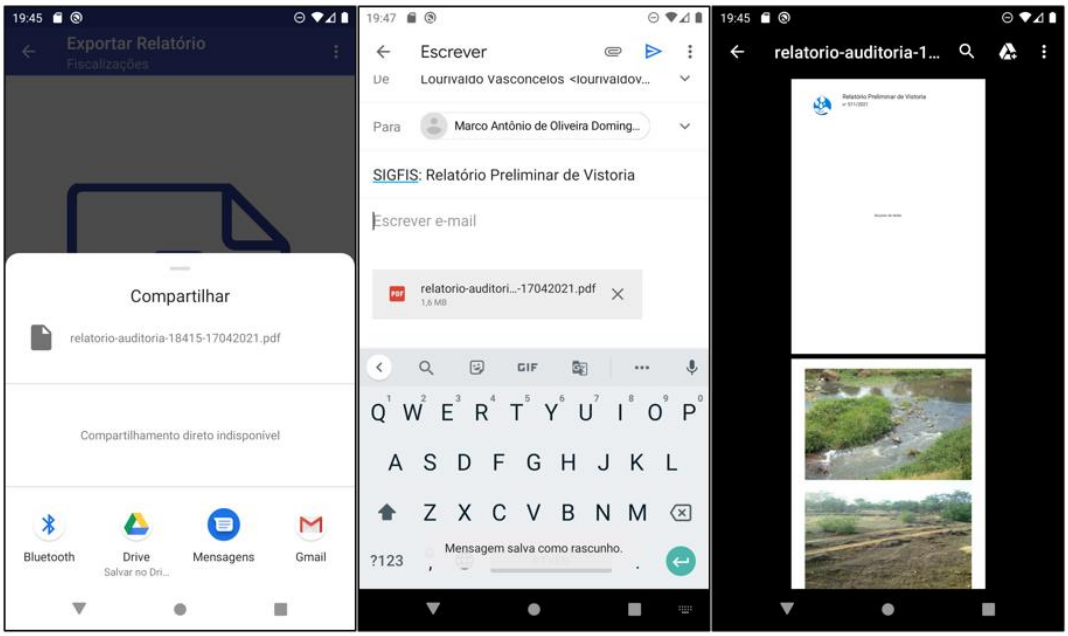
Figura 10 - Registro de uso de água



Fonte: elaborada pelo autor (2021)

Na Figura 10, é possível visualizar a tela para registro de uso de água, onde são selecionados a data e coordenadas no mapa. Nos formulários seguintes seleciona-se o tipo e subtipo de uso, que irá definir os próximos formulários a serem apresentados. No exemplo em questão, informa-se a situação e tipo de obra e dimensões, além das opções de busca por processos anteriores de fiscalização, com preenchimento dos dados do processo, outorga, em casos de processos existentes durante a busca.

Figura 11 - Gerar Relatório em PDF



Fonte: elaborada pelo autor (2021)

Na Figura 11, é possível visualizar a exportação do relatório de auditoria contendo todas informações preenchidas durante a vistoria. Sendo possível compartilhar por e-mail ou salvar no dispositivo. Essa funcionalidade foi criada para que seja possível prosseguir com o restante do processo de fiscalização. Vale ressaltar que é possível gerar o relatório a qualquer momento, não sendo necessário preencher todos formulários existentes.

6 CONSIDERAÇÕES

Observando os resultados obtidos por meio das pesquisas, análises e reuniões com potenciais usuários da solução, considera-se que os objetivos do desenvolvimento do SIGFIS foram alcançados. Durante simulações de uso o aplicativo proporcionou apoio à fiscalização dos recursos hídricos com agilidade e rapidez nos processos dos agentes fiscalizadores, diminuindo inclusive o retorno de visitas a campo, fato atrelado à possibilidade de validar informações geográficas durante a vistoria.

A utilização dos planos de informação proporcionará melhor conhecimento da cartografia e informações geográficas do território de Pernambuco, possibilitando consultar tais informações durante a ida a campo da equipe de fiscalização.

Como projeto futuro em relação ao uso do aplicativo pelos órgãos reguladores, o SIGFIS estará aberto a desenvolver aplicações em relação a impressões de documentos nos locais de vistoria com uso de impressoras portáteis, trazendo ainda mais agilidade aos fiscais.

Por fim a fiscalização dos recursos hídricos em conjunto com ferramentas de apoio fortalece a Política Nacional de Recursos Hídricos e contribui de forma efetiva com as tomadas de decisões.

REFERÊNCIAS

AFANACI JUNIOR, Marcelo. **Agreguei:** sistema agregador de ofertas para plataforma Android. 2012. 71 f. Trabalho de Conclusão de Curso (Especialização) – Universidade Tecnológica Federal do Paraná, Curitiba, 2012.

AGÊNCIA NACIONAL DE ÁGUAS. Superintendência de Planejamento de Recursos Hídricos**. A gestão de recursos hídricos em Pernambuco**. Brasília, 2019. Disponível em: https://progestao.ana.gov.br/portal/progestao/panorama-dos-estados/pe. Acesso em: 03 abr. 2021.

AGUIAR, Everson L. **Levantamento de ações de governo móvel no Brasil**. *In:* 2ª Conferência Web W3C Brasil, São Paulo. 2010.

ALMEIDA *et al.* **Uso de *Business Intelligence* na Gestão de Recursos Hídricos**: o caso da Fiscalização do Uso da Água. 2019. Disponível em: https://sol.sbc.org.br/index.php/wcama/article/view/6415. Acesso em: 07 abr. 2021.

ALVES *et al.* **Um método para gerenciamento do processo de fiscalização dos recursos hídricos.** 2018. Disponível em: https://sol.sbc.org.br/index.php/wcama/article/view/2936. Acesso em: 20 abr. 2021.

ALVES *et al.* **O uso de tecnologia da informação na fiscalização e denúncias do uso de recursos hídricos.** 2018. Disponível em: https://sol.sbc.org.br/index.php/wcama/article/view/6415. Acesso em: 20 abr. 2021.

ANDROID. **O que é o Android.** 2021. Disponível em: https://www.android.com/intl/pt-BR_br/what-is-android/. Acesso em: 27 abr. 2021.

ASPER BROTHERS. ***Cross-Platform App Development – Explore Frameworks, Technology and Business Benefits*.** 2020. Disponível em: https://asperbrothers.com/blog/cross-platform-app-development/#:~:text=Cross%2Dplatform%20application%20development%20is,app%20versions%20for%20each%20platform). Acesso em: 24 mar. 2021.

ASSAD, E.D.; SANO, E.E. **Sistemas de informações geográficas (Aplicações na Agricultura)**. 2. ed. Brasília: SPI/EMBRAPA-CPAC, 1998.

BORGHEZAN, Igor Liberato Fernandes *et al.* **Desenvolvimento do aplicativo educacional meu texto para plataforma android.** 2018. 43 f. TCC em Tecnologia da Informação e Comunicação. Universidade Federal de Santa Catarina. 2018.

BORSOI, Z. M. F.; TORRES, S. D. A. A política de recursos hídricos no Brasil. **Revista do BNDS**, Rio de Janeiro, v.4, n.8, p.143-166, dez. 1997.

BRASIL. **Constituição da República Federativa do Brasil, de 5 de outubro de 1988**. Disponível em: https://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm. Acesso em: 11 fev. 2021.

BRASIL. **Lei n. 9.433, de 8 de janeiro de 1997**. Disponível em: http:// www.camara.leg.br/legin/fed/lei/1997/lei-9433-8-janeiro-1997-374778-norma-pl.html. Acesso em: 12 fev. 2021.

BRASIL. **Lei n. 9.984, de 17 de julho de 2000**. Disponível em: http://www.planalto.gov.br/ccivil_03/leis/l9984.htm. Acesso em: 12 fev. 2021.

BRASIL. **Decreto n. 3.692, de 19 de dezembro de 2000**. Disponível em: http://www.planalto.gov.br/ccivil_03/decreto/D3692.htm. Acesso em: 12 fev. 2021.

CÂMARA, Gilberto; MEDEIROS, José S. Princípios básicos em Geoprocessamento. *In:* ASSAD, Eduardo D.; SANO, Edson E. **Sistemas de Informações Geográficas. Aplicações na agricultura.** 2. ed. Brasília: Embrapa, 1998.

CÂMARA, G.; MEDEIROS, J. S. de. Tendências de Evolução do Geoprocessamento. *In:* ASSAD, E. D.; SANO, E. (Org.) **Sistemas de Informações Geográficas**. 2. ed. Brasília: Embrapa, 2003.

CÂMARA, G., ORTIZ, M.J. Sistemas de informação geográfica para aplicações ambientais e cadastrais: uma visão geral. *In:* **Congresso Brasileiro de Engenharia Agrícola**. sn, 1998. p. 59-82.

CAVALCANTE, Rodrigo. **Introdução ao SIG.** Pró-Reitoria de Planejamento e Desenvolvimento. UFMG. 2015. Disponível em: https://www.ufmg.br/proplan/wp-content/uploads/Apostila-de-Introdu%C3%A7%C3%A3o-ao-SIG-Proplan-2015.pdf. Acesso em: 02 abr. 2021.

CIRILO, Reinaldo. **Dispositivos móveis - História e Evolução. 2020.** Disponível em: https://www.reinaldocirilo.com.br/post/2019/04/21/dispositivos-m%C3%B3veis-hist %C3%B3ria-e-evolu%C3%A7%C3%A3o#:~:text=Foi%20a%20primeira%20chamada %20m%C3%B3vel,e%2033%20cent%C3%ADmetros%20de%20altura. Acesso em: 03 mar. 2021.

COLUSSI. **Aplicativos Híbridos Vs Nativos:** prós e contras. 2020. Disponível em: https://www.keyworks.com.br/aplicativos-hibridos-vs-nativos-pros-e-contras/. Acesso em: 21 mar. 2021.

DÂMASO, Lívia. **O que é app? Quatro perguntas e respostas sobre aplicativos para celular.** TechTudo. 2019. Disponível em: https://www.techtudo.com.br/ noticias/2019/12/o-que-e-app-quatro-perguntas-e-respostas-sobre-aplicativos-para-celular.ghtml. Acesso em: 23 mar. 2021.

DERY, Kristine; KOLB, Darl; MACORNICK, Judith. **Trabalhando com fluxo conectivo:** como o uso de smartphones está evoluindo na prática. *European Journal of Information Systems*, n. 23, p. 558-570, mai. 2014.

DEVELOPER ANDROID. **Abordagem Kotlin do Android**. Disponível em: https://developer.android.com/kotlin/first?hl=pt. Acesso em: 30 mar. 2021.

DEVELOPER. ***Developer* Android.** 2020. Disponível em: https://developer.android.com/topic/libraries/support-library?hl=pt-br. Acesso em: 30 mar. 2021.

DEVELOPER ANDROID. **Pilha de Software do Android.** Disponível em: https://developer.android.com/guide/platform?hl=pt-br. Acesso em: 28 mar. 2021.

EMTU. **Relatório Mensal Páginas Dados Abertos.** 7º Relatório. Assessoria de Parcerias e Inovação. Chefia de Gabinete. Distrito Federal. 2019. Disponível em: https://www.emtu.sp.gov.br/EMTU/pdf/RelatorioDadosAbertosEMTU_janeiro2019.pdf . Acesso em: 02 abr. 2021.

FREITAS, Vladimir Passos de. **Águas – considerações gerais.** FREITAS, Vladimir Passos de (coord.). Águas: aspectos jurídicos e ambientais. 7. ed. Curitiba: Juruá, 2008.

GARBADE. ***Native vs. Cross-platform app development:*** *pros and cons*. 2018. Disponível em: https://codeburst.io/native-vs-cross-platform-app-development-pros-and-cons-49f397bb38ac. Acesso em: 24 mar. 2021.

GARCIA, Jean Carlo Vargas. **Plataformas android e arduino e tecnologia *bluetooth*.** Engenharia Elétrica Telemática-Pedra Branca, 2013.

GARTNER. **Futuro do desenvolvimento de aplicações é a oferta de soluções de multiexperiências**. Tiinside, 2019. Disponível em: https://tiinside.com.br/ 20/09/2019/para-o-gartner-futuro-do-desenvolvimento-de-aplicacoes-e-a-oferta-de-solucoes-de-multiexperiencias/. Acesso em: 06 mar. 2021.

GEEKHUNTER. **PWA: O que é? Vale a pena?** Disponível em: https://blog.geekhunter.com.br/pwa-o-que-e-vale-a-pena/. Acesso em: 23 mar. 2021

GEEKSFORGEEKS. ***Top 10 Programming Languages That Will Rule in 2020***. Disponível em: https://www.geeksforgeeks.org/top-10-programming-languages-that-will-rule-in-2021/#:~:text=The%20Stack%20Overflow%20survey%20states,the%20most%20loved%20programming%20languages. Acesso em: 30 mar. 2021.

Geoden. **Sistema de Informação Geográfica – SIG.** 2020. Disponível em: http://geoden.uff.br/geoprocessamento/. Acesso em: 04 abr. 2021.

GOOGLE INC. **Dev Guide**. Disponível em: http://developer.android.com/guide/basics/what-isandroid.html. Acesso em: 29 mar. 2021.

GRAÇA, João Paulo Barata da Rocha Gagliardini. **Avaliação potencialidades e impactos dos sistemas de informação geográfica na gestão pública de recursos hídricos**. 2009. 162f. Dissertação (Mestrado em Inovação e Empreendedorismo Tecnológico) – Faculdade de Engenharia, Universidade do Porto, Porto, 2009.

LECHETA, Ricardo R. **Google Android:** aprenda a criar aplicações para dispositivos móveis com o Android SDK. 2. ed. São Paulo: Novatec Editora, 2010.

LEPAGE. ***What are Progressive Web Apps?*** 2020. Disponível em: https://web.dev/what-are-pwas/. Acesso em: 23 mar. 2021.

LIMA, Cíntia Caldas Barcelar de. **Aplicativos móveis de interesse público:** limites e possibilidades para a cidadania no Brasil. Dissertação Mestrado. UNB. Brasília. 2017. Disponível em: https://repositorio.unb.br/bitstream/10482/23699/1/2017_C%C3%ADntiaCaldasBarcelardeLima.pdf. Acesso em: 19 abr. 2021.

LONGLEY, Paul A. *et al.* **Sistemas e Ciência da Informação Geográfica**. 3. ed. Porto Alegre: Bookman, 2011.

MATOS, Beatriz Rezener Dourado; DE BRITTO, João Gabriel. **Estudo comparativo entre o desenvolvimento de aplicativos móveis utilizando plataformas nativas e multiplataformas.** Trabalho de Conclusão de Curso de Engenharia de software, FGA/UnB. 2017.

MAXIMILIANO, Meyer. **A história do Android**. 2020. Disponível em: https://www.oficinadanet.com.br/post/13939-a-historia-do-android#:~:text=O%20primeiro%20dispositivo%20Android%20foi,Rubin%20definiu%20o%20Android%20Inc. Acesso em: 17 abr. 2021

MEIRELLES, Fernando. **31ª Pesquisa Anual do FGVcia**. FGV, 2020. Disponível em: https://portal.fgv.br/noticias/brasil-tem-424-milhoes-dispositivos-digitais-uso-revela31a -pesquisa-anual-fgvcia. Acesso em: 05 mar. 2021.

MELO, Adriany de Ávila. **Atlas geográfico escolar:** aplicação analógica e digital no ensino fundamental. 2006. 305f. Tese: Doutorado em Geografia. Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006

MENESES, H. B. **Interface Lógica em Ambiente SIG para Bases de Dados de Sistemas Centralizados de Controle do Tráfego Urbano em Tempo Real,** Dissertação de Mestrado, Centro de Tecnologia, Universidade Federal do Ceará, Fortaleza, 2003.

MEYER, Maximiliano. **A história do Android.** 2020. Disponível em: https://www.oficinadanet.com.br/post/13939-a-historia-do-android. Acesso em: 26 mar. 2021.

MIRANDA, *et al.* **Introdução Ao Gerenciamento De Recursos Hídricos**. 2021. Disponível em: https://lamorh.ufes.br/sites/lamorh.ufes.br/files/field/anexo/introducao_ao_gerenciamento_de_recursos_hidricos.pdf. Acesso em: 10 abr. 2021.

MOVILE. **Kotlin vs Java: motivos para trocar o Java pelo Kotlin ainda hoje.** Disponível em: https://movile.blog/motivos-para-trocar-o-java-pelo-kotlin-ainda-hoje/. Acesso em: 30 mar. 2021.

NAIDOO, Anil; DAVIDSON-HARDEN, Adam. **Água como recurso internacional estratégico:** as novas guerras da água**.** Disponível em: http://www.blueplanetproject.net/ documents/water_wars.pdf. Acesso em: 15 fev. 2021.

NICOLAI, Bruno Bernardeli *et al.* **Google android-a plataforma, seus componentes e suas versões.** Disponível em: http://www.williamluis.com.br/wp-content/uploads/2013/10/TCC-Google-Android-Final.pdf. Acesso em: 02 abr. 2021.

NOCERA, R.; Santo, F. **Testes de usabilidade entre os sistemas Android e IOS para identificar o melhor sistema para os idosos**. 2015. SIMTEC - Simpósio de Tecnologia da Fatec Taquaritinga, v. 4, n. 1, p. 15, 14 maio 2018.

NOGUEIRA, Rodolpho Bruno dos Santos. **Desenvolvimento de um sistema de apoio a eventos para uma Plataforma Android.** 2012. 164 f. Trabalho de Conclusão de Graduação Curso de Bacharelado em Ciência da Computação, Centro Universitário Eurípides de Marília, Marília, 2012.

OLIVEIRA, Ana Rachel Fonseca de; DE MENEZES ALENCAR, Maria Simone. O uso de aplicativos de saúde para dispositivos móveis como fontes de informação e educação em saúde. **RDBCI: Revista Digital de Biblioteconomia e Ciência da Informação**, v. 15, n. 1, p. 234-245, 2017.

OLIVEIRA Carlos Ueslei R.e ZEILHOFER Peter. **Sistema de Suporte à Decisão baseado em Lógica *Fuzzy* para Outorga de Recursos Hídricos Superficiais.** 2017. Disponível em: https://sol.sbc.org.br/index.php/wcama/article/view/3437. Acesso em: 09 abr. 2021.

PEREIRA, Lúcio Camilo Oliva; SILVA, Michel Lourenço da Silva. **Android para Desenvolvedores.** Rio de Janeiro: Brasport, 2009.

PERNAMBUCO, **Lei nº 11426 de 17 de janeiro de 1997.** Disponível em: https://legis.alepe.pe.gov.br/texto.aspx?tiponorma=1&numero=12984&complemento=0&ano=2005&tipo=&url=. Acesso em: 13 fev. 2021.

PERNAMBUCO, **Lei nº 13.205, de 19 de janeiro de 2007**. Disponível em: https://legis.alepe.pe.gov.br/texto.aspx?id=2212&tipo=textoatualizado. Acesso em: 12 fev. 2021.

PERNAMBUCO, **Lei nº 14.028, de 26 de março de 2010**. Disponível em: https://legis.alepe.pe.gov.br/texto.aspx?tiponorma=1&numero=14028&complemento=0&ano=2010&tipo=&url=. Acesso em: 12 fev. 2021.

PERNAMBUCO, **Decreto Estadual nº 38752 de 22 de outubro de 2012.** Disponível em: https://legis.alepe.pe.gov.br/texto.aspx?id=16290. Acesso em: 13 fev. 2021.

PICOLI, Cristian. **Aplicativos móveis:** entenda por que a empresa deve investir nisso. 2021. Disponível em: https://atmdigital.com.br/aplicativos-moveis-entenda-por-que-a-empresa-deve-investir-nisso/. Acesso em: 29 abr. 2021.

PIMENTA, Rafael da Silva. **Desenvolvimento de jogos para a plataforma android utilizando corona**. SDK. 2012.
QGIS Development Team. **QGIS *Geographic Information System*. Versão 3.16.5: Hannover.[s.l.].** Open Source Geospatial Foundation Project, 2020. Disponível em: http://qgis.osgeo.org. Acesso em: 05 abr. 2021

REIS, Antônio Carlos Serafim dos. **Um estudo comparativo entre modelos de desenvolvimento de aplicações móveis**. 2019. 39 f. Trabalho de Conclusão de Curso de Graduação em Engenharia de Software, Universidade Federal do Ceará, Campus de Quixadá, Quixadá, 2019.

RIGAUX, Philippe; SCHOOL, Michel; VOISARD, Agnès. **Bancos de dados espaciais:** com aplicação em SIG. Spatial Databases: with application to GIS. São Francisco: Elsevier, 2002,

ROSA, Vagner Santos da**.** Ambcare: monitoramento ambiental usando dispositivos móveis. **Revista de Empreendedorismo, Inovação e Tecnologia**, v. 1, n. 2, p. 43-49, 2015.

SANTOS, Rafael Rocha Rodrigues dos. **Sistema de Fiscalização da Agência Nacional de Águas.** 2013 UnB. Planaltina. Disponível em: https://bdm.unb.br/bitstream/10483/6705/1/2013_RafaelRochaRodriguesDosSantos.pdf. Acesso em: 02 abr. 2021.

SARTORI, Lucas Antonio Ramos. **Sistema de gerenciamento de licenças de posse e porte de armas de fogo**. 2018. 51 f. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná, 2018.

SILVA, J.SV. **Análise multivariada em zoneamento para planejamento ambiental. Estudo de caso:** Bacia Hidrográfica do Rio Taquari MS/MT. 2003. 307 f. Tese: Doutorado em Engenharia Agrícola. Universidade Estadual de Campinas, Campinas, 2003.

SILVA, Tárek Holanda. **Desenvolvendo um aplicativo híbrido para criação e comunicação de eventos em uma universidade**. 2017. 36 f. TCC Graduação em Sistemas de Informação, Universidade Federal do Ceará, Campus Quixadá, Quixadá, CE. 2017.

SILVA, João Vitor; RICARDO, André Wronscki. **Protótipo de aplicativo de relacionamento:** com base em eventos do Facebook. 2019. 54 f. Trabalho de Conclusão de Curso de Graduação em Sistemas de Informação, Universidade do Sul de Santa Catarina. Palhoça. 2019.

SILVA, Marcelo Moro da; SANTOS, Marilde Terezinha Prado. Os paradigmas de desenvolvimento de aplicativos para aparelhos celulares. **Revista TIS**, v. 3, n. 2, 2014.

SIRVINSKAS, Luís Paulo. **Manual de direito ambiental**. 13. ed. São Paulo: Saraiva, 2015.

SOURCE ANDROID. **Configuração para desenvolvimento em Android.** Disponível em: https://source.android.com/setup. Acesso em: 30 mar. 2021.

SOUZA JUNIOR, Wilson Cabral de. **Gestão das Águas no Brasil:** reflexões, diagnósticos e desafios. São Paulo: Petrópolis, 2004.

SOUZA, Matheus González Maia de. **Sistema para verificação da necessidade ou eficiência de proteção contra descargas atmosféricas através do gerenciamento de risco.** 2018. 83 f. Trabalho de conclusão de Graduação em Engenharia da Computação. Universidade Estadual do Maranhão, São Luís, 2018.

SUCCESSIVETECH. ***Benefits of Cross-Platform Development*.** 2019. Disponível em: https://medium.com/successivetech/benefits-of-cross-platform-developme nt-bfa5f708c0a4. Acesso em: 24 mar. 2021.

TERC. **Introdução em SIG**. *Intro to GIS*. 2017. Disponível em: https://serc.carleton.edu/eyesinthesky2/week5/intro_gis.html. Acesso em: 15 fev. 2021.

TOONEN. ***What is a progressive web app (PWA)? Why would you want one?*.** 2020. Disponível em: https://yoast.com/what-is-a-progressive-web-app-pwa/. Acesso em: 24 mar. 2021.

TUCCI, C. E. M.; HESPANHOL, I.; CORDEIRO NETTO, O. de M. **Gestão da água no Brasil.** Brasília: Unesco, 2001. Disponível em: http://www.crmariocovas.sp.gov.br/pdf/pol/gestao_agua.pdf. Acesso em: 18 fev. 2021.

UNESCO. **Diretrizes de políticas da UNESCO para a aprendizagem móvel**. 2014. Disponível em: http://unesdoc.unesco.org/images/0022/002277/227770por.pdf. Acesso em: 05 mar. 2021.

UNESCO. ***Integrated water resources management in action***. UNESCO, 2009. Disponível em: https://unesdoc.unesco.org/ark:/48223/pf0000181891. Acesso em: 18 fev. 2021.

VENTEU, Kelly Cristina; PINTO, Giuliano Scombatti. DESENVOLVIMENTO MÓVEL HÍBRIDO. **Revista Interface Tecnológica**, v. 15, n. 1, p. 86-96, 2018.

APÊNDICE A: Código da Atividade da Tela de solicitação de permissões do aplicativo SIGFIS

```
package com.lourivaldo.sigfis

import android.Manifest
import android.annotation.SuppressLint
import android.app.AlertDialog
import android.content.Context
import android.content.Intent
import android.location.LocationManager
import android.os.Bundle
import android.os.Handler
import android.provider.Settings
import android.util.Log
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity

class SplashScreenActivity : AppCompatActivity(),
PermissionsListener, PermissionListener {

    private var permissionsManager: PermissionsManager =
PermissionsManager(this)
    private lateinit var gpsDialog: AlertDialog.Builder

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.fragment_splash_screen)
    }

    @SuppressLint("MissingPermission")
    private fun runPermissions() {
        val gps = checkGpsStatus()
        if (!gps) return

        requestPermissions()
    }

    private fun navigate() {
        val handle = Handler()
        handle.postDelayed({ toLogin() }, 2000)
    }

    private fun toLogin() {
        val intent = Intent(this, MainActivity::class.java)
        startActivity(intent)
        finish()
    }

    private fun requestPermissions() {
        TedPermission.with(this)
            .setPermissionListener(this)
            .setDeniedMessage("Se você rejeitar a permissão, não
poderá usar o SIGFIS\n\nPor favor, ative as permissões em
[Configurações] > [Permissões]")
            .setPermissions(
```

```
                Manifest.permission.ACCESS_FINE_LOCATION,
                Manifest.permission.WRITE_EXTERNAL_STORAGE,
                Manifest.permission.READ_EXTERNAL_STORAGE)
            .check();
    }

    private fun checkGpsStatus(): Boolean {
        val locationManager =
this.getSystemService(Context.LOCATION_SERVICE) as LocationManager
        val isGpsOn =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)

        if (!isGpsOn) {
            openGpsDisabledDialog()
        }

        return isGpsOn
    }

    private fun openGpsDisabledDialog() {
        if (!this::gpsDialog.isInitialized) {
            gpsDialog = AlertDialog.Builder(this)
        }

        gpsDialog
            .setTitle("GPS")
            .setMessage("O serviço de localização está desativado.
Deseja mudar a configuração?")
            .setPositiveButton("Configurações") { dialog, id ->
                dialog.dismiss()
                val intent =
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS)
                startActivity(intent)
            }
            .setNegativeButton("Fechar") { dialog, id ->
                finish()
            }
            .show()
    }

    override fun onPermissionResult(granted: Boolean) {
        if (granted) runPermissions()
        else finish()
    }

    override fun onExplanationNeeded(permissionsToExplain:
List<String>) {
        Toast.makeText(this,
R.string.user_location_permission_explanation, Toast.LENGTH_LONG)
            .show()
    }

    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<String>,
        grantResults: IntArray
    ) {
        permissionsManager.onRequestPermissionsResult(requestCode,
permissions, grantResults)
    }

```

```
115     override fun onResume() {
116         super.onResume()
117         runPermissions()
118     }
119 
120     override fun onPermissionGranted() {
121         navigate()
122     }
123 
124     override fun onPermissionDenied(deniedPermissions:
125 MutableList<String>?) {
126         finish()
127     }
128 }
```

APÊNDICE B: Código da Atividade da tela inicial com mapa do aplicativo SIGFIS

```
package com.lourivaldo.sigfis.ui.home

import android.annotation.SuppressLint
import android.content.Context
import android.graphics.Color
import android.graphics.RectF
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.core.content.ContextCompat
import androidx.fragment.app.Fragment
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProviders
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import com.lourivaldo.sigfis.R
import kotlinx.android.synthetic.main.fragment_home.*
import java.net.URI

class HomeFragment : Fragment(), OnMapReadyCallback,
MapView.OnDidFinishRenderingMapListener,
    MapboxMap.OnMoveListener {

    private lateinit var mapboxMap: MapboxMap
    private lateinit var homeViewModel: HomeViewModel

    private lateinit var mContext: Context
    private lateinit var adapterAlerts:
AlertItemRecyclerViewAdapter;
    private val items = ArrayList<AlertModel>()

    private val sourcesMap = mapOf(
        "geojson-source-fiscalizacao" to "5ff3d198-5c1c-4107-b57d-
8a9d6e26a8fe",
        "geojson-source-municipio" to "ddc3b31f-bae5-43b4-9795-
08a6e3f0079e",
        "geojson-source-hidrografia" to "43c574c3-d85d-43c0-a607-
2a443e6ae63c",
        "geojson-source-municipios_emergencia" to "b32426d2-ad1a-
40ad-9955-ca91ba97bcb8",
        "geojson-source-federal_rios" to "9ae50c01-5fe1-41ea-bd10-
6876d3abfd44"
    )

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        homeViewModel =

ViewModelProviders.of(this).get(HomeViewModel::class.java)
        val root = inflater.inflate(R.layout.fragment_home,
container, false)
```

```
59          val textView: TextView = root.findViewById(R.id.text_home)
60          homeViewModel.text.observe(viewLifecycleOwner, Observer {
61              textView.text = it
62          })
63          return root
64      }
65
66      override fun onViewCreated(view: View, savedInstanceState:
67  Bundle?) {
68          super.onViewCreated(view, savedInstanceState)
69          bottomNavigation.setOnNavigationItemSelectedListener { item
70  ->
71              when (item.itemId) {
72                  R.id.nav_agenda -> {
73                      findNavController().navigate(R.id.nav_agenda)
74                  }
75                  R.id.nav_map -> {
76                      findNavController().navigate(R.id.nav_map)
77                  }
78                  R.id.nav_inspections -> {
79
80  findNavController().navigate(R.id.nav_inspections)
81                  }
82              }
83              false
84          }
85          fabAdd.setOnClickListener {
86
87  findNavController().navigate(R.id.inspectionsCreateFragment)
88          }
89
90          mapView.onCreate(savedInstanceState)
91          mapView.addOnDidFinishRenderingMapListener(this)
92          mapView.getMapAsync(this)
93      }
94
95      private fun initRecyclerView() {
96          if (recyclerView.adapter == null) {
97              val adapter = AlertItemRecyclerViewAdapter(items,
98  context!!)
99              adapterAlerts = adapter
100             recyclerView.adapter = adapter
101             recyclerView.layoutManager =
102 LinearLayoutManager(mContext)
103         }
104     }
105
106     override fun onAttach(context: Context) {
107         super.onAttach(context)
108         mContext = context
109     }
110
111     override fun onResume() {
112         super.onResume()
113         (activity as AppCompatActivity?)!!.supportActionBar!!.title
114 = "SIGFIS"
115         (activity as
116 AppCompatActivity?)!!.supportActionBar!!.subtitle = "Fiscalização de
117 Recursos Hídricos"
118         mapView?.onResume()
119     }
```

```
120
121     override fun onMapReady(mapboxMap: MapboxMap) {
122         this.mapboxMap = mapboxMap
123
124         mapboxMap.addOnMoveListener(this)
125         mapboxMap.setStyle(Style.SATELLITE) {
126             enableLocationComponent(it)
127             loadLayers(it)
128             setupLayers(it)
129         }
130
131         selection_box.setOnClickListener {
132             // Execute a consulta de recurso dentro da selection_box
133             updateInformation()
134         }
135     }
136
137     private fun updateInformation()
138     {
139         initRecyclerView()
140
141         val features: List<Feature> =
142 mapboxMap.queryRenderedFeatures(box,
143 *sourcesMap.keys.toTypedArray())
144
145         if (features.isNotEmpty()) {
146             items.clear()
147             val newItems = ArrayList<AlertModel>()
148
149             for (i in features) {
150                 val props = i.properties()
151                 when(i.properties()?.get("type")?.asString) {
152                     "municipios_emergencia" -> {
153                         val title =
154 i.properties()?.get("title")?.asString ?: ""
155                         val name = if
156 (props?.get("name")?.isJsonNull == true) "" else
157 props?.get("name")?.asString
158                         newItems.add(AlertModel(1,  title, level =
159 AlertLevelEnum.VERY_HIGH, description = "$name"))
160                     }
161                     "federal_rios" -> {
162                         val title =
163 i.properties()?.get("title")?.asString ?: ""
164                         val name = if
165 (props?.get("name")?.isJsonNull == true) "" else
166 props?.get("name")?.asString
167                         newItems.add(AlertModel(1,  title, level =
168 AlertLevelEnum.VERY_HIGH, description = "$name"))
169                     }
170                     "federal_espelhos_dagua" -> {
171                         val title =
172 i.properties()?.get("title")?.asString ?: ""
173                         val name = if
174 (props?.get("name")?.isJsonNull == true) "" else
175 props?.get("name")?.asString
176                         newItems.add(AlertModel(1,  title, level =
177 AlertLevelEnum.VERY_HIGH, description = "$name"))
178                     }
179                     "federal_terras_indigenas_pe" -> {
180                         val title =
```

```
i.properties()?.get("title")?.asString ?: ""
                        val name = if
(props?.get("name")?.isJsonNull == true) "" else
props?.get("name")?.asString
                        newItems.add(AlertModel(2,  title, level =
AlertLevelEnum.HIGH, description = "$name"))
                    }
                    else -> {
                        val title =
i.properties()?.get("title")?.asString ?: ""
                        val name =
i.properties()?.get("name")?.asString ?: ""
                        newItems.add(AlertModel(3,  title, level =
AlertLevelEnum.MODERATE, description = name))
                    }
                }
            }

            val sortedList =
newItems.sortedWith(compareBy(AlertModel::id, AlertModel::level))
            items.addAll(sortedList)
            adapterAlerts.notifyDataSetChanged();
        }
    }

    private fun loadLayers(loadedMapStyle: Style) {
        for (source in sourcesMap) {
            val source = GeoJsonSource(source.key,
URI("http://10.0.0.185:5000/layers/" + source.value +".geojson"))
            loadedMapStyle.addSource(source)
        }
    }

    private fun setupLayers(loadedMapStyle: Style) {
        val layer = CircleLayer("geojson-source-fiscalizacao",
"geojson-source-fiscalizacao")
        layer.setSourceLayer("contour")
        layer.setProperties(
            PropertyFactory.circleRadius(3f),

PropertyFactory.circleColor(Color.parseColor("#ff0022")),
            PropertyFactory.circleOpacity(0.2f)
        )
        loadedMapStyle.addLayer(layer)

        val layer1 = LineLayer("geojson-source-hidrografia",
"geojson-source-hidrografia")
        layer1.setSourceLayer("contour")
        layer1.setProperties(

PropertyFactory.backgroundColor(Color.parseColor("#3bb2d0")),
            PropertyFactory.lineJoin(Property.LINE_JOIN_ROUND),
            PropertyFactory.lineCap(Property.LINE_CAP_ROUND),
            PropertyFactory.lineColor(Color.parseColor("#69beff")),
            PropertyFactory.lineWidth(1f)
        )
        loadedMapStyle.addLayer(layer1)

   }

    @SuppressLint("MissingPermission")
```

```
242     private fun enableLocationComponent(loadedMapStyle: Style) {
243         val customLocationComponentOptions =
244 LocationComponentOptions.builder(context!!)
245             .trackingGesturesManagement(true)
246             .accuracyColor(
247                 ContextCompat.getColor(
248                     activity!!.applicationContext,
249                     R.color.colorPrimary
250                 )
251             )
252             .build()
253 
254         val locationComponentActivationOptions =
255             LocationComponentActivationOptions.builder(
256                 activity!!.applicationContext,
257                 loadedMapStyle
258             )
259 
260 .locationComponentOptions(customLocationComponentOptions)
261                 .build()
262 
263         mapboxMap.locationComponent.apply {
264 
265 activateLocationComponent(locationComponentActivationOptions)
266             isLocationComponentEnabled = true
267             cameraMode = CameraMode.TRACKING
268             renderMode = RenderMode.NORMAL
269         }
270     }
271 
272     override fun onStart() {
273         super.onStart()
274         mapView?.onStart()
275     }
276 
277     override fun onPause() {
278         super.onPause()
279         mapView?.onPause()
280     }
281 
282     override fun onStop() {
283         super.onStop()
284         mapView?.onStop()
285     }
286 
287     override fun onSaveInstanceState(outState: Bundle) {
288         super.onSaveInstanceState(outState)
289         mapView?.onSaveInstanceState(outState)
290     }
291 
292     override fun onDestroy() {
293         super.onDestroy()
294         mapView?.onDestroy()
295     }
296 
297     override fun onLowMemory() {
298         super.onLowMemory()
299         mapView?.onLowMemory()
300     }
301 
302     override fun onDidFinishRenderingMap(fully: Boolean) {
```

```
303             updateInformation()
304         }
305
306         override fun onMoveBegin(detector: MoveGestureDetector) {
307             //
308         }
309
310         override fun onMove(detector: MoveGestureDetector) {
311             //
312         }
313
314         override fun onMoveEnd(detector: MoveGestureDetector) {
315             updateInformation()
316         }
317
318 }
```

APÊNDICE C: Arquivo xml do layout da Atividade da tela inicial

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:mapbox="http://schemas.android.com/apk/res-auto"
    android:id="@+id/coordinator"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="start"
        android:orientation="vertical">

        <LinearLayout
            android:id="@+id/llMain"
            android:layout_width="match_parent"
            android:layout_height="100dp">

            <FrameLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:background="#FFFFFF">

                <ImageView
                    android:id="@+id/iv_movie_poster1"
                    android:layout_width="match_parent"
                    android:layout_height="60dp"
                    android:contentDescription="@string/app_name"
                    android:scaleType="fitXY"
                    android:src="@drawable/background_curved" />

                <RelativeLayout
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center_horizontal"
                    android:gravity="center_horizontal">

                    <androidx.constraintlayout.widget.ConstraintLayout
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
                        android:layout_alignParentBottom="true"
                        android:layout_marginStart="26dp"
                        android:layout_marginEnd="26dp"
                        android:layout_marginBottom="26dp">

                        <ImageView
                            android:id="@+id/iv_movie_poster_back"
                            android:layout_width="75dp"
                            android:layout_height="75dp"
                            android:layout_alignParentBottom="true"

android:contentDescription="@string/app_name"
                            android:scaleType="fitXY"
                            android:src="@drawable/avatar_circle_loro"

```

```
app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintStart_toStartOf="parent"
                                app:layout_constraintTop_toTopOf="parent"
                                app:layout_constraintVertical_bias="0.0"
/>

                            <TextView
                                android:id="@+id/text_home"
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:layout_marginStart="8dp"
                                android:layout_marginEnd="8dp"
                                android:text="Lourivaldo Vasconcelos"
                                android:textAlignment="center"

android:textColor="@color/textLightPrimary"
                                android:textSize="20sp"

app:layout_constraintBottom_toBottomOf="parent"
                                app:layout_constraintEnd_toEndOf="parent"
                                app:layout_constraintHorizontal_bias="0.0"

app:layout_constraintStart_toEndOf="@+id/iv_movie_poster_back"
                                app:layout_constraintTop_toTopOf="parent"
                                app:layout_constraintVertical_bias="0.0"
/>

</androidx.constraintlayout.widget.ConstraintLayout>

                    </RelativeLayout>

                </FrameLayout>

            </LinearLayout>

        </LinearLayout>

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_gravity="bottom"
            android:paddingTop="100dp">

            <ScrollView
                android:layout_width="match_parent"
                android:layout_height="0dp"
                android:scrollbars="none"
                app:layout_constraintBottom_toTopOf="@+id/linearLayout6"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent">

                <androidx.recyclerview.widget.RecyclerView
                    android:id="@+id/recyclerView"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:clipToPadding="false"
                    android:paddingBottom="72dp"
                    app:layout_constraintBottom_toBottomOf="parent"
                    app:layout_constraintEnd_toEndOf="parent"
```

```
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />
        </ScrollView>

        <LinearLayout
            android:id="@+id/linearLayout6"
            android:layout_width="match_parent"
            android:layout_height="250dp"
            android:layout_gravity="bottom|fill_horizontal"
            android:paddingBottom="20dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent">

            <RelativeLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent">

                <com.mapbox.mapboxsdk.maps.MapView
                    android:id="@+id/mapView"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    mapbox:mapbox_cameraZoom="12" />

            </RelativeLayout>
        </LinearLayout>
    </androidx.constraintlayout.widget.ConstraintLayout>

    <com.google.android.material.bottomappbar.BottomAppBar
        android:id="@+id/bottomBar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        app:backgroundTint="?attr/colorPrimary"
        app:contentInsetEnd="0dp"
        app:contentInsetStart="0dp"
        app:fabAlignmentMode="end">

<com.google.android.material.bottomnavigation.BottomNavigationView
            android:id="@+id/bottomNavigation"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            app:backgroundTint="@android:color/transparent"
            app:elevation="0dp"
            android:layout_marginEnd="100dp"
            app:itemIconTint="@android:color/white"
            app:itemTextColor="@android:color/white"
            app:menu="@menu/home_menu" />

    </com.google.android.material.bottomappbar.BottomAppBar>

<com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fabAdd"
        android:src="@drawable/ic_add"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_anchor="@id/bottomBar" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

APÊNDICE D: Código de geração de geojson da API de integração

```
@Injectable()
export class LayersRepository {

  async getFiscalizacoes(): Promise<string> {
    const entityManager = getManager();
    const fileQuery = entityManager.query(`
SELECT json_build_object(
    'crs',  json_build_object(
        'type',       'name',
        'properties', json_build_object(
            'name', 'EPSG:4674'
        )
    ),
    'features', json_agg(
        json_build_object(
            'type',       'Feature',
            'id',         "id",
            'geometry',   ST_AsGeoJSON(geom)::json,
            'properties', json_build_object(
                'type', 'fiscalizacao',
                'title', 'Fiscalizações',
                'pf', "proc.fisc.",
                'dt', "dt_entrada",
                'pa', "proc_outor"
            )
        )
    )
)
FROM fiscalizacoes`,
      []);

    const result = await fileQuery;

    if (result && result[0] && result[0]['json_build_object']) {
      return result[0]['json_build_object'];
    }

    return null;
  }

  async getMunicipios(): Promise<string> {
    const entityManager = getManager();
    const fileQuery = entityManager.query(`
SELECT json_build_object(
    'crs',  json_build_object(
        'type',       'name',
        'properties', json_build_object(
            'name', 'EPSG:4674'
        )
    ),
    'features', json_agg(
        json_build_object(
            'type',       'Feature',
            'id',         "id",
```

```
            'geometry',   ST_AsGeoJSON(geom)::json,
            'properties', json_build_object(
                'type', 'municipio',
                'title', 'Municipios',
                'name', "nm_municip"
            )
        )
    )
)
FROM municipios`,
      []);

    const result = await fileQuery;

    if (result && result[0] && result[0]['json_build_object']) {
      return result[0]['json_build_object'];
    }

    return null;
  }

  async getHidrografia(): Promise<string> {
    const entityManager = getManager();
    const fileQuery = entityManager.query(`
SELECT json_build_object(
    'crs',  json_build_object(
        'type',       'name',
        'properties', json_build_object(
            'name', 'EPSG:4674'
        )
    ),
    'features', json_agg(
        json_build_object(
            'type',       'Feature',
            'id',         "id",
            'geometry',   ST_AsGeoJSON(geom)::json,
            'properties', json_build_object(
                'type', 'hidrografia',
                'title', 'Hidrografia',
                'name', "name",
                'rio', "rioscsv_ba"
            )
        )
    )
)
FROM hidrografia`,
      []);

    const result = await fileQuery;

    if (result && result[0] && result[0]['json_build_object']) {
      return result[0]['json_build_object'];
    }

    return null;
  }

```

```
  async getMunicipiosEmergencia(): Promise<string> {
    const entityManager = getManager();
    const fileQuery = entityManager.query(`
SELECT json_build_object(
    'crs',  json_build_object(
        'type',       'name',
        'properties', json_build_object(
            'name', 'EPSG:4674'
        )
    ),
    'features', json_agg(
        json_build_object(
            'type',       'Feature',
            'id',         "id",
            'geometry',   ST_AsGeoJSON(geom)::json,
            'properties', json_build_object(
                'type', 'municipios_emergencia',
                'title', 'Municipios Emergencia',
                'name', "NM_MUNICIP"
            )
        )
    )
)
FROM municipios_emergencia`,
      []);

    const result = await fileQuery;

    if (result && result[0] && result[0]['json_build_object']) {
      return result[0]['json_build_object'];
    }

    return null;
  }

  async getFederalRios(): Promise<string> {
    const entityManager = getManager();
    const fileQuery = entityManager.query(`
SELECT json_build_object(
    'crs',  json_build_object(
        'type',       'name',
        'properties', json_build_object(
            'name', 'EPSG:4674'
        )
    ),
    'features', json_agg(
        json_build_object(
            'type',       'Feature',
            'id',         "id",
            'geometry',   ST_AsGeoJSON(geom)::json,
            'properties', json_build_object(
                'type', 'federal_rios',
                'title', 'Federal Rios',
                'name', "NORIOORIGI"
            )
        )
    )
```

```
)
FROM federal_rios`,
      []);

    const result = await fileQuery;

    if (result && result[0] && result[0]['json_build_object']) {
      return result[0]['json_build_object'];
    }

    return null;
  }

  async getFederalEspelhosDagua(): Promise<string> {
    const entityManager = getManager();
    const fileQuery = entityManager.query(`
SELECT json_build_object(
    'crs',  json_build_object(
        'type',       'name',
        'properties', json_build_object(
            'name', 'EPSG:4674'
        )
    ),
    'features', json_agg(
        json_build_object(
            'type',       'Feature',
            'id',         "id",
            'geometry',   ST_AsGeoJSON(geom)::json,
            'properties', json_build_object(
                'type', 'federal_espelhos_dagua',
                'title', 'Federal Espelhos D Agua',
                'name', "NOME_RESER"
            )
        )
    )
)
FROM federal_espelhos_dagua`,
      []);

    const result = await fileQuery;

    if (result && result[0] && result[0]['json_build_object']) {
      return result[0]['json_build_object'];
    }

    return null;
  }

  async getBacias(): Promise<string> {
    const entityManager = getManager();
    const fileQuery = entityManager.query(`
SELECT json_build_object(
    'crs',  json_build_object(
        'type',       'name',
        'properties', json_build_object(
            'name', 'EPSG:4674'
        )
```

```
    ),
    'features', json_agg(
        json_build_object(
            'type',       'Feature',
            'id',         "id",
            'geometry',   ST_AsGeoJSON(geom)::json,
            'properties', json_build_object(
                'type', 'bacias',
                'title', 'Bacias',
                'name', "Name"
            )
        )
    )
)
FROM bacias`,
      []);

    const result = await fileQuery;

    if (result && result[0] && result[0]['json_build_object']) {
      return result[0]['json_build_object'];
    }

    return null;
  }

  async getFederalTerrasIndigenasPe(): Promise<string> {
    const entityManager = getManager();
    const fileQuery = entityManager.query(`
SELECT json_build_object(
    'crs',  json_build_object(
        'type',       'name',
        'properties', json_build_object(
            'name', 'EPSG:4674'
        )
    ),
    'features', json_agg(
        json_build_object(
            'type',       'Feature',
            'id',         "id",
            'geometry',   ST_AsGeoJSON(geom)::json,
            'properties', json_build_object(
                'type', 'federal_terras_indigenas_pe',
                'title', 'Federal Terras Indigenas',
                'name', "terrai_nom"
            )
        )
    )
)
FROM federal_terras_indigenas_pe`,
      []);

    const result = await fileQuery;

    if (result && result[0] && result[0]['json_build_object']) {
      return result[0]['json_build_object'];
    }
```

```
303
304       return null;
305     }
306
307     async getUcProtecaoIntegral(): Promise<string> {
308       const entityManager = getManager();
309       const fileQuery = entityManager.query(`
310 SELECT json_build_object(
311     'crs',  json_build_object(
312         'type',       'name',
313         'properties', json_build_object(
314             'name', 'EPSG:4674'
315         )
316     ),
317     'features', json_agg(
318         json_build_object(
319             'type',       'Feature',
320             'id',         "id",
321             'geometry',   ST_AsGeoJSON(geom)::json,
322             'properties', json_build_object(
323                 'type', 'uc_protecao_integral',
324                 'title', 'UC Protecao Integral',
325                 'name', "Name"
326             )
327         )
328     )
329 )
330 FROM uc_protecao_integral`,
331       []);
332
333       const result = await fileQuery;
334
335       if (result && result[0] && result[0]['json_build_object']) {
336         return result[0]['json_build_object'];
337       }
338
339       return null;
340     }
341 }
342
343
344
```