CAPÍTULO I

1. INTRODUÇÃO

É notório a necessidade de subsídios estatais para quem reside na zona rural como meio de trabalho e também como opção de moradia. Devido a distância ou difícil acesso/localização a essas regiões, essas famílias podem ter que lidar com a falta celeridade nos atendimentos que deveriam ser diligentes. E essa demora pode ocasionar em uma desventura maior, como a família perder seu meio de renda, sua casa e até mesmo custar a vida de alguém.

Segundo estudos da CSP - Caderno de Saúde Pública, o acesso a saúde é menor nas áreas rurais em função da maior vulnerabilidade social de sua população e das maiores dificuldades de acesso que seus grupos sociais estão submetidos.(ARTIGO, Cad. Saúde Pública 2018).

Intentando as problemáticas apresentadas se faz necessário a criação de ferramentas que atendam as necessidades da população rural. A utilização de mídias sociais como “Whatsapp”, por meio de criação de grupos de comunicação entre os moradores foi aplicada com tal população. Contudo, sua efetividade não teve como resolução as carências presentes.

Em vista deste hiato nas necessidades apresentadas desenvolve-se um projeto com maior exatidão para resolutiva das demandas corriqueiras presentes da zona rural e também mais facilidade ao Estado de poder ter as ferramentas necessárias para promover segurança e saúde à sua população de maneira geral e mais abrangente.

O sistema tem como foco o envio da localização atual do usuário a partir do chamado feito via celular smartphone, o qual facilita a chegada dos profissionais de emergência solicitados. Nele poderá cadastrar dados pertinentes do usuário, como tipo sanguíneo, alergias e até mesmo contatos próximos de emergência. Para que tenha facilidade e maior celeridade no atendimento.

2. PROBLEMÁTICA

Notícias apontam que em casos de necessidades médicas, como o SAMU(Serviço de Atendimento Móvel de Urgência), a demora nos atendimentos em zonas rurais é comum devido a dificuldade de encontrar o endereço na região.

Funcionários desses serviços (condutor de ambulância, por exemplo) alegam que muitos chamados em zonas rurais geram preocupação antes mesmo de saber o fato ocorrido, pois atendimentos nessas áreas são dificultados pela indisponibilidade de comunicação e sinalização para saber o local correto de chegada. Os mesmo funcionários expressam em entrevistas que tempo é vida!. (GLOBO, 2017).

Outro ponto importante é a segurança na questão do policiamento ostensivo nessas áreas, onde muitas vezes é um local carente de rondas e por isso ocorrem muitos roubos e assaltos a residências, fazendas e chácaras.

Aplastados dessas situações e casos já confirmados de roubos, assaltos e invasões a residências, os moradores dessas regiões afirmam que pretendem se mudar para a zona urbana. Segundo algumas famílias, não há tranquilidade mais em ter uma propriedade no campo. (G1, 2016).

Dada as notícias acima como um exemplo da necessidade de serviço essencial, percebe-se demanda em frente a segurança da população rural e ao atendimento emergencial na região. A celeridade na resposta é indispensável para uma amplitude da qualidade de vida nesta região..

Os serviços emergenciais fornecidos até então pelo próprio governo ou por empresas privadas (muitas vezes financiados pelas empresas contratantes de funcionário rurais) tem sua aplicabilidade nos serviços presentes, contudo indubitavelmente podem ser melhorados se visados de forma mais abrangente.

Em suma dos argumentos apresentados se faz necessário o seguinte questionamento: Como pode-se dar maior celeridade nas respostas de atendimento de emergência para moradores de zonas rurais?

3. OBJETIVOS

Objetivo é o propósito de realizar algo, é aonde se quer chegar. É ele que fornece a direção do que se deseja e deve fazer, e serve como guia para que o sonho seja finalmente realizado. Tratando-se de projeto, se divide em duas partes: objetivo geral e objetivos específicos. (MARQUES, 2018)

3.1 Objetivo Geral

O trabalho tem como propósito desenvolver uma aplicação com o intuito de gerenciar e otimizar chamados de emergência feitos pela população.

3.2 Objetivos Específicos

- Integrar chamados feitos via COPOM para melhor gerenciamento.
- Utilizar GPS (API Google) para fornecer localização do usuário.
- Criar plataforma que permita conversação entre COPOM e usuário (chat).
- Utilizar servidor web para sincronizar dados em tempo real.
- Aplicar obrigatoriedade de CPF para que consiga efetuar chamados.

4. JUSTIFICATIVA

Vivenciado as notícias de jornais quanto a acidentes, roubos, e dificuldades vividas nos serviços emergenciais, a solução que poderia dar cabo dessas demandas de gravidade podem ser enxergadas através de um sistema onde tenha integração do usuário enviando a localização atual e exata junto ao serviço que ele deseja solicitar.

Sobre as dificuldades de localização, médicos afirmam e concordam com os empecilhos encontrados, e os mesmos alegam: "Tem lugar que a gente demora 40 minutos para chegar. É fundamental a localização, o endereço correto e o número correto". (EPTV - 2017).

Tendo em vista a necessidade de uma plataforma que agrega e permite melhor gestão nesses casos, pretende-se criar um aplicativo com o objetivo de assistir e auxiliar nos atendimentos emergenciais, com foco em zonas rurais, enviando a localização do usuário logo após a confirmação, utilizando API do Google e também abrindo uma plataforma que possibilita conversação entre o usuário e o serviço, plataforma essa que pode ser criada utilizando WebSocket. Informações pertinentes a um melhor atendimento do usuário (como tipo sanguíneo, alergias) ficam guardadas em banco após o cadastro, e são utilizadas quando confirmado o chamado. O trabalho tem o foco, portanto, de otimizar atendimentos em zonas de difícil acesso/localização.

CAPÍTULO II

5. METODOLOGIA E PESQUISA

Para Fonseca (2002), methodos significa organização, e logos, estudo sistemático, pesquisa, investigação; ou seja, metodologia é o estudo da organização, dos caminhos a serem percorridos, para se realizar uma pesquisa ou estudo, ou para se fazer ciência.

Pesquisa é o procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos. A pesquisa desenvolve-se por um processo constituído de várias fases, desde a formulação do problema até a apresentação e discussão dos resultados (Gil, 2007, p.17).

5.1 Técnicas de Levantamento de Requisitos

Em todo desenvolvimento de software, um aspecto fundamental é a captura dos requisitos dos usuários. Para apoiar este trabalho, diversas técnicas podem ser utilizadas. (KENDALL, 1992).

Requisito é uma condição ou capacidade que deve ser alcançada. É algo que um sistema ou componente deve possuir para satisfazer um contrato, padrão ou especificação.

Na fase de levantamento de requisitos, os profissionais delegados para a construção do software salientam o negócio que o sistema deve automatizar. Basicamente, é entender o que o cliente deseja ou o que ele pensa ser importante, além das regras e processos de negócio. O objetivo principal do levantamento de requisitos é que o usuário e os desenvolvedores tenham a mesma visão do problema a ser resolvido.(BEZERRA, 2007).

No processo de elicitação de requisitos, além da tecnologia, tem que ser levado em consideração o ambiente, contexto social e classe social do indivíduo. Com isso é possível a validação da necessidade do uso de ferramentas diversas para captar e sanar os problemas da aquisição de requisitos.

Existem várias técnicas de levantamento de requisitos que podem ser usadas, incluindo entrevistas, cenários, métodos soft systems, prototipagem e observação dos participantes.

Para melhor desempenho na aplicação do projeto, as seguintes técnicas de Levantamento de Requisitos foram escolhidas: Entrevista e prototipação.

5.1.1 Entrevista

A entrevista é uma das técnicas mais utilizadas para levantamento de requisitos, pois é fácil de utilizar e produz bons resultado na fase inicial de obtenção de dados. É interessante que o entrevistado tenha seu espaço para expor suas ideias e que o

entrevistador tenha uma plano a ser seguido para evitar dispersão da ideia principal.

Segundo Gerald Kotonya (1998), há dois tipos de entrevista: a) entrevistas fechadas onde o engenheiro de requisitos procura as perguntas para um conjunto um pré-definido de questões; b) entrevistas abertas onde não há agenda pré-definida e o engenheiro de requisitos discute, de modo aberto, o que os usuários querem do sistema.

Seguindo a ideia, para obter melhores resultados em entrevistas com os usuários, as seguintes condutas podem ser adotadas: abstrair um plano abrangente para a entrevista, verificar disponibilidade de diálogo com o usuário, planejar as questões a serem feitas para ter melhor uso do tempo evitando cansaço desnecessário que poderia afetar no resultado, utilizar de ferramentas automatizadas, focar no desejo que o usuário mais se interessa e adequar ao estilo de entrevista mais apropriado. Para um planejamento efetivo, devem ser angariados e compreendidos dados pertinentes a discussão antes da entrevista (formulários, relatórios e documentos). Assim o profissional responsável por analisar e levantar essas informações terá uma base mais fundamentada para obter resultados legítimos. Ressaltando que é importante foco na entrevista para que a duração da reunião não se estenda muito, pois o eixo e a concentração vão se dispersando conforme o cansaço do usuário se eleva.

Após definir a melhor técnica a ser implementada, é importante verificar as vantagens do método. No caso da entrevista além de ser uma técnica flexível, podemos angariar os dados estando mais próximo do usuário, e isso permite a obtenção de informações com mais propriedade. E também aproxima o engenheiro de requisitos do usuário do sistema, fazendo com que o usuário se sinta atuante do desenvolvimento do sistema. (MARTINS, 2001).

Baseado nas alegações acima, foi escolhida a entrevista para ser aplicada ao projeto. Para o aplicativo do SOS Rural, o analista investigou e apurou dados e informações obtidos através de notícias de jornais e relatos de moradores e famílias das regiões rurais do INCRA e Brazlândia situadas no Distrito Federal.

A reunião/entrevista com os habitantes residentes das zonas rurais foi realizada de maneira aberta e progrediu com tranquilidade. O analista se fixou a perguntas previamente selecionadas mas não se limitou somente a elas e se atentou aos detalhes das informações obtidas quanto à rondas policiais, velocidade de atendimento, chegada de emergência nessas regiões e relatos anteriores de fatos já consumados tanto de moradores quanto de profissionais que atendem os chamados em campo. A conversa seguiu um fluxo aberto apesar de também serem usadas as perguntas pré selecionadas. Com isso, houve as indagações quanto possíveis soluções que poderiam ser implementadas posteriormente no projeto.

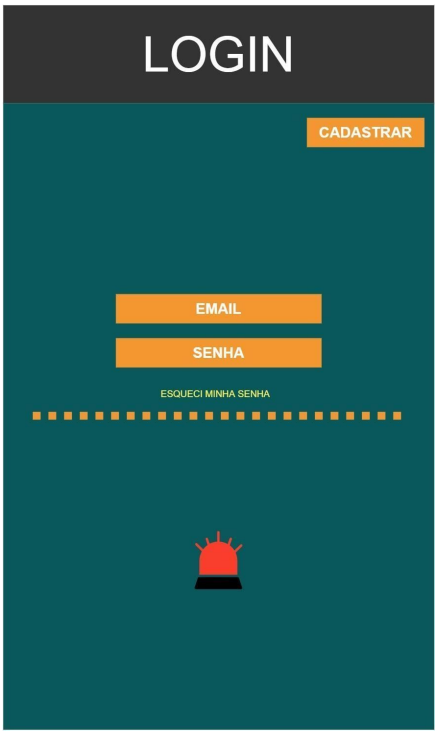
5.1.2 Prototipação

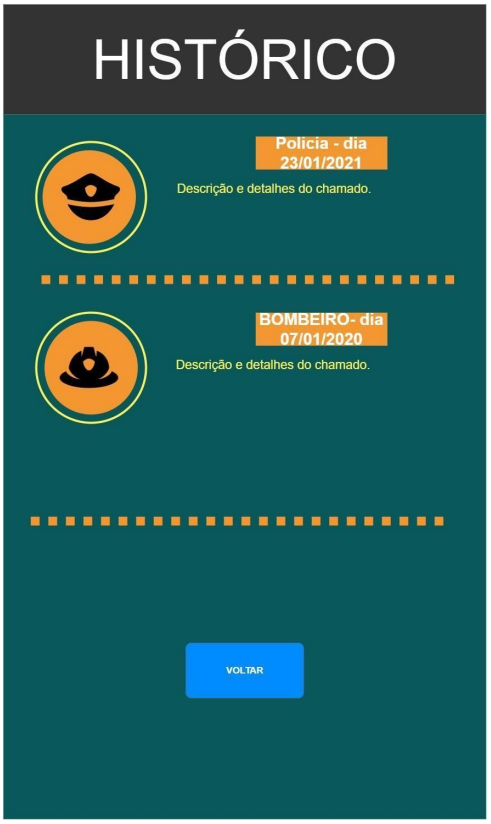
A técnica da prototipação foi escolhida com o interesse maior em validar os requisitos levantados através da ferramentas de entrevista.

A prototipação é uma técnica valiosa para se obter rapidamente informações específicas sobre requisitos de informação do usuário. Tipicamente, a prototipação permite obter: reações iniciais do usuário, sugestões do usuário para refinar, inovações e informações para revisão de planos. Nessa técnica são criados de forma rápida um conjunto de funcionalidades do produto, é uma versão inicial do sistema de software usado para demonstrar conceitos, experimentar opções de projeto, conhecer mais sobre o problema e suas possíveis soluções. baseado nos requisitos dispostos. Usa-se, normalmente,uma metodologia de desenvolvimento rápido e iterativo para que os clientes e/ou usuários possam usar o sistema o mais cedo possível (SOMMERVILLE, 2007)

Ainda segundo Ian Sommerville (2007), os protótipos permitem que os usuários experimentem o software, podendo eles, usuários, verificar em como o protótipo auxilia no seu trabalho. Além disso, durante o seu uso, podem revelar erros e omissões nos requisitos propostos, podendo os usuários apresentar novos requisitos para o sistema.

Considerando as informações acima e buscando contribuir com o entendimento dos processo envolvidos na aplicação, a técnica de prototipação foi atribuída para que os usuários pudessem elucidar um limitado número de funcionalidades do projeto. Algumas telas foram disponibilizadas por meio da ferramenta Draw.Io.





.

.

CAPÍTULO III

6. MODELO DE DESENVOLVIMENTO

6.1 Scrum

Scrum é uma metodologia ágil para gestão e planejamentos de projetos de software. É uma abordagem enxuta para gerenciar o desenvolvimento de software e é uma forma de organizar os negócios para que toda perda nos processos seja eliminada ou pelo menos fortemente reduzida. Scrum é um framework estrutural que está sendo usado para gerenciar o desenvolvimento de produtos complexos desde o início de 1990. Scrum não é um processo ou uma técnica para construir produtos; em vez disso, é um framework dentro do qual você pode empregar vários processos ou técnicas. O Scrum deixa claro a eficácia relativa das práticas de gerenciamento e desenvolvimento de produtos, de modo que você possa melhorá-las. (SCHAWBER, 2013).

O framework Scrum consiste nos times do Scrum associadas a papéis, eventos, artefatos e regras. Cada componente dentro do framework serve a um propósito específico e é essencial para o uso e sucesso do Scrum.

O Scrum é fundamentado no empirismo, que afirma a obtenção de conhecimento vem da experiência e de tomadas de decisões baseadas no que é conhecido. Ele emprega uma abordagem iterativa e incremental para aperfeiçoar e a previsibilidade e o controle de riscos. Possui 03(três) pilares principais: transparência, inspeção e adaptação. (SUTHERLAND, 2013).

Transparência: pontos significativos do processo devem ser padronizados para o entendimento de quem é responsável pelos resultados. Ex: Uma linguagem comum referindo-se ao processo deve ser compartilhada por todos os participantes.

Inspeção: deve haver uma frequência na inspeção de artefatos e progresso para detectar variações. Porém, apesar da frequência, é interessante que seja estabelecido pontos para que não atrapalhe na execução das tarefas.

Adaptação: caso o inspetor tenha verificado uma variação que prejudique o resultado final e o torne inaceitável, devem ser feitas alterações e ajustes em breve para minimizar futuros desvios.

6.1.1 Time Scrum

O Time Scrum é composto pelo Product Owner, o Time de Desenvolvimento e o Scrum Master. Eles mesmos organizam a melhor forma para completarem as tarefas, em vez de serem dirigidos por alguém de fora. Times funcionais possuem total competência para não precisarem de times que não fazem parte da equipe.

O Product Owner é o responsável por maximizar o valor do produto e do trabalho do time de desenvolvimento. Ele também é responsável por gerenciar o Backlog do produto expressando claramente os itens, ordenando os itens para melhor alcançar metas, garantir o valor do trabalho realizado e entendimento dos itens pelo time de desenvolvimento. Para que o Product Owner tenha sucesso, toda a organização deve respeitar as suas decisões. As decisões do Product Owner são visíveis no conteúdo e na priorização do Backlog do Produto. Ninguém tem permissão para falar com o time de desenvolvimento sobre diferentes configurações de prioridade, e o time de desenvolvimento não tem permissão para agir sobre o que outras pessoas disserem.

O time de desenvolvimento são os profissionais responsáveis pela entrega de uma versão que potencialmente incrementa o produto “Pronto” ao final de cada sprint. Algumas características do time de desenvolvimento são: auto-organizados, multifuncionais (possuem todas as habilidades necessárias, enquanto equipe, para criar o incremento do produto), todos possuem o título de “desenvolvedor”, e não possuem sub-times tais como teste ou análise de

negócios.

O Scrum Master é o responsável para que o Scrum seja entendido e aplicado corretamente. Ele é um servo-líder para o Time Scrum. O Scrum Master ajuda aqueles que estão fora do Time Scrum a entender quais as suas interações com o Time Scrum são úteis e quais não são. O Scrum Master ajuda todos a mudarem estas interações para maximizar o valor criado pelo Time Scrum.

Scrum Master com Product Owner: Encontra técnicas para o gerenciamento efetivo do Backlog do Produto; Comunica a visão, objetivos e itens do Backlog para o time de desenvolvimento; Ensinar a Time Scrum a criar itens de Backlog do produto de forma clara e concisa;

Scrum Master com Time de Desenvolvimento: Treinar o Time de Desenvolvimento em autogerenciamento e interdisciplinaridade; Ensinar e liderar o Time de Desenvolvimento na criação de produtos de alto valor; Facilitar os eventos Scrum conforme exigidos ou necessários; Treinar o Time de Desenvolvimento em ambientes organizacionais nos quais o Scrum não é totalmente adotado e compreendido.

Scrum Master com a Organização (Empresa): Planejando implementações Scrum dentro da organização; Ajudando funcionários e partes interessadas a compreender e tornar aplicável o Scrum e o desenvolvimento de produto empírico; Trabalhando com outros Scrum Masters para aumentar a eficácia da aplicação do Scrum nas organizações.

6.1.2 Eventos Scrum

Eventos prescritos são usados no Scrum para criar uma rotina e minimizar a necessidade de reuniões não definidas no Scrum. Todos os eventos são eventos time-boxed, de tal modo que todo evento tem uma duração máxima. Uma vez que a Sprint começa, sua duração é fixada e não pode ser reduzida ou aumentada. Os eventos restantes podem terminar sempre que o propósito do evento é alcançado, garantindo que uma quantidade adequada de tempo seja gasta sem permitir perdas no processo. Além da Sprint, que é um container para outros eventos, cada evento no Scrum é uma oportunidade de inspecionar e adaptar alguma coisa. Estes eventos são especificamente projetados para permitir uma transparência e inspeção criteriosa. A não inclusão de qualquer um dos eventos resultará na redução da transparência e da perda de oportunidade para inspecionar e adaptar. (SCHAWBER, 2013)

O Scrum determina 04 Eventos Formais para inspeção e adaptação: reunião de planejamento da Sprint, reunião diária, reunião de revisão da Sprint e retrospectiva da Sprint.

Sprint é o coração do Scrum. Um time-boxed de um mês ou menos, durante o qual um "Pronto", versão incremental potencialmente utilizável do produto, é criado. Sprints têm durações coerentes em todo o esforço de desenvolvimento. Uma nova Sprint inicia imediatamente após a conclusão da Sprint anterior.

Cada Sprint pode ser considerada um projeto com horizonte não maior que um mês. Como os projetos, as Sprints são utilizadas para realizar algo. Cada Sprint tem a definição do que é para ser construído, um plano projetado e flexível que irá guiar a construção, o trabalho e o resultado do produto. Sprints são limitadas a um mês corrido. Quando o horizonte da Sprint é muito longo, a definição do que será construído pode mudar, a complexidade pode aumentar e o risco pode crescer. Sprints permitem previsibilidade que garante a inspeção e adaptação do progresso em direção à meta pelo menos a cada mês corrido. Sprints também limitam o risco ao custo de um mês corrido. (SUTHERLAND, 2013).

Reunião de planejamento da Sprint: O trabalho a ser efetuado numa Sprint é definido nesta reunião com todo o Time Scrum. Possuem duração máxima de 8 horas para uma Sprint de 1 mês. Para Sprints menores, este evento é usualmente menor.

Algumas questões levantadas nessa reunião são: O que pode ser entregue como resultado do incremento da próxima Sprint?; Como o trabalho necessário para entregar o incremento será realizado?

Reunião Diária: Ocorre normalmente em 15 minutos e possui o objetivo de sincronizar o trabalho a ser feito com o time de desenvolvimento e inspecionar o trabalho desde a última reunião.

Algumas questões levantadas nessa reunião são: O que eu fiz ontem que ajudou o Time de Desenvolvimento a atender a meta da Sprint?; O que eu farei hoje para ajudar o Time de Desenvolvimento atender a meta da Sprint?; Eu vejo algum obstáculo que impeça a mim ou o Time de Desenvolvimento no atendimento da meta da Sprint?

Revisão da Sprint: Ocorre ao final da sprint com o intuito de inspecionar o incremento e adaptar o Backlog caso necessário. Esta é uma reunião informal, não uma reunião de status, e a apresentação do incremento destina-se a motivar e obter comentários e promover a colaboração. Possui normalmente a duração de 4 horas, podendo ser menor em projetos menores.

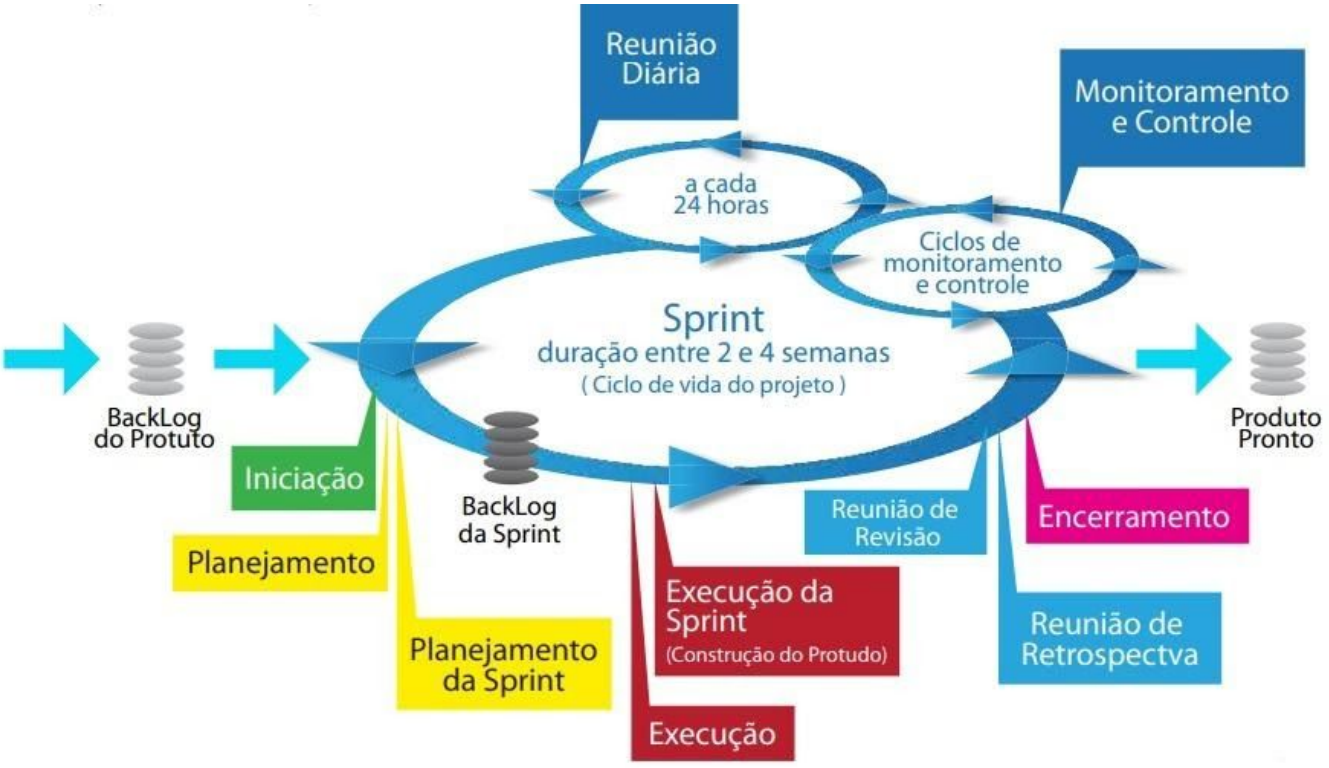
Algumas abordagens na revisão são: Os participantes incluem o Time Scrum e os Stakeholders chaves convidados pelo Product Owner; O Product Owner esclarece quais itens do Backlog do Produto foram “Prontos” e quais não foram “Prontos”; O Time de Desenvolvimento demonstra o trabalho que está “Pronto” e responde as questões sobre o incremento; O grupo todo colabora sobre o que fazer a seguir, e é assim que a Reunião de Revisão da Sprint fornece valiosas entradas para a Reunião de Planejamento da próxima Sprint;

Retrospectiva da Sprint: O Time Scrum inspeciona a si próprio e cria planos para

melhorarem as próximas Sprints. Ocorre depois da Revisão da Sprint e antes da reunião de planejamento da próxima Sprint. Esta é uma reunião time-boxed de 3 horas para uma Sprint de um mês. Para Sprint menores, este evento é usualmente menor.

Tem como propósito: Inspecionar como a última Sprint foi em relação às pessoas, aos relacionamentos, aos processos e às ferramentas; Criar um plano para implementar melhorias no modo que o Time Scrum faz seu trabalho;

A FIGURA 1 abaixo apresenta o modelo Scrum. A figura ilustra como as abordagens se encaixam de uma maneira natural, conectando-se perfeitamente com o objetivo de união dos pontos fortes com o intuito de diminuir as fraquezas ou limitações.



FONTE: GERENCIAMENTO ÁGIL DE PROJETOS COM SCRUM + PMBOK

(projectbuilder.com.br/Downloads/ebook-gratuito-scrum-pmbok.pdf)

6.1.3 Artefatos Scrum

Os artefatos do Scrum representam o trabalho ou o valor para o fornecimento de transparência e oportunidades para inspeção e adaptação. Os artefatos definidos para o Scrum são especificamente projetados para maximizar a transparência das informações chave de modo que todos tenham o mesmo entendimento dos artefatos. (SCHAWBER, 2013)

Backlog do Produto: É uma lista ordenada com todas as necessidades do produto. O responsável pelo seu conteúdo, disponibilidade e ordenação é o Product Owner.

Pode ser dito que o Backlog do produto nunca está completo. Ele evolui tanto quanto o produto e o ambiente no qual ele será inserido. O Backlog é dinâmico e pode constantemente mudar para identificar o que o produto necessita para ser mais apropriado, competitivo e útil. O Backlog do Produto existirá enquanto o produto também existir.

Ele lista todas as características, funções, requisitos, melhorias e correções que formam as mudanças que devem ser feitas no produto nas futuras versões. Os itens do Backlog do Produto possuem os atributos de descrição, ordem, estimativa e valor. Quanto mais utilizado o produto, maior e mais completo fica Backlog. Requisitos nunca param de mudar, então o Backlog do Produto é um artefato vivo.

Backlog da Sprint: O Backlog da Sprint é um conjunto de itens do Backlog do Produto selecionados para a Sprint, juntamente com o plano para entregar o incremento do produto e atingir o objetivo da Sprint. O Backlog da Sprint é um plano com detalhes suficientes que as mudanças no progresso sejam entendidas durante a Reunião Diária. O Time de Desenvolvimento modifica o Backlog da Sprint ao longo de toda a Sprint, e o Backlog da Sprint vai surgindo durante a Sprint.

6.2 Aplicação do Scrum

Após análise e observação dos processos do Scrum, foi preferível por utilizar algumas práticas pertencentes a essa metodologia, além dos três pilares básicos considerados por Jeff Sutherland e Ken Schwaber (2017). Estão listadas a seguir como cada prática escolhida foi aplicada no projeto:

- TIME SCRUM

O modelo de equipes no Scrum foi projetado para dar mais flexibilidade, engenhosidade e produtividade. O time Scrum provou ser cada vez mais eficaz para qualquer trabalho complexo.

Os times Scrum fazem entregas iterativas e incrementais, isso possibilita melhores análises e maior feedback. As entregas incrementais do produto, quando finalizadas, garantem que uma versão potencialmente útil do produto funcionando esteja sempre disponível.

Com auxílio de outros profissionais teríamos esse time dividido nas classes sugeridas de Product Owner, Scrum Master e Time de Desenvolvimento. Sendo 1 responsável, 2

profissionais e no mínimo 5 profissionais, respectivamente.

- EVENTOS SCRUM

Intentando estabelecer um padrão de encontros diários, semanais e mensais, foi escolhido também por programar eventos/reuniões. Cada uma com um propósito específico para destrinchar o projeto e seu andamento, assim, evitando falhas e dissipação desnecessária de tempo.

- SPRINT

Sendo o coração do Scrum, as Sprints são fundamentais para melhor detalhamento do produto. São períodos de tempo onde o próprio time scrum que define, mas foram escolhidos períodos com padrão sendo de 1 mês (podendo ser alterado conforme entendimento da equipe do que vai ser entregue e do trabalho/esforço necessário para isso).

O Product Owner deverá abstrair as funcionalidades coletadas no Backlog que deverão ser entregues naquele espaço de tempo, e o time scrum todo deve trabalhar para que isso seja desenvolvido. Todo o time scrum deve trabalhar para o entendimento coletivo daquela Sprint.

A equipe de desenvolvimento sendo auto-organizada quanto a realização do trabalho, define o tempo e o esforço necessário para produzir as funcionalidades delegadas. As dúvidas quanto os itens selecionados podem ser esclarecidas com o Product Owner e os itens a serem realizados podem ser trocados dependendo da decisão do time de desenvolvimento junto com o Product Owner.

- REVISÃO DE SPRINT

Com o intuito de examinar e controlar a entrega e adaptar o Backlog do Produto, devem ocorrer, ao final da Sprint, uma revisão do que foi feito e produzido naquele período de tempo. É uma reunião informal e com duração de 3 horas quando Sprints são de 1 mês.

Nessa revisão os itens do Backlog que foram implementados são verificados e definidos quais foram concluídos e quais estão pendentes, ocorrem discussões e tiradas de dúvidas com o time de desenvolvimento e elucidados os itens a serem feitos e datas-alvo a serem marcadas.

- DAILY SCRUM

Nessa reunião que ocorre diariamente, o objetivo é definir os pontos a serem realizados nas próximas 24h, ou seja, até o próximo encontro. Também é falado o que a equipe conseguiu atingir no encontro passado e previsões rápidas para os trabalhos futuros.

Essas reuniões possuem normalmente 15 minutos de duração.

A prática Daily Scrum otimiza os resultados obtidos pois conta com maior colaboração e desempenho da equipe devido a responsabilidade e entendimento quanto a sua auto-organização. É uma prática importante pois aumenta a probabilidade da equipe atingir o objetivo da Sprint.

O Scrum Master garante que a equipe de desenvolvimento tenha a reunião, mas a equipe de desenvolvimento é responsável pela condução do Daily Scrum. O Scrum Master ensina a Equipe de Desenvolvimento a manter o Daily Scrum dentro do prazo de 15 minutos.

6.3 Tecnologias em Geral

Nesta seção são exibidos os conhecimentos e métodos utilizados para o desenvolvimento da aplicação, como a programação utilizada, sistema operacional e base de dados.

6.3.1 Android

O mercado de celulares está maior do que nunca. Praticamente todos no mundo possuem um celular. Estudos apontam que hoje em dia 5,1 bilhões de pessoas possuem algum aparelho celular, e isso bem mais da metade da população mundial. (VALENTE, 2019)

As pessoas estão focadas em seus aparelhos constantemente, com isso a necessidade de sistemas online, sincronizados com informações de servidores confiáveis, produtos eletrônicos como smartphones, tablets, notebooks aumenta. Nesse nicho de mercado a disputa é crescente quanto a acompanhar as evoluções constantes.

O Android é a resposta da Google para ocupar esse mercado. É uma plataforma para tecnologia móvel completa, envolvendo um pacote com programas para celulares, já com um sistema operacional, middleware, aplicativos e interface de usuário. O Android foi construído com a intenção de permitir aos desenvolvedores criar aplicações móveis que possam tirar total proveito do que um aparelho portátil possa oferecer. (PEREIRA, SILVA, 2009)

Sendo construído para ser verdadeiramente open source, pode adaptar e incorporar novas tecnologias conforme surgirem. A evolução é constante pois a comunidade desenvolvedora está sempre inovando e fomentando as novas aplicações.

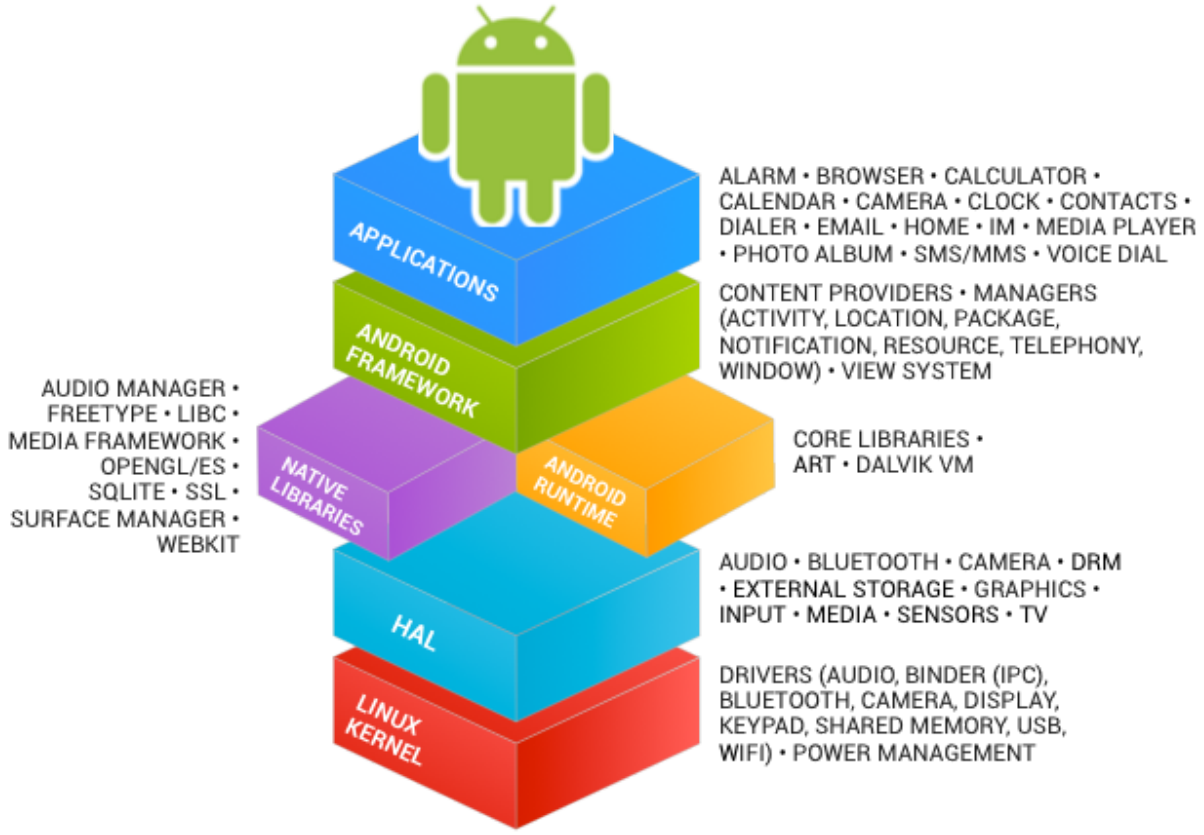
6.3.2 Android SDK

O Android SDK ou Kit de Desenvolvimento de Software para Android é o software

utilizado para desenvolver aplicações no Android, que tem um emulador para simular o celular, ferramentas utilitárias e uma API completa para a linguagem Java, com todas as classes necessárias para desenvolver as aplicações. ((LECHETA, 2013)

As aplicações do Android rodam por meio de máquinas virtuais ART e Dalvik. Os aplicativos instalados no Android interpretados pela máquina virtual Dalvik, tem suas informações enviadas até a interface gráfica. Uma máquina virtual com o Dalvik torna o Android mais amplo, traduzindo o código dos apps para que uma grande quantidade de aparelhos possa executá-los. Introduzido no Android KitKat como uma opção para consumidores avançados, o ART ainda está longe de ser uma interface de execução nativa, mas pode garantir melhores resultados por modificar a forma como ocorre a compilação dos dados. Em vez de trabalhar com o sistema "Just-in-Time" do Dalvik, é realizada uma compilação "Ahead-of-Time", que pode ser traduzida como "à frente do tempo". Ambas máquinas virtuais projetadas para dispositivos Android agem sobre o Kernel-Linux

Os pacotes são ligados e anexados nas pilha de software do Android, que por ser open source, flexível e abundante de aplicações, são divididos em camadas. Abaixo está a imagem da arquitetura do Android.



FONTE: ANDROID VIEW - PRÓLOGO

(medium.com/android-dev-br/android-view-pr%C3%B3logo-d44a89c14fd)

- Applications (Aplicações)

No topo da Figura 2 estão as aplicações do sistema. São conjuntos de aplicações padrões ou nativas como calendário, alarme, e-mail, navegador, que costumam vir agregadas a plataforma Android. Isso possibilita que as mesmas sejam utilizadas por várias outras funcionalidades. Também nessa camada de aplicações, estão os aplicativos instalados através da loja e de arquivos .APK gerados. O Android não faz distinção de aplicações nativas e aplicações desenvolvidas por outras pessoas.

Portanto, um dos motivos para o alto grau de flexibilidade e extensibilidade da plataforma Android é devido a esta característica de não ter distinção nas aplicações padrões ou feitas por outras pessoas, e também é um dos motivos para ter sido escolhida para este projeto.

- Android API Framework (API Java)

No segundo bloco da figura 2 ilustra o Framework a ser utilizado na plataforma, que são os pacotes e recursos necessários para desenvolver a aplicação. No caso, vamos utilizar Java. Os gerenciadores de serviços de telefonia, localização e notificação são alguns exemplos do que este framework disponibiliza. Com essas bibliotecas é possível criar aplicativos e simplificar a reutilização de componentes e serviços de sistemas modulares.

- Native Libraries (Bibliotecas Nativas)

Nesse bloco da figura 2 estão as bibliotecas nativas escritas em C/C++ que fazem parte da plataforma. Estão nesta camada APIs como o OpenGL ES (para renderização 3D), SQLite (gerenciador de bancos de dados) e suporte a diversos formatos de áudio e vídeo. O sistema Android fornece a Java Framework API, que expõe as funcionalidades de algumas dessas bibliotecas nativas aos aplicativos

- Android Runtime

Ao lado das bibliotecas, na figura 2, está o ambiente de tempo de execução, que dá condições para que as aplicações baseadas na plataforma sejam executadas. Um dos componentes desta camada são as *core libraries*, que disponibiliza uma API Java utilizada para programação (grande parte das funcionalidades encontradas no Java SE estão disponíveis para o Android). Já o outro componente é a *Dalvik Virtual Machine*, que é uma máquina virtual para suporte à execução de aplicações. 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 (ART). 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.

- HAL (Camada de Abstração de Hardware)

A camada de abstração de hardware (HAL) fornece interfaces padrão que expõem as capacidades de hardware do dispositivo para a estrutura da Java API de maior nível. 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.

- Linux Kernel (Kernel do Linux)

Sendo o último bloco da figura 2, o Linux Kernel é onde está situado o S.O da plataforma, que é baseado no Linux. É responsável por serviços de mais baixo nível e de suma importância da plataforma, como gerenciamento de memória e processos, segurança, etc. Com o Kernel do Linux o Android pode aproveitar os principais recursos de segurança, além de ser uma vantagem por ser um kernel conhecido, auxiliando os fabricantes dos dispositivos no desenvolvimento de drivers específicos.

6.3.3 Linguagem de Programação

Nesta seção será exibida a linguagem de programação utilizada para o desenvolvimento da aplicação Android e seus componentes necessários para execução, além de ferramentas externas que auxiliaram nesse processo. O desenvolvimento foi feito de forma nativa, explorando as camadas do Android, como foi visto no tópico anterior.

- Angular

Após análise e concepção do projeto quanto a melhor linguagem e desenvoltura para aplicar, foi designado o Angular para o projeto, devido sua qualidade com o design, fácil e clara manutenção do código fonte, oferece o desenvolvimento por módulos, recursos avançados como DI (injeção de dependência), roteamento (routes), animações, services e MVW (Model View Whatever), ou seja, você escolhe o que quer seja, MVC, MVP ou MVVM.

Angular é uma plataforma e framework para construção da interface de aplicações usando HTML, CSS e, principalmente, JavaScript, criada pelos desenvolvedores da Google. Dentre os principais, podemos destacar os componentes, templates, diretivas, roteamento,

módulos, serviços, injeção de dependências e ferramentas de infraestrutura que automatizam tarefas, como a de executar os testes unitários de uma aplicação. Alguns outros pontos dessa plataforma que merecem destaque são o fato de que ela é *open source* e possui uma grande comunidade, segundo Alexandre Afonso (2018). E esses foram motivos decisivos para escolher o framework.

- JavaScript

O JavaScript é uma linguagem de programação client-side, é uma linguagem que é executada no lado do cliente, ou seja, no computador do próprio usuário. Ela permite a você criar conteúdo que se atualiza dinamicamente, controlar multimídias, imagens animadas e até mesmo validar campos e criar menus.

Por ser uma linguagem amplamente utilizada em sites e aplicações, a maioria em peso dos navegadores - seja para celular, computador ou tablet - possuem interpretador de código Javascript. Ser uma linguagem interpretada e não tipada auxilia e deixa adequado para que o estilo de programação orientado a objeto seja implementado. (FLANAGAN, 2011)

- NodeJs

Para o Angular, também é necessário fazer uso da ferramenta Node.js, que é um interpretador de código JavaScript de modo assíncrono e orientado a eventos, focado em migrar a programação do JavaScript do lado do cliente para os servidores, criando assim, aplicações de alta escalabilidade, capazes de manipular milhares de conexões simultâneas em tempo real em uma única máquina física. (IBM, 2019)

Como a execução do JavaScript é orientada a eventos assíncronos, o Node foi projetado para criar aplicativos de rede escalonáveis. Dependendo da aplicação, muitas conexões podem ser tratadas simultaneamente, com o Node.js, em cada conexão o retorno chamado é acionado, mas se não houver um trabalho a ser feito, ele entrará em suspensão. (NODE, 2019)

6.3.3 Banco de Dados

Criar corretamente um banco de dados estruturado exige planejamento. O mais importante é planejar como os dados serão salvos e depois recuperados para tornar esse processo o mais fácil possível. Pensando nisso, e devido aos diversos fatores positivos e

facilitadores, foi escolhido para o projeto o Firebase como banco de dados.

O Firebase é uma plataforma do Google que contém diversas ferramentas e uma ótima infraestrutura para ajudar desenvolvedores, tanto web quanto mobile, a criar aplicações de alta performance. Algumas ferramentas do Firebase a serem ressaltadas para uso no projeto são as seguintes: Realtime Database, Authentication e Storage.

- Realtime Database

Todos os dados do Firebase Realtime Database são armazenados como objetos JSON. Pense no banco de dados como uma árvore JSON hospedada na nuvem. Ao contrário de um banco de dados SQL, não há tabelas nem registros. Quando você adiciona dados à árvore JSON, eles se tornam um nó na estrutura JSON com uma chave associada. É possível fornecer suas próprias chaves, como códigos de usuário e nomes semânticos, ou gerá-las. (GOOGLE, 201?)

O Firebase Realtime Database é um banco de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados. Quando se é criado aplicações em plataformas cruzadas com os SDKs para iOS, Android e JavaScript, todos os clientes compartilham uma instância do Realtime Database e recebem automaticamente atualizações com os dados mais recentes.

A característica de possuir dados atualizados em tempo real é fundamental para o projeto, pois o mesmo visa exatamente uso desse serviço quando necessário atualização na localização em tempo real do serviço de emergência que será prestado.

- Authentication

O Firebase Authentication fornece serviços de back-end, SDKs fáceis de usar e bibliotecas de IU prontas para autenticar usuários no aplicativo. O SDK do Firebase Authentication fornece métodos para criar e gerenciar usuário que utilizam os próprios endereços de e-mail e senha para login. Ele também lida com o envio de e-mails de redefinição de senha.

- Storage

Com os SDKs do Firebase para Cloud Storage, o uploads e downloads são feitos independentemente da qualidade da rede. Os uploads e downloads são mais confiáveis, o que significa que eles são reiniciados no ponto em que foram interrompidos, poupando tempo e largura de banda dos usuários.

Os SDKs estão integrados ao Firebase Authentication para fornecer uma autenticação

simples e intuitiva para os desenvolvedores. Use o nosso modelo de segurança declarativa para que o acesso seja concedido com base no nome, tamanho, tipo de conteúdo e outros metadados do arquivo. (GOOGLE, 201?)

REFERÊNCIAS

JORNAL EPTV 1º EDIÇÃO. **Samu vai usar sistema de segurança para atendimento de emergências na zona rural[...]**. Disponível em: <https://g1.globo.com/sp/piracicaba-regiao/noticia/samu-vai-usar-sistema-de-seguranca-para-atendimento-de-emergencias-na-zona-rural-de-piracicaba.ghtml>

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML.** Campus, 2007.

SCHAWBER, SUTHERLAND. **Scrum Guide, 2013**. Disponível em: <https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>

VALENTE, Jonas, AGÊNCIA BRASIL. **Mais de 5 bilhões de pessoas usam aparelho celular, revela pesquisa**, 2019. Disponível em: <https://agenciabrasil.ebc.com.br/geral/noticia/2019-09/mais-de-5-bilhoes-de-pessoas-usam-aparelho-celular-revela-pesquisa>

LECHETA, Ricardo. **Google Android: aprenda a criar aplicações para dispositivos móveis com Android SDK,** 3º Edição, 2016. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=NrVUAwAAQBAJ&oi=fnd&pg=PA21&dq=Artigos+sobre+Android&ots=QbUBFbRk_v&sig=3_JSOs59Nsm3GyfPXtPGhvVPDHw#v=onepage&q&f=false>

PEREIRA, SILVA. **Android para desenvolvedores**, 2009. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=8u9wJowXfdUC&oi=fnd&pg=PA1&dq=artigos+academicos+android&ots=LUlj24Ylo6&sig=HW1EgOR1vkMHurGfL4TR4nl28jk#v=onepage&q&f=false>

CAMPOS, Dannyrooh. **Por que adotar o Angular para desenvolvimento?**, 2017. Disponível em: <https://www.eng.com.br/artigo.cfm?id=5143&post=por-que-adotar-o-angular-para-desenvolvimento?>

AFONSO, Alexandre. **O que é Angular?**, 2018. Disponível em: <https://blog.algaworks.com/o-que-e-angular/>

FLANAGAN, **David. JavaScript: O Guia Definitivo**, 2011. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=zWNyDgAAQBAJ&oi=fnd&pg=PR1&dq=artigo+sobre+javascript&ots=IzDeD2P9dQ&sig=Y-sbSCwaZ2ECpL-JV-hN8vREzR8#v=onepage&q=artigo%20sobre%20javascript&f=false>

SOMMERVILLE, Ian. **Engenharia de Software,** 8º ed. São Paulo, 2007.

SOMMERVILLE, Ian. **Software Engineering,** 9. ed. São Paulo, 2011.

NODE. **Sobre o Node.js. NodeJs**, 2019. Disponível em: <https://nodejs.org/en/about/>

IBM. **Introdução ao Node JS**. Disponível em: <https://developer.ibm.com/technologies/node-js/>

GOOGLE, **Firebase, Estrutura do Banco de Dados.** Disponível em: <https://firebase.google.com/docs/database/ios/structure-data?hl=pt-br>

GOOGLE, **Firebase Authentication.** Disponível em: <https://firebase.google.com/docs/auth>

GOOGLE, **Firebase Storage**. Disponível em: <https://firebase.google.com/docs/storage>