CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Lucas Pedro Lopes Carvalho

NORTI: aplicativo móvel com informações sobre adoção de padrões de desenvolvimento, arquitetura de software e tecnologias por empresas de TI da região norte do Brasil.

Palmas – TO
2021

Lucas Pedro Lopes Carvalho

NORTI: aplicativo móvel com informações sobre adoção de padrões de desenvolvimento, arquitetura de software e tecnologias por empresas de TI da região norte do Brasil.

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso I (TCC I) do curso de bacharel em Engenharia de Software pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Me. Jackson Gomes de Souza.

Palmas – TO
2021

Lucas Pedro Lopes Carvalho

NORTI: aplicativo móvel com informações sobre adoção de padrões de desenvolvimento, arquitetura de software e tecnologias por empresas de TI da região norte do Brasil.

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso I (TCC I) do curso de bacharel em Ciência da Computação / Engenharia de Software / Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Me. Jackson Gomes de Souza.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. Me. Jackson Gomes de Souza

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. Esp. Fábio Castro Araújo

Centro Universitário Luterano de Palmas – CEULP

_____

Prof.a Dra. Parcilene Fernandes de Brito

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO
2021

AGRADECIMENTOS

A Deus, pela minha vida, por me dar condições físicas e psicológicas para suportar todo o caos provocado pela pandemia. Gratidão a Deus, por me ajudar a ultrapassar todos os obstáculos que me foram colocados ao longo dessa incrível jornada. Hoje meu sentimento é de alegria e muita emoção, por enfim ter alcançado a tão sonhada graduação em Engenharia de Software, meu coração se alegra em vivenciar esse momento de muita gratificação por todo empenho dedicado a este trabalho e durante todo o curso.

Aos meus pais (Hugo Dias e Olinda Gomes) e irmãos (Pryscilla Fernanda e Fabricio Rafael), que me incentivaram nos momentos difíceis e compreenderam a minha ausência enquanto eu me dedicava à realização deste trabalho. Em especial a minha mãe Olinda Gomes que foi meu refúgio, meu abrigo, minha sombra e água fresca, por essa pessoa que sempre acreditou em mim, no meu potencial e que sempre foi o pilar mais forte. Essa conquista que tanto comemoro hoje, só foi possível graças a minha mãe, Olinda.

Ao meu querido professor e amigo Jackson Gomes de Souza, por ter me orientado e ter desempenhado tal função com dedicação e amizade, por ter mim ensinado coisas que vão além de uma relação professor e aluno ou de amizade, mas sim, de uma relação pai e filho, por isso muito obrigado professor, Jack.

Aos professores do CEULP/Ulbra, pelas correções e ensinamentos que me permitiram apresentar um melhor desempenho no meu processo de formação profissional ao longo do curso, ofertando-me oportunidades de crescimento. Em especial aos professores Parcilene Fernandes Brito e Fábio Castro Araújo que compôs a banca examinadora, ajudando-me na entrega de um trabalho com mais qualidade.

Agradeço também ao professor Fabiano Fagundes por tornar as aulas mais divertidas, por aconselhar e acolher seus alunos (um diferencial incrível), professora Madianita por tratar todos os alunos como filhos, mostrando seus cuidados de mãe para cada um. Também aos professores Pierre Soares Brandão e Tayse Virgulino que foram pacientes e dedicaram seu tempo para me ajudar a formatar minha pesquisa de trabalho, meu muito obrigado.

Por último, mas não menos importante, quero agradecer a todos os empresários que me ajudaram respondendo minha pesquisa. Em especial aos empresários Sandro da Educarflix, Rodrigo da Neo Habitus, Henry Santos da Max Data, Jornilson Menezes da Rensoftware e Daniel da Tonolucro Delivery a todos vocês, meu muito obrigado.

RESUMO

CARVALHO, Lucas Pedro Lopes. **NORTI: aplicativo móvel com informações sobre adoção de padrões de desenvolvimento, arquitetura de software e tecnologias por empresas de TI da região norte de Palmas – TO.** 2020. 73 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Engenharia de Software, Centro Universitário Luterano de Palmas, Palmas/TO, 2020.

O software evoluiu de forma acelerada nos últimos anos, cada vez mais torna indispensável o uso dos softwares no cotidiano das pessoas. Para uma evolução contínua do *software*, novas e melhores formas de desenvolvê-lo também tiveram que evoluir, para atender um mercado que busca por soluções mais rápidas e eficientes a cada dia. Em virtude disso, surgiram metodologias de desenvolvimento que vão desde as tradicionais metodologias (clássico) até chegar nas atuais metodologias ágeis. Dessa forma, surge a problemática de como realizar um mapeamento da adoção de práticas de desenvolvimento de *software*, padrões de arquitetura de *software* e tecnologias de desenvolvimento utilizados por empresas de Tecnologia da Informação da região norte do país. Foi realizado a coleta de dados, através de entrevistas e questionários pré-estabelecidos, esses dados foram analisado e processados onde as informações resultantes desta análise, estão disponíveis também em um aplicativo móvel**.** Os resultados apresentado neste documento e no aplicativo, revelaram que as empresas de tecnologia da região norte do Brasil, possui características muito semelhantes as demais regiões do país.

Palavras-chave: desenvolvimento de software, tecnologia, metodologia, aplicativo móvel.

LISTA DE FIGURAS

LISTA DE TABELA

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

B2B2C – Business to business to consumer

B2C – Business to Consumer

CEO – Chief Executive Officer

CEULP – Centro Universitário Luterano de Palmas

CRUD – Create Retrieve Update Delete

CSS – Cascading Style Sheets

HTML – Hyper Text Markup Language

JS – JavaScript

JSON – JavaScript Object Notation

JSX – JavaScript e Extensible Markup Language

MVC – Model View Controller

RN – React Native

TIC - Tecnologias da Informação e Comunicação

TS – Typescript

TXT – Arquivo de Texto

ULBRA – Universidade Luterana do Brasil

WEB – World Wide Web

XML - Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da tecnologia e comercialização dos softwares, surgem com frequência novas aplicações para diferentes fins. O *software* se tornou um objeto necessário para a evolução do mercado de trabalho, onde empreendedores exploram vários meios de criar soluções com uso dessa tecnologia, desenvolvendo aplicativos para dispositivos móveis que geram grandes impactos na cultura e nos costumes da sociedade.

A Engenharia de Software (ES) tem como um dos seus principais objetivos a melhoria contínua do software, criando ou melhorando suas técnicas de atuação, visando incentivar o desenvolvimento da área. Podendo assim, construir sistemas maiores, mais complexos, e com entregas mais rápidas, evoluindo o *software* da melhor forma, conforme as necessidades vão surgindo.

Em virtude disso, surgiram metodologias de desenvolvimento que vão desde as tradicionais metodologias (clássico) até chegar nas atuais metodologias ágeis. Segundo Roger S. Pressman (2011, p. 40), "uma metodologia é o alicerce para processo de engenharia *software* completo, possuindo atividades estruturais aplicáveis a todos os projetos de *software*, independentes de tamanho ou complexidade".

A Engenharia de Software (ES) tradicional apresenta conceitos de planejamento bem definidos, assim como rigorosas documentações e planos estratégicos com pouca dinamicidade. O que gera complexidade por conta dos clássicos modelos de processos de desenvolvimento de *software*, como os modelos em *cascata*, *evolucionário* e *incremental*, que contam com quatro fases fundamentais no desenvolvimento de qualquer *software*: especificação, projeto, implementação, validação e evolução (SOMMERVILLE, 2007).

O *Scrum* é uma das metodologias ágeis utilizadas em projetos de *software*, criada na década de 1990 por Jeff Sutherland. Essa metodologia reúne algumas regras simples e atividades que proporcionam uma melhor comunicação e transparência entre os indivíduos de uma equipe. O *Extreme Programming* (XP) também é uma metodologia ágil, criada em 1996, por Kent Beck, apresenta valores, princípios e técnicas para entregar os melhores resultados em projeto de *software*. Essas e outras metodologias serão discutidas neste trabalho.

Os autores Garlan e Perry, ainda em 1995, foram os primeiros a sugerir a utilização de arquiteturas como uma representação de soluções de *software*. Essa representação poderia

facilitar a compreensão do *software* a ser desenvolvido, a ideia era levantar a importância da estruturação do *software* antes do seu desenvolvimento.

Os sistemas, plataformas e aplicativos de modo geral são construídos para atender as necessidades de uma empresa, cliente, usuários ou um público específico, buscando solucionar um problema. Essas aplicações são desenvolvidas utilizando algum tipo de arquitetura (solução) de *software* que atenda aos seus requisitos (problemas). As arquiteturas de *software*s têm profundo impacto no desenvolvimento, fazendo com que a equipe tome decisões iniciais que irão refletir em todo trabalho da engenharia de *software* e na entrega final, mostrando grande influência nos trabalhos posteriores, desde construção a manutenção do *software* (BASS, apud PRESSMAN, 2016).

A arquitetura de *software* é um termo considerado de difícil definição por alguns autores. Pressman apoia Bass na seguinte definição:

> (...) A arquitetura de *software* de um programa ou sistema computacional é a estrutura ou estruturas do sistema, que abrange os componentes de software, as propriedades externamente visíveis desses componentes e as relações entre eles. (BASS et. al. 2003, apud PRESSMAN, 2016, p. 230)

Um mapeamento da adoção de práticas de desenvolvimento e padrões da arquitetura de *software* pode ser realizado por meio da definição e da aplicação de questionários e análise das respostas. Para que se possa compreender e mensurar o atual cenário de TI (Tecnologia da Informação) da região norte do Brasil. Então ampliar o conhecimento de como são organizadas (estruturadas) as empresas, suas metodologias de desenvolvimento de *software*, arquiteturas de *software*, os tipos de trabalhos (aplicações) que são desenvolvidos e seu nível de inovação. Desta forma, o desenvolvimento do aplicativo NORTI se torna essencial para a apresentação desses dados.

O aplicativo NORTI realizará o processamento dos dados e apresentação da informação, assim como uma distribuição dessa informação para qualquer indivíduo que possua um dispositivo móvel como um *smartphone* (aparelho de celular), por exemplo. Este formato pode ter mais alcance que outros métodos tradicionais, por conta da popularidade dos aplicativos para dispositivos móveis, tornando assim a informação mais acessível ao público.

Por meio de um estudo empírico, esse trabalho busca realizar um mapeamento das práticas de desenvolvimento de *software*, padrões de arquitetura de *software* e tecnologia de desenvolvimento nas empresas de TI (Tecnologia da Informação) da Região Norte. É objeto

deste trabalho apresentar o mapeamento dessas empresas e fornecer um panorama das empresas de TI da Região Norte por meio de um aplicativo para dispositivo móvel.

Este documento está organizado da seguinte forma: No segundo capítulo será mostrado conceitos e definições sobre metodologia de desenvolvimento de *software* e arquitetura de *software*, que serviram de conteúdo base para o desenvolvimento deste projeto. Já no terceiro capítulo será apresentado a metodologia utilizada, o quarto e último capítulo é apresentado o cronograma de desenvolvimento do projeto.

2 REFERENCIAL TEÓRICO

Neste capítulo serão abordadas as referências teóricas para desenvolvimento deste trabalho. Os principais autores, a seguir, contribuíram com seus estudos para esse contexto, especialmente os livros: *Scrum e Agile em Projetos Guia Completo* (Fábio Cruz), *Engenharia de Software: Uma abordagem profissional* (Roger S. Pressman e Bruce R. Maxim).

2.1 METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE

Os softwares de computadores é uma das tecnologias que mais avançaram nas últimas décadas. Desde de 1950, o *software* vem evoluindo rapidamente, durante essa evolução, foram surgindo diversos problemas no âmbito de desenvolvimento do *software*, relacionados a manutenção, prazos de entrega, adaptação, aperfeiçoamento e principalmente pela qualidade dos softwares (SOMMERVILLE, 2007). Sobre o surgimento da Engenharia de Software é possível afirmar que:

> "(…) decorreu da análise feita na época sobre as condições da indústria de software que estava entrando em um período crítico de colapso que ficou conhecido pela alcunha de crise do software que teve seu início em meados da década de 1960, quando os programas existentes tornaram-se difíceis de serem mantidos, estendendo-se até o final da década de 1970" (PRESSMAN, 1995, p. 6)

Existem diversas definições para Engenharia de Software. O Instituto de Engenheiros Eletricistas e Eletrônicos - IEEE (1990, apud PRESSMAN, 2016) possui uma definição mais ampla: "a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção do software". Para Sommerville (2007, p. 5), "é uma disciplina de engenharia relacionada a todos os aspectos de produtos de software". Já para Roger S. Pressman & Bruce R. Maxim (2016, p. 27), "a engenharia de software engloba processos, métodos e ferramentas que possibilita a construção de sistemas complexos baseado em computador dentro do prazo e com qualidade". É comum nessas definições a preocupação com o método (ou processo) e com a medição, seja relacionada ao tempo ou à qualidade.

Os clássicos modelos de processos e métodos de construção de sistemas informatizados, a exemplos dos modelos em cascata, evolucionário e incremental abordados por Sommerville (2007), instituem quatro fundamentais fases no desenvolvimento de todos softwares: especificação, projeto e implementação, validação e evolução.

Outra mudança em relação aos processos tradicionais de Engenharia de Software é que as metodologias ágeis priorizam os indivíduos assim como as atividades e suas interações em

detrimento de ferramentas e processos. Para que o esforço de trabalho seja concentrado nas entregas constantes, a documentação seja o mais sucinta possível, ao contrário de seguir um planejamento fortemente definido (FOWLER, 2019).

Antes de apresentar uma visão sobre as práticas de processos de *software*, é necessário compreender com mais detalhes o termo *agilidade* no âmbito da Engenharia de Software. Agilidade, mudança e adaptação estão atreladas ao ambiente de desenvolvimento de projetos ágeis, assim como a palavra *mudanças* está diretamente ligada à natureza do *software*. Seguindo essa linha Ivar Jacobson traz a seguinte definição para agilidade no âmbito do desenvolvimento de *software*:

> "Uma equipe ágil é aquela rápida e capaz de responder de modo adequado às mudanças. Mudança tem tudo a ver com desenvolvimento de software. Mudança no software que está sendo criado, mudança nos membros da equipe, mudança devido a novas tecnologias, mudanças de todos os tipos que poderão ter um impacto no produto que está em construção ou no projeto que cria o produto." (JACOBSON, 2002 apud PRESSMAN, 2016)

Pressman (2016) vai além ao afirmar que agilidade é mais do que uma resposta à mudança. Ele defende o que Kent Beck (2001) afirmou junto com outros 16 renomados desenvolvedores, consultores e autores da área de *software* os seguintes valores:

- Indivíduos e interações acima de processos e ferramentas;
- Software operacional acima de documentação completa;
- Colaboração dos clientes acima de negociação contratual; e
- Respostas a mudanças acima de seguir um plano.

Esses valores foram batizados de *"Agile Alliance"* ou Aliança Ágil e fazem parte do "*Manifesto for Agile Software Development*" (Manifesto para o Desenvolvimento Ágil de Software). Vale salientar que embora o Manifesto Ágil tenha sido acordado em 2001, sendo referência para metodologias consideradas ágeis até nos dias atuais, essas ideias já vinham sendo discutidas há alguns anos, com principal intuito de suprir as fraquezas constatadas no processo de desenvolvimento de *software* tradicional (BECK, 2002 apud PRESSMAN, 2016).

Os estudos de Coleman e O'Connor (2008), mencionados em Paternoster et al. (2014), contribuem com esse trabalho ao afirmar que *Extreme Programming* (XP) é a metodologia de desenvolvimento de *software* mais aplicada nas empresas que estão em fase inicial, por vários motivos, mas principalmente por seu baixo custo e pouca exigência de documentação e, dentre outras metodologias, o *Scrum* é bastante utilizado por empresas de tecnologia.

Toda essa evolução do desenvolvimento ágil, atrelada aos princípios dos processos tradicionais da Engenharia de Software, provavelmente contribuiu e vem contribuindo para o desenvolvimento de *software*.

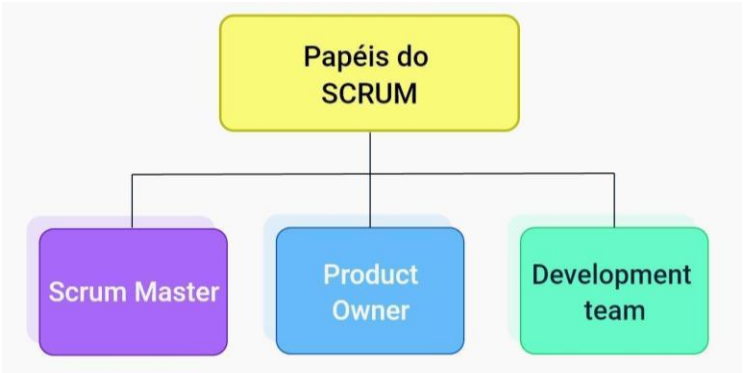
2.1.1 Scrum

O *Scrum* é uma das metodologias ágeis mais difundidas na área de desenvolvimento de *software* nos últimos tempos, foi concebido por Jeff Sutherland e sua equipe de desenvolvimento ainda no início dos anos 1990, entretanto somente se popularizou após o Manifesto Ágil (mencionado anteriormente). Fábio Cruz (2018, p. 52) define o Scrum como "um *framework* para desenvolver e manter produtos complexos que também pode ser utilizado no gerenciamento ágil de projetos que se destinam também à criação de produtos" e afirma que o *Scrum* pode ser utilizado no desenvolvimento de qualquer produto complexo. Por outro lado:

> "(…) Scrum é a forma pela qual uma pessoa, uma equipe ou organização se torna capaz de responder a essa complexidade, de responder a metamorfoses que não podem ser previstas, de se mover com agilidade e entusiasmo através de um espaço de problema em constante transformação." (SUTHERLAND, 2019).

A principal ideia do *Scrum* é aplicar um conjunto de regras simples e atividades que permitam uma maior comunicação e transparência entre os indivíduos de uma equipe (ou time) *Scrum*, dando-lhes autonomia para aprender por conta própria, auto gerenciar e aperfeiçoar a forma como trabalham, sem que haja a necessidade de um controle centralizado (SUTHERLAND, 2019). Os times Scrum possuem três papéis fundamentais, ilustrados pela Figura 1.

Figura 1: Ilustração dos papéis dentro de Time Scrum



É dever do *Scrum Master* assegurar que toda equipe desempenhe um bom papel dentro do Time Scrum, fazendo com todos os indivíduos consigam aplicar todas as práticas, as regras

e os valores que o *Scrum* demanda. Fabio Cruz (2018) afirma que o *Scrum Master* deve treinar o *Development Team* em ambientes onde o *Scrum* não é totalmente adotado ou compreendido, garantindo que todos sigam o *Scrum*, facilitando os eventos *Scrum* conforme necessário e removendo todos e quaisquer impedimentos que possam interferir no objetivo do time.

O *Product Owner (PO)* é o proprietário ou dono do produto. É de responsabilidade do PO gerenciar o *backlog* do produto, garantir que esteja visível a todos (o *backlog* do produto é apresentado adiante). É primordial que o dono do produto tenha o máximo de conhecimento sobre o produto e suas regras de negócio, é missão do PO, que toda a equipe compreenda o produto como um todo, ele também deve garantir a satisfação do cliente, pois o mesmo está diretamente ligado ao cliente (CRUZ, 2018).

Como em qualquer projeto de *software*, o *Scrum* também possui um time de desenvolvimento, chamado de *Development Team*. Seu papel é implementar e transformar o *backlog* do produto em incremento de recursos/funcionalidades para o produto. Fábio Cruz (2018, p. 62) afirma que este é "o papel mais importante para qualidade técnica e funcional do sistema ou produto que está sendo construído e entregue ao cliente" e deve garantir a qualidade técnica.

O desenvolvimento de *software* é algo desafiador, com várias atividades complexas. Um processo complexo geralmente exige uma comunicação bem ampla entre todos os membros de um time. Por isso o *Scrum* possui a *Daily Meeting* (reunião diária). Essa reunião é de curta duração (no máximo quinze minutos), tem como objetivo fazer com que as atividades do Time de Desenvolvimento sejam sincronizadas e consigam planejar as próximas 24 horas, uma *Daily Meeting* é composta por três perguntas chaves que são respondidas por todos os membros da equipe:

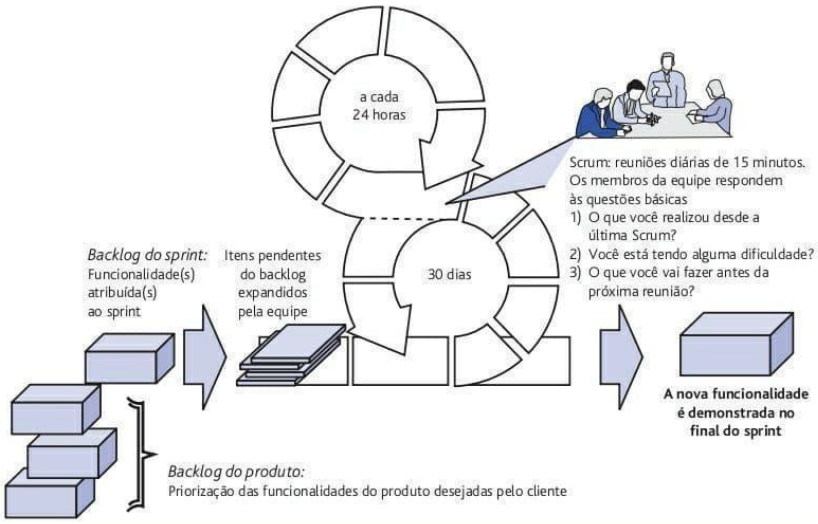
1. O que você realizou desde a última reunião de equipe?
2. Quais obstáculos está encontrando?
3. O que planeja realizar até a próxima reunião da equipe?

A reunião diária é conduzida pelo seu líder *Scrum Master* que avalia as respostas de cada integrante, isso ajuda a equipe a revelar problemas em potencial o mais cedo possível (PRESSMAN, 2018).

O *Scrum* possui um ciclo principal que vai desde o Planejamento da *Sprint* até Revisão da *Sprint*, esse ciclo é ditado principalmente pelas *Sprints*. A Figura 2 ajuda a compreender melhor o fluxo do processo *Scrum*, apresentando como cada iteração é realizada dentro *Scrum*.

A Figura 2 indica a existência dos papéis *Scrum Master*, *Product Owner* (PO) e *Development Team*

Figura 2: Fluxo do processo Scrum



Fonte**:** PRESSMAN, 2016.

A Figura 2 mostra com detalhes o fluxo completo do *Scrum* (alguns itens mencionados na imagem não foram discutidos anteriormente, mas que serão apresentados a seguir). Vale salientar que o Time Scrum se reveste de artefatos específicos, aplicando regras que unem tanto os eventos, como os papéis e artefatos, os quais são exatamente os itens mencionados na Figura 3:

- ***Backlog***: Scrum possui o backlog como o artefato mais importante, reunindo os itens/requisitos do produto a ser entregue; o *backlog* também deve oferecer um entendimento completo aos requisitos para que assim sejam reproduzidos em forma de funcionalidade para o produto, e por fim, entregar o produto (CRUZ, 2018);
- ***Sprint***: É um período com duração fixa e um evento iterativo, geralmente tem uma duração de, no máximo, um mês. Segundo Fábio Cruz (2018, p. 68), "*Sprint* é o coração do Scrum";
- ***Backlog da Sprint***: São os itens que estão "prontos" para compor a *Sprint*, em outras palavras, são um conjunto de itens a serem implementados dentro de uma *Sprint.*

Essas características do *Scrum* podem ser facilmente associadas às características que regem um projeto de *software*, onde os pilares de sua natureza (adaptabilidade, mudanças e flexibilidade) são apresentados como parte dos princípios e atividades que norteiam o *Scrum*.

Isso explica uma maior adoção por esses frameworks (ferramentas) ao invés de clássicos modelos de desenvolvimento de *software*.

2.1.2 Extreme Programming

Extreme Programming, também conhecido por Programação Extrema ou ainda pela sigla XP, é uma metodologia ágil, assim como Scrum, criada em 1996, por Kent Beck, direcionada a equipes de tamanho pequeno e médio. Seu desenvolvimento possui algumas diferenças em relação a outros modelos, podendo ser aplicada principalmente em projetos de alto risco e que possuem requisitos dinâmicos (CRUZ, 2018). Desta forma ele contribui para o desenvolvimento de *software* com maior qualidade, em menor tempo e de forma econômica.

O XP apresenta alguns valores, princípios e práticas das quais proporcionam o alcance do objetivo do projeto (KENT, 2001). Focado nas tarefas de programação propriamente ditas, o XP defende valores essenciais e práticas que são levadas ao extremo, para que seja possível atingir o sucesso em seus projetos (CRUZ, 2018). São estes os valores:

- Comunicação
- Feedback
- Coragem
- Respeito
- Simplicidade

Segundo Fábio Cruz (2018), essas premissas têm como objetivo primordial, direcionar toda a equipe quanto aos objetivos do projeto, com intuito de instigar o time a se conectar no que de fato é importante para o projeto, deixando para trás ações individuais.

A comunicação é sem dúvidas uma arma poderosa para minimizar problemas no âmbito do desenvolvimento de *software* ágil, que vão desde compreensão das regras de negócio, alcance dos resultados esperados, até conflitos de relacionamento (CRUZ, 2018).

Entre as mais diversas formas de comunicação atual, o *Extreme Programming* defende que toda a comunicação no escopo do projeto deve ser primordialmente realizada por meio de um diálogo face a face, mas sem descartar as outras maneiras de se comunicar. Isso porque o diálogo presencial oferece formas variadas de compreender uma informação, visto que elementos como postura, tom de voz, expressões faciais e até sentimentos acabam contribuindo para uma melhor compreensão. Para Fábio Cruz (2018, p. 210), “a comunicação é a principal forma de transmitir e trocar informações e conhecimento do projeto em projetos de *software*”. Esse pensamento vai ao encontro do próximo valor da Programação Extrema: o Feedback.

Fazendo uma breve retrospectiva das metodologias abordadas neste trabalho, o Scrum concede um elevado e amplo ganho na comunicação proporcionados principalmente pela *Daily Meeting* e revisão da *sprint*, segundo os valores do *Extreme Programming*.

O Feedback se mostra um valor fundamental para compreender e descobrir o mais rápido possível os problemas no desenvolvimento, reduzindo os prejuízos e consequentemente os riscos de impactos nos resultados (CRUZ, 2018).

Para Fábio Cruz, a técnica de feedback proporciona benefícios que levam a melhoria contínua do desenvolvimento de *software*:

> (...) O feedback é uma das melhores técnicas para aplicar um processo eficiente de melhoria contínua, especialmente quando o time considera que fornecer e receber feedback é uma das melhores oportunidades para aprender e identificar pontos de melhoria. (CRUZ, 2018, p. 211).

O *Extreme Programming* exige coragem de sua equipe, essa palavra é aplicada devido ao autor Kent Beck acreditar que a única constante dentro do XP é que a mudança sempre vai existir. Para tal é preciso coragem para enfrentar as constantes mudanças dentro de um projeto de *software*. Essas mudanças são quase sempre originadas pelos clientes que mudam de ideia com certa frequência, também são motivadas pelo fato de que a compreensão do produto se dá, em grande parte, durante o seu desenvolvimento (TELES *et al.*, 2007).

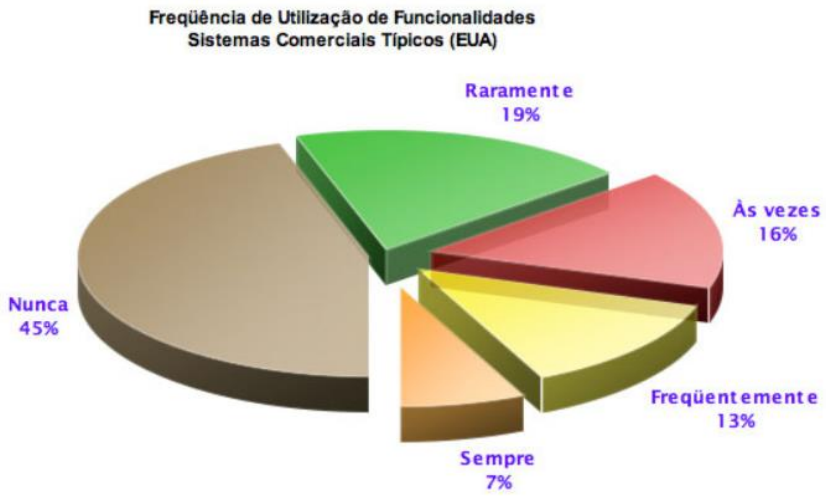
É possível traduzir a coragem imposta pela *Extreme Programming* em ter confiança ao assumir os riscos de um projeto, levando em consideração que eventualmente pode acontecer de quebrar o que vinha funcionando. Entretanto as práticas utilizadas por Kent Beck são voltadas, entre outras coisas, para a proteção do projeto de *software* contra esses potenciais eventos (TELES *et al.*, 2007).

A filosofia da simplicidade empenhada pelo XP, que visa manter o trabalho o mais simples e focado possível, busca entregar o que realmente é importante para o projeto. A Programação Extrema parte do princípio que não se deve perder tempo criando algo que não é necessário naquele momento. Para Fábio Cruz (2018, p. 213), “o maior desperdício é aquele de produzir algo que ninguém vai usar”, com certeza a simplicidade é um dos valores mais importante para XP, mantendo o foco no que realmente é, importa para o projeto.

Esse valor é muito importante no XP para que nenhum esforço da equipe seja desperdiçado sem que não sejam utilizadas. Uma pesquisa sobre projetos de *software* realizado por uma empresa americana chamada *Standish Group*, apresenta números bem interessantes,

dentre eles uma métrica sobre o grau de utilização dos recursos ou funcionalidade de um *software* que são colocados em produção. Esse estudo reuniu mais de 280 mil projetos de *software* nos EUA, a Figura 3 apresenta a frequência da utilização das funcionalidades de um sistema comercial.

Figura 3: Frequência de Utilização de Funcionalidades em Sistemas Comerciais Típicos (EUA)



Fonte**:** MELLO, 2007.

De acordo com gráfico apresentado na Figura 3 o estudo revela que 7% (sete por cento) das funcionalidades são sempre utilizadas e 13% (treze por cento) são utilizados frequentemente. Desta forma é possível concluir que o Princípio de Pareto é aplicado ao desenvolvimento de *software*, sendo que 20% (vinte por cento) das funcionalidades podem gerar 80% (oitenta por cento) do resultado esperado (CRUZ, 2018). Assim, conclui-se que a simplicidade enquanto valor da Programação Extrema possui um papel importantíssimo para com a entrega significativa dos resultados, focado principalmente na produtividade e esforço técnico do time.

O respeito também é um dos valores do XP responsável por conscientizar a equipe no que tange a responsabilidade de cada indivíduo para com outro dentro do time *Extreme Programming*. O respeito é um valor simples, mas que sustenta todos os outros valores, segundo Fábio Cruz (2018, p. 214), "saber ouvir e respeitar o ponto de vista do outro pode contribuir muito para o sucesso de um projeto de desenvolvimento de software".

Ainda de acordo com Cruz (2018), a comunicação é um ponto incomum entre as metodologias ágeis e os clássicos modelos de desenvolvimento de *software*. Logo, é

compreensível dizer que, a comunicação na equipe, é um fator determinante para alcançar o sucesso de um projeto de *software*, independente da metodologia utilizada.

Salienta-se, que embora a Engenharia de *Software* e seus já conhecidos processos de desenvolvimento não sejam totalmente difundidos dentre as metodologias ágeis. Algumas práticas são concebidas na utilização de seus princípios, mesmo que de formas diferentes, mas que produzem resultados semelhantes.

A Tabela 1 reúne de forma categorizada as características do conhecimento entre os ambientes ágeis e tradicionais em projetos de desenvolvimento de *software*. O conhecimento adquirido, percebido e gerenciado nas etapas dos projetos de TI (Tecnologia da Informação) é tratado tanto por métodos tradicionais quanto ágeis no desenvolvimento de *software* (ROSES; WINDMÖLLER; CARMO, 2016).

Tabela 1: Comparativo entre Desenvolvimento Tradicional e Desenvolvimento Ágil

Perspectivas	**Desenvolvimento Tradicional**	**Desenvolvimento Ágil**
Estilo de Gestão	Comando, controle e direção na execução dos projetos pelo gestor, interferindo no planejamento e distribuição dos trabalhos para as equipes. Os gerentes participam das decisões com os clientes.	Liderança, colaboração e comunicação. O gerente é um facilitador. Descentralização. Suporte aos líderes de equipe visando resultados. Os gerentes atuam como facilitadores. Coordenação do trabalho.
Comunicação	Formal	Informal
Modelo de Desenvolvimento	Ciclo de vida do modelo (cascata, espiral ou alguma variação)	Modelo de entrega evolutiva e contínua
Processos de trabalho	Regras e procedimentos operacionais padrão. Definição clara de funções e responsabilidades.	Fluidez dos processos de trabalho. Execução de atividades sem artefatos anteriores.
Características dos profissionais de TI	Especialista no assunto em diferentes equipes. Pouca flexibilidade para distribuição das atividades da equipe. Lealdade, obediência e poucos conflitos.	Diferentes funções de intercâmbio (generalistas) que trabalham no mesmo ambiente. Proximidade com cliente, comprometida com o objetivo é tolerante com os conflitos e a dialética. Solução

	de problemas em equipes com líder de projeto

Fonte*:* Roses, Windmöller e Carmo (2016).

A tabela apresenta ainda pontos importantes que contribuem para entender em qual cenário é mais adequado a utilização de cada metodologia.

2.1.3 RUP (Rational Unified Process)

A sigla RUP que é traduzido em Processo Unificado Rational ou comumente falado "Processo Unificado" foi criado pela Rational Software Corporation, mas em 2003 foi adquirida pela IBM, se trata de uma metodologia para desenvolvimento de *software* criada pela Rational Software, IBM, SofTeam, Unisys, Nihon Unisys, Alcatel e Q-Labs (PISKE, 2003).

Um dos principais sustentos do RUP é o conceito de *best practices* (melhores práticas), que são regras que buscam reduzir o risco e tornar o desenvolvimento mais prático. O RUP define seis *best practices* padrões, sendo elas:

- desenvolver iterativamente;
- gerenciar requerimentos;
- utilizar arquiteturas baseadas em componentes;
- modelar visualmente;
- verificação contínua de qualidade;
- controle de mudanças.

O processo tenta diminuir os riscos do desenvolvimento e efetivamente deixar o desenvolvimento mais eficiente, através das seis práticas básicas a serem executadas durante todo o processo de desenvolvimento (PISKE, 2003).

Ainda definido por Piske (2003) o processo define ainda sua organização em cima do desenvolvimento em 4 fases bem definidas, contendo em cada uma delas no mínimo uma iteração, ou seja, um ciclo de vida, são nessas interações que são mostradas ao cliente o andamento da produção para que ele possa validar e assim liberar a continuação do desenvolvimento. São elas: Concepção (e uma fase preliminar que define o escopo do *software*); Elaboração (onde ocorre o plano do projeto, especificação de características e arquitetura); Construção (nesta etapa ocorre a codificação do *software*) e Transição (que nada mais é que a implementação do *software*).

Segundo Piske (2003), o RUP prova ser um processo de desenvolvimento apesar de robusto é bem definido, embora bastante complexo para projetos de *software* de pequeno porte, ele pode ser bem melhor aproveitado para projetos onde é preciso manter registro constante do fluxo do projeto.

2.2 ARQUITETURA DE SOFTWARE

Ao longo dos últimos anos, os pesquisadores, autores e praticantes da engenharia de *software* têm buscado uma definição para a arquitetura de *software*, foram várias definições até nos dias atuais, isso mostra o interesse ou talvez a necessidade em se discutir mais sobre o assunto. Sendo algumas dessas definições:

> "(...) Arquitetura é o termo que muitas pessoas tentam definir com pouca concordância. Há dois elementos comuns: um deles trata do mais alto nível de quebra de um sistema em partes; o outro trata de uma decisão difícil de ser mudada." (FOWLER, 2002, p. 27);
>
> "(...) A arquitetura de software de um programa ou sistema computacional é a estrutura ou estruturas do sistema, que abrange os componentes de software, as propriedades externamente visíveis desses componentes e as relações entre eles." (BASS et. al. 2003, apud PRESSMAN, 2016, p. 230);
>
> "(...) A arquitetura não é o software operacional, mas sim, uma representação que nos permite (1) analisar a efetividade do projeto no atendimento dos requisitos declarados, (2) considerar alternativas de arquitetura em um estágio quando realizar mudanças de projeto ainda é relativamente fácil e (3) reduzir os riscos associados à construção do software." (PRESSMAN, 2016, p. 230)

Roger S. Pressman e Brunce R. Maxim (2016) enfatizam que arquitetura de *software* é um termo difícil de se descrever. Seguindo os dados apresentados, arquitetura de *software* tem por objetivo definir uma estrutura para o *software*, estabelecer a comunicação entre os módulos do *software* e suas respectivas propriedades, assim como atribuir responsabilidades para cada parte do *software*.

A arquitetura de *software* requer tomadas de decisões que implicam posteriormente no desenvolvimento da aplicação, as quais irão nortear todo o desenvolvimento de *software*. Segundo Martin Fowler (2019), "uma boa arquitetura é importante, caso contrário, torna-se mais lenta e mais cara para adicionar novas capacidades no futuro". Essa afirmação remete principalmente à capacidade do *software* sofrer mudanças sem prejudicar seu bom funcionamento, estando relacionado à capacidade de "crescer", recebendo novas funcionalidades de forma "saudável", sem complicações para o sistema.

Desde o surgimento da necessidade de compreender melhor como *software* deveria ser estruturado para suportar sua própria evolução, algumas arquiteturas foram desenvolvidas e aplicadas nos mais diversos sistemas nos últimos 60 anos. Esse termo arquitetura de *software* é relativamente novo, “padrão de software” foi por tempo (talvez ainda seja) utilizado para fazer referência às características que compõem uma arquitetura de um sistema.

Segundo Martin Fowler (2019), defende que uma boa arquitetura deve suportar sua própria evolução, estando também fortemente ligado com a sua codificação. Nas subseções a seguir serão apresentados fundamentos e características das principais arquiteturas de *software*.

2.2.1 Arquitetura em Camadas

Desde o surgimento da internet ainda nos anos 1990, que a estrutura básica de uma arquitetura em camadas vem sendo utilizada. Este é um padrão de arquitetura muito utilizado, cada uma dessas camadas exerce um papel diferente dentro da arquitetura.

Segundo Roger S. Pressman (2016, p. 237), “são definidas várias camadas diferentes, cada uma realizando operações que progressivamente se tornam mais próximas do conjunto de instruções de máquina”, embora a arquitetura não apresente uma quantidade mínima e máxima de camadas, e nem quais tipos de camadas deva existir em sua arquitetura, o autor Pressman sugere que essas arquiteturas, seja organizada da seguinte forma:

- Apresentação;
- Negócios;
- Persistência;
- Banco de dados.

Cada camada possui componentes que se relacionam de alguma forma. O autor Sommerville (2011), traz uma definição ampla sobre componentes de *software*, ele atribui as seguintes características a um componente de *software*:

- **Padronização**: significa que um componente usado em um processo *software* precisa obedecer a um modelo de componentes padrão;
- **Independente**: deve ser possível compor e implementá-lo sem precisar usar outros componentes específicos;
- **Passível de composição**: para um componente ser composto, todas as interações externas devem ter lugar por meio da interface publicamente definidas;
- **Implantável**: deve ser capaz de operar como uma entidade autônoma em uma plataforma de componentes que forneça uma implementação do modelo de

componentes, o que geralmente significa que o componente é binário e não tem como ser compilado antes de ser implantado;

- **Documentado**: os componentes devem ser completamente documentados para que os potenciais usuários possam decidir se satisfazem as suas necessidades. A sintaxe e, idealmente, a semântica de todas as interfaces de componentes deve ser especificada.

E essa definição irá contribuir para uma melhor compreensão dos sub tópicos seguintes. Pressman (2016), defende que as camadas isolam a responsabilidade de cada um dos componentes:

> "Na camada mais externa, os componentes atendem operações de interface do usuário. Na camada mais interna, os componentes realizam a interface com o sistema operacional. As camadas intermediárias fornecem serviços utilitários e funções de software de aplicação. (PRESSMAN, 2016, p. 236)"

Os componentes são organizados em camadas horizontais e exercem papéis diferentes. Dessa forma, cada camada na arquitetura não precisa saber em detalhes o que acontece em outra camada distinta.

Segundo Sommerville (2007, p. 167), "à medida que uma camada é desenvolvida, alguns serviços fornecidos por essa camada podem ser disponibilizados para os usuários", o autor contribui com trabalho ao afirmar que a arquitetura em camada é incremental, modificável e portável. Por acreditar que cada camada possui diferentes papéis, desta forma torna mais fácil a substituição de uma camada por outra equivalente.

2.2.2 Arquitetura Cliente-Servidor

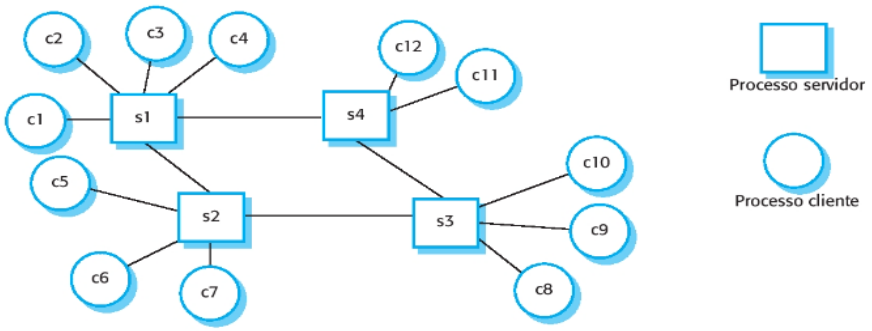
Uma arquitetura cliente-servidor ou modelo cliente-servidor é dentre outras coisas, um conjunto de serviços ou recursos. Esses recursos são normalmente disponibilizados por uma estrutura formada por um supercomputador (computadores com grandes capacidades de processamentos e armazenamento de dados). Esses computadores recebem o nome de servidores, já que são eles quem fornecem (servem) a outros computadores (clientes), os recursos e serviços que serão apresentados em algum momento para o usuário final.

Nos anos 70, era desenvolvido o modelo que atualmente é conhecido como arquitetura cliente-servidor. A rede mundial de computadores (internet) é composta por esse modelo (SOMMVERILLER, 2011).

Segundo Sommerville (2018, p 470), “em um sistema cliente-servidor, o usuário interage com um programa (recurso ou serviço) que está sendo executado em um computador local, como um navegador web ou aplicativo em uns dispositivos móveis”, os serviços de páginas web, são exemplos de serviços que os servidores disponibilizam aos computadores clientes.

É possível identificar dois elementos primordiais em uma arquitetura cliente-servidor, são eles: servidor e cliente. Esses elementos possuem papéis distintos, e são essenciais para compor o modelo cliente-servidor. Na Figura 4 é apresentado a representação de uma interação da arquitetura cliente-servidor.

Figura 4: Interação cliente-servidor



Fonte: SOMMERVILLE, 2018

São apresentados os papéis e características de cada elemento, são eles:

- **Servidor**: É geralmente um computador com características superiores, com grande poder de processamento de dados. Os servidores armazenam e disponibilizam os serviços que poderão ser utilizados por um computador cliente, são exemplos de servidores: servidores de impressão, aqueles que fornecem serviços de impressão; e servidor de compilação, são aqueles que fornecem serviços de compilação de código fonte; (SOMMERVILLE, 2011).
- **Cliente**: São computadores que podem consumir os serviços oferecidos por um ou mais servidores. Segundo Sommerville (2011, p. 129) "Uma rede que permite aos clientes acessar esses serviços. A maioria dos sistemas cliente-servidor é implementada como sistemas distribuídos, conectados através de protocolos de Internet".

Arquitetura cliente-servidor utiliza essencialmente o conceito de computação centralizada, seus recursos são distribuídos para diversas plataformas cliente. Geralmente um

cliente e um servidor se comunicam através da mesma rede de computadores (dois ou mais computadores interligados) ambos em máquinas distintas, porém o cliente e o servidor podem estar na mesma máquina (SOMMERVILLE, 2011).

2.2.3 Arquitetura MVC

O Model View Controller é uma arquitetura dividida em três camadas (Modelo, Visão e Controle), este é um dos padrões arquitetônicos mais utilizados no âmbito da internet nos últimos anos. O MVC foi implementado pela primeira vez em 1978, pelo seu criador Trygve Reenskaug, esse padrão foi desenvolvido inicialmente com intuito de ser utilizado por aplicações desktops (PRESSMAN, 2016). Entretanto, o MVC começou a ganhar forças após o uso em aplicações web, graças à adoção em diversos Frameworks, a exemplo:

- Django;
- Ruby Rails; e
- CakePHP.

Esses Frameworks são popularmente conhecidos por permitir o desenvolvimento de aplicações web com uso da arquitetura MVC. Há alguns anos, não existia uma separação das responsabilidades na criação de aplicativos web. Segundo Roger S. Pressman (2016, p. 348), o Model View Controller "é uma de vários modelos de infraestrutura sugeridos para WebApps que desassociam a interface do usuário da funcionalidade e do conteúdo de informações de uma WebApp", entenda "WebApp" como aplicações para internet, o autor defende que cada camada possui responsabilidade diferentes, separando os diferentes aspectos da aplicação.

O Framework Django juntamente com a linguagem de programação Python, utiliza o padrão MVC para criar aplicações para internet. Graças aos três principais componentes (Model, View e Controller) é possível realizar o desenvolvimento dessas aplicações em paralelo. Por exemplo, um time de desenvolvimento pode escolher trabalhar em paralelo, dividindo o time em três equipes menores, cada equipe trabalhando em uma camada.

- **Model:** O modelo, também conhecido como "objeto-modelo" contém todo o conteúdo das informações que são armazenadas, estejam elas em um banco de dados relacional, em um arquivo TXT, XML, JSON ou qualquer que seja. É de responsabilidade do Model realizar as operações de CRUD (*Create, Retrieve, Update e Delete*) são aqui que devem acontecer.
- **View**: A view ou simplesmente visão é responsável pela interação com usuário e por manter funcionalidades específicas da interface de usuário. É de responsabilidade desta

camada apresentar as informações da aplicação, ela contém todas as informações que um usuário pode visualizar. Geralmente, as views são documentos que utilizam tecnologias web, como JavaScript, HTML e CSS.

- **Controller**: O controlador é responsável por gerenciar o acesso entre o modelo e a visão e coordenar o fluxo de dados entre eles. Toda a lógica e regras de negócios são implementadas nessa camada, é essa camada que determina o comportamento da aplicação. Cada ação realizada pelo usuário (na view) pode refletir em uma mudança de estado, o controller recebe essa ação e decide o que fazer com ela, levando em consideração a lógica e as regras de negócios implementadas na camada. (ZEMEL, 2009).

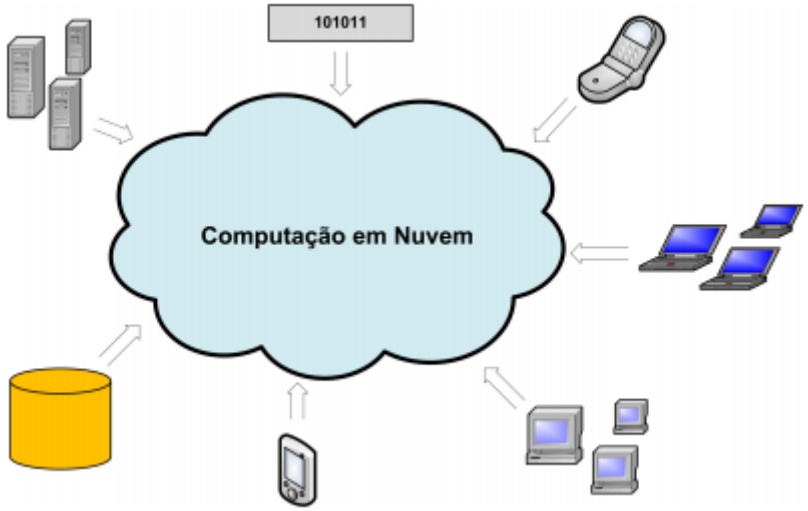
2.2.4 Computação em Nuvem

De acordo com Pedrosa e Nogueira, a computação na nuvem ou *Cloud Computing*, como é conhecida em por alguns autores, é um modelo de computação que permite ao usuário final acessar uma grande quantidade de aplicações e serviços em qualquer lugar de forma independentemente da plataforma, bastando para isso ter um terminal conectado à "nuvem" (o termo "nuvem" é representado pelo serviço de internet, ou seja, a infra-estrutura de comunicação composta por um conjunto de interfaces, softwares, redes de telecomunicação, hardwares, dispositivos de controle e de armazenamento que permitem a entrega da computação como serviço.).

Os *data centers* são necessários para reunir todas as aplicações e dados dos usuários em grandes centros de armazenamento, tornando esse modelo possível.

Anteriormente à computação em nuvem foi limitada a uma determinada classe de usuários ou focadas em tornar disponível uma demanda específica de recursos de TI, principalmente de informática (BUYYA et al. 2009). Computação em nuvem pretende ser global e prover serviços para as massas que vão desde o usuário final que hospeda seus documentos pessoais na Internet até empresas que terceirizam toda infraestrutura de TI para outras empresas (SOUSA et al., 2010). Não apenas recursos de computação e armazenamento são entregues sob demanda, mas toda a pilha de computação pode ser aproveitada na nuvem, como demonstrado na Figura 5.

Figura 5: Visão geral de uma nuvem computacional



Fonte: SOUSA et al., 2010.

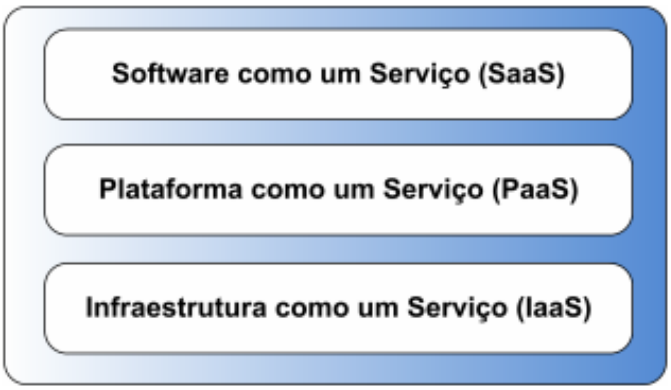
Definido como caso de utilização de processamento centralizado, com isso, os usuários estão movendo seus dados e aplicações para a nuvem e assim acessá-los de forma simples e de qualquer local (SOUSA et al., 2010).

Segundo Pedrosa e Nogueira para a utilização dos serviços, os usuários necessitam apenas ter em suas máquinas um sistema operacional, um navegador e acesso à Internet. Todos os recursos computacionais estão disponíveis na nuvem e as máquinas dos usuários não necessitam ter altos recursos computacionais, diminuindo o custo na aquisição de máquinas. A ferramenta mais famosa conhecida por plataforma como serviço é o *Google App Engine*. Ela oferece uma plataforma que possibilita o desenvolvimento de aplicações através da linguagem de programação Python, na infraestrutura da Google.

Todo hardware pode ser utilizado para realizar alguma tarefa que seja adequada ao seu poder de processamento. Novos recursos de hardware podem ser adicionados a fim de aumentar o poder de processamento e cooperar com os recursos existentes (SOUSA et al., 2010).

Ainda de acordo com a pesquisa de Sousa et al. (2010) o ambiente de computação em nuvem é composto de três modelos de serviços. Sendo eles tão importantes, pois definem um padrão arquitetural para soluções de computação em nuvem (Figura 6).

Figura 6: Modelos de Serviços



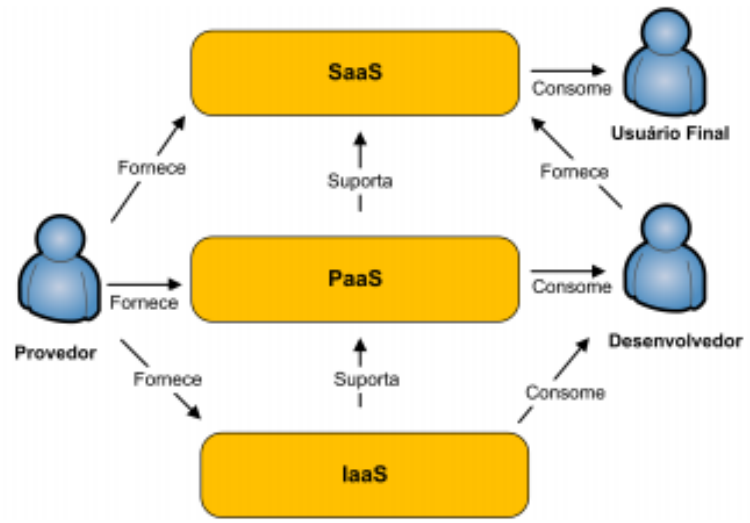
Fonte: SOUSA et al., 2010.

No modelo de SaaS, o usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistemas operacionais, armazenamento ou mesmo as características individuais da aplicação, exceto configurações específicas, proporciona ainda sistemas de *software* com propósitos específicos que estão disponíveis para os usuários através da Internet. Os sistemas de *software* são acessíveis a partir de vários dispositivos do usuário por meio de uma interface *thin client* como um navegador Web (SOUSA et al., 2010).

Já o sistema PaaS oferece uma infraestrutura de alto nível de integração para implementar e testar aplicações na nuvem, nesse sistema o usuário também não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistemas operacionais ou armazenamento, porém o diferencial é que tem controle sobre as aplicações implantadas e, possivelmente, as configurações das aplicações hospedadas nesta infraestrutura. O sistema fornece ainda um sistema operacional (SOUSA et al., 2010) linguagens de programação e ambientes de desenvolvimento para as aplicações, auxiliando a implementação de sistemas de *software*, já que contém ferramentas de desenvolvimento e colaboração entre desenvolvedores.

A Figura 7, demonstra sobre os papéis que cada serviço e sistema desempenham na computação em nuvem.

Figura 7: Papéis na Computação em Nuvem



Fonte: SOUSA et al., 2010.

Ainda segundo Sousa et al. (2010) O IaaS é a parte responsável por prover toda a infraestrutura necessária para a PaaS e o SaaS. O principal objetivo do IaaS é tornar mais fácil e acessível o fornecimento de recursos, tais como servidores, rede, armazenamento e outros recursos de computação fundamentais para construir um ambiente sob demanda, que podem incluir sistemas operacionais e aplicativos móveis, por exemplo.

2.2.5 Arquitetura Orientada a Serviço

Arquitetura orientada a serviços (SOA que é o acrônimo de *Service-Oriented Architecture*) é um tipo de design de *software* que torna os componentes reutilizáveis usando interfaces de serviços com uma linguagem de comunicação comum em uma rede, responsável por uma abordagem arquitetural corporativa que permite a criação de serviços de negócio interoperáveis que podem facilmente ser reutilizados e compartilhados entre aplicações e empresas, ou seja integra os componentes de *software* que foram implantados e são mantidos separadamente, permitindo que eles se comuniquem e trabalhem juntos para formar uma aplicação que funciona em sistemas diferentes (VICTORINO e BRÄSCHER, 2009).

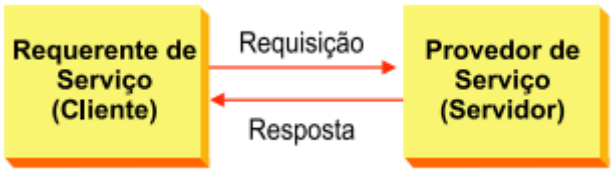
Ainda segundo o trabalho de Victorino e Bräscher (2009), as abordagens de modelagem de processos procuram atingir o máximo de reuso dos mesmos, que não dependam de automação. Para que os processos sejam decompostos em conjuntos de unidades bem definidas, dessa forma cada um com sua funcionalidade convenientemente descrita, e essas unidades são chamadas serviços. Para que uma organização tenha seu negócio modelado segundo esse

paradigma é necessário o uso de uma arquitetura para dar suporte, chamada Arquitetura Orientada a Serviços (SOA).

No final da década de 1990, a conexão de uma aplicação a serviços em um outro sistema era um processo complexo que envolvia uma intrincada integração *point-to-point* (P2P), incluindo conectividade, roteamento, modelos de tradução de dados etc. E em cada projeto novo, os desenvolvedores precisavam recriar essa integração (VICTORINO e BRÄSCHER, 2009).

Desde a primeira menção à arquitetura SOA que foi em um artigo publicado pela equipe de desenvolvimento de serviços web da IBM no site *developerWorks*. Desde então tem sido de grande interesse entre pesquisadores no meio científico e tem recebido uma forte aceitação no mercado de desenvolvimento de *software* (SOUZA, 2006), na Figura 8 é explicado de forma simplificada como funciona o SOA.

Figura 8: Versão simples da Arquitetura Orientada a Serviços



Fonte: SOUSA, 2006.

Mais do que uma tecnologia, SOA também influencia regras e processos de negócios, além de muitas vezes implicar reengenharia de *software* simultaneamente (VICTORINO e BRÄSCHER, 2009), porém a fim de utilizar eficientemente uma SOA, é necessário atender a alguns requisitos mínimos, sendo eles:

A interoperabilidade entre diferentes sistemas e linguagens de programação fornece a base para a integração entre aplicações em diferentes plataformas, através de um protocolo de comunicação. Um exemplo dessa comunicação depende do conceito de mensagens. Usando mensagens, através de canais de mensagens definidos, diminui a complexidade da aplicação final, permitindo que o desenvolvedor do aplicativo se concentre no banco de dados compartilhado. Isto permite que novas funcionalidades sejam desenvolvidas para um formato de negócio de referência comum para cada elemento de dados simultaneamente (VICTORINO e BRÄSCHER, 2009).

Em comparação a abordagem monolítica, o SOA possui vantagens claras, como: Menor *time to market* e maior flexibilidade, o uso de infraestrutura legada em novos mercados (com a

SOA, é mais fácil para os desenvolvedores escalar ou ampliar o uso de uma funcionalidade para plataformas ou ambientes novos), a redução de custos é resultado da maior agilidade e eficiência no desenvolvimento, a fácil manutenção (a maioria dos serviços são independentes e autossuficientes), a escalabilidade, sua maior confiabilidade (e mais fácil fazer o debug de serviços menores do que um grande código) e a maior disponibilidade (possui recursos disponíveis para todos.) (SOUZA, 2006).

2.2.6 Arquitetura de Microserviços

A arquitetura de *Microserviços* surge como resposta a problemas complicados de se resolver, já que a maioria dos desenvolvedores necessita definir dentre as diversidades de linguagens e qual atenderá melhor as necessidades finais do cliente, sendo que cada uma delas possui seu ponto forte, que pode ser necessário em diferentes partes do projeto, juntamente a isso, tem-se no atual ambiente de desenvolvimento é o contínuo aumento de usuários e números de requisições solicitadas ao sistema que será desenvolvido (RODRIGUES e PINTO, 2018).

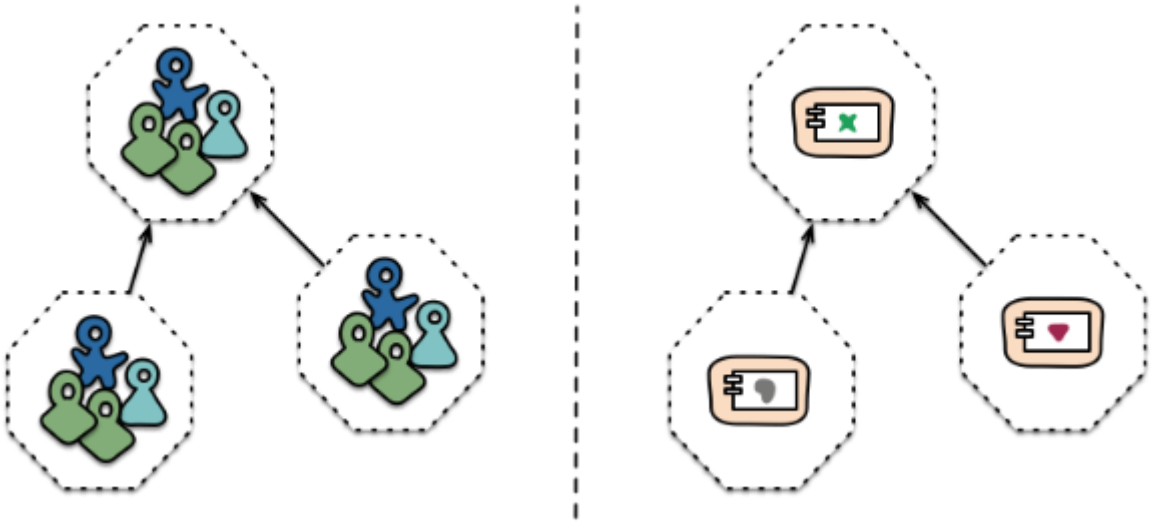
Apesar de ser novidade no mercado, é um sistema bastante discutido, pelo fato de que muitas grandes empresas e *startups* vêm utilizando esse método de desenvolvimento, como a Amazon, Spotify e Netflix.

Fowller e Lewis (2014) descrevem a arquitetura de *Microserviços* como sendo “uma abordagem para desenvolver uma única aplicação como uma suíte de serviços, cada um rodando em seu próprio processo e se comunicando através de mecanismos leves”. Dessa forma a arquitetura *Microserviços* é aplicado a “quebra” em diversos serviços que possuem uma alta coesão e baixo acoplamento, portanto, eles possuem pouca dependência ou até mesmo nenhuma de outros serviços da mesma aplicação.

Ainda segundo o trabalho de Fowller e Lewis (2014), a arquitetura de *Microserviços* apresenta algumas características únicas, sendo destaca-se dentre elas, a componentização por meio de serviços, como tal, a utilização da componentização por meio de serviços é dividir sua aplicação em componentes que se comunicarão entre si por meio das requisições (webservices ou chamadas de código remotas) de forma a transformá-los em um serviço, a sua organização através da área de negócios, também destaca-se, já que a equipe de desenvolvimento é criada para cada área de negócio deste modo ela é considerada *full stack* (em outras palavra esse termo representa um time de desenvolvimento com múltiplas habilidade ou um único indivíduo).

Na Figura 9, demonstra-se que cada equipe desenvolveu uma área de negócio inteira desde a base de dados até a sua interface.

Figura 9: Formação de equipes levando em consideração as regras de negócio



Fonte: Fowller e Lewis, 2014.

Outro termo que é importante ter conhecimento ao se falar sobre *Microserviços* é a escalabilidade. Neto (2015) define em seu trabalho que a escalabilidade como sendo "a habilidade que um sistema computacional tem de lidar, de forma transparente, com um número crescente de usuários ao mesmo tempo". Isso é relevante para a arquitetura de *Microserviços* pois em consequência da componentização por meio de serviços e a administração descentralizada de dados, esta arquitetura facilita a escalabilidade horizontal (utilização de diversos servidores, e aumento da potência de uso).

Moreira e Beder (2015) citam entre as vantagens da arquitetura de *Microserviços*: "heterogeneidade tecnológica, resiliência, escalabilidade e facilidade de implantação.". mencionam em seu trabalho ainda sobre facilidade de solucionar as falhas que podem ser apresentadas durante o desenvolvimento e utilização da aplicação.

Já sobre desvantagens os autores Moreira e Beder (2015) citam "complexidade de desenvolvimento, chamadas remotas e gerenciamento de múltiplos bancos de dados e transações", ainda no assunto, Santos (2017) diz que e necessário atentar-se para a necessidade de um excelente planejamento e documentação bem fundamentada para que haja gerenciamento correto, pois a sua complexidade pode fazer com que o sistema se torne desorganizado e de difícil entendimento da aplicação como um todo.

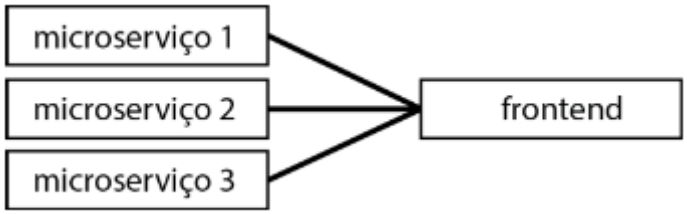
2.2.7 Aplicação Monolítica

Segundo Krishnamurthy (2018), de forma rasa, a aplicação monolítica pode ser descrita como uma única aplicação de *software* em camadas no qual a interface de usuário e código de acesso aos dados são combinados em um único programa a partir de uma única plataforma. Uma aplicação monolítica é autônoma e independente de outras aplicações de computação, uma arquitetura monolítica descreve uma aplicação de *software* de camada única, onde todos os serviços e componentes estão combinados em um único programa. Isso acaba impactando na complexidade da aplicação, visto que se cria um alto acoplamento entre estes módulos do sistema.

Já o autor Jackson (2019), diz que em aplicações monolíticas, quando diferentes times desejam trabalhar no mesmo sistema, um dos problemas que aparecem é que, devido ao alto acoplamento e complexidade existentes, podem haver conflitos nos trabalhos realizados, já que estes compartilham da mesma base de código, e isso acaba dificultando o desenvolvimento. Este tipo de arquitetura também dificulta a inclusão de novos recursos, tanto da linguagem de programação quanto de outras tecnologias (como *frameworks*), pois uma simples alteração pode comprometer o funcionamento de todo o sistema, uma vez que este utiliza uma estrutura monolítica.

Um dos principais pontos negativos, segundo Almeida (2015) é que, em aplicações monolíticas, quando ocorre um erro em determinada parte do sistema, pode acontecer de todo o restante (mesmo que não tenha nenhuma relação com a parte alterada) do sistema também parar de funcionar, o autor também cita é que, tendo uma base de código muito grande, o entendimento do sistema por novos membros da equipe acaba sendo um processo difícil, impactando na produtividade, a Figura 10 mostra um exemplo visual de uma arquitetura monolítica no *frontend*.

Figura 10: Arquitetura monolítica



Fonte: Nascimento e Sotto, 2020.

As principais linguagens de desenvolvimento de aplicações oferecem abstrações para quebrar a complexidade dos sistemas em módulos. Entretanto, são projetadas para a criação de um único executável monolítico, onde toda a modularização utilizada é executada em uma mesma máquina. Os módulos compartilham recursos de processamento, memória, bancos de dados e arquivos. Uma arquitetura monolítica típica de um sistema complexo, significa onde todas as funções do negócio estão implementadas em um único processo (NASCIMENTO e SOTTO, 2020).

Os autores, Jackson (2019), Nascimento e Sotto (2020), e Krishnamurthy (2018) apontam alguns pontos acerca das fraquezas que a arquitetura monolítica apresenta, algumas delas sendo: Complexidade, já que ao longo do tempo o sistema vai crescendo e tornando-se cada vez mais complexo, consumindo cada vez mais recursos. Surgem também alguns desafios substanciais para sua manutenção, ficando cada vez mais cara e lenta, pois os desenvolvedores têm que navegar em uma infinidade de código.

Alta dependência de componentes de código, já que muitas funções são interdependentes e entrelaçadas, de forma que a inclusão ou manutenção de componentes do sistema podem causar inconsistências ou comportamentos inesperados. A escalabilidade do sistema é limitada, e isso exige que todo o sistema seja replicado mesmo que apenas parte de sua funcionalidade seja necessária na nova instância. A falta de flexibilidade faz com que os desenvolvedores fiquem amarrados à tecnologia originalmente escolhida para o sistema, mesmo que em algumas situações não seja a melhor escolha.

Dificuldade para colocar alterações em produção, já que novas mudanças, por menor que sejam, requerem a reinicialização do sistema, incorrendo em riscos operacionais e necessitando de acompanhamento da equipe de desenvolvimento, de testes e de manutenção do sistema.

3 MATERIAIS E MÉTODOS

Nessa seção são apresentados os materiais e os métodos utilizados no fluxograma do processo de pesquisa e para desenvolvimento do aplicativo NORTI.

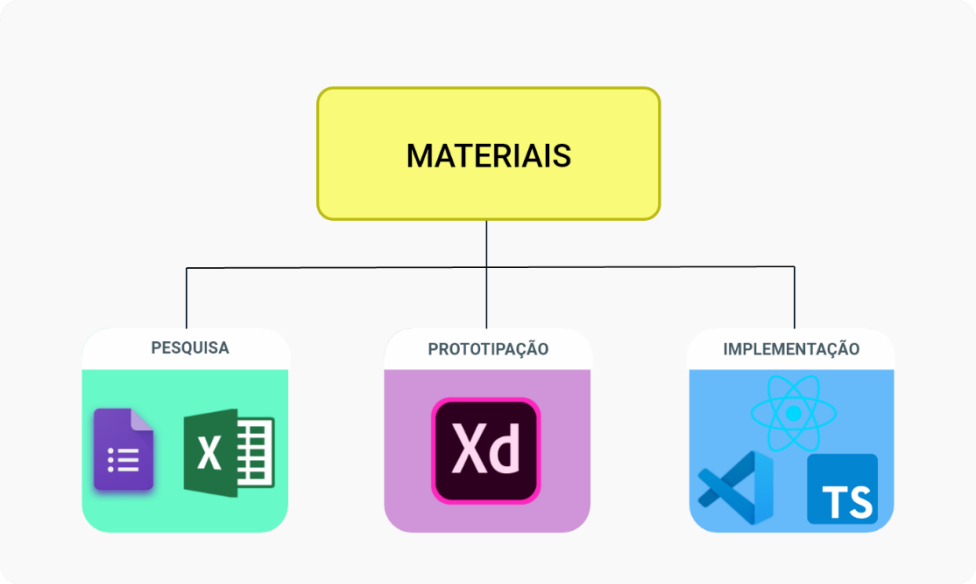
3.1 MATERIAIS

Os materiais que foram utilizados para o desenvolvimento do trabalho proposto foram divididos em três categorias: Pesquisa, Prototipação e Implementação do Aplicativo, organizados da seguinte forma:

- *Pesquisa:* Google Forms e Microsoft Excel
- *Prototipagem:* Adobe XD
- *Implementação do Aplicativo:* React Native, TypeScript e Visual Studio Code.

A Figura 11 ilustra a organização dos materiais, em seguida é apresentado os materiais em tópicos e seus respectivos papéis neste trabalho.

Figura 11: Organização dos Materiais



A Figura 11 apresenta ainda os matérias em três categorias, cada categoria possui responsabilidades diferentes dentro do processo metodológico (apresentado no tópico Métodos).

3.1.1 Google Forms

O Google Forms é uma ferramenta online, um aplicativo de gerenciamento de pesquisas lançado pelo Google e que faz parte do suíte de aplicativos online do Google, essa suíte é um conjunto de serviços online chamado *Google Workspace* (GOOGLE FORMS, 2020). O trabalho proposto fará uso desta ferramenta para criar e aplicar o questionário nas empresas. A escolha pelo Forms se dá principalmente por permitir que a pesquisa seja realizada independente da dispersão geográfica, além de conter atributos que contribuem de forma positiva para pesquisa, como: a criação de formulário com aspectos profissionais e personalizados, escolhas das várias opções de perguntas, de múltipla escolha a listas suspensas e escalas lineares.

3.1.2 Microsoft Excel

É um editor de planilhas produzido pela Microsoft para computadores que utilizam o sistema operacional Microsoft Windows, além de computadores Macintosh da Apple Inc. e dispositivos móveis como o Windows Phone, Android ou o iOS. O Excel permite além de cruzar informações de linhas e colunas, ele permite representação gráfica dos resultados para variáveis qualitativas e quantitativas, ela possui funções auxiliares prontas, que vai desde a soma de dois números a combinações de expressões regulares. Para este projeto o Microsoft Excel será utilizado na tabulação dos dados obtidos na pesquisa do projeto.

3.1.3 React Native

Trata-se de uma biblioteca moderna de interfaces de usuários para desenvolvimento de aplicativo *mobile* e *cross-platform.* O Facebook lançou o React Native em 2015 e mantém desde então, o React Native permite desenvolver aplicativos Android e iOS a partir de uma única base de código, fornecendo componentes nativos independente de qual plataforma, como visualização de texto e imagens, que mapeiam diretamente para os blocos de construção de *IU (User Interface)* nativos da plataforma (FACEBOOK, 2020). Para implementação dos códigos das aplicações, é utilizada a linguagem JavaScript por padrão, podendo fazer uso do TypeScript também. O React Native usa o *JSX*, uma sintaxe que permite escrever elementos HTML (Hyper Text Markup Language, que significa Linguagem de Marcação de Hipertexto) e CSS (Cascading Style Sheets ou simplesmente CSS) dentro do JavaScript.

Neste projeto o React Native foi utilizado para desenvolver a interface do aplicativo móvel. Entre os vários motivos pelo qual levaram a escolha dessa tecnologia para o

desenvolvimento deste projeto são: velocidade no desenvolvimento (um único código para as duas plataformas Android e iOS), por utilizar tecnologias da Web (JavaScript, HTML e CSS) para desenvolvimento móvel e por permitir TypeScript como linguagem alternativa. Embora o JavaScript seja a linguagem principal utilizada, neste projeto foi utilizado o Typescript para o desenvolvimento.

3.1.4 TypeScript

Desenvolvido pela Microsoft e lançado em outubro de 2012 na versão 0.8. É uma linguagem de programação de código aberto construída em cima do Javascript, contendo tudo do JavaScript, porém com um pouco mais de recurso (o que faz dela um *superconjunto* de javascript), adicionando definições de tipos por exemplo. Essa linguagem traz vários benefícios para o desenvolvimento de *software*, os seus recursos de tipos fornecem correções de escrita de código antes dele ser executado. Embora os tipos identifiquem erros antes da execução do código, isso não impede sua execução (TYPESCRIPT LANG, 2020).

A escolha do TypeScript para desenvolvimento do aplicativo se deu principalmente por conta de seus recursos de tipos, o que contribui para minimizar os erros na codificação, e consequentemente contribui com a produtividade do desenvolvimento.

3.1.5 Visual Studio Code

Foi desenvolvido pela Microsoft e lançado em abril de 2015. É bastante conhecido por *vscode,* um editor de código aberto, que permite desenvolver aplicações nas mais diversas linguagens de programação como Python, C++, JavaScript, TypeScript e entre outras, permite a instalação e o uso de extensões que auxiliam no desenvolvimento. Assim como o *TypeScript*, o Visual Studio Code também é um produto da Microsoft e que possui um recurso chamado de *IntelliSense*, é um termo geral para uma variedade de recursos de edição de código (VISUAL STUDIO, 2020).

O Visual Studio Code vai contribuir para codificação do código fonte da aplicação, a escolha desse editor de código se deu principalmente por seu recurso de *IntelliSense*, em uma combinação com o TypeScript onde ambas tecnologias são mantidas pela Microsoft, resultando em uma incrível experiência de produtividade.

3.1.6 Adobe XD

É um aplicativo da Adobe Inc., lançado em outubro de 2016 e mantido desde então. É uma ferramenta de design para experiência do usuário, baseada em vetor para aplicativos web

e aplicativos móveis que contribui desde a organização dos fluxos de telas, concepção do projeto, visualização do protótipo e compartilhamento. A ferramenta vai ser utilizada no desenvolvimento dos protótipos de tela do aplicativo.

3.2 MÉTODOS

A pesquisa foi realizada conforme o fluxograma da Figura 12. Cada uma das etapas é detalhada na sequência.

Figura 12: Fluxograma do processo de pesquisa

Primeiro ocorreu a revisão de literatura, baseando-se em trabalhos renomados de autores conceituados, o trabalho começa a ganhar forma realizando o mapeamento dos estudos existentes sobre desenvolvimento de *software*, além da pesquisa literária, na sequência ocorreu a definição dos critérios de avaliação das empresas. Por consequência deu início a fase de planejamento das entrevistas, neste ponto foi criado as questões que compõem o formulário eletrônico, optou-se por essa abordagem com formulário eletrônico devido à dispersão geográfica da amostra. As empresas convidadas receberam o formulário eletronicamente por contatos via e-mail ou forma de contato mais adequada aos padrões do entrevistado.

Deu-se início a fase de execução das entrevistas, seguindo algumas técnicas e ferramentas que foram utilizadas para a execução das tarefas: E-mails; vídeo conferências; ligações telefônicas; participações de eventos; aplicativo de troca de mensagens.

Ocorre então o detalhamento da pesquisa, nessa etapa, é crucial apresentar os dados coletados e identificar padrões de desenvolvimento das empresas. Na sequência a formatação e apresentação dos resultados obtidos com pesquisa.

3.3 DETALHAMENTO DA PESQUISA

3.3.1 Critérios de inclusão das empresas

Para o desenvolvimento do questionário, foi necessário definir quais os critérios de inclusão das empresas que atuam no ramo de tecnologia, nesse sentido foram definidos os seguintes critérios:

- Ter um produto ou serviço digital;
- O produto digital deve ser necessariamente um *software*, desenvolvido pela própria empresa ou parte dele;
- Possuir um time de desenvolvimento que seja formado por no mínimo dois profissionais de tecnologia.

Tais características foram apresentadas em reunião por videoconferência com a parceira AJEE Associação dos Jovens Empresários e Empreendedores do Tocantins realizada pela ferramenta *Google Meeting* juntamente com os diretores Flávio de Assis e Renan.

Esses critérios foram necessários para identificar quais empresas poderiam fazer parte desta pesquisa. Os membros da diretoria da AJEE apresentaram uma situação problema (com base em outra pesquisa realizada e aplicada por eles), onde uma empresa pode possuir um departamento de TI (Tecnologia da Informação), mas não necessariamente ser atuante no desenvolvimento de *software*. Assim, em discussão com os diretores da AJJE, onde ambos são especialistas do domínio, chegou-se ao entendimento que seria viável e interessante conceder o questionário para as empresas que atendem às descrições mencionadas anteriormente.

3.3.2 Elaboração do questionário

Para o desenvolvimento do formulário foi necessário definir uma estratégia inicial para elaboração das questões, a princípio o questionário foi estruturado e dividido em cinco principais categorias, que são elas:

- Identificação e informações básicas da entrevista;
- Caracterização da Empresa
- Desenvolvimento *software* e metodologias;
- Linguagens, *frameworks*, *cloud computing* e ferramentas;
- Profissionais, oportunidades e carreira.

Cada categoria possui um subconjunto de questões correlacionadas. Esse processo de elaboração do questionário teve como base uma pesquisa quali-quantitativa realizada com empresas palmenses em abril de 2017 por Ribeiro (2017). Além da pesquisa realizada por Ribeiro (2017), outras três pesquisas, duas de nível mundial e uma de nível nacional contribuíram para a elaboração do questionário, sendo elas a *Stack Overflow Survey 2020* realizada por Stack Overflow (site de perguntas e respostas) em fevereiro de 2020 com 65 mil desenvolvedores de *software* de 186 países ao redor do mundo.[1]

Outra referência é o *State of the Octoverse* (GITHUB, 2020), um relatório anual da comunidade de desenvolvimento de *software*. Nele é possível encontrar detalhes sobre os padrões e tendências que ajudam os desenvolvedores, equipes de trabalho e organizações empresariais.[2]

Por último, a pesquisa de nível nacional da ABStartup (Associação Brasileira de Startups), realizada no ano de 2020 com *startups* em seus vários *nichos* (áreas) de atuação no Brasil. Esta pesquisa foi útil para auxiliar na elaboração de algumas questões e alternativas relacionadas a categoria *caracterização da empresa*.[3]

O questionário completo está disponível no Apêndice A, todas as questões são de múltipla escolha podendo obter uma única resposta dentre várias alternativas ou mais de uma resposta dentre as alternativas da questão. O número de questões por categoria não segue uma quantidade mínima ou máxima, uma amostra do questionário é apresentada a seguir:

1. Qual é o modelo de receita da empresa?
    a. API
    b. Consumer
    c. E-commerce
    d. Marketplace

[1] Stack OverFlow Survy 2020 - https://insights.stackoverflow.com/survey/2020
[2] State of the Octoverse 2020 - https://octoverse.github.com/
[3] Associação Brasileira de Startups - https://startupbase.com.br/home/stats

    e. Outro
2. Qual o estilo do processo de desenvolvimento de software adotado pela empresa?
    a. Processo de desenvolvimento de *software* tradicional
    b. Processo de desenvolvimento de *software* ágil
3. Ao adotar metodologias ágeis nos projetos de software, quais foram os principais benefícios alcançados? (se necessário, selecione mais de uma opção)
    a. Transparência nas atividades realizadas pelo time
    b. Entendimento sobre o projeto
    c. Uso de artefatos para entendimento do problema
    d. Uso de diagramas para compreender a solução (o projeto)
    e. Documentação de fácil compreensão
    f. Redução de falhas
    g. Nenhum
    h. Outro
4. A empresa utiliza algum dos *frameworks backend* em seus projetos?
    a. .NET Core
    b. ASP.NET
    c. CakePHP
    d. CodeIgniter
    e. Django
    f. Express
    g. Laravel
    h. Outro

Outra característica do formulário é a ordenação das alternativas de cada questão ser apresentadas por ordem alfabética.

3.3.3 Aplicação do questionário

A aplicação do questionário ocorreu por meio do formulário eletrônico do Google Forms, todas as questões estão disponíveis neste formulário. A distribuição deste questionário foi realizada por estes meios:

- E-mail;
- Ligações telefônicas;

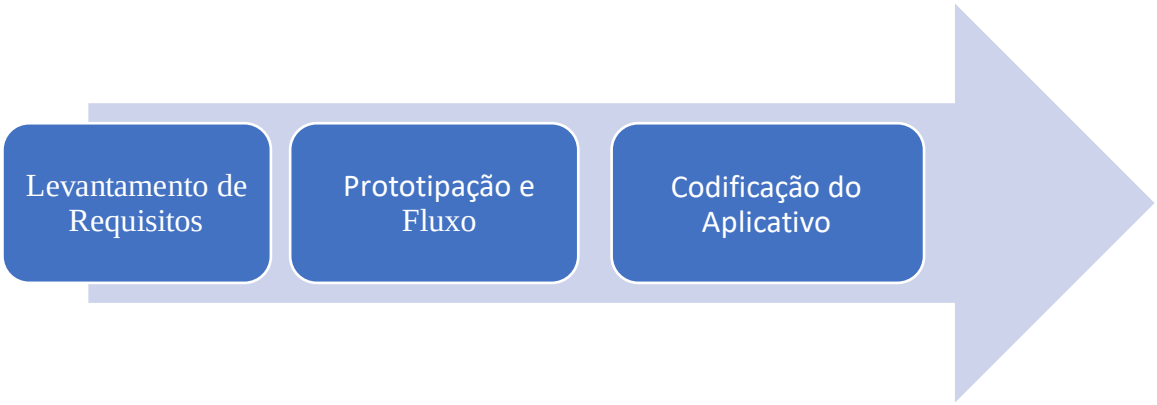
- WhatsApp (Aplicativo de mensagens instantâneas);
- Telegram (Aplicativo de mensagens instantâneas);
- Microsoft Teams (Plataforma unificada de comunicação e colaboração que combina bate-papo e videoconferências); e
- Google Meet (Serviço de comunicação por vídeo e bate-papo).

Estes foram os meios de comunicação utilizados durante as abordagens com as empresas convidadas, destas as ligações telefônicas e WhatsApp foram as mais utilizadas.

3.3 DETALHAMENTO DO DESENVOLVIMENTO DO APLICATIVO

Nesta etapa, são apresentadas todas as fases do processo de desenvolvimento do aplicativo. O fluxo apresentado na Figura 13 descreve a sequência destas fases.

Figura 13: Etapas para o desenvolvimento do aplicativo

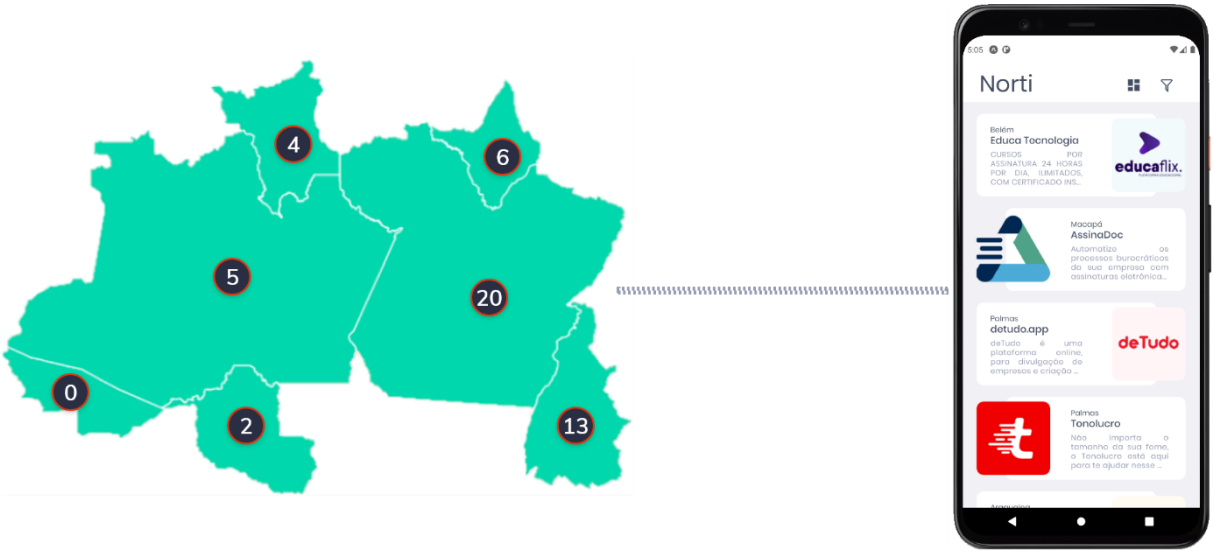


A Figura 13 apresenta os passos de levantamento de requisitos, que se resume no processo inicial do projeto de desenvolvimento do aplicativo, na sequência vem a criação do prototipagem do aplicativo e fluxo do usuário. Por último é apresentado a codificação, onde acontece a implementação do aplicativo móvel. Essas etapas foram fundamentais para um bom entendimento da solução que se pretendia desenvolver, NORTI.

4 RESULTADOS E DISCUSSÃO

Esta seção tem por objetivo descrever e apresentar o trabalho realizado para execução das etapas do fluxo do processo de pesquisa e desenvolvimento do aplicativo. Após o estudo detalhado sobre metodologia de desenvolvimento de *software* e arquiteturas de *software* e as tecnologias a serem utilizadas, essa seção traz consigo as justificativas para cada tomada de decisão realizada e os seus respectivos resultados obtidos durante e execução das etapas.

Figura 14 :Visão geral dos resultados



O trabalho através do questionário alcançou 6 dos 7 estados que compõem a região Norte do Brasil, com ausência apenas do Acre (Figura 14), e a maior presença de empresas atingidas foi no estado do Pará, seguido por Tocantins, Amapá, Amazonas, Roraima e Rondônia.

4.1 ANÁLISE DAS RESPOSTAS DO QUESTIONÁRIO

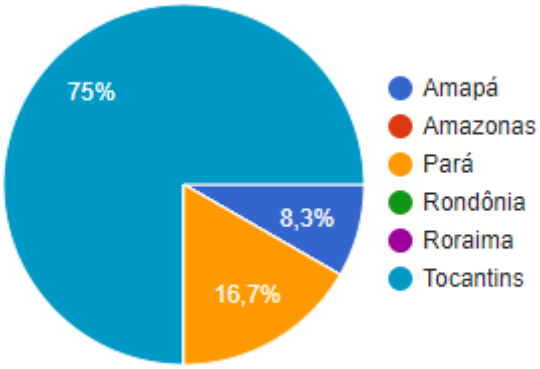
O questionário foi selecionado para ser aplicado em 50 empresas. O critério de escolha adotado foi seguido, as empresas foram contatadas, porém nem todas deram retorno, seja por motivos pessoais ou profissionais, o estudo não abordou a causa das escolhas. As seções a seguir apresentam os achados mais relevantes a partir das análises das respostas ao questionário.

4.1.1 Caracterização da Empresa/Organização

Das 50 empresas listadas para a aplicação do questionário, sendo 2 de Rondônia, 6 do Amapá, 5 do Amazonas, 4 de Roraima, 20 do Pará e 13 do Tocantins, 18 retornaram o contato informando ter recebido o questionário para resolução. Destas, 14 responderam de fato, totalizando uma participação de 28% do número amostral inicial, onde o Tocantins se destaca

por participações (Figura 15), liderando as respostas, se encontra Palmas compondo 57,1% dos dados amostrais.

Figura 15: Participação dos estados

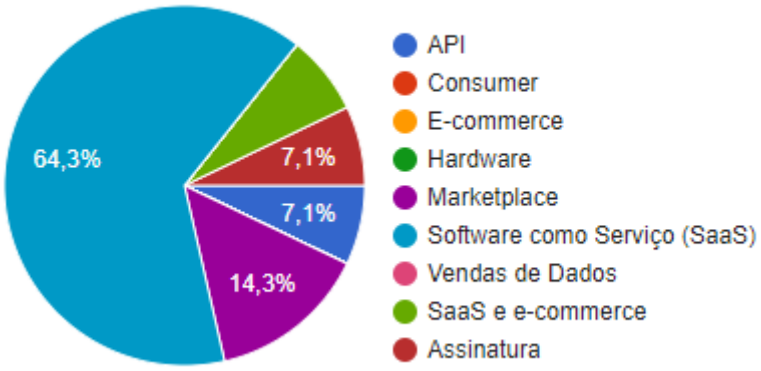


As empresas com sede nos municípios de Palmas e Belém (21,4%) demonstraram mais interesse em responder às perguntas, também foram as que mais facilmente se mantiveram em contato.

Tornou-se evidente a grande participação de CEOs (*Chief Executive Officer*), totalizando 33,8% dos representantes que responderam, desenvolvedores ocupam a segunda colocação com 24,9%. Esses resultados demonstram o interesse dos líderes pelos dados que serão expostos no aplicativo NORTI.

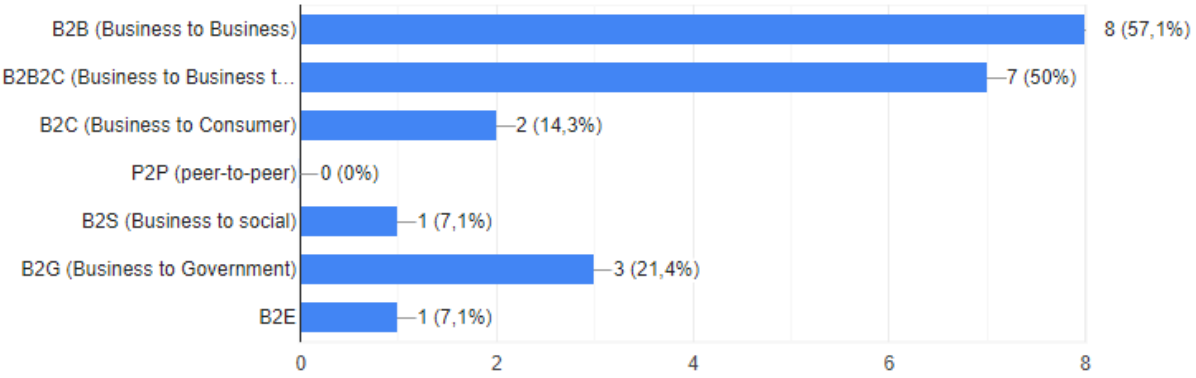
Na monetização dos serviços e produtos digitais, destaque para SaaS (Software como Serviço) que representa 84,3% das escolhas como modelo de receita mais escolhido entre as empresas (Figura 16).

Figura 16: Modelo de receita das empresas



A Figura 17 representa as respostas sobre público-alvo. Os modelos B2B (*Business to Business*) e B2C (*Business to Consumer*) ocupam as duas primeiras posições entre as opções mais escolhidas. O primeiro é um modelo de negócio em que o cliente final é outra empresa, enquanto no segundo, é uma pessoa física.

Figura 17: Público-alvo das Empresas

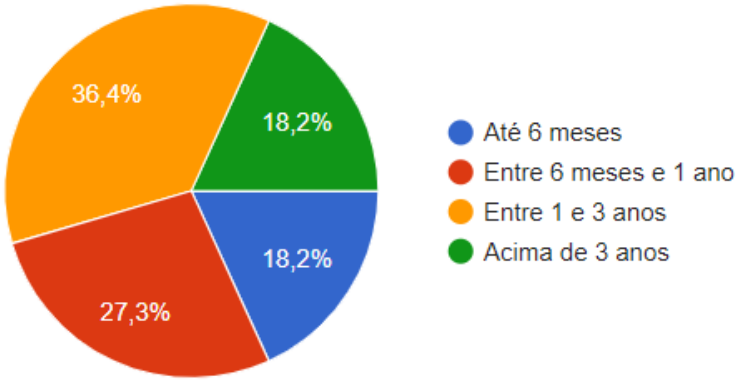


O modelo de negócio B2B tem como princípio a relação comercial de compra e venda de produtos ou serviços entre empresas, oposto ao modelo B2C, a venda B2B envolve apenas pessoas jurídicas. Já o modelo B2B2C (*Business to business to consumer*) é uma organização em cadeia onde a venda B2B tem uma dependência ligada à venda B2C, como por exemplo os *Marketplaces* onde as empresas oferecem um espaço para que outras empresas/pessoas vendam seus produtos para o consumidor final.

4.1.2 Desenvolvimento de software e metodologias

Nesta subseção são apresentados três resultados sobre o desenvolvimento de software com metodologias ágeis. A Figura 18 apresenta a quanto tempo as empresas trabalham com desenvolvimento de software. A maioria escolheram a opção “Entre 6 meses e um ano” com 36% da preferência.

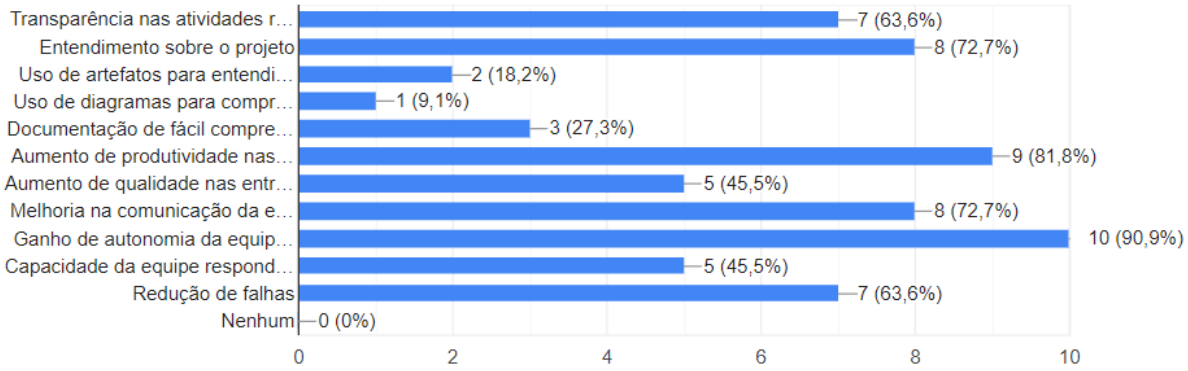
Figura 18: Tempo de desenvolvimento de software



A Figura 18 apresenta dados que preocupam tanto os profissionais quanto as empresas, mostrando que a empresa não consegue manter o profissional de TI por muito tempo em sua equipe, o que prejudica projetos a longo prazo, pois sua equipe se torna bastante volátil.

Sobre a temática metodologia ágeis (Figura 19), é notável que contar com tais métodos em processos de desenvolvimento de *software* tem vários benefícios .

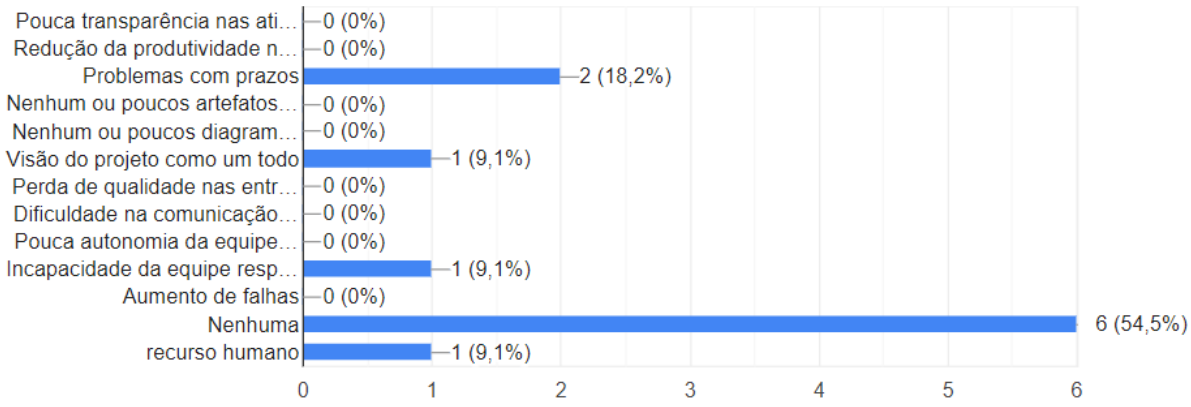
Figura 19: Vantagem das metodologias ágeis



Como destacado na Figura 19, as empresas que optaram por seguir metodologia ágeis apresentaram satisfação, principalmente sobre como suas equipes ganharam mais autonomia em seus projetos e sobre o aumento de produtividade apresentado por eles. Nesse aspecto uma característica que possibilita tais aumentos é, adaptabilidade do time e a rápida resposta as mudanças, destaque para as metodologias Scrum e XP.

Complementando os dados na Figura 19, a Figura 20 demonstra que mais de 54% das respostas contemplam apenas benefícios para seus métodos de atuação, como é possível uma mesma empresa ter mais de uma resposta, acredita-se que a mesma empresa possa ter tido mais de um problema. As metodologias ágeis apresentam uma melhor aceitação em relação a outras metodologias.

Figura 20: Desvantagens das metodologias ágeis

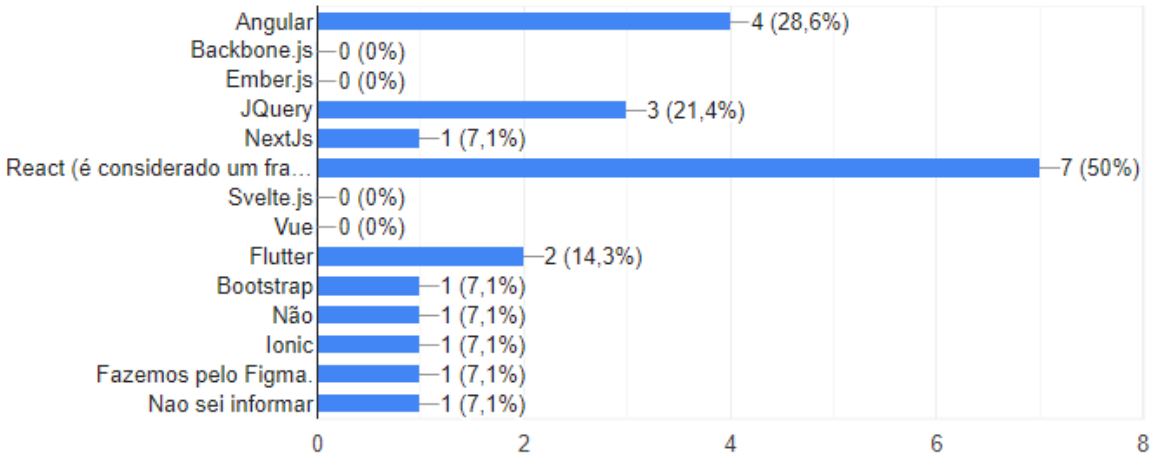


É possível notar que, embora seja muito satisfatório os benefícios ao adotar as práticas de metodologias ágeis, a pesquisa também apresenta o item "*Problemas com prazos*" recebendo 18,2% das escolhas das empresas (Figura 20).

4.1.3 Linguagens, frameworks, *cloud computing* e ferramentas

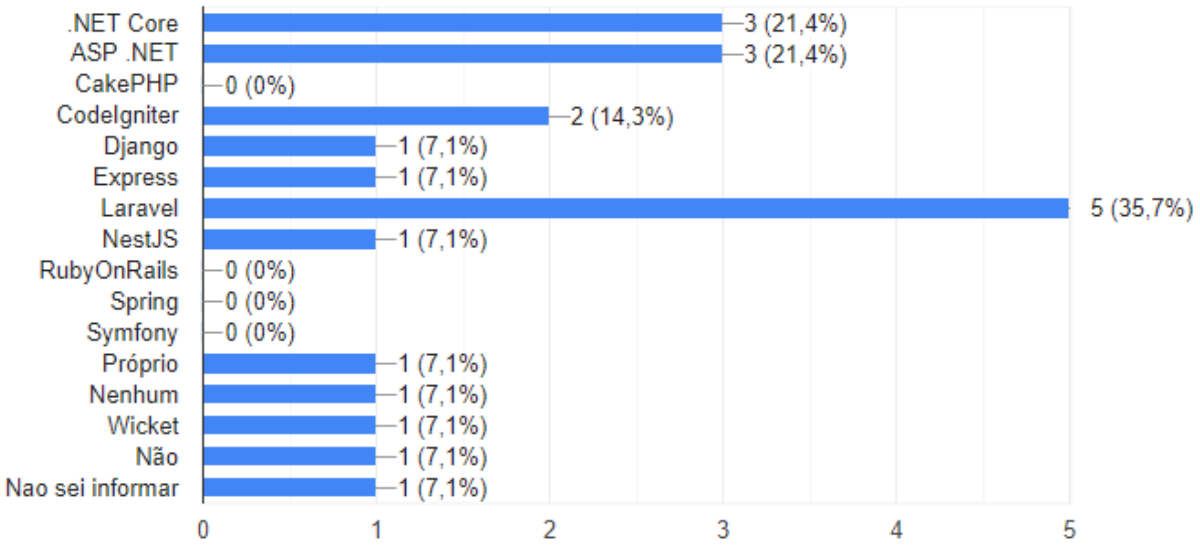
Os *frameworks frontend*, são ferramentas que auxiliam na criação de interfaces e na forma como os dados são exibidos. Por isso, eles se tornam muito relevantes, já que facilitam o processo de desenvolvimento trazendo diversas soluções já pré-definidas. Os números do *Stack Overflow Survey* (2020) revelam o *JQuery* como o *framework frontend* mais popular e aceito entre os desenvolvedores de *software*, registrando a liderança com seus 43.3% da preferência. Entretanto, conforme a pesquisa realizada neste trabalho, os frameworks que ganharam destaques foram o React, com 50% das escolhas, seguido pelo Angular, que aparece com 28.6% (Figura 21).

Figura 21: Frameworks frontend



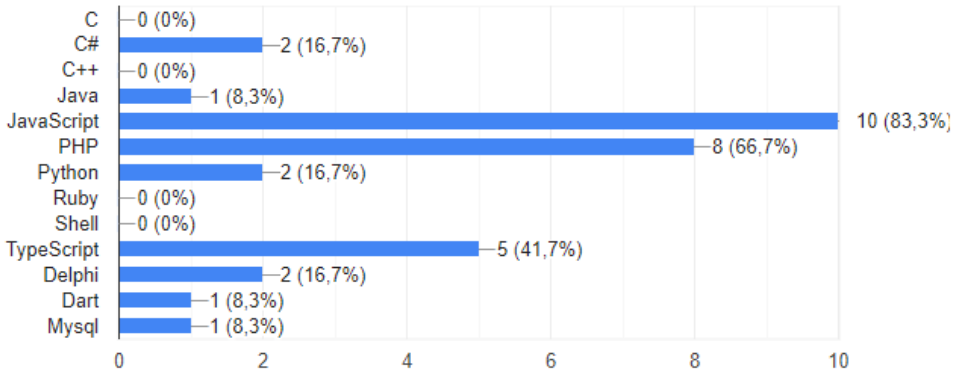
Os *frameworks backend* são uma estrutura de *software* de uma base onde os desenvolvedores podem fazer aplicativos de uma forma mais rápida e padronizada, sendo um de seus objetivos a economia de tempo, escalabilidade, robustez, segurança e integrações. As tecnologias em destaque foram Laravel, com 35%, .NET e ASP.NET, da Microsoft, ambas com 21,4% (Figura 22).

Figura 22: Frameworks backend



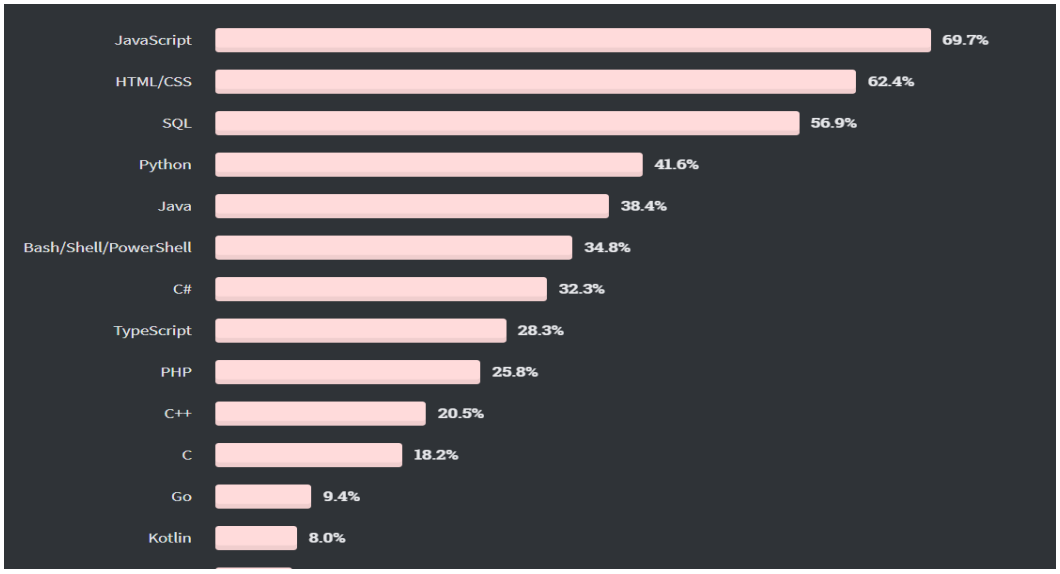
Outro fator que mostrou relevância, foi o uso da linguagem de escrita dos códigos (Figura 23), com a presença do *JavaScript* com mais de 83% como linguagem mais frequente nos projetos de *software*. Esse resultado demonstra que as empresas da região norte do Brasil estão coerentes com o direcionamento sobre as tecnologias mais aceitas por desenvolvedores no mundo, segundo os valores apresentados pela pesquisa *Stack Overflow Survey* (2020). Aprofundando um pouco mais nos resultados, aparece o PHP como segunda linguagem, demonstrando sua alta aceitação, mas contrariando o cenário mundial, descrito na Figura 23.

Figura 23: Linguagens de programação utilizadas no desenvolvimento de software



A Figura 24 apresenta uma inversão acentuada na escolha das tecnologias PHP e Typescript se comparadas com o *ranking* mundial (*Stack Overflow Survey*, 2020). Por um lado, o PHP obteve ganhos significativos, somando um total de 163,08% a mais em relação ao ranking mundial, por outro lado, o Typescript aparece com perdas moderadas e ocupando a terceira posição na lista. Além disso, C#, Delphi e Python aparecem empatados ambas na quarta colocação.

Figura 24: Linguagens de programação mais utilizadas no mundo



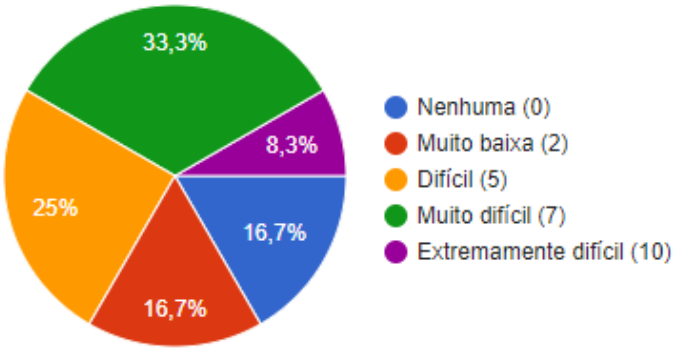
Fonte: Stack Overflow Survey, 2020.

Na Figura 24 é demonstrado o uso de Linguagens de Programação em nível mundial, nem todas as opções de linguagem de códigos foram apresentada. Entretanto é possível obter mais informações sobre a pesquisa, no link apresentado nas seções anteriores.

4.1.3 Profissionais, oportunidades e carreira

Um resultado que chamou a atenção foi a questão sobre o nível de dificuldade para contratar um profissional na área de tecnologia, visto que a questão fazia referência ao profissional responsável por codificar as aplicações, conhecido popularmente como desenvolvedor de *software*. Conforme apresentado na Figura 25, mais de 66% das empresas consideram difícil, muito difícil ou extremamente difícil a contratação deste profissional, demonstrando também a escassez desse profissional da área no mercado de trabalho.

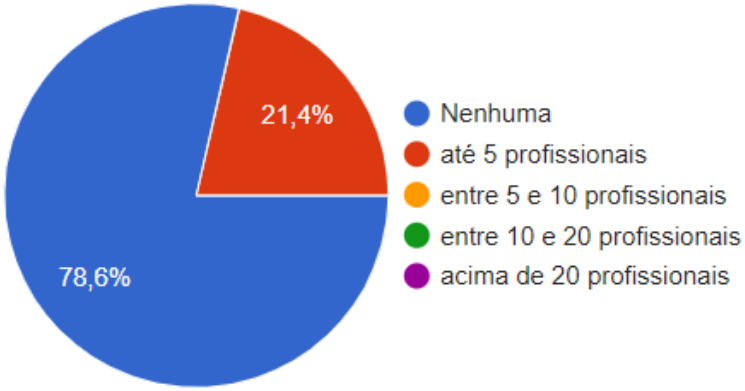
Figura 25: Nível de dificuldade para contratar um profissional na área



Brasscom (2019) realizou uma pesquisa em que aponta que, só no Brasil, a área de TI vai requerer 420 mil novos profissionais até 2024 e que, só no segmento de TICs (Tecnologias da Informação e Comunicação), existem 845 mil empregos em 2019, desses, 42 mil foram criados no ano anterior. Segundo as projeções da pesquisa, a necessidade vai saltar de ano em ano, até que de 2023 para 2024, 70 mil novos profissionais sejam requisitados (BRASSCOM, 2019).

O questionário apresenta ainda o lado social dos entrevistados (as empresas), demonstrando que 78,6% das empresas não possuem nenhuma mulher em sua equipe de profissionais de TI, mais precisamente na área de desenvolvimento.

Figura 26: Time de desenvolvimento é formado por quantas mulheres?



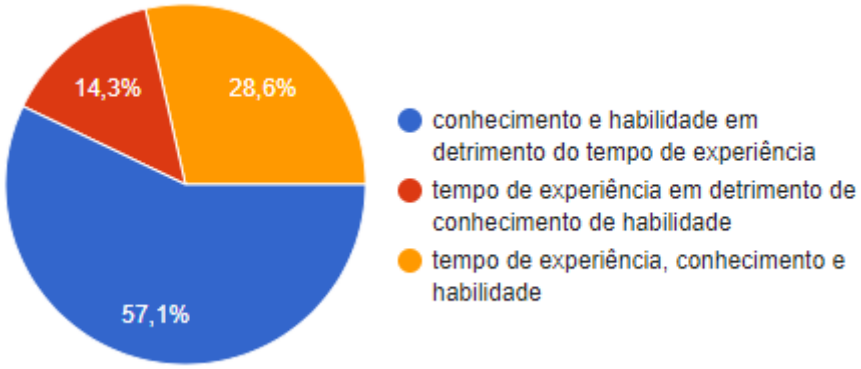
A Figura 26 revela o índice ruim para região Norte do Brasil, visto que há algumas décadas as mulheres contribuíram e vem contribuindo diretamente para a evolução do desenvolvimento de software. É o caso da Augusta Ada King, uma personagem marcante para a história computacional, sendo considerada frequentemente como a primeira criadora de programa de computador da história, ainda no século XIX (BIOGRAPHY, 2014). Outras personagens que contribuíram bastante com seus conhecimentos foram, Betty Snyder, Marlyn Wescoff, Fran Bilas, Kay McNulty, Ruth Lichterman e Adele Goldstine eram responsáveis pela configuração do ENIAC (considerado primeiro computador da história), dando a ele as instruções para realizar os cálculos necessários (DEMARTINI, 2016). Nesse sentido houve uma perda tanto computacional quanto social, levando em conta as contribuições passadas que proporcionaram ganhos tecnológicos que estão presentes até hoje.

ItGirls é um projeto dos cursos de Sistemas de Informação e Ciências da Computação do CEULP/ULBRA que objetiva inspirar jovens mulheres a participarem da área da Tecnologia da Informação (ITGIRLS, 2016). Projetos desta natureza possuem um papel fundamental para contribuir com a redução deste índice ruim. ItGirls apresenta iniciativas que vão desde mostrar a computação às meninas ainda no ensino fundamental e médio até o desenvolvimento de códigos e criação de novos paradigmas de programação. Desta forma, contribui com o aumento do número de profissionais mulheres na TI.

Ainda no quesito social, é fato que as empresas entrevistadas em sua maioria (mais de 50%) possuem menos de 6 profissionais de TI contratados, não possuem um plano de carreira atualmente ou estão em desenvolvimento. Tal como não possuem nenhum modelo de capacitação interna para os profissionais de tecnologia, demonstrando a dificuldade em manter e seguir com profissionais em potencial.

Na Figura 27 é apresentado os critérios mais cobrados pelas empresas na hora da contratação de novos colaboradores.

Figura 27: Processo de contratação



De acordo com Dias (2021), o Recursos Humanos possui uma área chamada de recrutamento e seleção, responsável pela seleção de profissionais para que possam ocupar as vagas solicitadas por uma empresa (DIAS, 2021). Nesse caso é comum que empresas optem por profissionais que já possuam experiência, desse modo economiza tempo e dinheiro na capacitação destes indivíduos, a Figura 27 mostra que esta abordagem é a preferência de 57.1% das empresas.

4.2 DESENVOLVIMENTO DO APLICATIVO

Esta etapa descreve as tomadas de decisões realizadas durante o desenvolvimento do aplicativo.

4.2.1 Levantamento dos requisitos

O processo de levantamento de requisitos foi realizado na tentativa de compreender as necessidades do cliente. Ressalta-se ainda, que o trabalho possui como público-alvo: empresas que possuem em sua estrutura uma área de TI atuante. Porém não é destinado a um único cliente final para quem será direcionado este aplicativo, mas todo e qualquer indivíduo ou empresa que queira ter acesso a tais informações.

Desta forma o levantamento de requisito foi iniciado em reuniões pontuais com membros da AJEE Tocantins juntamente com o orientador deste trabalho, e de forma informal, cada participante manifestou seu entendimento sobre o que esperava do aplicativo. Assim, os requisitos do aplicativo são apresentados pela Tabela 2.

Tabela 2: Requisitos

ID	Requisitos Funcionais	Descrições
R1	Ativar uma opção de filtro	A funcionalidade deve **ativar** uma (apenas uma) opção de filtro (Tocantins, Delivery, Scrum).
R2	Desativar uma opção de filtro	A funcionalidade deve **desativar** uma (apenas uma) opção de filtro (Tocantins, Delivery, Scrum).
R6	Selecionar todas as opções de filtros ativos por categorias	A funcionalidade deve buscar todas as opções de filtros por categorias que estão ativas
R7	Aplicar filtros na lista de empresas	A funcionalidade deve filtrar a lista de empresas de acordo com todas as opções de filtros ativos
R8	Limpar filtros por categorias	A funcionalidade deve **desativar** todas as opções de filtros ativos de uma (apenas uma) categoria
R9	Limpar filtros de todas as categorias	A funcionalidade deve **desativar** todas as opções de filtros de todas as categorias
ID	**Requisitos Não Funcionais**	**Descrições**
RN1	Buscar dados das empresas	Funcionalidade que realiza a busca dos dados do questionário na base de dados
RN2	Listar empresas	Funcionalidade que lista todas as empresas
RN3	Verificar se existe alguma categoria com filtro aplicado	Funcionalidade que verifica a existência de algum filtro, em alguma das categorias
RN4	Exibir quantidade de filtros aplicados por categorias	Funcionalidade que exibe o número de categorias com filtros aplicado

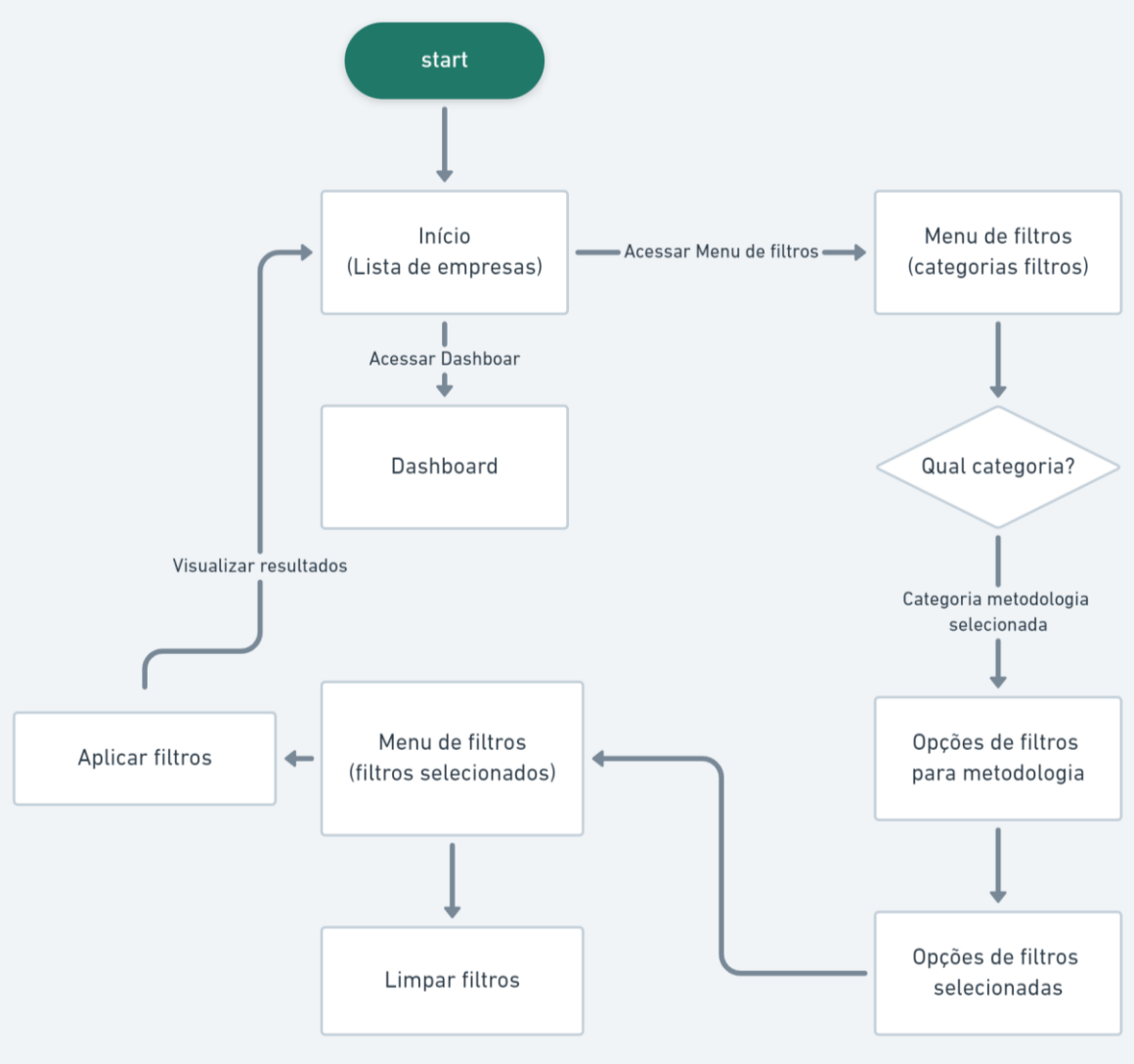
Os requisitos apresentados detalham os recursos principais no aplicativo, estes requisitos passaram pelo processo de elicitação, a elicitação de requisito em linhas gerais pode ser compreendida como as condições necessárias para alcançar um determinado objetivo.

4.4.2 Prototipação e Fluxo

Para desenvolvimento do protótipo foi necessário estruturar os requisitos para poder compreender em qual momento cada ação deveria ser realizada. A técnica de *User Flow* (Fluxo de Usuário) facilitou bastante o entendimento, esta é uma forma organizada e sequencial de ilustrar os passos da jornada do usuário para realizar uma tarefa específica dentro do aplicativo, NORTI.

Na Figura 28 é apresentado o fluxo necessário para aplicar as opções de filtros da categoria metodologia.

Figura 28: NORTI - Fluxo de usuário para aplicar filtros de metodologia

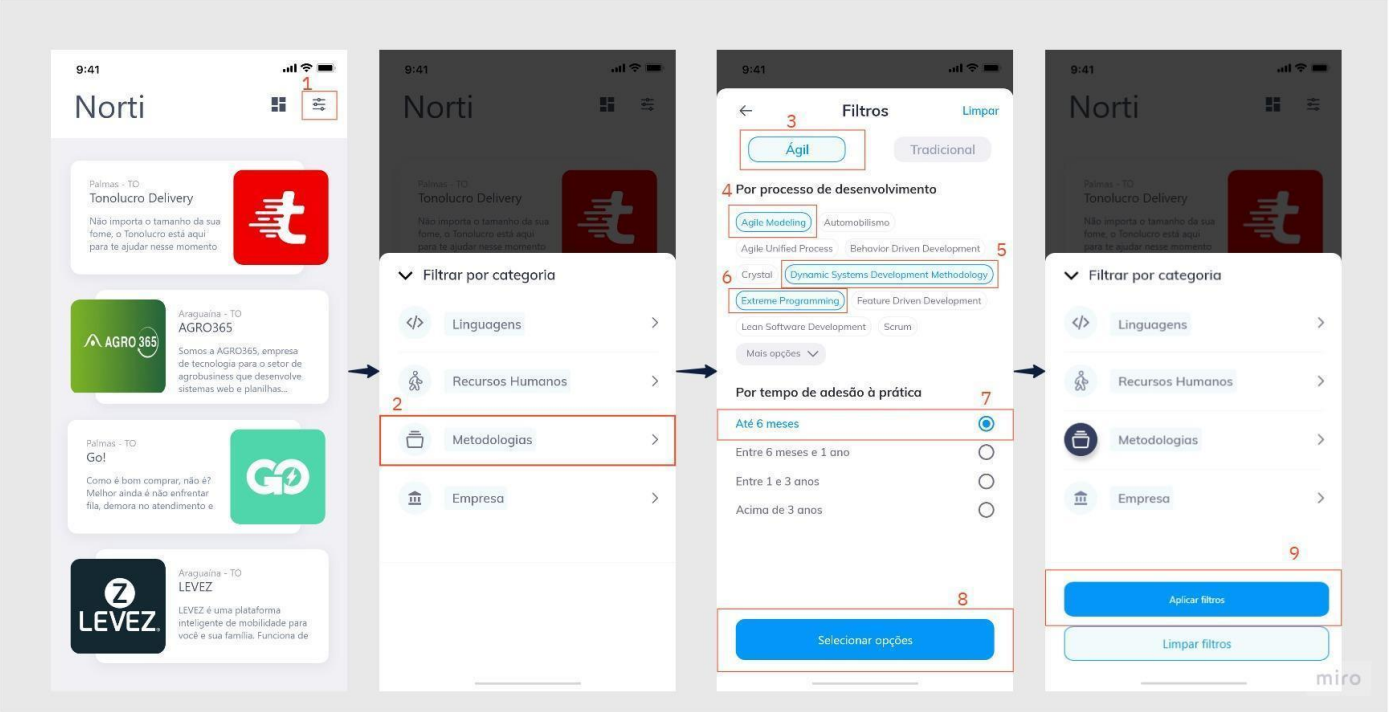


O desenho do fluxo apresentado acima trouxe ao processo de desenvolvimento, um entendimento mais completo sobre projeto NORTI, os fluxos de usuário (ou jornada usuário) enriquece a visão como um todo do projeto.

Esse fluxo é iniciado na “Lista de empresas”, em seguida o usuário acessa o “menu de filtros”, na sequência a categoria “metodologia” é selecionada, as opções de filtros são escolhida e confirmada em “Aplicar filtros”, por último retorna a “Lista de empresas” com resultado do filtro aplicado.

Para finalidade de desenvolvimento, ilustrações e interações do aplicativo foi criado um protótipo de alto nível, apoiando-se principalmente no fluxo de usuário apresentado na Figura 29.

Figura 29: Interações de tela do protótipo NORTI



A Figura 29 apresenta as interações de telas necessárias para o usuário realizar a tarefa de filtrar por metodologias, o fluxo apresentado na Figura 28 auxiliou no entendimento das interações de cada tela que compõe essa tarefa. Esse fluxo de telas será apresentado com mais riqueza de detalhes na seção *Componentes e telas no aplicativo NORTI.*

É importante mencionar que as mesmas categorias apresentadas na seção de *detalhamento da pesquisa*, também foram utilizadas no aplicativo NORTI, entretanto os nomes de cada categoria foram adaptados, tornando simples a visualização destes nome em tela. A Tabela 3 apresenta essa adaptação.

A tela *Home* é responsável por exibir as empresas cadastradas no aplicativo. A tela de menu de filtros passo 2 apresenta quatro categorias principais, as categorias dos filtros possuem a mesma representatividade que as categorias do questionário, entretanto os nomes de cada categoria foram adaptados no NORTI para tornar simples a visualização em tela, assim como a Tabela 3 apresenta.

Tabela 3: Nomes das categorias no questionário e no aplicativo

Nomes das categorias	
Questionário	**Aplicativo NORTI**

Caracterização da Empresa;	Empresa
Desenvolvimento *software* e metodologias	Metodologias
Linguagens, *frameworks*, *cloud computing* e ferramentas	Linguagem
Profissionais, oportunidades e carreira	Recursos Humanos

Essa escolha foi realizada em conformidade com a documentação do material *design* para *android*, mais especificamente na coleção de componentes que aborda a anatomia dos *Buttons* (botões no *android*), a documentação sugere que um *Button* possua nome simples, curto e que não ultrapasse mais que uma linha [3].

Dando continuidade no tema de iteração de telas, do passo 3 até o passo 7 são adotadas as escolhas de opções de filtros da categoria metodologia, já no passo 8 estas escolhas são confirmadas, e por fim no passo 9 o filtro é aplicado a lista de empresas.

4.4.3 Estrutura do projeto e codificação do aplicativo

Para codificação do aplicativo, NORTI, foi utilizado o protótipo como principal referência.

O NORTI é um projeto *open source*. Segundo GitHub (2021) isso significa que qualquer um pode ver, usar, modificar e distribuir o projeto. Desta forma o aplicativo NORTI pode ganhar contribuições de profissionais TI de qualquer lugar do mundo, se transformar em algo maior e receber novas atualizações ou correções de erros. Por esses motivos, o código-fonte está disponível para contribuições da comunidade de desenvolvimento *software*, por meio do site do GitHub.[4]

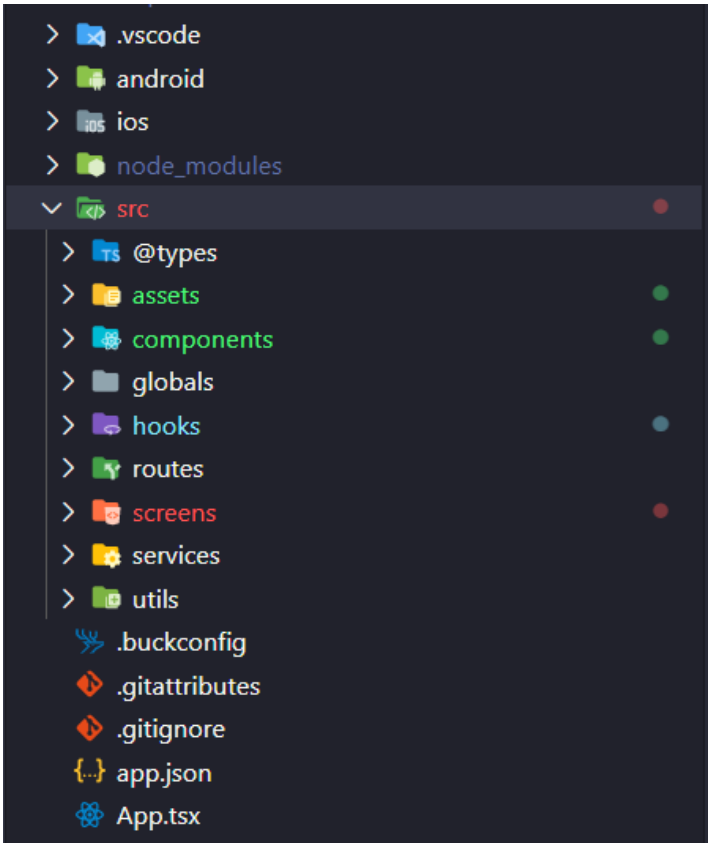
Figura 30: Iniciando projeto React Native com typescript

```
npx react-native init norti-app --template react-native-template-typescript
```

A linha de comando apresentada na Figura 30 é o ponto de entrada para o desenvolvimento com React Native. O React Native é bastante flexível no que diz respeito à organização de projeto, desta forma existem várias maneiras de organizá-lo.

[4] Projeto Open Source – NORTI: https://github.com/lucaspedronet/norti-app

Figura 31: Estrutura de pastas e arquivos



A Figura 31 apresenta a estrutura e organização de pastas e arquivo do aplicativo NORTI, na sequência essa organização é apresentada com mais detalhes:

- **android** - Responsável por armazenar todas as informações relacionadas ao código fonte da plataforma Android. Dentro deste diretório existem outras subpastas e arquivos como por exemplo: **.gradle, *.java e *.xml.*
- **ios** - Do mesmo modo que a pasta **Android**, aqui contém as informações relacionadas ao código fonte da plataforma iOS. Dentro deste diretório estará o projeto xcode, arquivos como por exemplo: **.plist, *.h, *.m,* e outros.
- **src** - Este diretório foi criado de forma manual e é nele que todas as subpastas apresentadas a seguir foram armazenadas.
- **src/@types -** Esta pasta é essencial para projetos que trabalham com Typescript, é nesta pasta que se consegue declarar tipos para bibliotecas que porventura não possui uma tipagem própria. O desenvolvedor consegue por meio desta pasta declarar tipos de dados para uma biblioteca específica.
- **src/assets** - Responsável por armazenar todos os arquivos de imagens do projeto, neste projeto todas as imagens foram armazenadas no formato *.png* (Portable Network Graphics).
- **src/components** - Essa pasta é comumente utilizada pela comunidade do React para guardar todos os componentes globais da aplicação, é nela que o componente *BusinessCardLeft* está armazenada, por exemplo. Somente os componentes que são reutilizáveis estão armazenados nesta pasta, aproveitando ao máximo a reutilização de código.

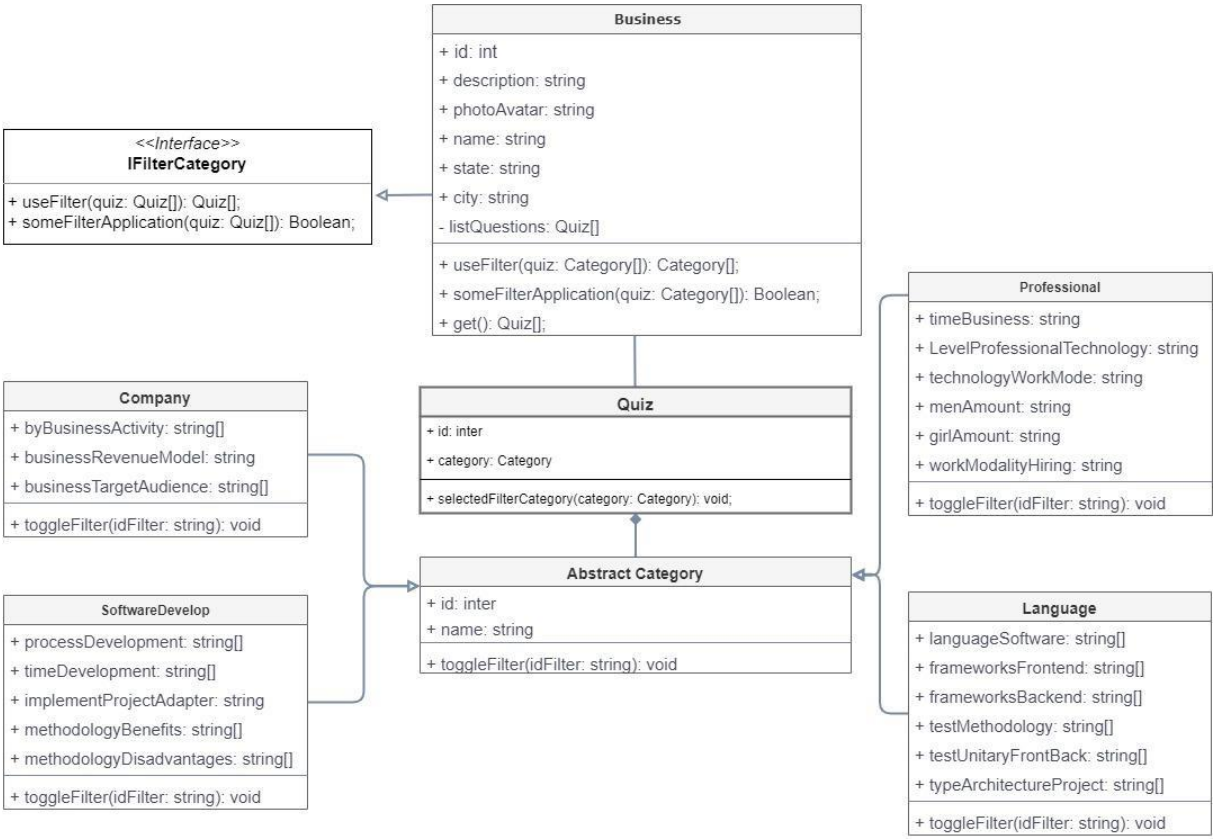
- **src/globals** - Neste repositório é armazenado os arquivos globais que estão ligados diretamente ao tema e estilização do aplicativo NORTI. Métricas de estilização de componentes e de telas são compartilhados pelos seguintes arquivos: *color.ts, theme.ts e styled.d.ts*.
- **src/hooks** - Nesta pasta são armazenados os arquivos de hooks personalizados para o NORTI, os Hooks permitem extrair a lógica de um componente em funções reutilizáveis. Desta forma é possível ter o máximo de reaproveitamento de código, são exemplos de arquivos hook no projeto: *useBusiness.tsx, useFilterBusiness.tsx e useModal.tsx*
- **src/routes** - Esta pasta armazena todos os arquivos de navegação, nela contém o seguinte arquivo: *app.router.tsx*.
- **src/screens** - Responsável por guardar todos os arquivos e pastas que são mantidos como telas na aplicação, esses componentes de telas também são utilizados dentro dos arquivos de rotas.
- **src/services** - Essa pasta é a única que possui arquivos responsáveis por realizar o acesso direto à base de dados do NORTI, por meio do arquivo *data.json* o hook *useBussiness.tsx* consegue fazer a leitura dos dados e mantê-los no estado global da aplicação.
- **src/utils** - Na pasta **utils** é guardado os arquivos úteis na aplicação, mas que não se enquadra no contexto das demais pastas vista anteriormente, como por exemplo: *iconsFilterCategory.ts*.

Todos os arquivos e pastas dentro do repositório *src* foram criados pelo autor, o arquivo *App.tsx* é o arquivo principal da aplicação, sendo este o primeiro a ser inicializado. Os componentes contidos nas pastas *screens* e *components* possui dois arquivos, o primeiro é o *index.tsx* responsável pela estrutura do componente, o segundo é o arquivo *styles.ts* responsável pelos estilos.

4.4.4 Estrutura da informação e modelagem

Nesta seção é apresentado a estrutura da informação, utilizando-se de recursos da Linguagem de modelagem unificada (UML) para ter uma representação de modelagem do sistema (NORTI). Desse modo, foi criado um diagrama de classe com intuito de ilustrar o modelo de dados, entender melhor a visão geral da aplicação, assim como expressar visualmente as características específicas do NORTI.

Figura 32: Diagrama de classe



Business é a principal classe do diagrama acima (Figura 32), *Business* traz um conjunto de atributos, destes o mais importante é o *"listQuestions"* uma lista que guarda todas as respostas da empresa. O diagrama faz uso de conceito importante para codificação e representação de modelagem de sistema, é o caso de *herança*, *composição*, *interfaces e abstração*. A classe *Abstract Category* fornece seus atributos (modelo) base para suas subclasses, esse recurso é chamado de herança, ainda na classe *Category* é possível identificar um relacionamento de *composição* com *Quiz*, mostrando que a class *Quiz* necessita da *Category*.

As Tabelas 4 a 10 apresentam a estrutura e organização dos dados para cada classe no diagrama.

Tabela 4: Estrutura e organização dos dados da classe Business

Business		
Atributos/Métodos	**Descrição**	**Tipo**
name	Nome da empresa	String
state	Estado que empresa foi criado	String
city	Cidade da empresa	String
photoAvatar	URI da Logo da empresa	String
quiz	Respostas da empresa	Array Category

Tabela 5: Estrutura e organização dos dados da classe Quiz

Quiz		
Atributos/Métodos	**Descrição**	**Tipo**
id	Identificador único	number

Tabela 6: Estrutura e organização dos dados da classe Abstract Category

Abstract Category		
Atributos/Métodos	**Descrição**	**Tipo**
toggleFilter	Alternar uma opção de filtro entre ativado e desativado	Function

Tabela 7: Estrutura e organização dos dados da classe Company

Company		
Atributos/Métodos	**Descrição**	**Tipo**
byBusinessActivity	Atividades que empresa atua	Array String
businessRevenueModel	Modelo de receita da empresa	String

businessTargetAudience	Público alvo da empresa	Array String

Tabela 8: Estrutura e organização dos dados da classe SoftwareDevelop

SoftwareDevelop		
Atributos/Métodos	**Descrição**	**Tipo**
processDevelopment	Atividades que empresa atua	Array String
timeDevelopment	Modelo de receita da empresa	String
implementProjectAdapter	Público alvo da empresa	Array String
methodologyBenefits	Metodologias e suas vantagens	Array String
methodologyDisadvantages	Metodologias e suas desvantagens	Array String

Tabela 9: Estrutura e organização dos dados da classe Language

Language		
Atributos/Métodos	**Descrição**	**Tipo**
languageSoftware	Linguagens de programação	Array String
frameworksFrontend	Framework frontend	Array String
frameworksBackend	Framework bakend	Array String
testMethodology	Metodologias de testes	Array String
testUnitaryFrontBack	Teste unitários no frontend e backend	Array String
typeArchitectureProject	Tipo de arquiteturas de software	Array String

Tabela 10: Estrutura e organização dos dados da classe Professional

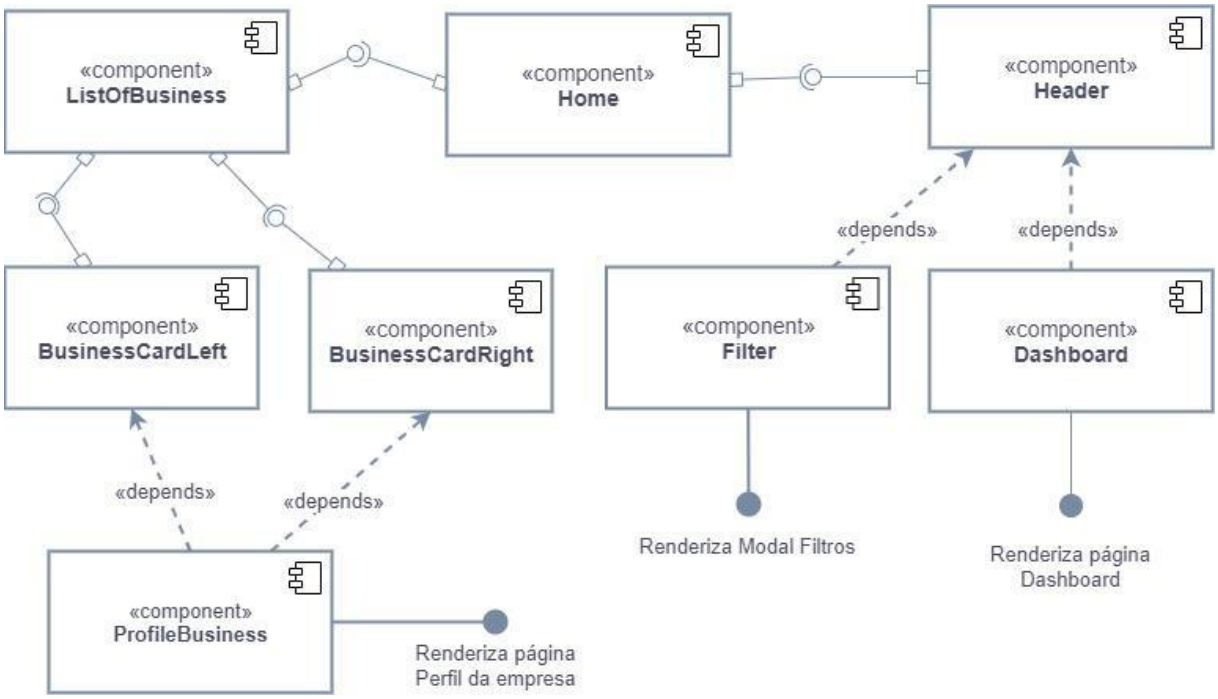
Professional		
Atributos/Métodos	**Descrição**	**Tipo**
timeBusiness	Time de um colaborador na empresa	String
levelProfessionalTechnology	Nível de dificuldade para contratar profissionais de tecnologia	String
technologyWorkMode	Framework bakend	String
menAmount	Quantidade de homem no time de tecnologia	String
girlAmount	Quantidade de mulheres no time de tecnologia	String
workModalityHiring	Modalidade de trabalho	String

Os métodos e atributos apresentados nas tabelas de 4 a 10, foram essenciais para compreender as necessidades de desenvolvimento do NORTI. Após a criação do diagrama de classe foi possível dar início no desenvolvimento de software, assunto que foi discutido com mais detalhes na seção anterior e que também está presente na seção seguinte.

4.4.5 Componentes e telas no aplicativo NORTI

Como dito anteriormente, o aplicativo NORTI foi desenvolvido com React Native (RN), sendo composto por vários componentes, pode-se dizer que tudo no RN é componente. Por esse motivo é importante que seja apresentado às dependências dos principais componentes do aplicativo NORTI. Desta forma, a Figura 33 apresenta o diagrama de componentes.

Figura 33: Diagrama de componente



A criação de componentes personalizados no RN é realizada com a composição de outros componentes menores, sejam eles nativos (componentes do próprio React Native) ou por outros componentes personalizados. Diante deste cenário, faz-se necessário apresentar as dependências de cada componente e suas relações ou conexões existentes. Para isso, existe uma abordagem chamada diagrama de componentes (Figura 33).

Na Figura 33 são apresentados os principais componentes e suas dependências (componentes menores). O componente *Home* possui duas dependências diretas, uma com *ListOfBusines* e a outra com *Header*, ambas são respectivamente uma lista de empresas e o cabeçalho da tela *Home*.

O *BusinessCardLeft* é um exemplo de componente de interface de usuário, criado para compor a tela *Home*, principal tela do aplicativo.

Figura 34: Componente BusinessCardLeft do aplicativo NORTI

A Figura 34 apresenta anatomia do componente *BusinessCardLeft*, visto que seus elementos visuais estão identificados, as informações (elementos) da empresa estão organizadas e em conformidade com o protótipo apresentado nas seções anteriores. Todos os componentes foram criados respeitando integralmente as métricas do protótipo NORTI, a seguir é apresentado cada elemento identificado na Figura 33.

1. Container ou caixa retângular
2. Cidade e estado
3. Nome fantasia
4. Logomarca ou logotipo
5. Descrição da área de atuação

A Figura 35 apresenta um componente visual e personalizado, baseado em função. Atualmente existem duas maneiras de criar componentes RN, são elas: classe e função. Segundo o React Native, o componente de função é a maneira mais moderna de criar componentes (REACT NATIVE, 2021). Por isso, todos os componentes do aplicativo NORTI, foram desenvolvidos como componentes de função.

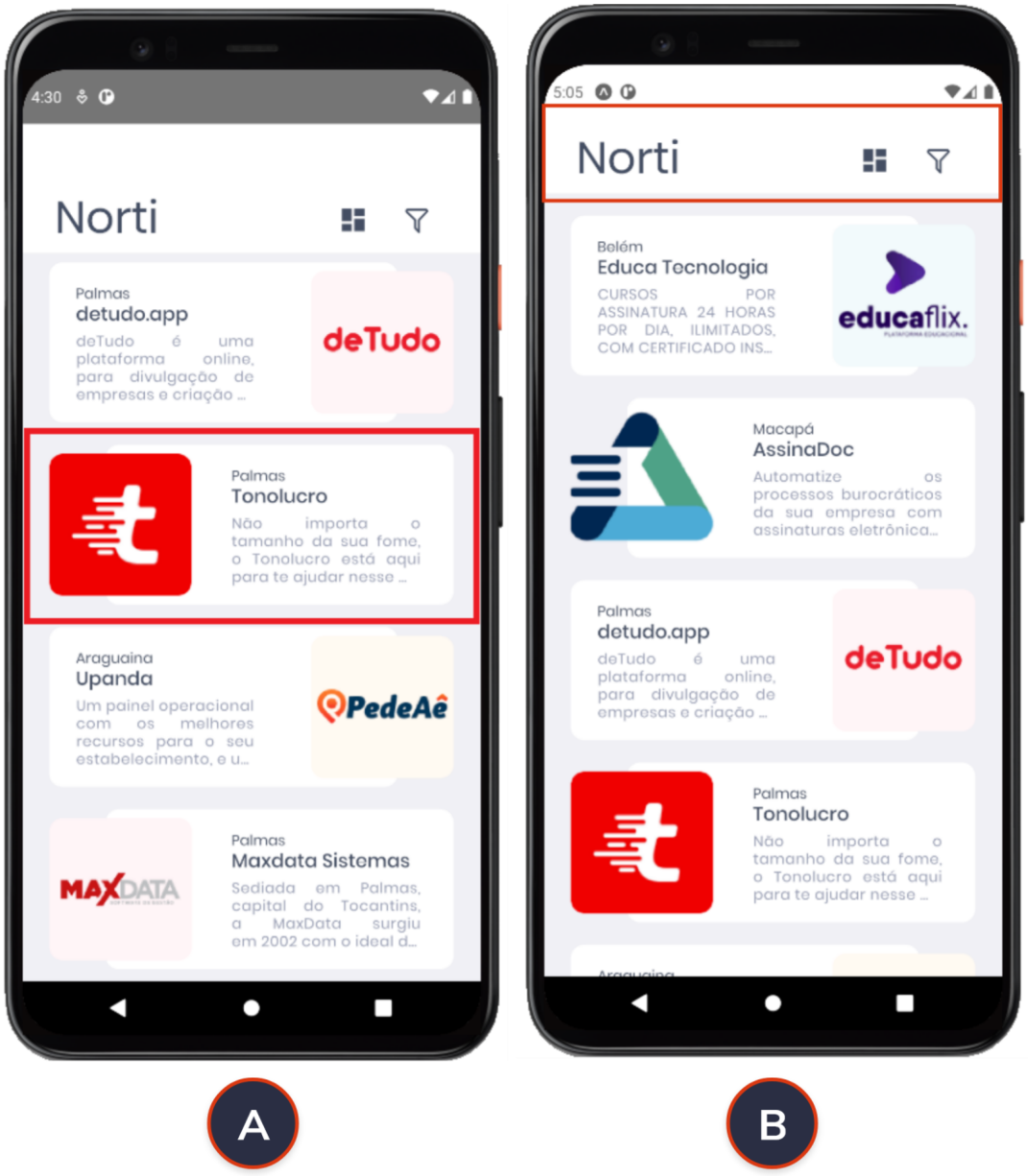
Figura 35: Código do componente BusinessCardLeft

```
interface IBusinessCarProps {
   business: IBusinessQuiz;
}

const BusinessCardLeft: React.FC<IBusinessCarProps> = ({ business }: IBusinessCarProps) => {
   return (
      <BackgroundItem>
         <BusinessContainer>
            <BusinessHeader>
               <City>{business.businessCategory.city}</City>
               <BusinessName>{business.businessCategory.name}</BusinessName>
            </BusinessHeader>
            <BusinessDescription>{business.description}</BusinessDescription>
         </BusinessContainer>
         <BusinessAvatar source={business.photoAvatar} />
      </BackgroundItem>
   );
}
```

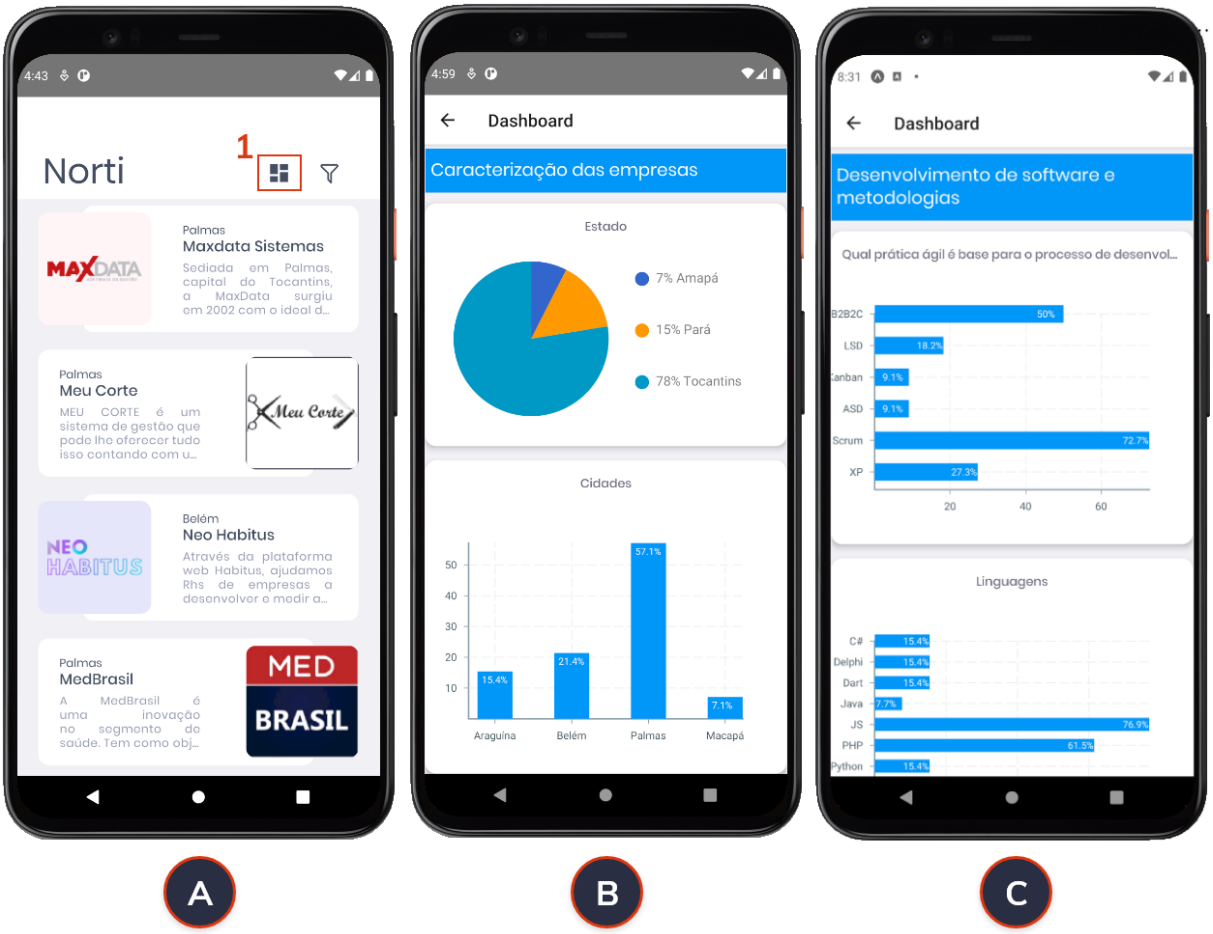
O parâmetro *business* contém dados relacionados à empresa, como: nome da empresa, descrição, foto, cidade e estado.Já na Figura 36 é apresentada a tela principal do aplicativo *mobile* com a lista de empresas cadastrada, a lista é composta por componentes *BusinessCardLeft* e *BusinessCardRight*. Para permitir uma interatividade do usuário no aplicativo, a tela *Home* foi desenvolvida com uma lista deslizante para exibir todas as empresas.

Figura 36: Tela Home do aplicativo NORTI



É possível identificar o componente *BusinessCardLeft* que aparece em destaque (retângulo de linha sólida e cor vermelha) na Figura 36-A. Ainda na tela *Home* é possível identificar outros elementos que a compõem, é o caso do *Header* cabeçalho da página, composto por um título e outros dois botões, o cabeçalho aparece em destaque na Figura 36-B. Os *buttons* no componente *Header* são respectivamente *Dashboard* e *FilterButtonIcon* ambos representados por ícones.

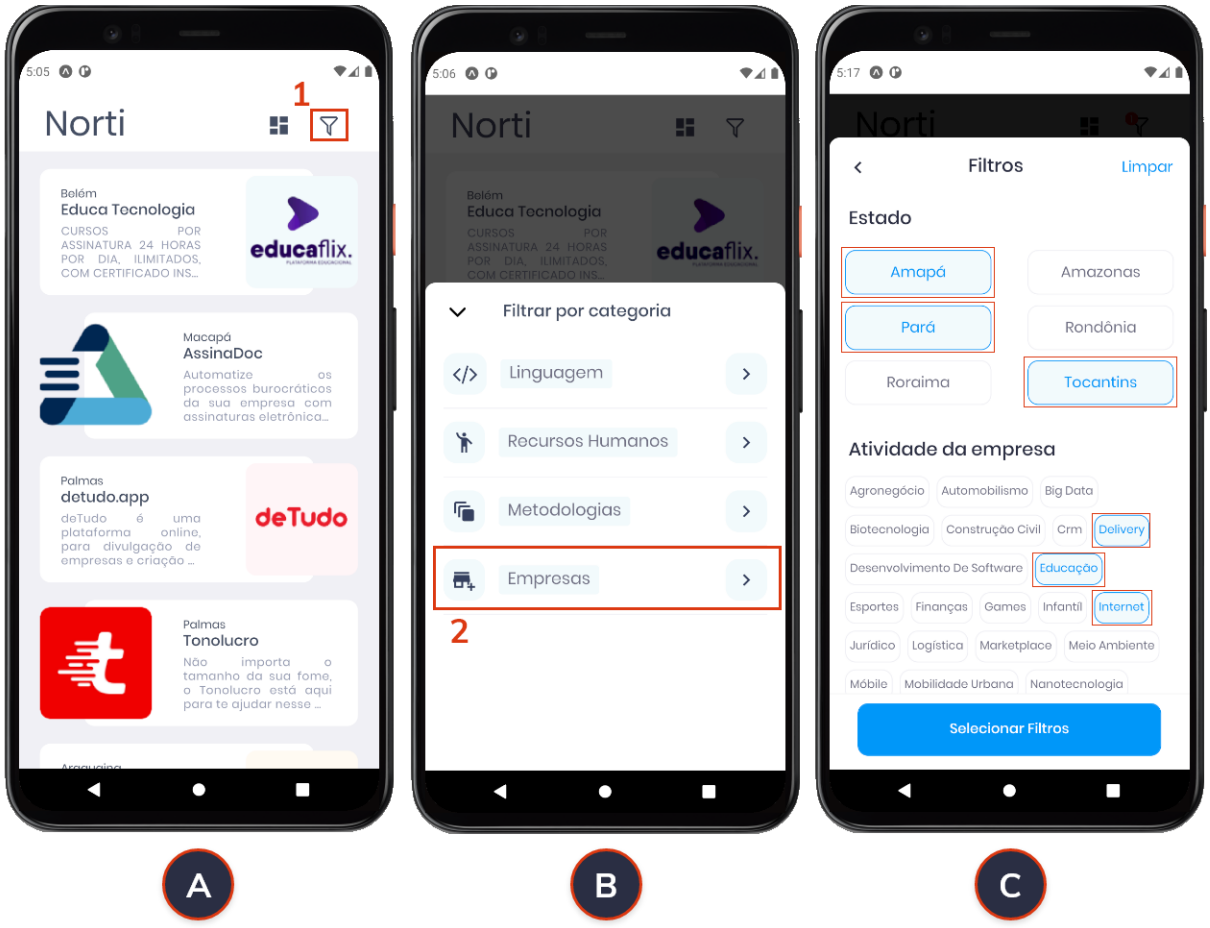
Figura 37: Telas apresentação do Dashboard



Ao pressionar o botão *DashboardButtonIcon*, que aparece em destaque na Figura 37-A (retângulo de linha sólida e cor vermelha), o usuário é redirecionado para a tela de *Dashboard*. Na Figura 37-B é apresentado a tela de *Dashboard* com as informações sobre a "caracterização das empresas", esta categoria contém informações relacionadas a cidades e estados das empresas, na Figura 37-C é apresentado as informações sobre "desenvolvimento de software e metodologias". Todas as categorias do questionário estão presentes na tela de Dashboard, entretanto, somente algumas questões foram disponibilizadas no NORTI, contabilizando três resultados para cada categoria.

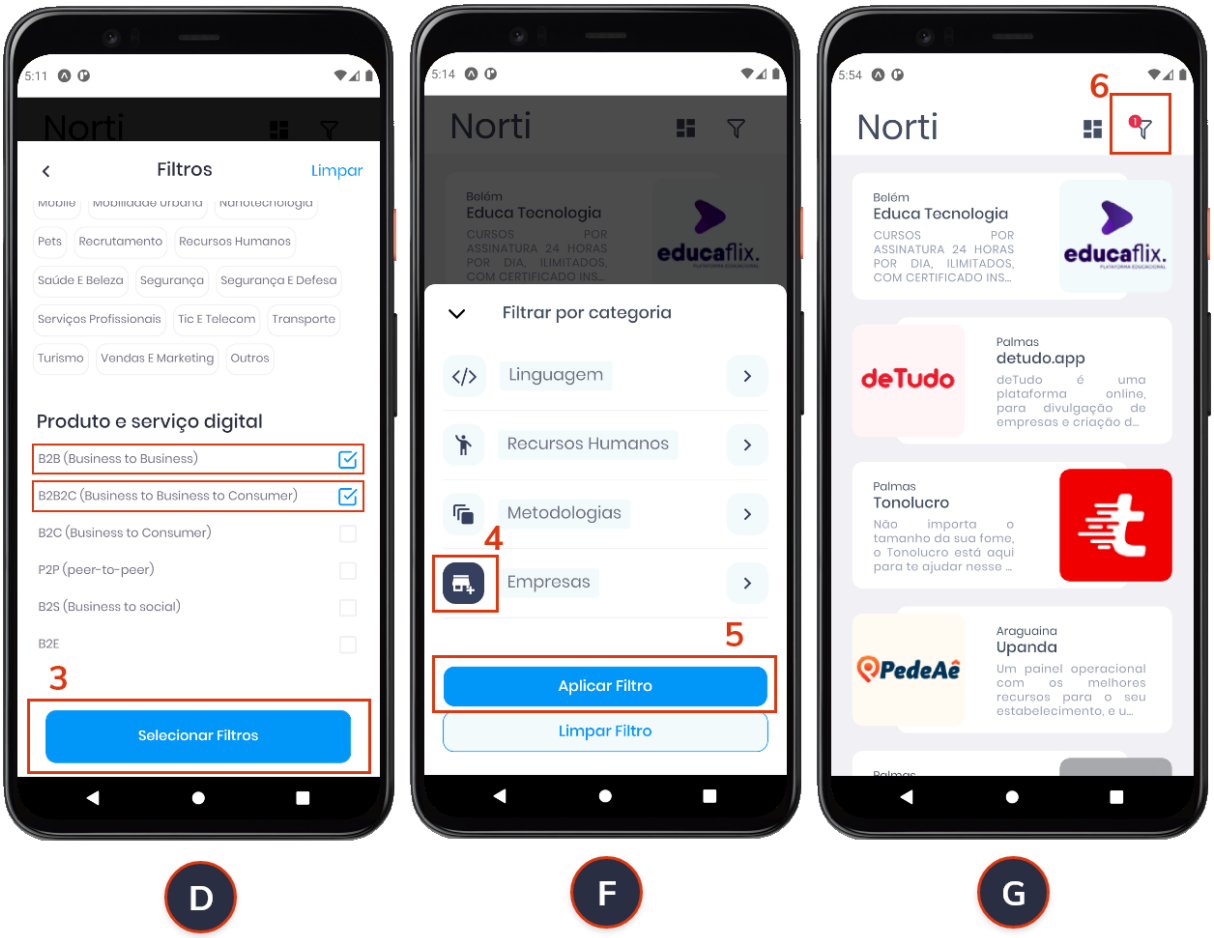
Como o próprio nome sugere, a tela de *Dashboard* possui a responsabilidade de apresentar um Dashboard, fornecendo informações de maneira organizada, objetiva e clara. Desta forma, o usuário consegue ter um panorama das empresas de tecnologia da região Norte do Brasil.

Figura 38: Fluxo de telas para aplicar filtros na categoria empresa no NORTI - PARTE 1



Na Figura 38-A é possível identificar o botão *FilterButtonIcon*, que aparece na região em destaque com uma sinalização numérica (1). Este botão é responsável por apresentar o modal de categorias de filtros, na sequência a Figura 38-B apresenta o modal "Filtrar por categoria" destacando a categoria "Empresas". As regiões em destaque na Figura 38-C não receberam uma sinalização numérica, pois a seleção ou escolha dos filtros não seguem uma sequência pré-definida ou uma ordem de seleção. Nesse sentido o usuário poderá selecionar as opções de filtros na ordem que desejar.

Figura 39: Fluxo de telas para aplicar filtros na categoria empresa no NORTI - PARTE 2

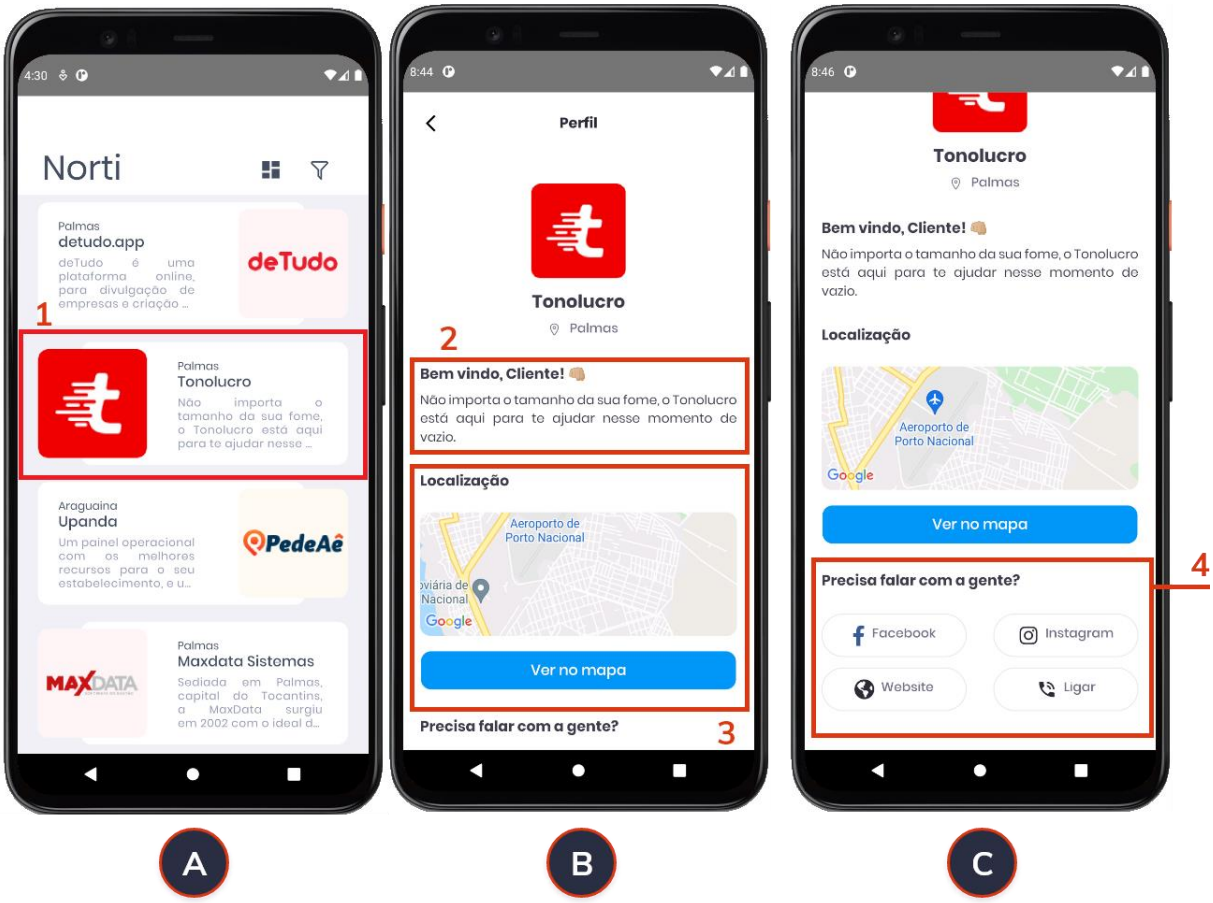


Percebe-se que na Figura 39-D é realizada uma rolagem de baixo para cima até chegar na categoria “Público-alvo”, onde duas regiões aparecem em destaque: B2B (*Business to Business*) e B2B2C (*Business to Business to Client*). As opções de filtros são confirmadas quando o usuário pressiona o botão “Selecionar filtros”, este botão aparece em destaque na região de passo 3.

Esta ação faz com que o modal “Filtros” seja fechado e na sequência apresenta o modal “Filtrar por categoria”. Um aspecto importante a ser mencionado é a animação, indicando que há filtros selecionados na categoria, isso pode ser conferido na sinalização no passo 4 que aparece em destaque na Figura 39-F. No passo 5, o usuário consegue enfim aplicar os filtros selecionados, na sequência o usuário é redirecionado para tela *Home*.

Na Figura 39-G é apresentado o resultado com todas as empresas que passaram pelo filtro. Ainda na Figura 39-G, é possível notar que no passo 6 na região em destaque, o usuário é informado com a quantidade de categorias de filtros aplicados. Isso torna melhor a experiência do usuário.

Figura 40: Telas de apresentação do *ProfileBusiness*



A última tela do aplicativo NORTI se resume no perfil da empresa, com informações de cidade, estado, contato e localização geográfica, sendo possível obter as coordenadas de rotas caso o usuário tenha em seu *smartphone* (aparelho de celular) um aplicativo de rotas semelhante ao Google Maps, por exemplo.

Ao pressionar o botão *BusinessCardLeft* indicado no passo 1 que aparece em destaque na Figura 40-A, o usuário é redirecionado para a tela de *ProfileBusiness.* A região em destaque (passe 2) na Figura 40-B apresenta uma descrição do negócio ou área de atuação da empresa (comumente conhecido como *bio*). Um mapa com a localização da sede da empresa (matriz), ainda conta com o botão "Ver no mapa" que pode redirecionar o usuário para um aplicativo de mapas (Figura 40-B).

Por último são apresentados as informações de contatos e redes sociais da empresa, localizadas na região 4 (Figura 40-C). Um vídeo com apresentação do funcionamento do NORTI e todos os fluxos possíveis está disponível no YouTube.[5]

[5] Vídeo de apresentação do NORTI - https://youtu.be/-Ak_7Kta8bk

5 CONSIDERAÇÕES FINAIS

Este trabalho foi desenvolvido com o objetivo de auxiliar a sociedade, o poder público e novas empresas na tomada de decisões, apresentando um panorama do aspecto tecnológico das empresas de tecnologia da região norte do Brasil. Sendo este trabalho o único, que apresenta informações em um aplicativo (NORTI) sobre práticas de desenvolvimento de *software*, padrões de arquiteturas de *software*, abordando ainda um aspecto social da empresa.

Apesar de apenas 28% das empresas abordadas responderem o questionário (14 empresas) os dados obtidos foram de suma importância para se ter uma visão preliminar do mercado de TI na região norte do Brasil. Porém, devido ao pequeno espaço amostral, recomenda-se mais pesquisas sobre o tema antes de conclusões definitivas.

No quesito social, são apresentados números preocupantes, pois a maioria das empresas não possuem políticas e práticas de planos de carreira, e nem de inclusão de gênero. Na parte de contratação, este trabalho revelou que a profissão de desenvolvedor de *software* está em alta procura, mas há uma grande limitação, pois a maioria das empresas declararam dificuldades em contratar colaboradores, e que os mesmos permaneçam por longos períodos, assim como dão preferência a quem possuam habilidades em detrimento do tempo de experiência.

O desenvolvimento do aplicativo NORTI ocorreu dentro do tempo previsto. Desta forma, é possível disponibilizar o aplicativo para as plataformas Android e iOS em ambas as plataformas com todos os recursos compilados de forma nativa, inclusive estes serão os próximos passos do trabalho.

Em virtude dos resultados apresentados na seção anterior, o objetivo principal deste trabalho foi alcançado, através da aplicação do questionário e desenvolvimento do aplicativo NORTI. As respostas do questionário alimentaram a base de dados do aplicativo, assim foram desenvolvidas funcionalidades para auxiliarem no entendimento do panorama das empresas de Tecnologia da Informação da região norte do Brasil.

O NORTI ficará disponível de forma gratuita a quem precisar, nesse sentido o aplicativo receberá novas funcionalidades que serão disponibilizadas na medida com que cada novo recurso for desenvolvido e testado. Essas atualizações serão realizadas tanto pelo autor deste trabalho, como também poderá receber contribuições da comunidade de desenvolvimento de *software*. Como foi apresentado nas seções anteriores, o NORTI possui seu código-fonte aberto e é um projeto *open source*.

O questionário ficará aberto para respostas até o fim do ano de 2021, e as empresas que não participaram ainda poderão ser abordadas de maneira mais direta, em prol da coleta de dados para a definição e caracterização das empresas.

Para trabalhos futuros, além de disponibilizar o NORTI nas lojas de aplicativos, poderão ser adicionadas novas funcionalidades, como por exemplo: possibilidade de criar funcionalidades para cadastro de usuário e cadastro de empresas permitindo que as empresas consigam atualizar seus dados dentro do NORTI. Desenvolvimento de uma API e criação de um banco de dados onde as informações poderão ser persistidas, esta Interface Application Programming pode fornecer todas as informações ao aplicativo. Adicionar essa a API nos serviços de *cloud computering*, como por exemplo: Google Cloud, AWS (Amazo Web Service) ou Microsoft.

Recursos de integração com sistemas de empresas parceiras, a exemplo do SEBRAE, permitindo conectar com outras bases de dados. Diante de tantas possibilidades e recursos tecnológicos, não é exagero dizer que o NORTI pode se tornar um negócio digital

REFERÊNCIAS

BLANK, S., “Why the Lean Start-Up Changes Everything”, Harvard Business Review, 2013."IEEE Standard Glossary of Software Engineering Terminology," in IEEE Std 610.12-1990 , vol., no., pp.1-84, 31 Dec. 1990, doi: 10.1109/IEEESTD.1990.101064.

Beck, K. Extreme Programming Explained: Embrace Change. **Addison Wesley Professional**, 1st edition. 224 pag. 1999

Beck, K. and Fowler, M. (2000). Planning Extreme Programming. **Addison Wesley Professional**.1st edition. 160 pág. 2000

BUYYA, R., YEO, C. S., VENUGOPAL, S., BROBERG, J., and BRANDIC, I. (2009b). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. **Future Gener. Comput. Syst**., 25(6):599–616.

BRASSCOM. **TI precisa de 420 mil novos profissionais até 2024**. 2019. Disponível em https://brasscom.org.br/ti-precisa-de-420-mil-novos-profissionais-ate-2024/ Acesso em: 26 de jun. 2021.

CARRILO, Ana Flávia. **Crescimento das startups: veja o que mudou nos últimos cinco anos!** 2020. Disponível em: https://abstartups.com.br/crescimento-das-startups/. Acesso em: 30 out. 2020.

DIAS, Mariana. **Guia do Recrutamento e Seleção de pessoas (R&S): o que é, como fazer o processo e melhores técnicas** 2021. Disponível em: https://www.gupy.io/blog/recrutamento-e-selecao. Acesso em: 12 jul. 2021.

CASTELLS, M. **A galáxia da Internet**: reflexões sobre a internet, os negócios e a sociedade. Rio de Janeiro: Jorge Zahar, 2003.

COLEMAN, Gerry; O’CONNOR, Rory V. **An Investigation into Software Development Process Formation in Software Start-ups**. 2008. Disponível em: https://www.researchgate.net/publication/220306357_An_investigation_into_software_development_process_formation_in_software_start-ups. Acesso em: 23 nov. 2020.

CRUZ, Fábio. **Scrum e Agile em Projetos Guia Completo**: conquiste sua certificação e aprenda a usar métodos ágeis no seu dia a dia. 2. ed. Rio de Janeiro, Rj: Brasport Livros e Multimídia Ltda, 2018. 298 p.

FACEBOOK. **React Native Learn once, write anywhere**. 2020. Disponível em: https://facebook.github.io/react-native/. Acesso em: 21 set. 2020.

LUCIDCHART. **Why User Flow Diagrams Are Worth Your Time**. 2021. Disponível em: https://www.lucidchart.com/blog/why-user-flow-diagrams-are-worth-your-time. Acesso em: 12 jun. 2021.

FOWLER, Martin; LEWIS, James. **Microservices a definition of this new architectural term.**[S.L], jun. 2014. Disponível em: https://martinfowler.com/articles/microservices.html. Acesso em: 2020.

DEMARTINI, Felipe. **As dez mulheres mais importantes da história da tecnologia.** 08 de Março de 2016. Disponível em: https://canaltech.com.br/internet/as-dez-mulheres-mais-importantes-da-historia-da-tecnologia-59485/. Acesso em: 16 jul. 2021.

FERGUSSON, Kymberly. **UML class diagrams in draw.io.** set. 2018. Disponível em: https://drawio-app.com/uml-class-diagrams-in-draw-io/. Acesso em: 2021.

MICROSOFT. **Criar um diagrama de componente UML.** Disponível em: https://support.microsoft.com/pt-br/office/criar-um-diagrama-de-componente-uml-aa924ecb-e4d2-4172-976e-a78fa157b074. Acesso em: 2021.

FOWLER, Martin. **Agile Software Guide**. 2019. Disponível em: https://www.martinfowler.com/agile.html. Acesso em: 20 nov. 2020.

FOWLER, Martin. **Software Architecture Guide**. 2019. Disponível em: https://martinfowler.com/architecture/. Acesso em: 30 nov. 2020.

FOWLER, Martin. Patterns of Enterprise Architecture. Addision-Wesley, 2002.

GOOGLE. **Google Forms**. 2020. Disponível em: https://www.google.com/forms/about/. Acesso em: 29 out. 2020.

BIOGRAPHY, Editores. **Biografia de Ada Lovelace**. 2 de abril de 2014. Disponível em: https://www.biography.com/scholar/ada-lovelace. Acesso em: 13 de julho de 2021.

GITHUB. **The 2020 State of the OCTOVERSE**. © 2020 GitHub, Inc. Disponível em https://octoverse.github.com/static/github-octoverse-2020-productivity-report.pdf#page=10 Acesso em: 12 de jun. 2021.

GITHUB. **Open Source Guides**. © 2021 GitHub, Inc. Disponível em https://opensource.guide/pt/starting-a-project/ Acesso em: 13 de jul. 2021.

ITGIRLS. **Mulheres na TI**. 2016 ItGirls. Disponível em http://ulbra-to.br/itgirls/#biografia Acesso em: 02 de jul. 2021.

MICROSOFT. **TypeScript é um idioma da Microsoft que se baseia no JavaScript.** 2012. Disponível em: https://www.typescriptlang.org/pt/why-create-typescript. Acesso em: 02 nov. 2020.

MICROSOFT. **Getting Started**. 2020. Disponível em: https://code.visualstudio.com/docs/. Acesso em: 02 nov. 2020.

MOREIRA, P. F. M.; BEDER, D. M. Desenvolvimento de Aplicações e Micro Serviços: Um estudo de caso. T.I.S. São Carlos,**São Carlos, v. 4**, n. 3, p. 209-215, nov./dez. 2015. ISSN 231 6-2872

NASCIMENTO, C. P.; SOTTO, E. C. S. MICROFRONTEND: um estudo sobre o conceito e aplicação no frontend. **Revista Interface Tecnológica**, *[S. l.]*, v. 17, n. 1, p. 153-165, 2020.

NETO, Manoel Veras de Sousa. Virtualização: Tecnologia Central do Datacenter. 2ed. **Brasil: Brasport**, 2015. 224p.

OSTERWALDER, A.; PIGNEUR, Y. Business Model Generation. London: Wiley John&Sons. 2010.

PATERNOSTER, Nicolò et al. Software Development in Startup Companies: a systematic mapping study. **Electronic Research Archive Of Blekinge Institute Of Technology**, [S.I.], v. 1, p. 112-113, 14 abr. 2014. Mensal. Disponível em: https://www.bth.se/fou/. Acesso em: 28 nov. 2020.

PISKE, Otávio Rodolfo. **RUP – Rational Unified Process**. Universidade do Contestado – UNC, Unidade Universitária de Mafra. 2003.

PEDROSA, C Ra,. H., & NOGUEIRA, T. (n.d.). Computação em Nuvem Palavras-Chave, 3–6. Retrieved from http://www.ic.unicamp.br/~ducatte/mo401/1s2011/T2/Artigos/G04-095352-120531-t2.pdf

PRESSMAN, Roger S. *et al*. **Engenharia de Software**: uma abordagem profissional. 8. ed. Porto Alegre, RS: Amgh Editora Ltda, 2016. 933 p. João Eduardo Nobrega Tortello.

RODRIGUES, J. L.; PINTO, G. S. ANÁLISE DA ARQUITETURA DE MICROSERVIÇOS. SIMTEC - Simpósio de Tecnologia da Fatec Taquaritinga, v. 5, n. 1, p. 39-49, 21 dez. 2018.

ROSES, Luís Kalb; WINDMÖLLER, Arno; CARMO, Eliana Almeida do. Favorability conditions in the adoption of agile method practices for software development in a public banking. Journal Of Information Systems And Technology Management, [S.L.], v. 13, n. 3, p. 443-445, 30 dez. 2016. TECSI. http://dx.doi.org/10.4301/s1807-17752016000300005.

SANTOS, Lucas. **Microserviços: dos grandes monólitos às pequenas rotas.** Training Center, [S.L], jun. 2017. Disponível em: https://medium.com/trainingcenter/microservi%C3%A7os-dos-grandes-mon%C3%B3litos-%C3%A0s-pequenas-rotas-adb70303b6a3. Acesso em 2020.

SOUSA, Flávio & MOREIRA, Leonardo & MACHADO, Javam. (2009). Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios. Universidade Federal do Ceará (UFC), **ERCEMAPI, 2009**. EDUFPI.

SOUZA, Victor Alexandre Siqueira Marques de. **Uma arquitetura orientada a serviços para desenvolvimento, gerenciamento e instalação de serviços de rede**. – Campinas, SP: Dissertação de Mestrado [s.n.], 2006.

SUTHERLAND, Jeff J.. **SCRUM Guia prático**: maior produtividade. melhores resultados. aplicação imediata.. Rio de Janeiro, Rj: Gmt Editores Ltda, 2019. 352 p.

TELES, Vinícius Manhães et al. Desenvolvimento ágil. 2007. Elaborado por Vinícius Manhães Teles, ilustrações de Leandro Mello. Disponível em: https://www.desenvolvimentoagil.com.br/xp/desenvolvimento_tradicional. Acesso em: 22 nov. 2020.

VASCO, Carlos G. VITHOFT, Marcelo H. ESTANTE Paulo R. **Comparação entre Metodologias Rational Unified Process (RUP) e eXtreme Programming(XP)**. Fundamentos de Engenharia de Software PPGIA. 2006.

VICTORINO, Marcio e BRÄSCHER, Marisa. Organização da Informação e do Conhecimento, Engenharia de Software e Arquitetura Orientada a Serviços: uma Abordagem Holística para o Desenvolvimento de Sistemas de Informação Computadorizados. DataGramaZero - **Revista de Ciência da Informação** - v.10 n.3 jun/09.

APÊNDICE A

Pesquisa sobre a utilização dos processos de desenvolvimento de software

Bom dia (boa tarde/boa noite), me chamo Lucas Pedro, bacharelando em Engenharia de Software pelo CEULP/ULBRA em Palmas - TO. Estou desenvolvendo, como parte do meu trabalho de conclusão de curso, um aplicativo capaz de apresentar o panorama do mercado das empresas de tecnologia da região Norte do Brasil.

De acordo com uma pesquisa realizada pela ABStartups (Associação Brasileira de Startups), a região Norte é a que menos desenvolve softwares e produtos voltados à tecnologia, sendo a região menos inovadora e com menos cases de sucesso.

Sua resposta a este questionário, como participante do cenário de negócios em tecnologia, contribuirá no sentido de auxiliar no desenvolvimento de uma base de informações capaz de proporcionar conhecimento sobre o cenário de negócios em tecnologia na região Norte. Entendendo que a informação é chave para a tomada de decisões, presumo que esta base de informações também poderá ser utilizada para motivar estudantes, empreendedores, o poder público e a população na tomada de decisões e ações que possam contribuir para a melhoria do cenário de negócios em tecnologia na região Norte.

A sua contribuição para esta pesquisa, é muito importante, desde já agradeço!

1. **Identificação e informações básicas da entrevista:**
    a. Nome
    b. Endereço de e-mail
    c. Papel(éis) ou função(ões) na empresa

2. **Caracterização da Empresa**
    (Esta seção contém informações sobre a empresa em questão):
    a. Nome da empresa
    b. Estado
    c. Cidade
    d. Site
    e. Perfil na rede social utilizada com maior frequência?
    f. CNPJ
    g. Quais as áreas de atividades em que a empresa atua?
        i. Agronegócio
        ii. Automobilismo
        iii. Big Data
        iv. Biotecnologia
        v. Construção civil

vi. CRM
vii. Delivery
viii. Desenvolvimento de Software
ix. Educação
x. Esportes
xi. Finanças
xii. Games
xiii. Infantil
xiv. Internet
xv. Jurídico
xvi. Logística e Mobilidade Urbana
xvii. Marketplace
xviii. Meio Ambiente
xix. Mobile
xx. Mobilidade urbana
xxi. Nanotecnologia
xxii. Pets
xxiii. Recrutamento
xxiv. Recursos Humanos
xxv. Saúde e beleza
xxvi. Segurança
xxvii. Segurança e Defesa
xxviii. Serviços Profissionais
xxix. TIC e Telecom
xxx. Transporte
xxxi. Turismo
xxxii. Vendas e Marketing
xxxiii. Outra

h. O produto/serviço digital que a empresa possui no mercado, é exatamente ou semelhante a qual destes?
   i. Aplicativo para smartphones
   ii. Sistema/aplicativo Web
   iii. Sistema/aplicativo Desktop para computadores sem conexão de internet (exemplo PDV)
   iv. Sistema/aplicativo Desktop para computadores com conexão de internet
   v. Outro

i. Qual é o modelo de receita da empresa?
   i. API
   ii. Consumer
   iii. E-commerce
   iv. Hardware
   v. Marketplace

vi. Software como Serviço (SaaS)
vii. Vendas de Dados
viii. Outro

j. Qual público alvo da empresa?
   i. B2B (Business to Business)
   ii. B2B2C (Business to Business to Consumer)
   iii. B2C (Business to Consumer)
   iv. P2P (peer-to-peer)
   v. B2S (Business to social)
   vi. B2G (Business to Government)
   vii. Outro

3. Desenvolvimento de software e metodologias

a. Qual o estilo do processo de desenvolvimento de software adotado pela empresa na maioria dos projetos?
   i. Processo de desenvolvimento de software tradicional
   ii. Processo de desenvolvimento de software ágil

b. Específico para empresas que trabalham com uma prática ágil de desenvolvimento de software? (levando em consideração a maiorias dos projetos na empresas)

c. Qual prática ágil é base para o processo de desenvolvimento?
   i. Agile Modeling (AM)
   ii. Agile Unified Process (AUP)
   iii. Behavior Driven Development (BDD)
   iv. Crystal
   v. Dynamic Systems Development Methodology (DSDM)
   vi. Extreme Programming (XP)
   vii. Feature Driven Development (FDD)
   viii. Lean Software Development (LSD)
   ix. Microsoft Solutions Framework (MSF)
   x. Rapid Application Development (RAD)
   xi. Rational Unified Process (RUP)
   xii. Scrum
   xiii. Test Driven Development (TDD)
   xiv. Outra

d. Quanto tempo de adesão à prática de desenvolvimento de software?
   i. Até 6 meses
   ii. Entre 6 meses e 1 ano
   iii. Entre 1 e 3 anos
   iv. Acima de 3 anos

e. A metodologia utilizada pela empresa é implementada por completo em seus projetos ou adaptada para atender as necessidades de cada projeto?
   i. Implementada por completo nos projetos
   ii. Implementada parcialmente e adaptada aos projetos

f. Ao adotar metodologias ágeis nos projetos de software, quais foram os principais benefícios alcançados? (se necessário, selecione mais de uma opção)
   i. Transparência nas atividades realizadas pelo time
   ii. Entendimento sobre o projeto
   iii. Uso de artefatos para entendimento do problema
   iv. Uso de diagramas para compreender a solução (o projeto)
   v. Documentação de fácil compreensão
   vi. Aumento de produtividade nas entregas
   vii. Aumento de qualidade nas entregas
   viii. Melhoria na comunicação da equipe
   ix. Ganho de autonomia da equipe e indivíduos
   x. Capacidade da equipe responder rápido a mudanças no projetos
   xi. Redução de falhas
   xii. Nenhum
   xiii. Outro

g. Ao adotar metodologias ágeis nos projetos de software, quais foram as desvantagens? (se necessário, selecione mais de uma opção)
   i. Pouca transparência nas atividades realizadas pelo time
   ii. Redução da produtividade nas entregas
   iii. Problemas com prazos
   iv. Nenhum ou poucos artefatos para entendimento do problema
   v. Nenhum ou poucos diagramas para compreender a solução (o projeto)
   vi. Visão do projeto como um todo
   vii. Aumento de qualidade nas entregas
   viii. Melhoria na comunicação da equipe
   ix. Ganho de autonomia da equipe e indivíduos
   x. Capacidade da equipe responder rápido a mudanças no projetos
   xi. Redução de falhas
   xii. Nenhuma
   xiii. Outra

h. Específico para empresas que trabalham com modelo ou prática tradicional de desenvolvimento de software

i. Qual prática tradicional é base para o processo de desenvolvimento de software?
   i. Domain Driven Design (DDD)
   ii. Joint Application Development (JAD)

iii. Model Driven Architecture (MDA)
iv. Modelo Cascata
v. Modelo de Prototipagem CleanRoom
vi. Modelo Espiral
vii. Rational Unified Process (RUP)
viii. Service Oriented Modeling (SOM)
ix. V-Model
x. Outra

j. Quanto tempo de adesão à prática de desenvolvimento de software?
   i. Até 6 meses
   ii. Entre 6 meses e 1 ano
   iii. Entre 1 e 3 anos
   iv. Acima de 3 anos

k. Ao adotar metodologias tradicionais nos projetos de software, quais foram os principais benefícios alcançados? (se necessário, selecione mais de uma opção)
   i. Transparência nas atividades realizadas pelo time
   ii. Entendimento sobre o projeto
   iii. Uso de artefatos para entendimento do problema
   iv. Uso de diagramas para compreender a solução (o projeto)
   v. Documentação de fácil compreensão
   vi. Aumento de produtividade nas entregas
   vii. Aumento de qualidade nas entregas
   viii. Melhoria na comunicação da equipe
   ix. Ganho de autonomia da equipe e indivíduos
   x. Capacidade da equipe responder rápido a mudanças no projetos
   xi. Redução de falhas
   xii. Nenhum
   xiii. Outro

l. Ao adotar metodologias tradicionais nos projetos de software, quais foram as desvantagens? (se necessário, selecione mais de uma opção)
   i. Pouca transparência nas atividades realizadas pelo time
   ii. Redução da produtividade nas entregas
   iii. Problemas com prazos
   iv. Nenhum ou poucos artefatos para entendimento do problema
   v. Nenhum ou poucos diagramas para compreender a solução (o projeto)
   vi. Documentação confusa
   vii. Pouca compreensão do projeto como um todo
   viii. Perda de qualidade nas entregas
   ix. Falha na comunicação da equipe
   x. Perda de autonomia da equipe e indivíduos
   xi. Aumento de falhas

xii. Nenhuma
xiii. Outra

4. Linguagens, frameworks, cloud computing e ferramentas

a. Quais linguagens de programação são utilizadas no desenvolvimento de software?
   i. C
   ii. C#
   iii. C++
   iv. Java
   v. JavaScript
   vi. PHP
   vii. Python
   viii. Ruby
   ix. Shell
   x. TypeScript
   xi. Outra

b. A empresa utiliza algum destes frameworks front-end em seus projetos?
   i. Angular
   ii. Backbone.js
   iii. Ember.js
   iv. JQuery
   v. NextJs
   vi. React (é considerado um framework dentro de um ecossistema de React)
   vii. Svelte.js
   viii. Vue
   ix. Outro

c. A empresa utiliza algum destes frameworks backend em seus projetos?
   i. .NET Core
   ii. ASP.NET
   iii. CakePHP
   iv. CodeIgniter
   v. Django
   vi. Express
   vii. Laravel
   viii. NestJS
   ix. RubyOnRails
   x. Spring
   xi. Symfony
   xii. Outro

d. Qual metodologia de teste é utilizada pela empresa? (se necessária marque mais de uma opção)
    i. TDD (Test-driven Development)
    ii. BDD (Behavior Driven Development)
    iii. DDD (Domain Driven Development)
    iv. Outra

e. Quais framework/libs/ferramenta são utilizados para automatização de teste unitários no backend/frontend? (se necessário, selecione mais de uma opção)
    i. appveyor.com
    ii. circleci.com
    iii. Cypress
    iv. Faker
    v. FreezeGun
    vi. Jasmine
    vii. Jenkins
    viii. Jest
    ix. Junit
    x. Mock
    xi. NSubstitute
    xii. PyUnit
    xiii. QUnit
    xiv. Rhino Mocks
    xv. Sinon.JS
    xvi. travis-ci.org
    xvii. Unittest
    xviii. VCR.py
    xix. Outro

f. A empresa utiliza algum framework próprio, desenvolvido pelo time de Tecnologia?
    i. Sim
    ii. Não

g. Quais plataformas/serviços de cloud computing (computação em nuvem) a empresa utiliza em seus produtos/serviços?
    i. AWS (Amazon Web Services)
    ii. Digital Ocean
    iii. Google Cloud
    iv. Microsoft Azure
    v. Oracle Cloud
    vi. Outra

h. Quais destas ferramentas são utilizadas para gerenciamento/organização de tarefas?
    i. Asana
    ii. Git/Gitlab
    iii. Jira
    iv. Microsoft Project
    v. Miro
    vi. Notion
    vii. Project.net
    viii. Redmine
    ix. Trello
    x. Zoho Projects
    xi. Outra

i. Quais destas ferramentas/IDEs é a mais utilizada na codificação dos projetos de software?
    i. Atom
    ii. Eclipse
    iii. IntelliJ IDE
    iv. Jupyter
    v. Netbeans
    vi. NotePad++
    vii. Pycharm
    viii. Sublime Text
    ix. Visual Studio
    x. Visual Studio Code
    xi. Outra

j. Quais ferramentas de design são utilizadas para criação de protótipos de software (web, aplicativos e etc)?
    i. Adobe Illustrator
    ii. Adobe Photoshop
    iii. Adobe XD
    iv. Canva
    v. Corel Draw
    vi. Figma
    vii. Papel e caneta
    viii. Prototype
    ix. Outra

k. Quais ferramentas de videoconferência (chamada por voz/vídeo) são utilizadas pela empresa para realizar reuniões de equipe, alinhar estratégia, consultorias e etc?
    i. Discord

ii. Google Meet
iii. Hangouts
iv. Microsoft Teams
v. Skype
vi. Slack
vii. Zoom
viii. Outra

l. Quais ferramentas de versionamento de código e/ou automatização de atividades de projetos de software a empresa utiliza?
   i. Azure DevOps Services
   ii. Bitbucket
   iii. Git
   iv. GitHub
   v. GitLab
   vi. Outra

m. Quais tipos de arquitetura(s) a empresa utiliza com maior frequência em seus projetos?
   i. Arquiteturas em camadas
   ii. Arquitetura MVC (Model-View-Controller)
   iii. Arquitetura de Microservices
   iv. Arquitetura Monolítica

5. Profissionais, oportunidades e carreira

a. Em média, quanto tempo um profissional de Tecnologia permanece na empresa?
   i. Menos de 1 ano
   ii. Entre 1 e 3 anos
   iii. Entre 3 e 5 anos
   iv. Entre 5 e 10 anos
   v. Acima de 10 anos

b. Qual o nível de dificuldade para contratar um profissional na área de Tecnologia (desenvolvimento)?
   i. Nenhuma (0)
   ii. Muito baixa (2)
   iii. Difícil (5)
   iv. Muito difícil (7)
   v. Extremamente difícil (10)

c. A empresa utiliza qual modalidade de trabalho da equipe de tecnologia?
   i. Presencial

ii. Remoto
iii. Híbrida

d. No processo de contratação, geralmente, a empresa adota preferência por:
   i. conhecimento e habilidade em detrimento do tempo de experiência
   ii. tempo de experiência em detrimento de conhecimento de habilidade
   iii. tempo de experiência, conhecimento e habilidade

e. Quantos profissionais na área de TI a empresa possui?
   i. Até 5 profissionais
   ii. Entre 5 e 10 profissionais
   iii. Entre 10 e 20 profissionais
   iv. Acima de 20 profissionais

f. O squad (time ou equipe) de desenvolvimento é formado por quantos homens?
   i. até 5 profissionais
   ii. entre 5 e 10 profissionais
   iii. entre 10 e 20 profissionais
   iv. acima de 20 profissionais

g. O squad (time ou equipe) de desenvolvimento é formado por quantas mulheres?
   i. até 5 profissionais
   ii. entre 5 e 10 profissionais
   iii. entre 10 e 20 profissionais
   iv. acima de 20 profissionais

h. A empresa possui práticas voltadas para a inclusão ou diversidade de gênero?
   i. sim
   ii. não

i. A empresa adota divisão dos profissionais de TI nas áreas de trabalho a seguir:
   i. Desenvolvimento (dev)
   ii. DevOps
   iii. Redes/Infraestrutura
   iv. Produto e Experiência de Usuário
   v. Q&A (Software Quality and Assurance ou Garantia da Qualidade de Software) e Teste
   vi. Outra

j. A empresa oferece plano de carreira para os profissionais de tecnologia?
   i. Sim
   ii. Não
   iii. Estamos desenvolvendo um plano

k. A empresa possui algum modelo de capacitação interna para os profissionais de tecnologia?
    i. Oferecemos treinamento profissional em nossas instalações
    ii. Os profissionais recebem treinamento de uma outra empresa especializado na capacitação dos profissionais de tecnologia
    iii. Estamos desenvolvendo um plano de capacitação para os profissionais de tecnologia.
    iv. Nenhum

l. A empresa trabalha com qual modalidade de contratação?
    i. CLT (Consolidação das Leis Trabalhistas)
    ii. PJ (Pessoa Jurídica)
    iii. Ambas