UNIVERSIDADE FEDERAL DE SANTA CATARINA

Desenvolvimento de ferramenta para coleta de dados em pesquisas quantitativas utilizando dispositivos móveis

Caio Pereira Oliveira

Florianópolis
2019

Caio Pereira Oliveira

Desenvolvimento de ferramenta para coleta de dados em pesquisas quantitativas utilizando dispositivos móveis

Trabalho de Conclusão de Curso submetido ao Departamento de Informática e Estatística para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador: Prof. Raul Sidnei Wazlawick

Florianópolis, dezembro de 2019

Caio Pereira Oliveira

Desenvolvimento de ferramenta para coleta de dados em pesquisas quantitativas utilizando dispositivos móveis

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para a obtenção do Título de "Bacharel em Ciências da Computação" e e aprovado em sua forma final pelo Curso de Ciências da Computação da Universidade Federal de Santa Catarina.

Florianópolis, dezembro de 2019

Prof. José Francisco Fletes
Coordenador do Curso

Banca Examinadora:

Prof. Raul Sidnei Wazlawick
Orientador

Prof. José Eduardo de Lucca

Bel. Rahony Goulart Ricardo

The most dangerous phrase in the language is, "We've always done it this way."

(Grace Hopper)

Resumo

O uso de questionários em pesquisas científicas é frequente tanto no cenário nacional, quanto internacional. No entanto, quando os questionários são preenchidos em papel geram um grande volume de dados que precisam ser lançados em planilhas individualmente. Esse processo, além de exigir tempo, está sujeito a erros de digitação, o que pode impactar nos resultados do estudo. Diante disso, muitos pesquisadores têm optado pelo uso de questionários informatizados em pesquisas. Os benefícios da utilização de dispositivos móveis para o preenchimento de questionários envolvem a redução de erros e do tempo de preenchimento pelos participantes, eliminação do uso de papel e agilidade na organização dos dados para análise. Contudo, a informatização também possui limitações como indisponibilidade do sinal de internet, problemas de usabilidade e preocupações sobre a segurança dos dados dos participantes. Assim, esta pesquisa teve por objetivo desenvolver uma nova ferramenta de gerenciamento de questionários, visando reduzir o impacto das limitações indicadas na literatura quanto ao uso de dispositivos móveis em pesquisas científicas.

Palavras-chave: Questionário, Aplicativo, REST, Android, iOS.

Abstract

The use of questionnaires in scientific research is frequent in the entire world. However, questionnaires filled with paper-and-pen generate a huge volume of data that must be typed. This process, besides requiring time, is subject to typing errors, which may impact the study results. Given this, many researchers have opted for the use of computerized questionnaires in research. The benefits of using mobile devices to complete questionnaires involve reducing errors and responding time, eliminating paper usage and faster data organization for analysis. However, computerization also has limitations such as internet signal unavailability, usability issues, and concerns about participant data security. Thus, this research aimed to develop a new questionnaire management tool to reduce the impact of the limitations indicated by the literature regarding the use of mobile devices in scientific research.

Keywords: Questionnaire, Survey, Application, REST, Android, iOS.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
JSON	*JavaScript Object Notation*
REST	*Representational State Transfer*
GUI	*Graphical User Interface*
SGBD	*Sistema de Gerenciamento de Banco de Dados*
TCC	Trabalho de Conclusão de Curso
UFSC	Universidade Federal de Santa Catarina

Sumário

1 Apresentação

A motivação para a realização deste trabalho me ocorreu no início de 2018, quando estava auxiliando voluntariamente um grupo de pesquisa do Departamento de Psicologia da UFSC. O grupo de pesquisa – composto por três doutorandas e duas bolsistas de Iniciação Científica orientadas pela Profa. Dra. Maria Aparecida Crepaldi, do PPGP-UFSC – estava iniciando o planejamento da coleta de dados de uma pesquisa que visava conseguir, ao longo de 2 anos, o preenchimento de questionários em três etapas para a avaliação de um programa de intervenção. A previsão era ter mais de 100 participantes e um conjunto de questionários com aproximadamente 250 perguntas.

O grupo havia recentemente adquirido 12 tablets para utilizar durante a coleta de dados, substituindo o uso de questionários impressos. Fui requisitado para auxiliar o grupo a escolher uma plataforma de questionários informatizados, inserir as perguntas na plataforma escolhida, e por fim exportar o conjunto de dados obtido para o software de análise de dados estatísticos *IBM SPSS Statistics*.

Durante a busca por uma plataforma que atendesse as necessidades do grupo - que buscava uma plataforma com funcionamento *offline*, já que os questionários seriam aplicados especialmente em locais sem acesso à internet confiável - descobrimos que a UFSC possuía uma parceria com a empresa que desenvolve a plataforma de questionários *QuestionPro* [1], porém, ao tentar realizar o cadastro descobrimos que a parceria havia terminado há pouco tempo. Um representante da empresa entrou em contato informando que para continuar, o grupo precisaria pagar um valor muito alto pelo software.

Ao direcionar nossas buscas para plataformas gratuitas encontramos a *Zoho Forms*, fizemos alguns testes e inicialmente o programa parecia atender as necessidades da coleta de dados da pesquisa. Realizamos então a segunda etapa, na qual foram inseridas na plataforma todas as 250 perguntas do questionário de avaliação do programa de intervenção. Em seguida, efetuamos um teste da versão informatizada dos questionários, com um grupo pessoas do laboratório de pesquisa, que preencheram os questionários para simular o preenchimento que seria realizado com participantes da pesquisa. Com a realização desses testes, percebemos que não havia confiabilidade, pois ao tentar enviar os questionários para a base de dados, o aplicativo apresentava erro algumas vezes e perdia os dados preenchidos. Temendo que a falta de confiabilidade da plataforma ocasionasse perda de dados dos participantes da pesquisa, o que ocasionaria um prejuízo grave para a pesquisa, tomamos a decisão de não utilizar a plataforma *Zoho Forms*.

Todo esse processo de procura e escolha do software que seria utilizado, lançamento

[1] <https://noticias.ufsc.br/2017/05/ufsc-fecha-parceria-com-plataforma-de-pesquisas-e-questionarios-on-line>

dos questionários na plataforma e testes foram realizados em alguns meses e com isso aproximou-se a data em que os a coleta de dados iria iniciar. E devido aos problemas encontrados optou-se por desistir do uso do software e os questionários foram impressos. A coleta de dados desta pesquisa teve seu término em Outubro de 2019 e foram utilizadas mais de 6.000 folhas impressas nas três etapas da coleta de dados.

Tendo sido preenchidos no modo impresso pelos participantes da pesquisa, os questionários precisaram ser lançar manualmente no software SPSS. O tempo médio de lançamento dos questionários foi de 20 minutos, sendo 15 minutos por participante e mais cinco minutos para a conferência da digitação, para evitar erros. Desse modo, o lançamento dos dados de aproximadamente 300 envelopes de coletas de dados (160 de avaliação pré-intervenção; 95 de pós-avaliação e 45 de *follow-up*) foi realizado em aproximadamente 101 horas de trabalho, executado em duplas para diminuir as chances de erro. Tempo que teria sido otimizado pelo grupo de pesquisa se os questionários tivessem sido preenchidos através do uso de tablets. Além disso, devido ao fato de alguns dados sociodemográficos terem sido deixados em branco pelos participantes, um novo contato foi realizado com eles para evitar itens em branco, que prejudicam as análises. Considerando a experiência com esse grupo de pesquisa, encontrei motivação para desenvolver um software que ofereça a pesquisadores uma ferramenta de livre acesso que atenda as necessidades de coletas de dados de pesquisas.

2 Introdução

O questionário constitui-se como uma ferramenta de coleta de dados de respondentes através de uma série de perguntas, sendo considerado como um dos instrumentos de pesquisa mais utilizados nas ciências sociais [Lewis-Beck, Bryman e Liao 2004]. As pesquisas em Psicologia com frequência são baseadas em dados obtidos através de questionários, o que o torna um instrumento considerado confiável por muitos pesquisadores [Gault 1907].

As vantagens de utilizar questionários são muitas, e incluem: baixo custo financeiro; baixo custo de tempo, se comparados com entrevistas presenciais ou por telefone; redução de viés do entrevistador, já que todos os participantes receberão as questões da mesma forma; possibilidade de ser aplicado em múltiplas pessoas ao mesmo tempo; facilidade de analisar questões fechadas; entre outras [Gillham 2008].

Porém, questionários também possuem alguns desafios, como a necessidade de formulação de questões breves e claras, dificuldades com respondentes analfabetos, avaliação do impacto do modo como as perguntas foram formuladas nas respostas, impossibilidade de verificar a seriedade e honestidade das respostas, impossibilidade de corrigir equívocos na interpretação, entre outros [Gillham 2008].

Por conta da importância dos questionários, diversas ferramentas surgiram com o intuito de informatizá-los, como Google Forms [1], SurveyMonkey [2], etc. Trazer questionários para plataformas informatizadas possui benefícios, tais como: redução do tempo de preenchimento, redução do número de erros no preenchimento, redução do uso de papel, e eliminação tempo necessário para inserção dos dados em sistemas de análise de dados [Mwita, Pearson e Zwickle 2019, Pereira et al. 2017, Silveira et al. 2014, Dy et al. 2012]. O estudo em Azevedo, Miazaki e Porfirio 2014 encontrou redução do tempo em até 50% no preenchimento e digitação dos questionários, além da eliminação de erro humano no processo.

Entretanto, a informatização também possui limitações como oscilação e indisponibilidade do sinal de redes WiFi e 3G/4G, problemas relacionados a layout mal adaptados para dispositivos móveis, preocupações com a segurança dos dados, segurança dos dispositivos, duração da bateria, entre outras. Para reduzir o impacto destas limitações, o presente estudo se propõe a desenvolver uma nova ferramenta de gerenciamento de questionários.

[1] google.com/forms
[2] surveymonkey.com

2.1 Objetivos

O objetivo geral deste trabalho é desenvolver um sistema que permita cadastro, preenchimento, armazenamento e exportação dos dados de questionários.

Os objetivos específicos são (1) identificar as dificuldades, problemas e desafios enfrentados por pesquisadores e participantes ao realizar pesquisas com questionários informatizados através do uso de dispositivos móveis, (2) desenvolver um servidor RESTful para gerenciamento, armazenamento e exportação de questionários, (3) desenvolver uma aplicação híbrida para iOS/Android para cadastro e preenchimento de questionários que utilize o servidor desenvolvido e (4) disponibilizar integralmente o código fonte da aplicação e do servidor em licença livre e *open source*.

3 Procedimento metodológico

Este trabalho almeja atingir os objetivos propostos na Seção 2.1 através das seguintes etapas:

1. **Levantamento de problemas existentes**: Revisar sistematicamente a literatura com o objetivo de identificar as dificuldades, problemas e desafios enfrentados por pesquisadores e participantes ao realizar pesquisas com questionários informatizados através do uso de dispositivos móveis.

2. **Análise de requisitos**: Realizar a análise de requisitos para uma plataforma visando resolver os problemas encontrados.

3. **Análise de alternativas**: Relacionar requisitos elaborados com as funcionalidades das principais plataformas de questionários informatizados existentes.

4. **Desenvolvimento do servidor**: Desenvolvimento de um servidor RESTful que permita gerenciamento, armazenamento e exportação de questionários.

5. **Desenvolvimento do aplicativo**: Desenvolvimento de uma aplicação híbrida para iOS/Android para cadastro e preenchimento de questionários que utilize o servidor desenvolvido.

6. **Disponibilização do código fonte**: Tornar o código fonte e os demais artefatos relevantes da aplicação desenvolvida em licença *open source*.

4 Revisão da Literatura

A revisão sistemática da literatura deste trabalho teve como objetivo identificar as dificuldades, problemas e desafios enfrentados por pesquisadores e participantes ao realizar pesquisas com questionários informatizados através do uso de dispositivos móveis.

4.1 String de busca

Foi utilizado o motor de busca Scopus[1] com a *string* de busca *"(questionnaire* OR survey*) AND mobile AND (*phone* OR tablet*) AND (problem* OR challenge* OR difficult* OR issue*)"* limitando a artigos publicados desde 2015. Esta *string* foi escolhida após diversas tentativas com *strings* menos restritivas, mas que apresentavam muitos resultados, o que tornaria inviável a conferência de todos os artigos.

A parte *"(questionnaire* OR survey*)"* está presente para buscar por artigos que mencionem questionários, utilizando sinônimos e caracteres-curinga (*wildcard*) para não excluir indevidamente algum artigo possivelmente relevante. Já a parte *"mobile AND (*phone* OR tablet*)"* tem o propósito de restringir a busca apenas para artigos que utilizem tecnologias móveis, através de *smartphones* e *tablets*. Por fim, o trecho *"(problem* OR challenge* OR difficult* OR issue*)"* restringe ainda mais a busca para artigos que mencionem os problemas, desafios ou dificuldades.

4.2 Protocolo de inclusão e exclusão

Esta busca recuperou 1.762 artigos, submetidos a três etapas de seleção para a presente revisão de literatura. Na primeira etapa inseriu-se as informações de todos os artigos em uma planilha e todos os títulos foram lidos. Foram selecionados os artigos considerados pertinentes com a presente revisão e nos casos em onde houve dúvidas, por conta de títulos muito abrangentes, incluiu-se para posterior revisão na segunda etapa. Desse modo, foram selecionados 130 artigos.

A segunda etapa consistiu na leitura do título e do resumo de todos os artigos selecionados na primeira etapa, novamente excluindo artigos não relacionados com esta revisão. Ao final desta etapa, permaneceram 61 artigos.

Na terceira e última etapa o artigo foi lido integralmente e determinado se pertence à presente revisão, caso contrário, o artigo foi removido. Após esta etapa, restaram 34 artigos.

[1] <https://www.scopus.com>

Os artigos excluídos em geral tratavam de outros temas, tais como: aplicações de *mHealth*, revisões de literatura (em inglês chamadas de *surveys*) diversas, utilização de dispositivos móveis para obtenção de dados que não por questionários, questionários através de chamadas telefônicas, relação entre utilização de dispositivos móveis e aspectos comportamentais, entre outros.

4.3 Resultados

Os artigos selecionados foram lidos na íntegra com o objetivo de identificar as dificuldades, problemas e desafios enfrentados por pesquisadores ao realizar pesquisas com questionários informatizados através do uso de celulares ou tablets. A partir desta análise, foram organizadas 6 categorias com o propósito de agrupar por similaridade as dificuldades, problemas e desafios encontrados nos artigos analisados, as quais referem-se a: conectividade, *layout*, familiaridade com tecnologia, erros do aplicativo, segurança dos dados, segurança dos dispositivos e bateria, sumarizados na tabela 1.

Artigo	Conectividade	Layout	Familiaridade com Tecnologia	Erros do aplicativo	Segurança dos dados	Segurança dos dispositivos	Bateria
Piau et al. 2019			×				
Mehdi et al. 2019							×
Gummer, Quoß e Roßmann 2019		×					
Grady, Greenspan e Liu 2019		×					
Abdel-All et al. 2019							
Hershman et al. 2019				×			
Mwita, Pearson e Zwickle 2019	×						
Dietrich et al. 2018	×						
Angues et al. 2018	×			×			×
Toepoel e Funke 2018		×					
Zijlstra, Wijgergangs e Hoogendoorn-Lanser 2018	×	×					
Olmsted-Hawala et al. 2018		×					
King et al. 2017			×				
Couper e Peterson 2017	×	×					

Continuação da Tabela 1

Artigo	Conectividade	Layout	Familiaridade com Tecnologia	Erros do aplicativo	Segurança dos dados	Segurança dos dispositivos	Bateria
Marcano-Belisario et al. 2017	×				×	×	
Antoun, Couper e Conrad 2017		×					
Meyers et al. 2017	×						
Mercader et al. 2017							
Yu et al. 2017		×					
Bourezgue 2017					×	×	×
O'Reilly-Shah 2017	×						
Guyon et al. 2016	×					×	
Rajan et al. 2016		×					
Kuehne et al. 2016	×						
Zhang et al. 2016					×		
Ramsey et al. 2016			×	×			
King et al. 2016				×			
Revilla et al. 2016	×	×					
Gichohi 2016	×		×		×		×

Continuação da Tabela 1

Artigo	Conectividade	Layout	Familiaridade com Tecnologia	Erros do aplicativo	Segurança dos dados	Segurança dos dispositivos	Bateria
Backman et al. 2015			×	×			
Schick-Makaroff e Molzahn 2015			×		×		
Schobel, Pryss e Reichert 2015		×					
Lane et al. 2015	×						
Thrul, Bühler e Ferguson 2015	×						
Total	14	11	6	5	5	3	4

Tabela 1 – Artigos analisados nesta revisão

A tabela 1 apresenta a referência dos artigos analisados e quais os tipos de dificuldades, problemas e desafios foram encontrados. A seguir apresenta-se a descrição das categorias de acordo com a frequência na qual foram referidas nos artigos.

4.3.1 Conectividade

Os problemas relacionados à conexão com a Internet foram os mais citados, tendo sido citados em 14 artigos.

Os pesquisadores Mwita, Pearson e Zwickle 2019, Zijlstra, Wijgergangs e Hoogendoorn-Lanser 2018, Marcano-Belisario et al. 2017, O'Reilly-Shah 2017, Guyon et al. 2016, Kuehne et al. 2016, Lane et al. 2015 e Thrul, Bühler e Ferguson 2015 enfrentaram como um problema a falta de cobertura de redes móveis. Mesmo com avanços recentes, este problema afeta especialmente áreas rurais e países de média ou baixa renda, que possuem cobertura reduzida e de menor velocidade, quando comparados com países de alta renda [GSMA 2019].

Gichohi 2016 e Couper e Peterson 2017 mencionaram que a baixa velocidade da conexão à Internet pode afetar negativamente o respondente, aumentando o tempo necessário para responder o questionário. Já Revilla et al. 2016 citou que uma parcela significativa de usuários deixam de utilizar dispositivos móveis para acessar a internet por preocupações relacionadas com o custo do serviço.

Em Meyers et al. 2017, um problema de infraestrutura (modem defeituoso) impediu a coleta de dados durante um mês.

Angues et al. 2018 relataram problemas com a renovação de pacotes de dados com a operadora do serviço, o que fez com que parte dos entrevistadores ficassem temporariamente impossibilitados de trabalhar.

4.3.2 Layout

Problemas relacionados ao layout do questionário foram citados em 11 artigos.

Gummer, Quoß e Roßmann 2019, Zijlstra, Wijgergangs e Hoogendoorn-Lanser 2018, Antoun, Couper e Conrad 2017 e Revilla et al. 2016 defendem que questionários *web* devem ser adaptados para dispositivos utilizando layouts diferentes da versão para computador. Zijlstra, Wijgergangs e Hoogendoorn-Lanser 2018 relataram que questionários adaptados aumentam a satisfação do usuário e diminuem o número de usuários que desistem de responder. Também evidencia que elementos complexos, como questões de matriz, imagens, *drop downs* estão associados com maior taxa de desistência.

Couper e Peterson 2017 citaram que o tempo de resposta de um questionário em dispositivos móveis é maior do que num computador, e que grande parte dessa diferença

se deve ao tempo de rolagem entre as questões, especialmente em questões de matriz.

Toepoel e Funke 2018 concluiu através de um experimento que escalas do tipo *slider* (Figura 1) introduzem viés às respostas, por exemplo, se a posição inicial do marcador for na esquerda, os resultados serão enviesados para valores na esquerda, quando comparados com a mesma pergunta utilizando uma escala diferente.

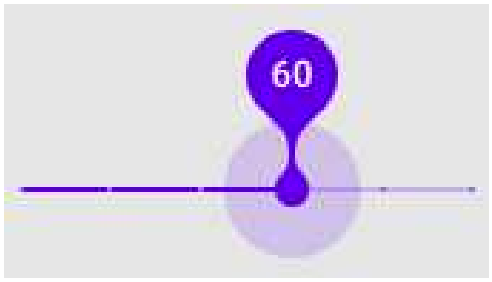

Figura 1 – Exemplo de *slider*. Fonte: <https://material.io/components/sliders/>

Olmsted-Hawala et al. 2018 demonstrou que pessoas idosas podem ter dificuldades em campos complexos de escolha única e dúvidas em relação ao significado de ícones. No primeiro caso, o design utilizado por padrão em dispositivos iOS levava significativamente mais tempo para ser respondido e foi preterido pelos usuários (Figura 2). Já no segundo caso, os usuários preferiram fortemente o uso de palavras nos botões de navegação do questionário ("Anterior"/"Próximo") no lugar de ícones (← / →), já que populações mais velhas podem não estar familiarizadas com o significado de ícones comuns em interfaces.

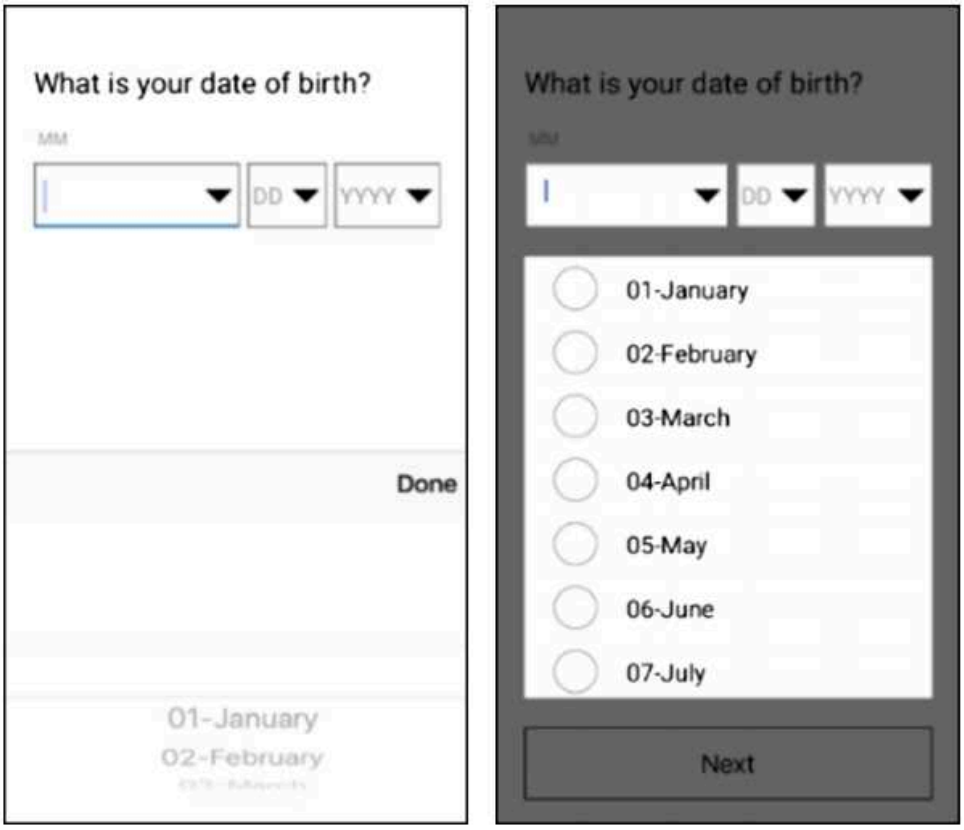

Figura 2 – Campos de escolha padrões nas plataformas iOS (esquerda) e Android (direita). Fonte: Olmsted-Hawala et al. 2018

Yu et al. 2017 menciona que pessoas com comprometimento de destreza comentaram sobre o tamanho dos botões utilizados e expressaram que prefeririam botões maiores, mais fáceis de pressionar em telas pequenas [Fitts 1954].

Rajan et al. 2016 reporta que campos de texto livre geram consideravelmente mais erros que outros tipos de campos que não envolvem digitação e recomenda a busca de soluções que evitem seu uso.

Schobel, Pryss e Reichert 2015 recomenda o uso de elementos de controle padrões (ex. botões) para garantir a familiaridade do usuário com a aplicação.

4.3.3 Familiaridade com Tecnologia

Gichohi 2016 e Backman et al. 2015 relatam que os usuários tiveram níveis muito variados de familiaridade com tecnologia. Backman et al. 2015 expõem que nem todos os recenseadores estavam inteiramente aptos a completar os questionários, portanto foi necessário enviar mais recenseadores para áreas que precisassem de ajuda. Gichohi 2016 e Schick-Makaroff e Molzahn 2015 enfatizam que o questionário deve ser adaptado para permitir seu uso por diferentes níveis de familiaridade.

Piau et al. 2019, desenvolveram uma aplicação de mHealth para ser usada por idosos e esperavam que os pacientes necessitassem do apoio de familiares ou cuidadores para

sua utilização. No entanto, precisaram realizar adaptações para facilitar o uso por idosos isolados e com baixa familiaridade com tecnologia. Ramsey et al. 2016 relataram que erros de operação dos dispositivos impediram alguns questionários de serem respondidos, como o participante habilitar o modo avião e portanto bloquear a comunicação da aplicação, necessitando que os pesquisadores entrassem em contato com o participante para resolver o problema.

King et al. 2017 observaram que pessoas mais velhas ou com níveis mais baixos de escolaridade tendem a demorar mais para completar o questionário, e que pessoas mais velhas necessitaram de mais ajuda. Apesar disso, todos os grupos preferiram utilizar a versão computadorizada do questionário e a avaliaram como mais usável.

4.3.4 Erros do aplicativo

Hershman et al. 2019 relataram que por conta de um problema na integração do HealthKit[2] com o ResearchKit[3], as informações demográficas de alguns dos primeiros participantes foram perdidas, o que fez com que os pesquisadores necessitassem entrar em contato com estes para recuperar as informações perdidas.

Angues et al. 2018 enfrentaram dificuldades com o software escolhido, como questões e respostas duplicadas e mensagens de erro. Relataram que foi necessário entrar em contato com a equipe de suporte do software para que os problemas fossem eliminados.

Backman et al. 2015 e King et al. 2016 expuseram dificuldades com dados incompletos por conta de problemas técnicos, como erros de lógica e erros no salvamento de questões fechadas, respectivamente.

Ramsey et al. 2016 também enfrentou problemas técnicos envolvendo erros não especificados no aplicativo escolhido, que foram resolvidos pelos pesquisadores.

4.3.5 Segurança dos dados

Cinco artigos relataram que a segurança dos dados respondidos nos questionários gerou preocupações.

Bourezgue 2017, Marcano-Belisario et al. 2017 e Zhang et al. 2016 citam que os dados guardados na aplicação podem ser confidenciais, e portanto é necessário que sejam armazenados e transmitidos de forma segura, a fim de impossibilitar seu acesso por pessoas não autorizadas. Marcano-Belisario et al. 2017 também menciona a importância de *compliance* à regras governamentais sobre segurança de dados quando se trabalha com dados médicos de pacientes.

[2] https://developer.apple.com/healthkit/
[3] http://researchkit.org/

Schick-Makaroff e Molzahn 2015 relatou que para cumprir os padrões de ética foi necessário utilizar senhas fortes para os dispositivos e níveis altos de criptografia.

Gichohi 2016 expõe que por conta do aumento de casos de cibercrime no Quênia, alguns participantes se sentiram desconfortáveis com suas informações sendo enviadas através da internet, para estes casos foi necessário utilizar questionários impressos, que foram posteriormente digitados e integrados com o restante da base de dados.

4.3.6 Segurança dos dispositivos

Marcano-Belisario et al. 2017, Bourezgue 2017 e Guyon et al. 2016 citaram que furto ou roubo dos dispositivos foi uma preocupação durante o desenvolvimento de suas pesquisas. Em Guyon et al. 2016 também houve precauções para proteção dos equipamentos eletrônicos contra chuva.

Marcano-Belisario et al. 2017 enfatizaram que além das implicações econômicas, roubo pode acarretar em quebra de confidencialidade, caso os dados do paciente estejam armazenados no dispositivo.

4.3.7 Bateria

O uso contínuo de certas funcionalidades de dispositivos móveis como WiFi, GPS [Angues et al. 2018] e sensores [Mehdi et al. 2019] pode causar problemas de alto consumo de bateria. Mehdi et al. 2019 recomendam que aplicações que dependam de dados coletados a partir de sensores implementem técnicas que os habilitem e desabilitem automaticamente, otimizando o consumo da bateria e eximindo o usuário da obrigação de gerenciar os sensores manualmente.

Gichohi 2016 relatam que a capacidade da bateria de alguns celulares não era suficiente para durar o dia inteiro, o que se tornou um desafio para recenseadores que trabalhavam em áreas rurais sem acesso à energia elétrica. Estratégias para resolver este problema incluíram utilizar mais de um dispositivo ou utilizar o celular do respondente. Bourezgue 2017 também citou preocupações com a autonomia da bateria dos dispositivos e recomendou o uso de carregadores portáteis.

4.4 Limitações

Entre as limitações desta revisão, reconhece-se que existem outras bases de dados que poderiam ser consultadas para aumentar o número de artigos analisados. Uma *string* de busca diferente também poderia encontrar artigos relevantes diferentes, que não foram recuperados pela *string* utilizada.

5 Análise de Requisitos

Com o objetivo de mitigar as dificuldades, problemas e desafios encontrados no Capítulo 4, foi elaborada uma lista de requisitos para uma ferramenta de questionários. Estes requisitos serão comparados com ferramentas existentes no Capítulo 6

1. Autenticação. Com o objetivo de restringir a visualização e modificação das informações inseridas na ferramenta apenas aos usuários que possuem permissão para tal, esta deverá implementar algum modo de autenticação, seja por usuário e senha, autenticação em dois fatores, *social login*, entre outros.

Requisito 1.1: O usuário deverá se cadastrar e se autenticar para utilizar o sistema.

Requisito 1.2: Apenas o usuário poderá ver suas informações e seus questionários.

2. Cadastro de Questionários. Se refere a criação de um questionário e inserção de suas perguntas. O usuário dessa funcionalidade será o pesquisador.

Requisito 2.1: O usuário poderá cadastrar questionários, e para cada questionário incluir as perguntas que desejar.

Requisito 2.2: Cada pergunta deverá especificar um tipo de resposta, como "escolha múltipla", "escolha única", "numérica", "textual", entre outras. Adicionalmente, cada tipo de pergunta pode ter campos de configuração específicos, como mínimo e máximo para campos numéricos.

Requisito 2.3: Cada pergunta poderá ser obrigatória ou opcional.

Requisito 2.4: Não deverá haver limite no número máximo de questionários e perguntas cadastradas por usuário.

3. Preenchimento de Questionários. Com o objetivo de efetivamente coletar os dados, a ferramenta deve permitir que o questionário seja preenchido e armazenado. O usuário dessa funcionalidade será principalmente o participante da pesquisa.

Requisito 3.1: O respondente poderá preencher questionários de acordo com as regras configuradas em seu cadastro.

Requisito 3.2: O aplicativo deverá permitir o preenchimento de questionários independentemente de haver conexão com a internet.

Requisito 3.3: O aplicativo deverá registrar, para cada resposta, o horário de término do preenchimento e de envio ao servidor.

Requisito 3.4: Não deverá haver limite no número máximo de respostas cadastra-

das.

Requisito 3.5: Quando o participante finalizar o preenchimento de um questionário, a ferramenta deve armazenar as respostas de forma cifrada, que possa ser decifrada apenas pelo servidor.

Requisito 3.6: O *layout* deve utilizar elementos conhecidos e seguir padrões estabelecidos de interação e tipografia.

4. Sincronização com o Servidor. Como o aplicativo armazena as respostas dos questionários no dispositivo, deve haver um passo em que estes dados são enviados para o servidor, para serem agrupados com outras respostas do mesmo questionário e posteriormente exportados.

Requisito 4.1: O usuário deverá sincronizar o aplicativo com o servidor, enviando as respostas cadastradas desde a última sincronização.

Requisito 4.2: A sincronização só poderá ser realizada enquanto houver conexão com a internet.

Requisito 4.3: A conexão entre o servidor e o aplicativo deve utilizar o protocolo seguro.

5. Exportação dos dados. Com o objetivo de permitir que o pesquisador escolha a ferramenta que mais o agrada para realizar a análise dos dados obtidos, a ferramenta deve exportar os dados para algum formato aberto para dados tabulares, como CSV [Shafranovich 2005].

Requisito 5.1: O usuário poderá exportar todas as respostas de cada um de seus questionários.

Requisito 5.2: Os dados serão entregues em um arquivo compatível com ferramentas de planilha eletrônica.

6. Precificação. Com o objetivo de permitir que o maior número possível de pesquisadores tenham acesso à ferramenta, o aplicativo e servidor devem ser gratuitos e seu código-fonte deve ser disponibilizado.

Requisito 6.1: A ferramenta deve ser gratuita e sem limites de utilização.

Requisito 6.2: O código-fonte da ferramenta deve ser disponibilizado com uma licença livre e *open source*[1].

[1] <https://www.gnu.org/philosophy/free-sw.html>

5.1 Relação entre requisitos e dificuldades encontradas

A funcionalidade 1 se refere à dificuldade 4.3.5: Segurança dos dados, já que os dados devem ser protegidos de terceiros não autorizados. Autenticação segura pode ser um requisito para aprovação de uma pesquisa por um comitê de ética [Schick-Makaroff e Molzahn 2015].

Os requisitos 3.2, 4.1 e 4.2 estão relacionados com as dificuldades 4.3.1: Conectividade e 4.3.7: Bateria. O problema mais citado nos artigos revisados foi a falta de cobertura de redes móveis, o que pode ser parcialmente mitigado armazenando as respostas no dispositivo e posteriormente as transmitindo para o servidor. Esta solução ainda necessita de internet eventualmente para realizar a sincronização, mas não impede que dados sejam coletados por falta de conexão. A redução da frequência de conexão com a internet também é uma otimização benéfica para a duração da bateria[2].

O requisito 3.5 está associado à dificuldade 4.3.5: Segurança dos dados. Utilizando criptografia assimétrica é possível armazenar os dados de forma que nem uma pessoa com acesso físico ao dispositivo possa se apropriar dos dados nele armazenados. Pode-se gerar o par de chaves no servidor e enviar para a aplicação apenas a chave pública, que será utilizada para cifrar as respostas quando o participante finalizar o preenchimento do questionário. A aplicação armazena apenas os dados cifrados, posteriormente enviando-os para o servidor, que por possuir a chave privada, pode decifrar os dados, armazenando-os em texto claro em um banco de dados seguro.

O requisito 3.6 se refere à dificuldade 4.3.2: Layout e 4.3.3: Familiaridade com tecnologia. O uso de elementos de controle padrões aumenta a familiaridade da aplicação [Schobel, Pryss e Reichert 2015]. As interações e a tipografia devem ter tamanhos adequados para garantir acessibilidade de pessoas com dificuldades motoras [Yu et al. 2017] ou visuais. O Material Design[3], linguagem de design desenvolvida pela Google usada como padrão do sistema operacional Android, por exemplo, possui recomendações de tamanho mínimo para fontes e elementos interativos[4].

O requisito 4.3 está relacionado com a dificuldade 4.3.5: Segurança dos dados, já que protocolos de comunicação seguros, como HTTPS, impedem a interceptação das informações e ataques como *man-in-the-middle* e *session hijacking* [Rescorla 2000].

Os requisitos 6.1 e 6.2 não estão associados à dificuldades encontradas durante a revisão, mas foram incluídos por saber que o preço de um software excerce influencia na decisão de usá-lo ou não em uma pesquisa. O modelo gratuito, portanto, permite que o software seja utilizado por quem não poder pagar.

[2] <https://developer.android.com/topic/performance/power>
[3] <https://material.io/>
[4] <https://material.io/design/usability/accessibility.html>

Os requisitos restantes não estão relacionados com nenhuma dificuldade específica, mas são integrais para o funcionamento de uma ferramenta de questionários.

6 Trabalhos Correlatos

Com o objetivo de identificar a existência de uma ferramenta de questionários que cumpra os requisitos elaborados no Capítulo 5, foi realizado um levantamento das principais ferramentas de questionário existentes atualmente.

As ferramentas foram encontradas através do motor de busca Google, utilizando-se as *strings* de busca "survey software"e "questionnaire software". As ferramentas localizadas nas primeiras 3 páginas de cada consulta foram adicionadas à comparação, ignorando resultados provenientes de anúncios. É sabido que o Google personaliza os resultados de acordo com o usuário, mas para esta busca é inviável utilizar um motor de busca determinístico, como o Scopus, já que este busca apenas por artigos e trabalhos científicos. Para reduzir o impacto deste problema, todas as buscas foram feitas a partir de um navegador sem *cookies*, logins e buscas prévias e utilizando um IP fornecido pela universidade. Após realizar a busca, 30 ferramentas foram localizadas, quais sejam: Checkbox, CheckMarket, Cloud Cherry, Fluid Surveys, Google Forms, Id Survey, Ingress Survey, LimeSurvey, Novi Survey, Opinio, ProProfs, Qualtrics, Questant, QuestionPro, Rotator Survey, Sawtooth, SmartSurvey, Snap Survey, SoGo Survey, StatPac, Survey Analytics, Survey Expression, SurveyGizmo, SurveyMonkey, SurveySparrow, SurveySystem, Survicate, Survio, TypeForm e Voxco.

As ferramentas utilizadas pelos artigos selecionados no Capítulo 4 também foram adicionadas à analise: Qualtrics [Mwita, Pearson e Zwickle 2019, Zijlstra, Wijgergangs e Hoogendoorn-Lanser 2018, Zijlstra, Wijgergangs e Hoogendoorn-Lanser 2018], Open Data Kit [Kuehne et al. 2016, Zhang et al. 2016], Magpi [Guyon et al. 2016, Angues et al. 2018], Snap [Marcano-Belisario et al. 2017], Blaise [Antoun, Couper e Conrad 2017] e Epi Info [Mercader et al. 2017].

Após localizar as ferramentas, todas que não possuíam planos gratuitos foram excluídas da comparação, visto que seria inviável pagar por todas para testar as funcionalidades. As ferramentas removidas por este motivo foram: Blaise[1], Checkbox[2], CheckMarket[3], Cloud Cherry[4], Id Survey[5], Ingress Survey[6], Novi Survey[7], Opinio[8], ProProfs[9], Qual-

[1] <https://blaise.com/>
[2] <https://www.checkbox.com/>
[3] <https://www.checkmarket.com/>
[4] <https://cloudcherry.com/>
[5] <https://www.idsurvey.com/en/>
[6] <https://www.ingress-survey.co.uk/>
[7] <https://novisurvey.net/>
[8] <http://www.objectplanet.com/opinio/>
[9] <https://www.proprofs.com/>

trics[10], Sawtooth[11], Snap Survey[12], SoGo Survey[13], Survey Analytics[14], SurveySparrow[15], SurveySystem[16] e Voxco[17].

Além disto, Fluid Surveys[18] e StatPac[19] foram removidas por terem sido descontinuadas. Questant[20] foi removida por não permitir a exportação dos dados no plano gratuito.

A análise das ferramentas foi feita com base nos requisitos e funcionalidades (Capítulo 5) consideradas diferenciais: precificação, adaptação do layout, preenchimento de questionários sem conexão à internet (offline), ciframento das respostas enquanto estiverem armazenadas no dispositivo.

Precificação. Esta característica está relacionada ao requisito 6.1 e é relativa ao modelo de cobrança da ferramenta. Somente foram analisadas ferramentas que possuíssem algum plano ou modo gratuito.

Adaptação do Layout. Esta característica está associada ao requisito 3.6 e está relacionada à adaptação do layout da ferramenta para dispositivos móveis. Somente foi analisado se o layout da funcionalidade de preenchimento de questionários estava adaptada, visto que é a única funcionalidade com que o participante do questionário terá contato. A adaptação do layout pode ser feita através de um aplicativo ou de um site responsivo.

Preenchimento de questionários sem conexão à internet (offline). Esta característica é referente aos requisitos 3.2, 4.1 e 4.2 e está vinculada à possibilidade de preencher os questionários na ferramenta enquanto não houver conexão à internet e a posterior sincronização do dispositivo móvel com o servidor.

Ciframento das respostas enquanto estiverem armazenadas no dispositivo. Esta característica está associada ao requisito 3.5 e está relacionado com o armazenamento das respostas obtidas no dispositivo de forma que apenas o servidor seja capaz de decifrar. Esta característica só é possível se a ferramenta também permitir preenchimento offline.

[10] <https://www.qualtrics.com/lp/survey-platform/>
[11] <http://www.sawtoothsoftware.com/survey-software>
[12] <https://www.snapsurveys.com/>
[13] <https://www.snapsurveys.com/>
[14] <https://www.surveyanalytics.com/>
[15] <https://surveysparrow.com/>
[16] <https://www.surveysystem.com/>
[17] <https://www.voxco.com/>
[18] <http://fluidsurveys.com/>
[19] <https://www.statpac.com/index.htm>
[20] <https://questant.jp/en/>

Ferramenta	Precificação	Layout Adaptado	Offline	Ciframento no armazenamento das respostas
Epi Info[21]	Gratuito	Sim	Sim	Não encontrado
Google Forms[22]	Gratuito	Sim	Não	N/A
LimeSurvey[23]	Possui plano gratuito com limite de respostas	Sim	Não	N/A
Magpi[24]	Possui plano gratuito com limite de perguntas e respostas	Sim	Sim	Não
Open Data Kit[25]	Sim	Sim	Apenas para Android	Sim
QuestionPro[26]	Possui plano gratuito com limite de perguntas e respostas	Sim	Apenas no plano pago	Não encontrado
Rotator Survey[27]	Possui plano gratuito com limite de perguntas e respostas	Sim	Apenas no plano pago	Não

[21] <https://www.cdc.gov/epiinfo/index.html>
[22] <https://www.google.com/forms/about/>
[23] <https://www.limesurvey.org/>
[24] <https://home.magpi.com/>
[25] <https://opendatakit.org/>
[26] <https://www.questionpro.com/pt-br/>
[27] <https://rotatorsurvey.com/pt/index.html>

Continuação da Tabela 2

Ferramenta	Precificação	Layout Adaptado	Offline	Ciframento no armazenamento das respostas
SmartSurvey[28]	Possui plano gratuito com limite de perguntas e respostas	Sim	Apenas no plano pago	Apenas no plano pago
Survey Expression[29]	Possui plano gratuito com limite de respostas	Não	Não	N/A
SurveyGizmo[30]	Possui plano gratuito com limite de respostas	Sim	Apenas no plano pago	Apenas no plano pago
SurveyMonkey[31]	Possui plano gratuito com limite de perguntas e respostas	Sim	Apenas no plano pago	Apenas no plano pago
Survicate[32]	Possui plano gratuito com limite de respostas	Sim	Não	N/A
Survio[33]	Possui plano gratuito com limite de respostas	Sim	Não	N/A

[28] <https://www.smartsurvey.co.uk/>
[29] <https://www.surveyexpression.com/>
[30] <https://www.surveygizmo.com/>
[31] <https://pt.surveymonkey.com/>
[32] <https://survicate.com/>
[33] <https://www.survio.com/br/>

Continuação da Tabela 2

Ferramenta	Precificação	Layout Adaptado	Offline	Ciframento no armazenamento das respostas
TypeForm[34]	Possui plano gratuito com limite de perguntas e respostas	Sim	Não	N/A

Tabela 2 – Comparação das ferramentas de questionário selecionadas

As informações presentes na Tabela 2 foram extraídas do site de cada uma das ferramentas. Em casos onde o site não especificava se o layout era adaptado para dispositivos móveis, a aplicação foi testada com um questionário de exemplo em um celular com Android.

Nos casos em que a informação sobre o ciframento das respostas enquanto armazenadas no dispositivo não foi encontrada, foi enviado um email para a empresa responsável pela ferramenta pedindo esclarecimentos sobre esta funcionalidade.

É possível notar através da Tabela 2 que nenhuma das ferramentas comparadas cumpre integralmente todas as 4 características desejadas. As que mais se aproximaram foram Epi Info e Open Data Kit, porém ambos possuem características que vão de encontro aos requisitos elaborados.

A ferramenta Epi Info tem como propósito específico a coleta e análise de dados relativos apenas à epidemiologia, não sendo então uma ferramenta de questionários de uso geral.

Já Open Data Kit é uma coleção de ferramentas relacionadas à coleta e gerenciamento de dados, entre estas ferramentas estão ODK Collect, aplicativo Android para preenchimento de questionários, e ODK Aggregate, servidor que armazena as respostas coletadas pelo aplicativo. Os dois principais problemas do Open Data Kit são sua complexidade e falta de suporte à plataforma iOS. Por conta da quantidade de ferramentas envolvidas, os usuários necessitarão de conhecimento técnico para integrá-las, o que pode tornar seu uso inviável, em favor de outra ferramenta que seja mais amigável ao usuário leigo. Não oferecer suporte ao iOS é um problema, visto que a plataforma possui uma expressiva fatia de mercado, cerca de 22% [35].

[34] <https://www.typeform.com/>
[35] <https://www.macworld.co.uk/feature/iphone/iphone-vs-android-market-share-3691861/>

7 Desenvolvimento

Este capítulo descreve o desenvolvimento da nova ferramenta de questionários informatizados para dispositivos móveis, denominada *Sage*. A ferramenta possui dois módulos principais: o servidor, desenvolvido na linguagem Python; e o aplicativo, desenvolvido na linguagem Dart, utilizando o framework Flutter.

7.1 Tecnologias utilizadas

Nesta seção serão explicados alguns conceitos utilizados no desenvolvimento da ferramenta.

7.1.1 API

Uma *Application Programming Interface* (API) é um conjunto de rotinas que representam comunicação entre componentes de software. APIs podem ser de diversos tipos, como de sistemas operacionais, de bibliotecas de software, para a Web, entre outros. Para este trabalho, destacaremos APIs para a Web.

7.1.1.1 REST

Representational State Transfer (REST) é um padrão arquitetural para APIs para a Web. O padrão define um conjunto de exigências que, quando aplicadas por completo, melhoram a escalabilidade, confiabilidade e segurança [Fielding 2000]. Uma API que respeita o padrão REST é denominada RESTful.

As 5 exigências obrigatórias do padrão REST definidas em Fielding 2000 são:

- **Cliente-Servidor**: deve utilizar a arquitetura Cliente-Servidor para minimizar o acoplamento da interface com o usuário (cliente) do armazenamento e gerenciamento dos dados (servidor). Isto simplifica e aumenta a portabilidade de ambos os componentes e permite que eles evoluam separadamente.
- ***Stateless***: as interações do cliente com o servidor não devem utilizar nenhum contexto armazenado no servidor, ou seja, toda requisição deve conter todos os dados necessários para seu entendimento.
- **Cache**: Para utilizar a rede de forma eficiente, cada resposta do servidor deve indicar se é possível realizar cache sobre seus dados. Se for possível, uma cache implementada no cliente pode reutilizar estes dados caso a mesma requisição seja feita no futuro.

- **Interface Uniforme**: recursos são unicamente identificados através de seu Uniform Resource Identifier (URI).

- **Sistema em camadas**: impede que o cliente tenha conhecimento de outros servidores além do que está efetivamente o servindo, o que reduz a complexidade e promove independência entre componentes do sistema. Pode ser usado para criar sub-sistemas que se comunicam sem o cliente precisar conhecer todas as partes. Também pode ser usado para encapsular sistemas legado.

7.1.2 JSON

JavaScript Object Notation (JSON) é um formato de texto para serialização de dados estruturados [Bray 2017].

Foi padronizado na RFC8259 e pode representar 4 tipos primitivos: *number*, *string*, *boolean* e *null*; e 2 tipos estruturados: *array* (também conhecido como o tipo *List* ou *Vector* em algumas linguagens de programação) e *object* (também conhecido como *Map* ou *Dictionary*).

```
{
    "city": "Florianópolis",
    "state": "Santa Catarina",
    "country": "Brasil",
    "population": 477798,
    "is_capital": true,
    "beaches": ["Joaquina", "Ingleses", "Campeche"]
}
```

Figura 3 – Exemplo de documento JSON

Trabalhos como Lin et al. 2012 mostram que JSON é um formato mais eficiente para transferência de dados em APIs para a Web quando comparado com XML, produzindo mensagens menores e desserialização mais rápida.

7.1.3 Programação Reativa para Interfaces

Programação Reativa é uma abordagem para o desenvolvimento de interfaces gráficas com o usuário (GUI) que proporciona abstrações alto-nível, declarativas, baseadas em composição [Czaplicki e Chong 2013]. É um paradigma utilizado pelos *frameworks* React e Flutter, e pela linguagem de programação Elm.

É caracterizada pela existência de funções puras que descrevem a interface de acordo com o estado atual da aplicação, também conhecidas como componentes. Estes

componentes podem utilizar outros componentes mais simples, enfatizando o reúso.

Funções puras são funções livres de efeitos colaterais que para um dado conjunto de argumentos, retornam sempre o mesmo valor.

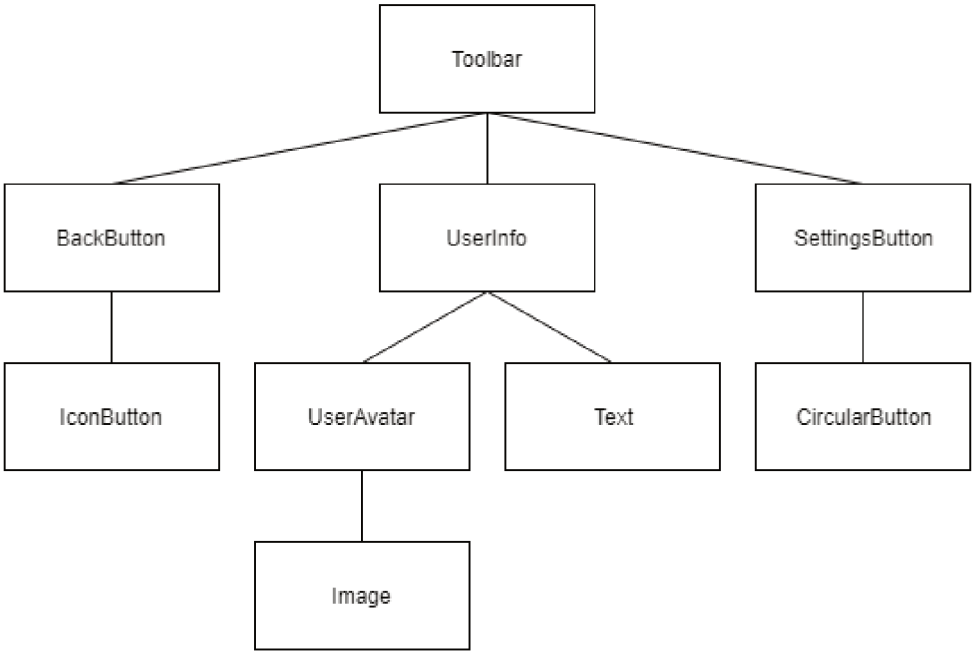

Figura 4 – Exemplo de composição de componentes para formar o componente *Toolbar*

7.1.3.1 Gerenciamento de Estado

O gerenciamento do estado é importante em aplicações utilizando o paradigma de Programação Reativa, já que toda a interface é construída de acordo com o estado. Uma solução existente para isso é a arquitetura Business Logic Component (BLoC) [Boelens 2018].

Assim como a arquitetura *Model–View–Controller* (MVC) [Reenskaug e Coplien 2009], BLoC promove a separação de responsabilidades em 3 principais componentes:

1. **Provedor de Dados**: APIs REST, bancos de dados, ou qualquer fonte de dados da aplicação. Normalmente acessado de forma assíncrona.
2. **BLoC**: implementa as regras de negócio e armazena e atualiza o estado.
3. **Interface com o Usuário**: recebe o estado do BLoC, que usa para definir como construir a interface. Também emite os eventos realizados pelo usuário para o BLoC.

A comunicação entre o BLoC e a Interface com o Usuário é feita de forma assíncrona através de *Streams*, fluxos de dados que ocorrem com o decorrer do tempo. Um BLoC

deve implementar no mínimo duas *Streams*, uma que envia o estado para a Interface com o Usuário e outra que recebe as ações executadas pelo usuário.

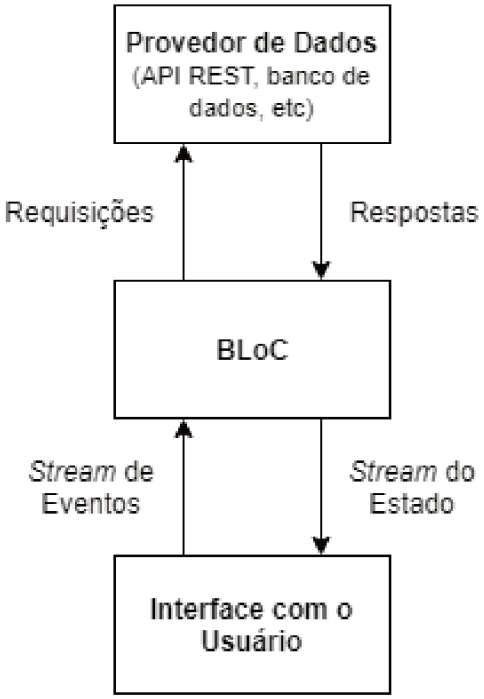

Figura 5 – Fluxo de dados da arquitetura BLoC

Por exemplo, em um aplicativo de mensagens, quando o usuário pressionar o botão de enviar mensagem, um novo evento do tipo "envio de mensagem" é adicionado à *stream* de eventos. O BLoC receberá este evento, emitirá um novo estado de carregamento na *stream* correspondente e então iniciará o envio da mensagem ao servidor. Quando a transmissão ao servidor terminar, o BLoC emitirá mais um estado de sucesso ou falha, permitindo que a interface avise ao usuário do resultado da operação.

7.1.4 Flutter

Flutter[1] é um framework para desenvolvimento de aplicativos multiplataforma, utiliza a linguagem de programação Dart[2], ambos criados pela Google. É implementado utilizando a biblioteca gráfica Skia[3], utilizada no Google Chrome, Android, Mozilla Firefox, etc.

Flutter implementa o paradigma de programação reativa para interfaces, popularizado pela biblioteca React[4], criada pelo Facebook.

[1] <https://flutter.dev/>
[2] <https://dart.dev/>
[3] <https://skia.org/>
[4] <https://reactjs.org/>

7.1.5 Flask

Flask[5] é um framework para servidores web para Python. Implementa a especificação WSGI (Web Server Gateway Interface), padrão utilizado por frameworks web que facilita a portabilidade das aplicações em diferentes *web servers*. É um dos pacotes mais utilizados do ecosistema Python[6].

7.1.6 Firebase

Firebase[7] é uma plataforma para desenvolvimento de aplicativos móveis e *web apps* que implementa diversas funcionalidades comumente encontradas nestas aplicações. Entre seu rol de serviços estão: *Auth*, permitindo cadastro e autenticação facilitado, sem precisar implementar esta funcionalidade no servidor da aplicação, suporta *social login*; *Crashlytics*, funcionalidade que permite que um aplicativo que sofreu um *crash* – encerramento inesperado por conta de erro – envie as informações sobre o problema aos seus desenvolvedores; *Analytics*, que permite aos desenvolvedores extrairem informações de como seus usuários utilizam o aplicativo; *Cloud Messaging*, que permite o envio facilitado de notificações para os dispositivos.

7.2 Modelagem de dados

A ferramenta desenvolvida possui dois modelos de dados similares, um para o servidor e outro para a aplicação. A necessidade de dois modelos diferentes surgiu do fato que o aplicativo precisa armazenar o conteúdo dos questionários de forma cifrada e só precisa armazenar dados relativos ao usuário atual.

[5] <https://palletsprojects.com/p/flask/>
[6] <https://pypistats.org/packages/flask>
[7] <https://firebase.google.com/>

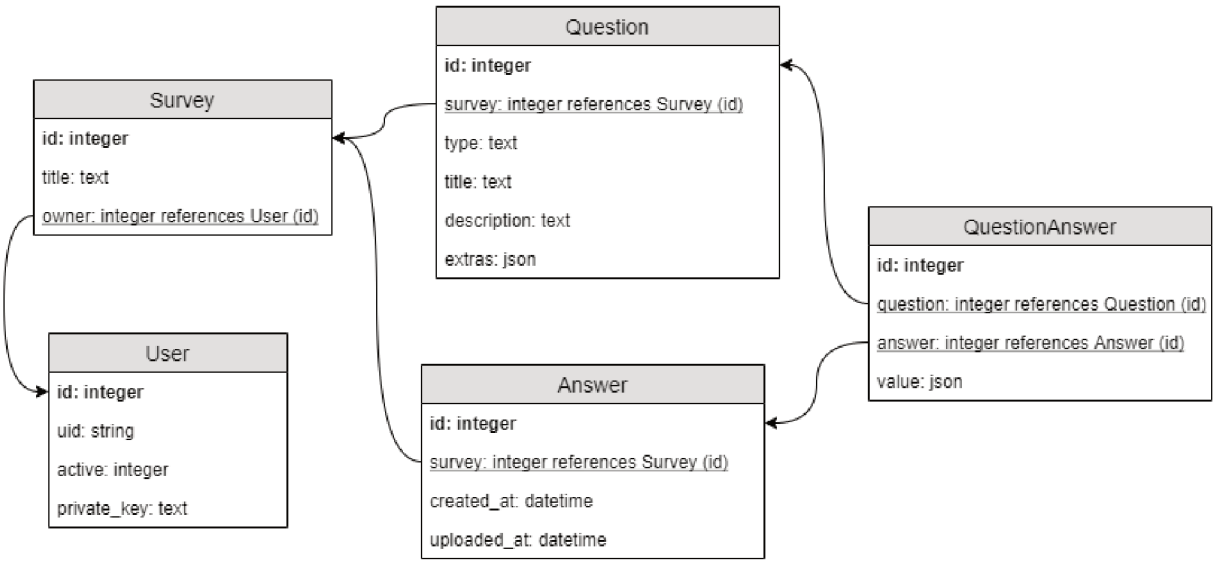

Figura 6 – Diagrama Entidade-Relacionamento do banco de dados do servidor

O modelo de dados do servidor, exposto na Figura 6, é inspirado em um modelo EAV (Entidade-Atributo-Valor), que permite que cada Entidade (*Survey*) possua quantos Atributos (*Question*) desejar. Cada resposta (*Answer*) terá exatamente um Valor (*QuestionAnswer*) para cada Atributo.

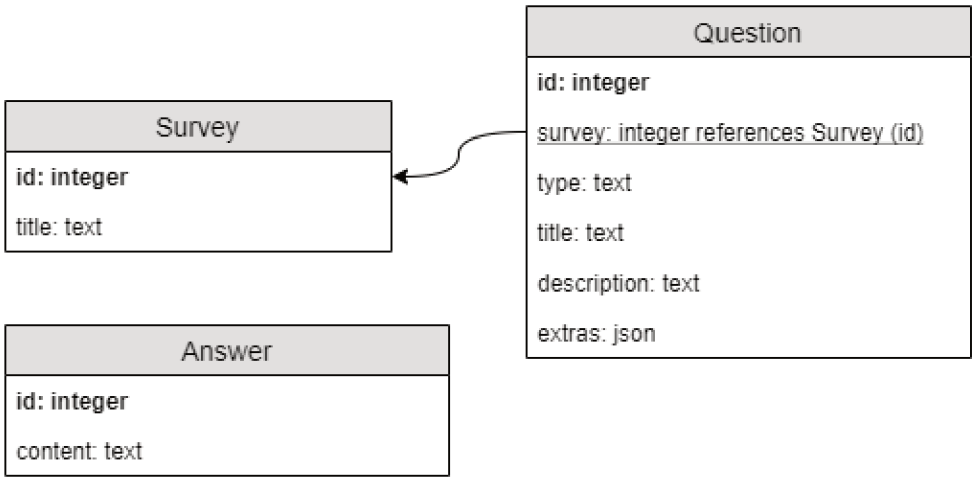

Figura 7 – Diagrama Entidade-Relacionamento do banco de dados do aplicativo

O campo *extras* da tabela *Question* utiliza o tipo de dados JSON, que permite o armazenamento de dados semi-estruturados dentro da tabela. Este tipo de dados foi incluído na oitava revisão da especificação da linguagem de consulta SQL, ISO/IEC

9075:2016[8] e é suportado por SGBDs como Postgres[9], Oracle Database[10], SQLite[11], entre outros.

O campo *extras* armazena configurações da pergunta que dependem do valor do campo *type*, por exemplo, um campo numérico pode ter um valor mínimo e máximo, já um campo de seleção múltipla necessita armazenar as alternativas possíveis. Portanto, como o tipo de dados JSON permite armazenar dados semi-estruturados, foi escolhido para este campo por sua flexibilidade, permitindo um *schema* implícito, controlado pelo servidor, não pelo SGBD.

A tabela *Answer* no banco de dados do aplicativo não possui relacionamento com nenhuma outra, visto que todas as informações da resposta estão armazenados de forma cifrada em seu campo *content*.

7.3 Autenticação

A autenticação do usuário foi implementada utilizando Firebase Auth[12], visto que proporciona grande facilidade para implementação desta funcionalidade. Foi escolhido o Google como provedor de identidade para *social login*, entre suas vantagens está a simplificação do cadastro e do login, já que o usuário precisa apenas apertar o botão "Entrar com Google" e seus principais dados (nome e email) são enviados para a aplicação, não necessitando que o usuário aprenda mais um login e senha.

[8] <https://standards.iso.org/ittf/PubliclyAvailableStandards/c067367_ISO_IEC_TR_19075-6_2017.zip>
[9] <https://www.postgresql.org/docs/10/datatype-json.html>
[10] <https://docs.oracle.com/database/121/ADXDB/json.htm#ADXDB6246>
[11] <https://www.sqlite.org/json1.html>
[12] <https://firebase.google.com/docs/auth>

Figura 8 – Tela de Login

Para autenticar o usuário no servidor, o aplicativo gera um *token* para o usuário atual através do Firebase SDK e o envia através de HTTPS para o servidor. Ao receber o *token*, o servidor utiliza a função `verify_id_token`[13] do Firebase Admin para verificar se o token é válido e corresponde à um usuário cadastrado, posteriormente o servidor pode recuperar o usuário e seus questionários utilizando o *uid* retornado pela função supracitada. O *token* é obrigatório para realizar qualquer operação sobre a API, caso não seja enviado, o servidor retornará o código de erro HTTP 401 Unauthorized.

Uma vez que o usuário efetue o login, sua sessão é mantida até que este escolha realizar o logout.

7.4 Cadastro de questionários

A funcionalidade de cadastro de questionários permite que o usuário crie novos questionários e insira perguntas dos tipos "escolha única", "escolha múltipla", "texto livre"e "numérica". O questionário requer um título e no mínimo uma pergunta para ser válido.

[13] <https://firebase.google.com/docs/auth/admin/verify-id-tokens>

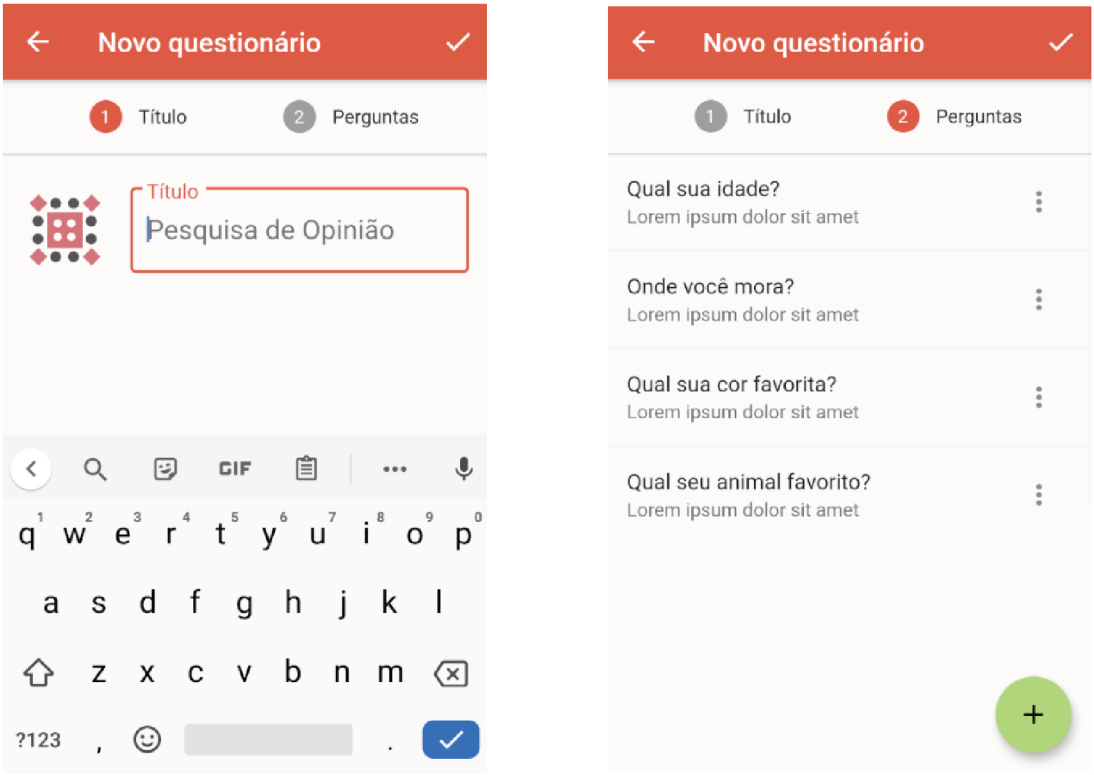

Figura 9 – Tela de Cadastro de questionários

Só é possível cadastrar um novo questionário se o dispositivo estiver conectado à internet.

7.5 Preenchimento de questionários

Para implementar essa funcionalidade, foi escolhido um layout que apresenta uma pergunta por vez ao usuário, e só permite que ele avance caso a resposta seja válida, com o objetivo de impedir preenchimentos inválidos. O usuário pode navegar livremente pelas questões já respondidas. Esta funcionalidade foi projetada para que o aplicativo seja entregue ao participante para que preencha o questionário de forma autônoma, porém, o pesquisador pode também escolher ler as perguntas e respostas ao participante e então registrar suas escolhas.

Os campos de resposta foram implementados de acordo com as especificações do Material Design[14], as questões de escolha única utilizam uma lista de *Radio Buttons*[15], as questões de escolha múltipla utilizam uma lista de *Checkboxes*[16] e as questões numéricas e de texto livre utilizam um campo de texto[17]. Questões numéricas possuem validação no campo de texto para assegurar que o conteúdo se trata de um número.

[14] <https://material.io/>
[15] <https://material.io/components/selection-controls/#radio-buttons>
[16] <https://material.io/components/selection-controls/#checkboxes>
[17] <https://material.io/components/text-fields/>

Os textos das perguntas, e de todo o aplicativo, se adaptam às configurações de acessibilidade do dispositivo, aumentando a fonte automaticamente caso o usuário deseje. Todos os elementos interativos seguem as especificações de acessibilidade do Material Design[18], como tamanho mínimo de áreas clicáveis.

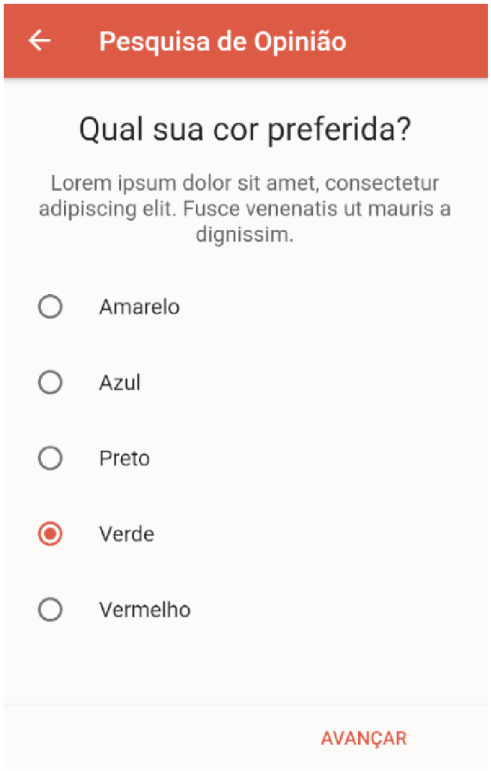

Figura 10 – Tela de preenchimento de questionário

7.6 Armazenamento das respostas no dispositivo

Para armazenar as respostas dos questionários no dispositivo de forma segura, o aplicativo utiliza criptografia assimétrica. O algoritmo RSA/OAEP [Kaliski e Staddon 1998] com chave de 4096 bits foi escolhido por prover alta segurança e estar disponível tanto no servidor quanto no aplicativo.

Durante o login, o aplicativo obtém do servidor a chave pública que deve ser utilizada para cifrar os dados. Ao finalizar o preenchimento do questionário, todos os dados são serializados em JSON e então cifrados utilizando a chave previamente obtida, então, os dados cifrados são inseridos em um registro da tabela *Answer* e a versão em texto claro dos dados é excluída.

[18] <https://material.io/design/usability/accessibility.html>

7.7 Sincronização com o Servidor

O passo de sincronização é necessário para enviar os dados cifrados que estão armazenados no dispositivo. O aplicativo realiza uma requisição HTTP POST para cada resposta armazenada; o servidor, ao receber a requisição, decifra os dados recebidos utilizando a chave privada do usuário, obtendo então um documento JSON com as respostas do questionário, que são inseridas no banco de dados. Ao receber resposta do servidor, o aplicativo exclui a resposta cifrada que acabou de enviar.

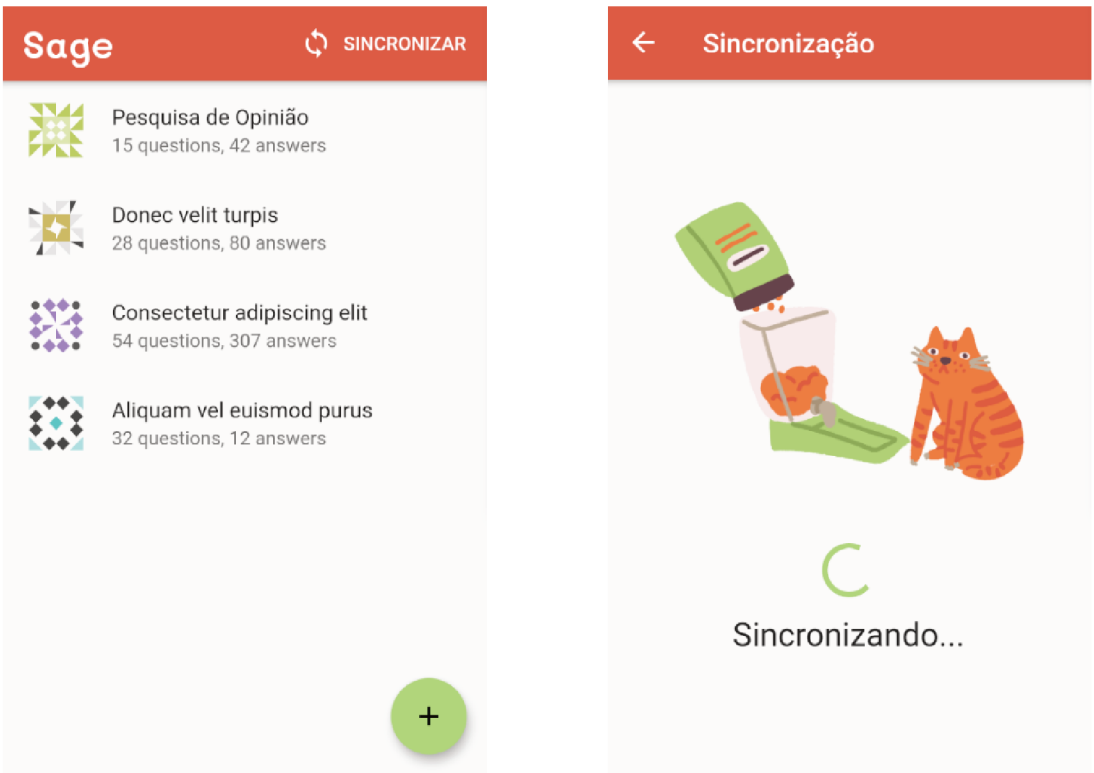

Figura 11 – Telas de listagem de questionários e de sincronização

7.8 Exportação dos dados

Para a exportação dos dados foi utilizado o formato de aquivo CSV (Comma-separated values) [Shafranovich 2005], por ser um formato aberto, simples e aceito por ferramentas de planilha eletrônica populares, como Microsoft Excel, Google Sheets e LibreOffice Calc.

Como o aplicativo é compatível apenas com dispositivos móveis e a análise dos dados do questionário provavelmente será feita em um computador de mesa ou *notebook*, foi implementado o envio do arquivo contendo os dados através de email para o pesquisador responsável pelo questionário. O email é obtido por meio dos dados disponibilizados à ferramenta quando o usuário se cadastra utilizando uma conta Google.

8 Conclusão

A informatização de questionários traz diversos benefícios em comparação com questionários em papel, mas ainda pode gerar obstáculos aos seus usuários. Ao longo deste trabalho foram identificados diversas dificuldades, problemas e desafios enfrentados por pesquisadores e participantes ao realizar pesquisas com questionários informatizados através do uso de dispositivos móveis, que se referiam principalmente à conectividade, *layout*, familiaridade com tecnologia, erros do aplicativo, segurança dos dados, segurança dos dispositivos e bateria.

A partir disso foi elaborado um conjunto de requisitos para uma ferramenta de questionários visando reduzir o impacto dessas dificuldades, problemas e desafios. Estes requisitos foram comparados com ferramentas existentes, não identificando-se nenhuma que os cumprisse integralmente.

Identificada esta lacuna, foi desenvolvida uma ferramenta de questionários que satisfizesse os requisitos elaborados. A ferramenta consiste em dois módulos, um servidor e um aplicativo, que foram disponibilizados sob a licença MIT[1] no GitHub[2] e no apêndice 2 deste trabalho.

8.1 Trabalhos futuros

Como recomendação de esforços futuros que complementem o presente trabalho, sugere-se a avaliação de usabilidade da funcionalidade de preenchimento de questionários, visto que esta foi desenvolvida a partir de recomendações da literatura, e não de entrevistas com potenciais usuários. Sabe-se que determinados coortes tem necessidades diferentes quanto a usabilidade de um sistema, como por exemplo, pessoas idosas, daltônicos, pessoas que utilizam leitores de tela. Um estudo avaliativo da usabilidade da ferramenta com estes diferentes grupos pode encontrar pontos de melhoria e dificuldades para estas populações.

Além disso, é possível realizar trabalhos técnicos de extensão da ferramenta, como:

- **Adição de novos tipos de pergunta.** Como perguntas de avaliação (nota de 1 até 5), escolha de imagens, escala Likert, etc.
- **Adição de ramificação de perguntas.** Permitir que o pesquisador registre caminhos lógicos que alteram quais perguntas devem ser respondidas de acordo com as respostas registradas.

[1] <https://choosealicense.com/licenses/mit/>
[2] <https://github.com/caiopo/sage>

- **Registro da localização em que o questionário foi respondido.** A pesquisa de Mwita, Pearson e Zwickle 2019 utilizou uma ferramenta para preencher os questionários e outra para registrar a localização atual dos recenseadores, caso a ferramenta de questionários permitisse registrar a localização atual, a segunda ferramenta não seria necessária.

- **Versão web da ferramenta.** Desenvolvimento de um *web app* que permita preencher o questionário através de um navegador. Poucas modificações seriam necessárias no servidor, visto que já é genérico o suficiente para suportar também uma ferramenta web.

Referências

ABDEL-ALL, M. et al. The development of an Android platform to undertake a discrete choice experiment in a low resource setting. *Archives of Public Health*, v. 77, n. 1, p. 20, dez. 2019. ISSN 2049-3258. Disponível em: <https://archpublichealth.biomedcentral.com/articles/10.1186/s13690-019-0346-0>. Citado na página 29.

ANGUES, R. V. et al. A real-time medical cartography of epidemic disease (Nodding syndrome) using village-based lay mHealth reporters. *PLOS Neglected Tropical Diseases*, v. 12, n. 6, p. e0006588, jun. 2018. ISSN 1935-2735. Disponível em: <http://dx.plos.org/10.1371/journal.pntd.0006588>. Citado 5 vezes nas páginas 29, 32, 35, 36 e 41.

ANTOUN, C.; COUPER, M. P.; CONRAD, F. G. Effects of Mobile versus PC Web on Survey Response Quality. *Public Opinion Quarterly*, v. 81, n. S1, p. 280–306, 2017. ISSN 0033-362X, 1537-5331. Disponível em: <https://academic.oup.com/poq/article-lookup/doi/10.1093/poq/nfw088>. Citado 3 vezes nas páginas 30, 32 e 41.

AZEVEDO, L. J. de Melo de; MIAZAKI, M.; PORFIRIO, A. J. Questionário eletrônico em ambiente android para coleta de dados. 2014. Citado na página 23.

BACKMAN, C. et al. Implementation of an electronic data collection tool to monitor nursing-sensitive indicators in a large academic health sciences centre. *Nursing leadership (Toronto, Ont.)*, v. 28, p. 77–91, 02 2015. Citado 3 vezes nas páginas 31, 34 e 35.

BOELENS, D. *Reactive Programming - Streams - BLoC*. 2018. <https://medium.com/flutter-community/reactive-programming-streams-bloc-6f0d2bd2d248>. Visitado em Junho/2019. Citado na página 49.

BOUREZGUE, T. Using tablets for the 2018 Algerian Census: Census data management and quality assessment. *Statistical Journal of the IAOS*, v. 33, n. 3, p. 777–784, ago. 2017. ISSN 18747655, 18759254. Disponível em: <http://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/SJI-160286>. Citado 3 vezes nas páginas 30, 35 e 36.

BRAY, T. *The JavaScript Object Notation (JSON) Data Interchange Format*. [S.l.], 2017. Citado na página 48.

COUPER, M. P.; PETERSON, G. J. Why Do Web Surveys Take Longer on Smartphones? *Social Science Computer Review*, v. 35, n. 3, p. 357–377, jun. 2017. ISSN 0894-4393, 1552-8286. Disponível em: <http://journals.sagepub.com/doi/10.1177/0894439316629932>. Citado 2 vezes nas páginas 29 e 32.

CZAPLICKI, E.; CHONG, S. Asynchronous functional reactive programming for guis. 2013. Citado na página 48.

DIETRICH, J. J. et al. Mobile Phone Questionnaires for Sexual Risk Data Collection Among Young Women in Soweto, South Africa. *AIDS and Behavior*,

v. 22, n. 7, p. 2312–2321, jul. 2018. ISSN 1090-7165, 1573-3254. Disponível em: <http://link.springer.com/10.1007/s10461-018-2080-y>. Citado na página 29.

DY, C. J. et al. The use of a tablet computer to complete the dash questionnaire. *The Journal of Hand Surgery*, v. 37, n. 12, p. 2589 – 2594, 2012. ISSN 0363-5023. Disponível em: <http://www.sciencedirect.com/science/article/pii/S036350231201372X>. Citado na página 23.

FIELDING, R. T. Architectural styles and the design of network-based software architectures. 2000. Citado na página 47.

FITTS, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, v. 47, n. 6, p. 381–391, 1954. ISSN 0022-1015. Disponível em: <http://doi.apa.org/getdoi.cfm?doi=10.1037/h0055392>. Citado na página 34.

GAULT, R. H. A history of the questionnaire method of research in psychology. *The Pedagogical Seminary*, Routledge, v. 14, n. 3, p. 366–383, 1907. Disponível em: <https://doi.org/10.1080/08919402.1907.10532551>. Citado na página 23.

GICHOHI, B. W. Web surveys for offline rural communities. *Statistical Journal of the IAOS*, v. 32, n. 4, p. 627–630, nov. 2016. ISSN 18747655, 18759254. Disponível em: <http://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/SJI-160992>. Citado 4 vezes nas páginas 30, 32, 34 e 36.

GILLHAM, B. *Developing a Questionnaire*. Bloomsbury Publishing, 2008. (Real World Research). ISBN 9781441154866. Disponível em: <https://books.google.com.br/books?id=EpKvAwAAQBAJ>. Citado na página 23.

GRADY, R. H.; GREENSPAN, R. L.; LIU, M. What Is the Best Size for Matrix-Style Questions in Online Surveys? *Social Science Computer Review*, v. 37, n. 3, p. 435–445, jun. 2019. ISSN 0894-4393, 1552-8286. Disponível em: <http://journals.sagepub.com/doi/10.1177/0894439318773733>. Citado na página 29.

GSMA. *The State of Mobile Internet Connectivity*. 2019. <https://www.gsma.com/mobilefordevelopment/wp-content/uploads/2019/07/GSMA-State-of-Mobile-Internet-Connectivity-Report-2019.pdf>. Visitado em Outubro/2019. Citado na página 32.

GUMMER, T.; QUOß, F.; ROßMANN, J. Does Increasing Mobile Device Coverage Reduce Heterogeneity in Completing Web Surveys on Smartphones? *Social Science Computer Review*, v. 37, n. 3, p. 371–384, jun. 2019. ISSN 0894-4393, 1552-8286. Disponível em: <http://journals.sagepub.com/doi/10.1177/0894439318766836>. Citado 2 vezes nas páginas 29 e 32.

GUYON, A. et al. Mobile-Based Nutrition and Child Health Monitoring to Inform Program Development: An Experience From Liberia. *Global Health: Science and Practice*, v. 4, n. 4, p. 661–670, dez. 2016. ISSN 2169-575X. Disponível em: <http://www.ghspjournal.org/lookup/doi/10.9745/GHSP-D-16-00189>. Citado 4 vezes nas páginas 30, 32, 36 e 41.

HERSHMAN, S. G. et al. Physical activity, sleep and cardiovascular health data for 50,000 individuals from the MyHeart Counts Study. *Scientific Data*, v. 6, n. 1, p. 24, dez. 2019. ISSN 2052-4463. Disponível em: <http://www.nature.com/articles/s41597-019-0016-7>. Citado 2 vezes nas páginas 29 e 35.

KALISKI, B.; STADDON, J. *PKCS #1: RSA Cryptography Specifications Version 2.0*. [S.l.], 1998. Citado na página 56.

KING, C. et al. The quality and diagnostic value of open narratives in verbal autopsy: a mixed-methods analysis of partnered interviews from Malawi. *BMC Medical Research Methodology*, v. 16, n. 1, p. 13, dez. 2016. ISSN 1471-2288. Disponível em: <http://www.biomedcentral.com/1471-2288/16/13>. Citado 2 vezes nas páginas 30 e 35.

KING, C. M. et al. Tablet computers and forensic and correctional psychological assessment: A randomized controlled study. *Law and Human Behavior*, v. 41, n. 5, p. 468–477, out. 2017. ISSN 1573-661X, 0147-7307. Disponível em: <http://doi.apa.org/getdoi.cfm?doi=10.1037/lhb0000245>. Citado 2 vezes nas páginas 29 e 35.

KUEHNE, A. et al. Mortality, Morbidity and Health-Seeking Behaviour during the Ebola Epidemic 2014–2015 in Monrovia Results from a Mobile Phone Survey. *PLOS Neglected Tropical Diseases*, v. 10, n. 8, p. e0004899, ago. 2016. ISSN 1935-2735. Disponível em: <http://dx.plos.org/10.1371/journal.pntd.0004899>. Citado 3 vezes nas páginas 30, 32 e 41.

LANE, B. et al. Mobile field data collection for post bushfire analysis and african farmers. v. 448, p. 160–168, 03 2015. Citado 2 vezes nas páginas 31 e 32.

LEWIS-BECK, M.; BRYMAN, A.; LIAO, T. F. *The SAGE Encyclopedia of Social Science Research Methods*. 2455 Teller Road, Thousand Oaks California 91320 United States of America: Sage Publications, Inc., 2004. ISBN 9780761923633 9781412950589. Disponível em: <http://methods.sagepub.com/reference/the-sage-encyclopedia-of-social-science-research-methods>. Citado na página 23.

Lin, B. et al. Comparison between json and xml in applications based on ajax. In: *2012 International Conference on Computer Science and Service System*. [S.l.: s.n.], 2012. p. 1174–1177. Citado na página 48.

MARCANO-BELISARIO, J. S. et al. Implementation of depression screening in antenatal clinics through tablet computers: results of a feasibility study. *BMC Medical Informatics and Decision Making*, v. 17, n. 1, p. 59, dez. 2017. ISSN 1472-6947. Disponível em: <http://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-017-0459-8>. Citado 5 vezes nas páginas 30, 32, 35, 36 e 41.

MEHDI, M. et al. Towards Automated Smart Mobile Crowdsensing for Tinnitus Research. In: *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*. Cordoba, Spain: IEEE, 2019. p. 75–80. ISBN 9781728122861. Disponível em: <https://ieeexplore.ieee.org/document/8787495/>. Citado 2 vezes nas páginas 29 e 36.

MERCADER, H. F. G. et al. Female respondent acceptance of computer-assisted personal interviewing (CAPI) for maternal, newborn and child health coverage surveys in rural Uganda. *International Journal of Medical Informatics*, v. 98, p. 41–46, fev. 2017. ISSN 13865056. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1386505616302660>. Citado 2 vezes nas páginas 30 e 41.

MEYERS, D. J. et al. Management challenges in mHealth: failures of a mobile community health worker surveillance programme in rural Nepal. *BMJ Innovations*, v. 3, n. 1, p. 19–25, fev. 2017. ISSN 2055-8074, 2055-642X. Disponível em: <http://innovations.bmj.com/lookup/doi/10.1136/bmjinnov-2015-000102>. Citado 2 vezes nas páginas 30 e 32.

MWITA, E. J.; PEARSON, A. L.; ZWICKLE, A. Effectiveness of tablet computers in spatial data collection and surveys in rural Tanzania. *The Electronic Journal of Information Systems in Developing Countries*, jul. 2019. ISSN 1681-4835, 1681-4835. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/isd2.12106>. Citado 5 vezes nas páginas 23, 29, 32, 41 e 60.

OLMSTED-HAWALA, E. et al. Optimal Data Entry Designs in Mobile Web Surveys for Older Adults. In: ZHOU, J.; SALVENDY, G. (Ed.). *Human Aspects of IT for the Aged Population. Acceptance, Communication and Participation*. Cham: Springer International Publishing, 2018. v. 10926, p. 335–354. ISBN 9783319920337 9783319920344. Disponível em: <http://link.springer.com/10.1007/978-3-319-92034-4_26>. Citado 4 vezes nas páginas 13, 29, 33 e 34.

O'REILLY-SHAH, V. N. Factors influencing healthcare provider respondent fatigue answering a globally administered in-app survey. *PeerJ*, v. 5, p. e3785, set. 2017. ISSN 2167-8359. Disponível em: <https://peerj.com/articles/3785>. Citado 2 vezes nas páginas 30 e 32.

PEREIRA, I. M. et al. Tecnologia móvel para coleta de dados de pesquisas em saúde. 2017. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-21002017000500479&lng=pt&tlng=pt>. Citado na página 23.

PIAU, A. et al. A smartphone Chatbot application to optimize monitoring of older patients with cancer. *International Journal of Medical Informatics*, v. 128, p. 18–23, ago. 2019. ISSN 13865056. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1386505619302102>. Citado 2 vezes nas páginas 29 e 34.

RAJAN, J. V. et al. Understanding the barriers to successful adoption and use of a mobile health information system in a community health center in São Paulo, Brazil: a cohort study. *BMC Medical Informatics and Decision Making*, v. 16, n. 1, p. 146, dez. 2016. ISSN 1472-6947. Disponível em: <http://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-016-0385-1>. Citado 2 vezes nas páginas 30 e 34.

RAMSEY, A. T. et al. Feasibility and Acceptability of Smartphone Assessment in Older Adults with Cognitive and Emotional Difficulties. *Journal of Technology in Human Services*, v. 34, n. 2, p. 209–223, abr. 2016. ISSN 1522-8835, 1522-8991. Disponível em: <http://www.tandfonline.com/doi/full/10.1080/15228835.2016.1170649>. Citado 2 vezes nas páginas 30 e 35.

REENSKAUG, T.; COPLIEN, J. O. *The DCI Architecture: A New Vision of Object-Oriented Programming*. 2009. <https://www.artima.com/articles/dci_vision.html>. Visitado em Junho/2019. Citado na página 49.

RESCORLA, E. *HTTP Over TLS*. [S.l.], 2000. <http://www.rfc-editor.org/rfc/rfc2818.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc2818.txt>. Citado na página 39.

REVILLA, M. et al. Do online access panels need to adapt surveys for mobile devices? *Internet Research*, v. 26, n. 5, p. 1209–1227, out. 2016. ISSN 1066-2243. Disponível em: <https://www.emerald.com/insight/content/doi/10.1108/IntR-02-2015-0032/full/html>. Citado 2 vezes nas páginas 30 e 32.

SCHICK-MAKAROFF, K.; MOLZAHN, A. Strategies to use tablet computers for collection of electronic patient-reported outcomes. *Health and Quality of Life Outcomes*, v. 13, n. 1, p. 2, 2015. ISSN 1477-7525. Disponível em: <http://hqlo.biomedcentral.com/articles/10.1186/s12955-014-0205-1>. Citado 4 vezes nas páginas 31, 34, 35 e 39.

SCHOBEL, J.; PRYSS, R.; REICHERT, M. Using Smart Mobile Devices for Collecting Structured Data in Clinical Trials: Results from a Large-Scale Case Study. In: *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*. Sao Carlos, Brazil: IEEE, 2015. p. 13–18. ISBN 9781467367752. Disponível em: <http://ieeexplore.ieee.org/document/7167446/>. Citado 3 vezes nas páginas 31, 34 e 39.

SHAFRANOVICH, Y. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. [S.l.], 2005. <http://www.rfc-editor.org/rfc/rfc4180.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc4180.txt>. Citado 2 vezes nas páginas 38 e 57.

SILVEIRA, C. E. et al. Sistema móvel para realização de pesquisas de demanda turística - a interdisciplinaridade entre turismo e sistemas de informação gerando resultados aplicados. 2014. Disponível em: <http://www.seer.ufal.br/index.php/ritur/article/view/1029>. Citado na página 23.

THRUL, J.; BüHLER, A.; FERGUSON, S. G. An Internet-Based Ecological Momentary Assessment Study Relying on Participants' Own Mobile Phones: Insights from a Study with Young Adult Smokers. *European Addiction Research*, v. 21, n. 1, p. 1–5, 2015. ISSN 1022-6877, 1421-9891. Disponível em: <https://www.karger.com/Article/FullText/363231>. Citado 2 vezes nas páginas 31 e 32.

TOEPOEL, V.; FUNKE, F. Sliders, visual analogue scales, or buttons: Influence of formats and scales in mobile and desktop surveys. *Mathematical Population Studies*, v. 25, n. 2, p. 112–122, abr. 2018. ISSN 0889-8480, 1547-724X. Disponível em: <https://www.tandfonline.com/doi/full/10.1080/08898480.2018.1439245>. Citado 2 vezes nas páginas 29 e 33.

YU, D. X. et al. Accessibility needs and challenges of a mHealth system for patients with dexterity impairments. *Disability and Rehabilitation: Assistive Technology*, v. 12, n. 1, p. 56–64, jan. 2017. ISSN 1748-3107, 1748-3115. Disponível em: <https://www.tandfonline.com/doi/full/10.3109/17483107.2015.1063171>. Citado 3 vezes nas páginas 30, 34 e 39.

ZHANG, J. et al. A feasibility study on using smartphones to conduct short-version verbal autopsies in rural China. *Population Health Metrics*, v. 14, n. 1, p. 31, dez. 2016. ISSN 1478-7954. Disponível em: <http://pophealthmetrics.biomedcentral.com/articles/10.1186/s12963-016-0100-6>. Citado 3 vezes nas páginas 30, 35 e 41.

ZIJLSTRA, T.; WIJGERGANGS, K.; HOOGENDOORN-LANSER, S. Traditional and mobile devices in computer assisted web-interviews. *Transportation Research*

Procedia, v. 32, p. 184–194, 2018. ISSN 23521465. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S235214651830187X>. Citado 3 vezes nas páginas 29, 32 e 41.

Apêndices

APÊNDICE A – Artigo

Uma Ferramenta para Extração de Esquemas de Bancos de Dados NoSQL do Tipo Grafos

Caio Pereira Oliveira[1], **Raul Sidnei Wazlawick**[1]

[1]Universidade Federal de Santa Catarina

Abstract. *The use of questionnaires in scientific research is frequent in the entire world. However, questionnaires filled with paper-and-pen generate a huge volume of data that must be typed. This process, besides requiring time, is subject to typing errors, which may impact the study results. Given this, many researchers have opted for the use of computerized questionnaires in research. The benefits of using mobile devices to complete questionnaires involve reducing errors and responding time, eliminating paper usage and faster data organization for analysis. However, computerization also has limitations such as internet signal unavailability, usability issues, and concerns about participant data security. Thus, this research aimed to develop a new questionnaire management tool to reduce the impact of the limitations indicated by the literature regarding the use of mobile devices in scientific research.*

Resumo. *O uso de questionários em pesquisas científicas é frequente tanto no cenário nacional, quanto internacional. No entanto, quando os questionários são preenchidos em papel geram um grande volume de dados que precisam ser lançados em planilhas individualmente. Esse processo, além de exigir tempo, está sujeito a erros de digitação, o que pode impactar nos resultados do estudo. Diante disso, muitos pesquisadores têm optado pelo uso de questionários informatizados em pesquisas. Os benefícios da utilização de dispositivos móveis para o preenchimento de questionários envolvem a redução de erros e do tempo de preenchimento pelos participantes, eliminação do uso de papel e agilidade na organização dos dados para análise. Contudo, a informatização também possui limitações como indisponibilidade do sinal de internet, problemas de usabilidade e preocupações sobre a segurança dos dados dos participantes. Assim, esta pesquisa teve por objetivo desenvolver uma nova ferramenta de gerenciamento de questionários, visando reduzir o impacto das limitações indicadas na literatura quanto ao uso de dispositivos móveis em pesquisas científicas.*

1. Introdução

O questionário constitui-se como uma ferramenta de coleta de dados de respondentes através de uma série de perguntas, sendo considerado como um dos instrumentos de pesquisa mais utilizados nas ciências sociais [Lewis-Beck et al. 2004]. As pesquisas em Psicologia com frequência são baseadas em dados obtidos através de questionários, o que o torna um instrumento considerado confiável por muitos pesquisadores [Gault 1907].

As vantagens de utilizar questionários são muitas, e incluem: baixo custo financeiro; baixo custo de tempo, se comparados com entrevistas presenciais ou por telefone; redução de viés do entrevistador, já que todos os participantes receberão as questões da

mesma forma; possibilidade de ser aplicado em múltiplas pessoas ao mesmo tempo; facilidade de analisar questões fechadas; entre outras [Gillham 2008].

Porém, questionários também possuem alguns desafios, como a necessidade de formulação de questões breves e claras, dificuldades com respondentes analfabetos, avaliação do impacto do modo como as perguntas foram formuladas nas respostas, impossibilidade de verificar a seriedade e honestidade das respostas, impossibilidade de corrigir equívocos na interpretação, entre outros [Gillham 2008].

Por conta da importância dos questionários, diversas ferramentas surgiram com o intuito de informatizá-los, como Google Forms [1], SurveyMonkey [2], etc. Trazer questionários para plataformas informatizadas possui benefícios, tais como: redução do tempo de preenchimento, redução do número de erros no preenchimento, redução do uso de papel, e eliminação tempo necessário para inserção dos dados em sistemas de análise de dados [Mwita et al. 2019, Pereira et al. 2017, Silveira et al. 2014, Dy et al. 2012]. O estudo em [de Melo de Azevedo et al. 2014] encontrou redução do tempo em até 50% no preenchimento e digitação dos questionários, além da eliminação de erro humano no processo.

Entretanto, a informatização também possui limitações como oscilação e indisponibilidade do sinal de redes WiFi e 3G/4G, problemas relacionados a layout mal adaptados para dispositivos móveis, preocupações com a segurança dos dados, segurança dos dispositivos, duração da bateria, entre outras. Para reduzir o impacto destas limitações, o presente estudo se propõe a desenvolver uma nova ferramenta de gerenciamento de questionários.

1.1. Objetivos

O objetivo geral deste trabalho é desenvolver um sistema que permita cadastro, preenchimento, armazenamento e exportação dos dados de questionários.

Os objetivos específicos são (1) identificar as dificuldades, problemas e desafios enfrentados por pesquisadores e participantes ao realizar pesquisas com questionários informatizados através do uso de dispositivos móveis, (2) desenvolver um servidor RESTful para gerenciamento, armazenamento e exportação de questionários, (3) desenvolver uma aplicação híbrida para iOS/Android para cadastro e preenchimento de questionários que utilize o servidor desenvolvido e (4) disponibilizar integralmente o código fonte da aplicação e do servidor em licença livre e *open source*.

2. Procedimento metodológico

Este trabalho almeja atingir os objetivos propostos na Seção 1.1 através das seguintes etapas:

1. **Levantamento de problemas existentes**: Revisar sistematicamente a literatura com o objetivo de identificar as dificuldades, problemas e desafios enfrentados por pesquisadores e participantes ao realizar pesquisas com questionários informatizados através do uso de dispositivos móveis.

[1] google.com/forms
[2] surveymonkey.com

2. **Análise de requisitos**: Realizar a análise de requisitos para uma plataforma visando resolver os problemas encontrados.
3. **Análise de alternativas**: Relacionar requisitos elaborados com as funcionalidades das principais plataformas de questionários informatizados existentes.
4. **Desenvolvimento do servidor**: Desenvolvimento de um servidor RESTful que permita gerenciamento, armazenamento e exportação de questionários.
5. **Desenvolvimento do aplicativo**: Desenvolvimento de uma aplicação híbrida para iOS/Android para cadastro e preenchimento de questionários que utilize o servidor desenvolvido.
6. **Disponibilização do código fonte**: Tornar o código fonte e os demais artefatos relevantes da aplicação desenvolvida em licença *open source*.

3. Revisão da Literatura

A revisão sistemática da literatura deste trabalho teve como objetivo identificar as dificuldades, problemas e desafios enfrentados por pesquisadores e participantes ao realizar pesquisas com questionários informatizados através do uso de dispositivos móveis.

3.1. *String* de busca

Foi utilizado o motor de busca Scopus[3] com a *string* de busca *"(questionnaire* OR survey*) AND mobile AND (*phone* OR tablet*) AND (problem* OR challenge* OR difficult* OR issue*)"* limitando a artigos publicados desde 2015. Esta *string* foi escolhida após diversas tentativas com *strings* menos restritivas, mas que apresentavam muitos resultados, o que tornaria inviável a conferência de todos os artigos.

A parte *"(questionnaire* OR survey*)"* está presente para buscar por artigos que mencionem questionários, utilizando sinônimos e caracteres-curinga (*wildcard*) para não excluir indevidamente algum artigo possivelmente relevante. Já a parte *"mobile AND (*phone* OR tablet*)"* tem o propósito de restringir a busca apenas para artigos que utilizem tecnologias móveis, através de *smartphones* e *tablets*. Por fim, o trecho *"(problem* OR challenge* OR difficult* OR issue*)"* restringe ainda mais a busca para artigos que mencionem os problemas, desafios ou dificuldades.

3.2. Protocolo de inclusão e exclusão

Esta busca recuperou 1.762 artigos, submetidos a três etapas de seleção para a presente revisão de literatura. Na primeira etapa inseriu-se as informações de todos os artigos em uma planilha e todos os títulos foram lidos. Foram selecionados os artigos considerados pertinentes com a presente revisão e nos casos em onde houve dúvidas, por conta de títulos muito abrangentes, incluiu-se para posterior revisão na segunda etapa. Desse modo, foram selecionados 130 artigos.

A segunda etapa consistiu na leitura do título e do resumo de todos os artigos selecionados na primeira etapa, novamente excluindo artigos não relacionados com esta revisão. Ao final desta etapa, permaneceram 61 artigos.

Na terceira e última etapa o artigo foi lido integralmente e determinado se pertence à presente revisão, caso contrário, o artigo foi removido. Após esta etapa, restaram 34 artigos.

[3] `https://www.scopus.com`

Os artigos excluídos em geral tratavam de outros temas, tais como: aplicações de *mHealth*, revisões de literatura (em inglês chamadas de *surveys*) diversas, utilização de dispositivos móveis para obtenção de dados que não por questionários, questionários através de chamadas telefônicas, relação entre utilização de dispositivos móveis e aspectos comportamentais, entre outros.

3.3. Resultados

Os artigos selecionados foram lidos na íntegra com o objetivo de identificar as dificuldades, problemas e desafios enfrentados por pesquisadores ao realizar pesquisas com questionários informatizados através do uso de celulares ou tablets. A partir desta análise, foram organizadas 6 categorias com o propósito de agrupar por similaridade as dificuldades, problemas e desafios encontrados nos artigos analisados, as quais referem-se a: conectividade, *layout*, familiaridade com tecnologia, erros do aplicativo, segurança dos dados, segurança dos dispositivos e bateria. A seguir apresenta-se a descrição das categorias de acordo com a frequência na qual foram referidas nos artigos.

3.3.1. Conectividade

Os problemas relacionados à conexão com a Internet foram os mais citados, tendo sido citados em 14 artigos.

Os pesquisadores [Mwita et al. 2019], [Zijlstra et al. 2018], [Marcano-Belisario et al. 2017], [O'Reilly-Shah 2017], [Guyon et al. 2016], [Kuehne et al. 2016], [Lane et al. 2015] e [Thrul et al. 2015] enfrentaram como um problema a falta de cobertura de redes móveis. Mesmo com avanços recentes, este problema afeta especialmente áreas rurais e países de média ou baixa renda, que possuem cobertura reduzida e de menor velocidade, quando comparados com países de alta renda [GSMA 2019].

[Gichohi 2016] e [Couper and Peterson 2017] mencionaram que a baixa velocidade da conexão à Internet pode afetar negativamente o respondente, aumentando o tempo necessário para responder o questionário. Já [Revilla et al. 2016] citou que uma parcela significativa de usuários deixam de utilizar dispositivos móveis para acessar a internet por preocupações relacionadas com o custo do serviço.

Em [Meyers et al. 2017], um problema de infraestrutura (modem defeituoso) impediu a coleta de dados durante um mês.

[Valdes Angues et al. 2018] relataram problemas com a renovação de pacotes de dados com a operadora do serviço, o que fez com que parte dos entrevistadores ficassem temporariamente impossibilitados de trabalhar.

3.3.2. Layout

Problemas relacionados ao layout do questionário foram citados em 11 artigos.

[Gummer et al. 2019], [Zijlstra et al. 2018], [Antoun et al. 2017] e [Revilla et al. 2016] defendem que questionários *web* devem ser adaptados para dispositivos utilizando layouts diferentes da versão para computador. [Zijlstra et al. 2018]

relataram que questionários adaptados aumentam a satisfação do usuário e diminuem o número de usuários que desistem de responder. Também evidencia que elementos complexos, como questões de matriz, imagens, *drop downs* estão associados com maior taxa de desistência.

[Couper and Peterson 2017] citaram que o tempo de resposta de um questionário em dispositivos móveis é maior do que num computador, e que grande parte dessa diferença se deve ao tempo de rolagem entre as questões, especialmente em questões de matriz.

[Toepoel and Funke 2018] concluiu através de um experimento que escalas do tipo *slider* introduzem viés às respostas, por exemplo, se a posição inicial do marcador for na esquerda, os resultados serão enviesados para valores na esquerda, quando comparados com a mesma pergunta utilizando uma escala diferente.

[Olmsted-Hawala et al. 2018] demonstrou que pessoas idosas podem ter dificuldades em campos complexos de escolha única e dúvidas em relação ao significado de ícones. No primeiro caso, o design utilizado por padrão em dispositivos iOS levava significativamente mais tempo para ser respondido e foi preterido pelos usuários. Já no segundo caso, os usuários preferiram fortemente o uso de palavras nos botões de navegação do questionário (”Anterior”/”Próximo”) no lugar de ícones ($\leftarrow$ / $\rightarrow$), já que populações mais velhas podem não estar familiarizadas com o significado de ícones comuns em interfaces.

[Yu et al. 2017] menciona que pessoas com comprometimento de destreza comentaram sobre o tamanho dos botões utilizados e expressaram que prefeririam botões maiores, mais fáceis de pressionar em telas pequenas [Fitts 1954].

[Rajan et al. 2016] reporta que campos de texto livre geram consideravelmente mais erros que outros tipos de campos que não envolvem digitação e recomenda a busca de soluções que evitem seu uso.

[Schobel et al. 2015] recomenda o uso de elementos de controle padrões (ex. botões) para garantir a familiaridade do usuário com a aplicação.

3.3.3. Familiaridade com Tecnologia

[Gichohi 2016] e [Backman et al. 2015] relatam que os usuários tiveram níveis muito variados de familiaridade com tecnologia. [Backman et al. 2015] expõem que nem todos os recenseadores estavam inteiramente aptos a completar os questionários, portanto foi necessário enviar mais recenseadores para áreas que precisassem de ajuda. [Gichohi 2016] e [Schick-Makaroff and Molzahn 2015] enfatizam que o questionário deve ser adaptado para permitir seu uso por diferentes níveis de familiaridade.

[Piau et al. 2019], desenvolveram uma aplicação de mHealth para ser usada por idosos e esperavam que os pacientes necessitassem do apoio de familiares ou cuidadores para sua utilização. No entanto, precisaram realizar adaptações para facilitar o uso por idosos isolados e com baixa familiaridade com tecnologia. [Ramsey et al. 2016] relataram que erros de operação dos dispositivos impediram alguns questionários de serem respondidos, como o participante habilitar o modo avião e portanto bloquear a comunicação

da aplicação, necessitando que os pesquisadores entrassem em contato com o participante para resolver o problema.

[King et al. 2017] observaram que pessoas mais velhas ou com níveis mais baixos de escolaridade tendem a demorar mais para completar o questionário, e que pessoas mais velhas necessitaram de mais ajuda. Apesar disso, todos os grupos preferiram utilizar a versão computadorizada do questionário e a avaliaram como mais usável.

3.3.4. Erros do aplicativo

[Hershman et al. 2019] relataram que por conta de um problema na integração do HealthKit[4] com o ResearchKit[5], as informações demográficas de alguns dos primeiros participantes foram perdidas, o que fez com que os pesquisadores necessitassem entrar em contato com estes para recuperar as informações perdidas.

[Valdes Angues et al. 2018] enfrentaram dificuldades com o software escolhido, como questões e respostas duplicadas e mensagens de erro. Relataram que foi necessário entrar em contato com a equipe de suporte do software para que os problemas fossem eliminados.

[Backman et al. 2015] e [King et al. 2016] expuseram dificuldades com dados incompletos por conta de problemas técnicos, como erros de lógica e erros no salvamento de questões fechadas, respectivamente.

[Ramsey et al. 2016] também enfrentou problemas técnicos envolvendo erros não especificados no aplicativo escolhido, que foram resolvidos pelos pesquisadores.

3.3.5. Segurança dos dados

Cinco artigos relataram que a segurança dos dados respondidos nos questionários gerou preocupações.

[Bourezgue 2017], [Marcano-Belisario et al. 2017] e [Zhang et al. 2016] citam que os dados guardados na aplicação podem ser confidenciais, e portanto é necessário que sejam armazenados e transmitidos de forma segura, a fim de impossibilitar seu acesso por pessoas não autorizadas. [Marcano-Belisario et al. 2017] também menciona a importância de *compliance* à regras governamentais sobre segurança de dados quando se trabalha com dados médicos de pacientes.

[Schick-Makaroff and Molzahn 2015] relatou que para cumprir os padrões de ética foi necessário utilizar senhas fortes para os dispositivos e níveis altos de criptografia.

[Gichohi 2016] expõe que por conta do aumento de casos de cibercrime no Quênia, alguns participantes se sentiram desconfortáveis com suas informações sendo enviadas através da internet, para estes casos foi necessário utilizar questionários impressos, que foram posteriormente digitados e integrados com o restante da base de dados.

[4]https://developer.apple.com/healthkit/
[5]http://researchkit.org/

3.3.6. Segurança dos dispositivos

[Marcano-Belisario et al. 2017], [Bourezgue 2017] e [Guyon et al. 2016] citaram que furto ou roubo dos dispositivos foi uma preocupação durante o desenvolvimento de suas pesquisas. Em [Guyon et al. 2016] também houve precauções para proteção dos equipamentos eletrônicos contra chuva.

[Marcano-Belisario et al. 2017] enfatizaram que além das implicações econômicas, roubo pode acarretar em quebra de confidencialidade, caso os dados do paciente estejam armazenados no dispositivo.

3.4. Bateria

O uso contínuo de certas funcionalidades de dispositivos móveis como WiFi, GPS [Valdes Angues et al. 2018] e sensores [Mehdi et al. 2019] pode causar problemas de alto consumo de bateria. [Mehdi et al. 2019] recomendam que aplicações que dependam de dados coletados a partir de sensores implementem técnicas que os habilitem e desabilitem automaticamente, otimizando o consumo da bateria e eximindo o usuário da obrigação de gerenciar os sensores manualmente.

[Gichohi 2016] relatam que a capacidade da bateria de alguns celulares não era suficiente para durar o dia inteiro, o que se tornou um desafio para recenseadores que trabalhavam em áreas rurais sem acesso à energia elétrica. Estratégias para resolver este problema incluíram utilizar mais de um dispositivo ou utilizar o celular do respondente. [Bourezgue 2017] também citou preocupações com a autonomia da bateria dos dispositivos e recomendou o uso de carregadores portáteis.

3.5. Limitações

Entre as limitações desta revisão, reconhece-se que existem outras bases de dados que poderiam ser consultadas para aumentar o número de artigos analisados. Uma *string* de busca diferente também poderia encontrar artigos relevantes diferentes, que não foram recuperados pela *string* utilizada.

4. Análise de Requisitos

Com o objetivo de mitigar as dificuldades, problemas e desafios encontrados no Capítulo 3, foi elaborada uma lista de requisitos para uma ferramenta de questionários. Estes requisitos serão comparados com ferramentas existentes na Seção 6

1. Autenticação. Com o objetivo de restringir a visualização e modificação das informações inseridas na ferramenta apenas aos usuários que possuem permissão para tal, esta deverá implementar algum modo de autenticação, seja por usuário e senha, autenticação em dois fatores, *social login*, entre outros.

Requisito 1.1: O usuário deverá se cadastrar e se autenticar para utilizar o sistema.

Requisito 1.2: Apenas o usuário poderá ver suas informações e seus questionários.

2. Cadastro de Questionários. Se refere a criação de um questionário e inserção de suas perguntas. O usuário dessa funcionalidade será o pesquisador.

Requisito 2.1: O usuário poderá cadastrar questionários, e para cada questionário incluir as perguntas que desejar.

Requisito 2.2: Cada pergunta deverá especificar um tipo de resposta, como ”escolha múltipla”, ”escolha única”, ”numérica”, ”textual”, entre outras. Adicionalmente, cada tipo de pergunta pode ter campos de configuração específicos, como mínimo e máximo para campos numéricos.

Requisito 2.3: Cada pergunta poderá ser obrigatória ou opcional.

Requisito 2.4: Não deverá haver limite no número máximo de questionários e perguntas cadastradas por usuário.

3. Preenchimento de Questionários. Com o objetivo de efetivamente coletar os dados, a ferramenta deve permitir que o questionário seja preenchido e armazenado. O usuário dessa funcionalidade será principalmente o participante da pesquisa.

Requisito 3.1: O respondente poderá preencher questionários de acordo com as regras configuradas em seu cadastro.

Requisito 3.2: O aplicativo deverá permitir o preenchimento de questionários independentemente de haver conexão com a internet.

Requisito 3.3: O aplicativo deverá registrar, para cada resposta, o horário de término do preenchimento e de envio ao servidor.

Requisito 3.4: Não deverá haver limite no número máximo de respostas cadastradas.

Requisito 3.5: Quando o participante finalizar o preenchimento de um questionário, a ferramenta deve armazenar as respostas de forma cifrada, que possa ser decifrada apenas pelo servidor.

Requisito 3.6: O *layout* deve utilizar elementos conhecidos e seguir padrões estabelecidos de interação e tipografia.

4. Sincronização com o Servidor. Como o aplicativo armazena as respostas dos questionários no dispositivo, deve haver um passo em que estes dados são enviados para o servidor, para serem agrupados com outras respostas do mesmo questionário e posteriormente exportados.

Requisito 4.1: O usuário deverá sincronizar o aplicativo com o servidor, enviando as respostas cadastradas desde a última sincronização.

Requisito 4.2: A sincronização só poderá ser realizada enquanto houver conexão com a internet.

Requisito 4.3: A conexão entre o servidor e o aplicativo deve utilizar o protocolo seguro.

5. Exportação dos dados. Com o objetivo de permitir que o pesquisador escolha a ferramenta que mais o agrada para realizar a análise dos dados obtidos, a ferramenta deve exportar os dados para algum formato aberto para dados tabulares, como CSV [Shafranovich 2005].

Requisito 5.1: O usuário poderá exportar todas as respostas de cada um de seus

questionários.

Requisito 5.2: Os dados serão entregues em um arquivo compatível com ferramentas de planilha eletrônica.

6. Precificação. Com o objetivo de permitir que o maior número possível de pesquisadores tenham acesso à ferramenta, o aplicativo e servidor devem ser gratuitos e seu código-fonte deve ser disponibilizado.

Requisito 6.1: A ferramenta deve ser gratuita e sem limites de utilização.

Requisito 6.2: O código-fonte da ferramenta deve ser disponibilizado com uma licença livre e *open source*[6].

5. Relação entre requisitos e dificuldades encontradas

A funcionalidade 1 se refere à dificuldade 3.3.5: Segurança dos dados, já que os dados devem ser protegidos de terceiros não autorizados. Autenticação segura pode ser um requisito para aprovação de uma pesquisa por um comitê de ética [Schick-Makaroff and Molzahn 2015].

Os requisitos 3.2, 4.1 e 4.2 estão relacionados com as dificuldades 3.3.1: Conectividade e 3.4: Bateria. O problema mais citado nos artigos revisados foi a falta de cobertura de redes móveis, o que pode ser parcialmente mitigado armazenando as respostas no dispositivo e posteriormente as transmitindo para o servidor. Esta solução ainda necessita de internet eventualmente para realizar a sincronização, mas não impede que dados sejam coletados por falta de conexão. A redução da frequência de conexão com a internet também é uma otimização benéfica para a duração da bateria[7].

O requisito 3.5 está associado à dificuldade 3.3.5: Segurança dos dados. Utilizando criptografia assimétrica é possível armazenar os dados de forma que nem uma pessoa com acesso físico ao dispositivo possa se apropriar dos dados nele armazenados. Pode-se gerar o par de chaves no servidor e enviar para a aplicação apenas a chave pública, que será utilizada para cifrar as respostas quando o participante finalizar o preenchimento do questionário. A aplicação armazena apenas os dados cifrados, posteriormente enviando-os para o servidor, que por possuir a chave privada, pode decifrar os dados, armazenando-os em texto claro em um banco de dados seguro.

O requisito 3.6 se refere à dificuldade 3.3.2: Layout e 3.3.3: Familiaridade com tecnologia. O uso de elementos de controle padrões aumenta a familiaridade da aplicação [Schobel et al. 2015]. As interações e a tipografia devem ter tamanhos adequados para garantir acessibilidade de pessoas com dificuldades motoras[Yu et al. 2017] ou visuais. O Material Design[8], linguagem de design desenvolvida pela Google usada como padrão do sistema operacional Android, por exemplo, possui recomendações de tamanho mínimo para fontes e elementos interativos[9].

O requisito 4.3 está relacionado com a dificuldade 3.3.5: Segurança dos dados, já que protocolos de comunicação seguros, como HTTPS, impedem a interceptação das

[6] `https://www.gnu.org/philosophy/free-sw.html`
[7] `https://developer.android.com/topic/performance/power`
[8] `https://material.io/`
[9] `https://material.io/design/usability/accessibility.html`

informações e ataques como *man-in-the-middle* e *session hijacking* [Rescorla 2000].

Os requisitos 6.1 e 6.2 não estão associados à dificuldades encontradas durante a revisão, mas foram incluídos por saber que o preço de um software excerce influencia na decisão de usá-lo ou não em uma pesquisa. O modelo gratuito, portanto, permite que o software seja utilizado por quem não poder pagar.

Os requisitos restantes não estão relacionados com nenhuma dificuldade específica, mas são integrais para o funcionamento de uma ferramenta de questionários.

6. Trabalhos Correlatos

Com o objetivo de identificar a existência de uma ferramenta de questionários que cumpra os requisitos elaborados no Capítulo 4, foi realizado um levantamento das principais ferramentas de questionário existentes atualmente.

As ferramentas foram encontradas através do motor de busca Google, utilizando-se as *strings* de busca "survey software"e "questionnaire software". As ferramentas localizadas nas primeiras 3 páginas de cada consulta foram adicionadas à comparação, ignorando resultados provenientes de anúncios. É sabido que o Google personaliza os resultados de acordo com o usuário, mas para esta busca é inviável utilizar um motor de busca determinístico, como o Scopus, já que este busca apenas por artigos e trabalhos científicos. Para reduzir o impacto deste problema, todas as buscas foram feitas a partir de um navegador sem *cookies*, logins e buscas prévias e utilizando um IP fornecido pela universidade. Após realizar a busca, 30 ferramentas foram localizadas, quais sejam: Checkbox, CheckMarket, Cloud Cherry, Fluid Surveys, Google Forms, Id Survey, Ingress Survey, LimeSurvey, Novi Survey, Opinio, ProProfs, Qualtrics, Questant, QuestionPro, Rotator Survey, Sawtooth, SmartSurvey, Snap Survey, SoGo Survey, StatPac, Survey Analytics, Survey Expression, SurveyGizmo, SurveyMonkey, SurveySparrow, SurveySystem, Survicate, Survio, TypeForm e Voxco.

As ferramentas utilizadas pelos artigos selecionados no Capítulo 3 também foram adicionadas à analise: Qualtrics [Mwita et al. 2019, Zijlstra et al. 2018, Zijlstra et al. 2018], Open Data Kit [Kuehne et al. 2016, Zhang et al. 2016], Magpi [Guyon et al. 2016, Valdes Angues et al. 2018], Snap [Marcano-Belisario et al. 2017], Blaise [Antoun et al. 2017] e Epi Info [Mercader et al. 2017].

Após localizar as ferramentas, todas que não possuíam planos gratuitos foram excluídas da comparação, visto que seria inviável pagar por todas para testar as funcionalidades. As ferramentas removidas por este motivo foram: Blaise[10], Checkbox[11], CheckMarket[12], Cloud Cherry[13], Id Survey[14], Ingress Survey[15], Novi Survey[16], Opinio[17],

[10] https://blaise.com/
[11] https://www.checkbox.com/
[12] https://www.checkmarket.com/
[13] https://cloudcherry.com/
[14] https://www.idsurvey.com/en/
[15] https://www.ingress-survey.co.uk/
[16] https://novisurvey.net/
[17] http://www.objectplanet.com/opinio/

ProProfs[18], Qualtrics[19], Sawtooth[20], Snap Survey[21], SoGo Survey[22], Survey Analytics[23], SurveySparrow[24], SurveySystem[25] e Voxco[26].

Além disto, Fluid Surveys[27] e StatPac[28] foram removidas por terem sido descontinuadas. Questant[29] foi removida por não permitir a exportação dos dados no plano gratuito.

A análise das ferramentas foi feita com base nos requisitos e funcionalidades (Capítulo 4) consideradas diferenciais: precificação, adaptação do layout, preenchimento de questionários sem conexão à internet (offline), ciframento das respostas enquanto estiverem armazenadas no dispositivo.

Precificação. Esta característica está relacionada ao requisito 6.1 e é relativa ao modelo de cobrança da ferramenta. Somente foram analisadas ferramentas que possuíssem algum plano ou modo gratuito.

Adaptação do Layout. Esta característica está associada ao requisito 3.6 e está relacionada à adaptação do layout da ferramenta para dispositivos móveis. Somente foi analisado se o layout da funcionalidade de preenchimento de questionários estava adaptada, visto que é a única funcionalidade com que o participante do questionário terá contato. A adaptação do layout pode ser feita através de um aplicativo ou de um site responsivo.

Preenchimento de questionários sem conexão à internet (offline). Esta característica é referente aos requisitos 3.2, 4.1 e 4.2 e está vinculada à possibilidade de preencher os questionários na ferramenta enquanto não houver conexão à internet e a posterior sincronização do dispositivo móvel com o servidor.

Ciframento das respostas enquanto estiverem armazenadas no dispositivo. Esta característica está associada ao requisito 3.5 e está relacionado com o armazenamento das respostas obtidas no dispositivo de forma que apenas o servidor seja capaz de decifrar. Esta característica só é possível se a ferramenta também permitir preenchimento offline.

Nos casos em que a informação sobre o ciframento das respostas enquanto armazenadas no dispositivo não foi encontrada, foi enviado um email para a empresa responsável pela ferramenta pedindo esclarecimentos sobre esta funcionalidade.

Nenhuma das ferramentas comparadas cumpre integralmente todas as 4 características desejadas. As que mais se aproximaram foram Epi Info e Open Data Kit, porém ambos possuem características que vão de encontro aos requisitos elaborados.

---

[18] https://www.proprofs.com/
[19] https://www.qualtrics.com/lp/survey-platform/
[20] http://www.sawtoothsoftware.com/survey-software
[21] https://www.snapsurveys.com/
[22] https://www.snapsurveys.com/
[23] https://www.surveyanalytics.com/
[24] https://surveysparrow.com/
[25] https://www.surveysystem.com/
[26] https://www.voxco.com/
[27] http://fluidsurveys.com/
[28] https://www.statpac.com/index.htm
[29] https://questant.jp/en/

A ferramenta Epi Info tem como propósito específico a coleta e análise de dados relativos apenas à epidemiologia, não sendo então uma ferramenta de questionários de uso geral.

Já Open Data Kit é uma coleção de ferramentas relacionadas à coleta e gerenciamento de dados, entre estas ferramentas estão ODK Collect, aplicativo Android para preenchimento de questionários, e ODK Aggregate, servidor que armazena as respostas coletadas pelo aplicativo. Os dois principais problemas do Open Data Kit são sua complexidade e falta de suporte à plataforma iOS. Por conta da quantidade de ferramentas envolvidas, os usuários necessitarão de conhecimento técnico para integrá-las, o que pode tornar seu uso inviável, em favor de outra ferramenta que seja mais amigável ao usuário leigo. Não oferecer suporte ao iOS é um problema, visto que a plataforma possui uma expressiva fatia de mercado, cerca de 22% [30].

7. Desenvolvimento

Este capítulo descreve o desenvolvimento da nova ferramenta de questionários informatizados para dispositivos móveis, denominada *Sage*. A ferramenta possui dois módulos principais: o servidor, desenvolvido na linguagem Python; e o aplicativo, desenvolvido na linguagem Dart, utilizando o framework Flutter.

7.1. Tecnologias utilizadas

Nesta seção serão explicados alguns conceitos utilizados no desenvolvimento da ferramenta.

7.1.1. API

Uma *Application Programming Interface* (API) é um conjunto de rotinas que representam comunicação entre componentes de software. APIs podem ser de diversos tipos, como de sistemas operacionais, de bibliotecas de software, para a Web, entre outros. Para este trabalho, destacaremos APIs para a Web.

7.1.2. REST

Representational State Transfer (REST) é um padrão arquitetural para APIs para a Web. O padrão define um conjunto de exigências que, quando aplicadas por completo, melhoram a escalabilidade, confiabilidade e segurança [Fielding 2000]. Uma API que respeita o padrão REST é denominada RESTful.

As 5 exigências obrigatórias do padrão REST definidas em [Fielding 2000] são:

- **Cliente-Servidor**: deve utilizar a arquitetura Cliente-Servidor para minimizar o acoplamento da interface com o usuário (cliente) do armazenamento e gerenciamento dos dados (servidor). Isto simplifica e aumenta a portabilidade de ambos os componentes e permite que eles evoluam separadamente.

[30] `https://www.macworld.co.uk/feature/iphone/iphone-vs-android-market-share-3691861/`

- ***Stateless***: as interações do cliente com o servidor não devem utilizar nenhum contexto armazenado no servidor, ou seja, toda requisição deve conter todos os dados necessários para seu entendimento.
- **Cache**: Para utilizar a rede de forma eficiente, cada resposta do servidor deve indicar se é possível realizar cache sobre seus dados. Se for possível, uma cache implementada no cliente pode reutilizar estes dados caso a mesma requisição seja feita no futuro.
- **Interface Uniforme**: recursos são unicamente identificados através de seu Uniform Resource Identifier (URI).
- **Sistema em camadas**: impede que o cliente tenha conhecimento de outros servidores além do que está efetivamente o servindo, o que reduz a complexidade e promove independência entre componentes do sistema. Pode ser usado para criar sub-sistemas que se comunicam sem o cliente precisar conhecer todas as partes. Também pode ser usado para encapsular sistemas legado.

7.1.3. JSON

JavaScript Object Notation (JSON) é um formato de texto para serialização de dados estruturados [Bray 2017].

Foi padronizado na RFC8259 e pode representar 4 tipos primitivos: *number*, *string*, *boolean* e *null*; e 2 tipos estruturados: *array* (também conhecido como o tipo *List* ou *Vector* em algumas linguagens de programação) e *object* (também conhecido como *Map* ou *Dictionary*).

```
{
    "city": "Florianópolis",
    "state": "Santa Catarina",
    "country": "Brasil",
    "population": 477798,
    "is_capital": true,
    "beaches": ["Joaquina", "Ingleses", "Campeche"]
}
```

Figura 1. Exemplo de documento JSON

Trabalhos como [Lin et al. 2012] mostram que JSON é um formato mais eficiente para transferência de dados em APIs para a Web quando comparado com XML, produzindo mensagens menores e desserialização mais rápida.

7.1.4. Programação Reativa para Interfaces

Programação Reativa é uma abordagem para o desenvolvimento de interfaces gráficas com o usuário (GUI) que proporciona abstrações alto-nível, declarativas, baseadas em composição [Czaplicki and Chong 2013]. É um paradigma utilizado pelos *frameworks* React e Flutter, e pela linguagem de programação Elm.

É caracterizada pela existência de funções puras que descrevem a interface de acordo com o estado atual da aplicação, também conhecidas como componentes. Estes componentes podem utilizar outros componentes mais simples, enfatizando o reúso.

Funções puras são funções livres de efeitos colaterais que para um dado conjunto de argumentos, retornam sempre o mesmo valor.

7.1.5. Gerenciamento de Estado

O gerenciamento do estado é importante em aplicações utilizando o paradigma de Programação Reativa, já que toda a interface é construída de acordo com o estado. Uma solução existente para isso é a arquitetura Business Logic Component (BLoC) [Boelens 2018].

Assim como a arquitetura *Model–View–Controller* (MVC) [Reenskaug and Coplien 2009], BLoC promove a separação de responsabilidades em 3 principais componentes:

1. **Provedor de Dados**: APIs REST, bancos de dados, ou qualquer fonte de dados da aplicação. Normalmente acessado de forma assíncrona.
2. **BLoC**: implementa as regras de negócio e armazena e atualiza o estado.
3. **Interface com o Usuário**: recebe o estado do BLoC, que usa para definir como construir a interface. Também emite os eventos realizados pelo usuário para o BLoC.

A comunicação entre o BLoC e a Interface com o Usuário é feita de forma assíncrona através de *Streams*, fluxos de dados que ocorrem com o decorrer do tempo. Um BLoC deve implementar no mínimo duas *Streams*, uma que envia o estado para a Interface com o Usuário e outra que recebe as ações executadas pelo usuário.

Por exemplo, em um aplicativo de mensagens, quando o usuário pressionar o botão de enviar mensagem, um novo evento do tipo ”envio de mensagem” é adicionado à *stream* de eventos. O BLoC receberá este evento, emitirá um novo estado de carregamento na *stream* correspondente e então iniciará o envio da mensagem ao servidor. Quando a transmissão ao servidor terminar, o BLoC emitirá mais um estado de sucesso ou falha, permitindo que a interface avise ao usuário do resultado da operação.

7.1.6. Flutter

Flutter[31] é um framework para desenvolvimento de aplicativos multiplataforma, utiliza a linguagem de programação Dart[32], ambos criados pela Google. É implementado utilizando a biblioteca gráfica Skia[33], utilizada no Google Chrome, Android, Mozilla Firefox, etc.

Flutter implementa o paradigma de programação reativa para interfaces, popularizado pela biblioteca React[34], criada pelo Facebook.

[31] `https://flutter.dev/`
[32] `https://dart.dev/`
[33] `https://skia.org/`
[34] `https://reactjs.org/`

7.1.7. Flask

Flask[35] é um framework para servidores web para Python. Implementa a especificação WSGI (Web Server Gateway Interface), padrão utilizado por frameworks web que facilita a portabilidade das aplicações em diferentes *web servers*. É um dos pacotes mais utilizados do ecosistema Python[36].

7.1.8. Firebase

Firebase[37] é uma plataforma para desenvolvimento de aplicativos móveis e *web apps* que implementa diversas funcionalidades comumente encontradas nestas aplicações. Entre seu rol de serviços estão: *Auth*, permitindo cadastro e autenticação facilitado, sem precisar implementar esta funcionalidade no servidor da aplicação, suporta *social login*; *Crashlytics*, funcionalidade que permite que um aplicativo que sofreu um *crash* – encerramento inesperado por conta de erro – envie as informações sobre o problema aos seus desenvolvedores; *Analytics*, que permite aos desenvolvedores extrairem informações de como seus usuários utilizam o aplicativo; *Cloud Messaging*, que permite o envio facilitado de notificações para os dispositivos.

7.2. Modelagem de dados

A ferramenta desenvolvida possui dois modelos de dados similares, um para o servidor e outro para a aplicação. A necessidade de dois modelos diferentes surgiu do fato que o aplicativo precisa armazenar o conteúdo dos questionários de forma cifrada e só precisa armazenar dados relativos ao usuário atual.

O modelo de dados do servidor é inspirado em um modelo EAV (Entidade-Atributo-Valor), que permite que cada Entidade (*Survey*) possua quantos Atributos (*Question*) desejar. Cada resposta (*Answer*) terá exatamente um Valor (*QuestionAnswer*) para cada Atributo.

O campo *extras* da tabela *Question* utiliza o tipo de dados JSON, que permite o armazenamento de dados semi-estruturados dentro da tabela. Este tipo de dados foi incluído na oitava revisão da especificação da linguagem de consulta SQL, ISO/IEC 9075:2016[38] e é suportado por SGBDs como Postgres[39], Oracle Database[40], SQLite[41], entre outros.

O campo *extras* armazena configurações da pergunta que dependem do valor do campo *type*, por exemplo, um campo numérico pode ter um valor mínimo e máximo, já um campo de seleção múltipla necessita armazenar as alternativas possíveis. Portanto, como o tipo de dados JSON permite armazenar dados semi-estruturados, foi escolhido para este campo por sua flexibilidade, permitindo um *schema* implícito, controlado pelo servidor, não pelo SGBD.

[35] https://palletsprojects.com/p/flask/
[36] https://pypistats.org/packages/flask
[37] https://firebase.google.com/
[38] https://standards.iso.org/ittf/PubliclyAvailableStandards/c067367_ISO_IEC_TR_19075-6_2017.zip
[39] https://www.postgresql.org/docs/10/datatype-json.html
[40] https://docs.oracle.com/database/121/ADXDB/json.htm\#ADXDB6246
[41] https://www.sqlite.org/json1.html

A tabela *Answer* no banco de dados do aplicativo não possui relacionamento com nenhuma outra, visto que todas as informações da resposta estão armazenados de forma cifrada em seu campo *content*.

7.3. Autenticação

A autenticação do usuário foi implementada utilizando Firebase Auth[42], visto que proporciona grande facilidade para implementação desta funcionalidade. Foi escolhido o Google como provedor de identidade para *social login*, entre suas vantagens está a simplificação do cadastro e do login, já que o usuário precisa apenas apertar o botão "Entrar com Google" e seus principais dados (nome e email) são enviados para a aplicação, não necessitando que o usuário aprenda mais um login e senha.

Para autenticar o usuário no servidor, o aplicativo gera um *token* para o usuário atual através do Firebase SDK e o envia através de HTTPS para o servidor. Ao receber o *token*, o servidor utiliza a função `verify_id_token`[43] do Firebase Admin para verificar se o token é válido e corresponde à um usuário cadastrado, posteriormente o servidor pode recuperar o usuário e seus questionários utilizando o *uid* retornado pela função supracitada. O *token* é obrigatório para realizar qualquer operação sobre a API, caso não seja enviado, o servidor retornará o código de erro HTTP 401 Unauthorized.

Uma vez que o usuário efetue o login, sua sessão é mantida até que este escolha realizar o logout.

7.4. Cadastro de questionários

A funcionalidade de cadastro de questionários permite que o usuário crie novos questionários e insira perguntas dos tipos "escolha única", "escolha múltipla", "texto livre"e "numérica". O questionário requer um título e no mínimo uma pergunta para ser válido. Só é possível cadastrar um novo questionário se o dispositivo estiver conectado à internet.

7.5. Preenchimento de questionários

Para implementar essa funcionalidade, foi escolhido um layout que apresenta uma pergunta por vez ao usuário, e só permite que ele avance caso a resposta seja válida, com o objetivo de impedir preenchimentos inválidos. O usuário pode navegar livremente pelas questões já respondidas. Esta funcionalidade foi projetada para que o aplicativo seja entregue ao participante para que preencha o questionário de forma autônoma, porém, o pesquisador pode também escolher ler as perguntas e respostas ao participante e então registrar suas escolhas.

Os campos de resposta foram implementados de acordo com as especificações do Material Design[44], as questões de escolha única utilizam uma lista de *Radio Buttons*[45], as questões de escolha múltipla utilizam uma lista de *Checkboxes*[46] e as questões numéricas e de texto livre utilizam um campo de texto[47]. Questões numéricas possuem validação no campo de texto para assegurar que o conteúdo se trata de um número.

[42] `https://firebase.google.com/docs/auth`
[43] `https://firebase.google.com/docs/auth/admin/verify-id-tokens`
[44] `https://material.io/`
[45] `https://material.io/components/selection-controls/\#radio-buttons`
[46] `https://material.io/components/selection-controls/\#checkboxes`
[47] `https://material.io/components/text-fields/`

Os textos das perguntas, e de todo o aplicativo, se adaptam às configurações de acessibilidade do dispositivo, aumentando a fonte automaticamente caso o usuário deseje. Todos os elementos interativos seguem as especificações de acessibilidade do Material Design[48], como tamanho mínimo de áreas clicáveis.

7.6. Armazenamento das respostas no dispositivo

Para armazenar as respostas dos questionários no dispositivo de forma segura, o aplicativo utiliza criptografia assimétrica. O algoritmo RSA/OAEP [Kaliski and Staddon 1998] com chave de 4096 bits foi escolhido por prover alta segurança e estar disponível tanto no servidor quanto no aplicativo.

Durante o login, o aplicativo obtém do servidor a chave pública que deve ser utilizada para cifrar os dados. Ao finalizar o preenchimento do questionário, todos os dados são serializados em JSON e então cifrados utilizando a chave previamente obtida, então, os dados cifrados são inseridos em um registro da tabela *Answer* e a versão em texto claro dos dados é excluída.

7.7. Sincronização com o Servidor

O passo de sincronização é necessário para enviar os dados cifrados que estão armazenados no dispositivo. O aplicativo realiza uma requisição HTTP POST para cada resposta armazenada; o servidor, ao receber a requisição, decifra os dados recebidos utilizando a chave privada do usuário, obtendo então um documento JSON com as respostas do questionário, que são inseridas no banco de dados. Ao receber resposta do servidor, o aplicativo exclui a resposta cifrada que acabou de enviar.

7.8. Exportação dos dados

Para a exportação dos dados foi utilizado o formato de aquivo CSV (Comma-separated values)[Shafranovich 2005], por ser um formato aberto, simples e aceito por ferramentas de planilha eletrônica populares, como Microsoft Excel, Google Sheets e LibreOffice Calc.

Como o aplicativo é compatível apenas com dispositivos móveis e a análise dos dados do questionário provavelmente será feita em um computador de mesa ou *notebook*, foi implementado o envio do arquivo contendo os dados através de email para o pesquisador responsável pelo questionário. O email é obtido por meio dos dados disponibilizados à ferramenta quando o usuário se cadastra utilizando uma conta Google.

8. Conclusão

A informatização de questionários traz diversos benefícios em comparação com questionários em papel, mas ainda pode gerar obstáculos aos seus usuários. Ao longo deste trabalho foram identificados diversas dificuldades, problemas e desafios enfrentados por pesquisadores e participantes ao realizar pesquisas com questionários informatizados através do uso de dispositivos móveis, que se referiam principalmente à conectividade, *layout*, familiaridade com tecnologia, erros do aplicativo, segurança dos dados, segurança dos dispositivos e bateria.

[48] `https://material.io/design/usability/accessibility.html`

A partir disso foi elaborado um conjunto de requisitos para uma ferramenta de questionários visando reduzir o impacto dessas dificuldades, problemas e desafios. Estes requisitos foram comparados com ferramentas existentes, não identificando-se nenhuma que os cumprisse integralmente.

Identificada esta lacuna, foi desenvolvida uma ferramenta de questionários que satisfizesse os requisitos elaborados. A ferramenta consiste em dois módulos, um servidor e um aplicativo, que foram disponibilizados sob a licença MIT[49] no GitHub[50].

8.1. Trabalhos futuros

Como recomendação de esforços futuros que complementem o presente trabalho, sugere-se a avaliação de usabilidade da funcionalidade de preenchimento de questionários, visto que esta foi desenvolvida a partir de recomendações da literatura, e não de entrevistas com potenciais usuários. Sabe-se que determinados coortes tem necessidades diferentes quanto a usabilidade de um sistema, como por exemplo, pessoas idosas, daltônicos, pessoas que utilizam leitores de tela. Um estudo avaliativo da usabilidade da ferramenta com estes diferentes grupos pode encontrar pontos de melhoria e dificuldades para estas populações.

Além disso, é possível realizar trabalhos técnicos de extensão da ferramenta, como:

- **Adição de novos tipos de pergunta.** Como perguntas de avaliação (nota de 1 até 5), escolha de imagens, escala Likert, etc.
- **Adição de ramificação de perguntas.** Permitir que o pesquisador registre caminhos lógicos que alteram quais perguntas devem ser respondidas de acordo com as respostas registradas.
- **Registro da localização em que o questionário foi respondido.** A pesquisa de [Mwita et al. 2019] utilizou uma ferramenta para preencher os questionários e outra para registrar a localização atual dos recenseadores, caso a ferramenta de questionários permitisse registrar a localização atual, a segunda ferramenta não seria necessária.
- **Versão web da ferramenta.** Desenvolvimento de um *web app* que permita preencher o questionário através de um navegador. Poucas modificações seriam necessárias no servidor, visto que já é genérico o suficiente para suportar também uma ferramenta web.

Referências

Antoun, C., Couper, M. P., and Conrad, F. G. (2017). Effects of Mobile versus PC Web on Survey Response Quality. *Public Opinion Quarterly*, 81(S1):280–306.

Backman, C., Vanderloo, S., Momtahan, K., Entremont, B., Freeman, L., Kachuik, L., Rossy, D., Mille, T., Mojaverian, N., Lemire Rodger, G., and Forster, A. (2015). Implementation of an electronic data collection tool to monitor nursing-sensitive indicators in a large academic health sciences centre. *Nursing leadership (Toronto, Ont.)*, 28:77–91.

[49] `https://choosealicense.com/licenses/mit/`

[50] `https://github.com/caiopo/sage`

Boelens, D. (2018). Reactive programming - streams - bloc. https://medium.com/flutter-community/reactive-programming-streams-bloc-6f0d2bd2d248. Visitado em Junho/2019.

Bourezgue, T. (2017). Using tablets for the 2018 Algerian Census: Census data management and quality assessment. *Statistical Journal of the IAOS*, 33(3):777–784.

Bray, T. (2017). The javascript object notation (json) data interchange format. STD 90, RFC Editor.

Couper, M. P. and Peterson, G. J. (2017). Why Do Web Surveys Take Longer on Smartphones? *Social Science Computer Review*, 35(3):357–377.

Czaplicki, E. and Chong, S. (2013). Asynchronous functional reactive programming for guis.

de Melo de Azevedo, L. J., Miazaki, M., and Porfirio, A. J. (2014). Questionário eletrônico em ambiente android para coleta de dados.

Dy, C. J., Schmicker, T., Tran, Q., Chadwick, B., and Daluiski, A. (2012). The use of a tablet computer to complete the dash questionnaire. *The Journal of Hand Surgery*, 37(12):2589 – 2594.

Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391.

Gault, R. H. (1907). A history of the questionnaire method of research in psychology. *The Pedagogical Seminary*, 14(3):366–383.

Gichohi, B. W. (2016). Web surveys for offline rural communities. *Statistical Journal of the IAOS*, 32(4):627–630.

Gillham, B. (2008). *Developing a Questionnaire*. Real World Research. Bloomsbury Publishing.

GSMA (2019). The state of mobile internet connectivity. https://www.gsma.com/mobilefordevelopment/wp-content/uploads/2019/07/GSMA-State-of-Mobile-Internet-Connectivity-Report-2019.pdf. Visitado em Outubro/2019.

Gummer, T., Quoß, F., and Roßmann, J. (2019). Does Increasing Mobile Device Coverage Reduce Heterogeneity in Completing Web Surveys on Smartphones? *Social Science Computer Review*, 37(3):371–384.

Guyon, A., Bock, A., Buback, L., and Knittel, B. (2016). Mobile-Based Nutrition and Child Health Monitoring to Inform Program Development: An Experience From Liberia. *Global Health: Science and Practice*, 4(4):661–670.

Hershman, S. G., Bot, B. M., Shcherbina, A., Doerr, M., Moayedi, Y., Pavlovic, A., Waggott, D., Cho, M. K., Rosenberger, M. E., Haskell, W. L., Myers, J., Champagne, M. A., Mignot, E., Salvi, D., Landray, M., Tarassenko, L., Harrington, R. A., Yeung,

A. C., McConnell, M. V., and Ashley, E. A. (2019). Physical activity, sleep and cardiovascular health data for 50,000 individuals from the MyHeart Counts Study. *Scientific Data*, 6(1):24.

Kaliski, B. and Staddon, J. (1998). Pkcs #1: Rsa cryptography specifications version 2.0. RFC 2437, RFC Editor.

King, C., Zamawe, C., Banda, M., Bar-Zeev, N., Beard, J., Bird, J., Costello, A., Kazembe, P., Osrin, D., Fottrell, E., and for the VacSurv Consortium (2016). The quality and diagnostic value of open narratives in verbal autopsy: a mixed-methods analysis of partnered interviews from Malawi. *BMC Medical Research Methodology*, 16(1):13.

King, C. M., Heilbrun, K., Kim, N. Y., McWilliams, K., Phillips, S., Barbera, J., and Fretz, R. (2017). Tablet computers and forensic and correctional psychological assessment: A randomized controlled study. *Law and Human Behavior*, 41(5):468–477.

Kuehne, A., Lynch, E., Marshall, E., Tiffany, A., Alley, I., Bawo, L., Massaquoi, M., Lodesani, C., Le Vaillant, P., Porten, K., and Gignoux, E. (2016). Mortality, Morbidity and Health-Seeking Behaviour during the Ebola Epidemic 2014–2015 in Monrovia Results from a Mobile Phone Survey. *PLOS Neglected Tropical Diseases*, 10(8):e0004899.

Lane, B., Car, N., Leonard, J., Lipkin, F., and Siggins, A. (2015). Mobile field data collection for post bushfire analysis and african farmers. 448:160–168.

Lewis-Beck, M., Bryman, A., and Futing Liao, T. (2004). *The SAGE Encyclopedia of Social Science Research Methods*. Sage Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States of America.

Lin, B., Chen, Y., Chen, X., and Yu, Y. (2012). Comparison between json and xml in applications based on ajax. In *2012 International Conference on Computer Science and Service System*, pages 1174–1177.

Marcano-Belisario, J. S., Gupta, A. K., O'Donoghue, J., Ramchandani, P., Morrison, C., and Car, J. (2017). Implementation of depression screening in antenatal clinics through tablet computers: results of a feasibility study. *BMC Medical Informatics and Decision Making*, 17(1):59.

Mehdi, M., Schwager, D., Pryss, R., Schlee, W., Reichert, M., and Hauck, F. J. (2019). Towards Automated Smart Mobile Crowdsensing for Tinnitus Research. In *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*, pages 75–80, Cordoba, Spain. IEEE.

Mercader, H. F. G., Kabakyenga, J., Katuruba, D. T., Hobbs, A. J., and Brenner, J. L. (2017). Female respondent acceptance of computer-assisted personal interviewing (CAPI) for maternal, newborn and child health coverage surveys in rural Uganda. *International Journal of Medical Informatics*, 98:41–46.

Meyers, D. J., Filkins, M., Harsha Bangura, A., Sharma, R., Baruwal, A., Pande, S., Halliday, S., Schwarz, D., Schwarz, R. K., and Maru, D. S. R. (2017). Management challenges in mHealth: failures of a mobile community health worker surveillance programme in rural Nepal. *BMJ Innovations*, 3(1):19–25.

Mwita, E. J., Pearson, A. L., and Zwickle, A. (2019). Effectiveness of tablet computers in spatial data collection and surveys in rural Tanzania. *The Electronic Journal of Information Systems in Developing Countries*.

Olmsted-Hawala, E., Nichols, E., Falcone, B., Figueroa, I. J., Antoun, C., and Wang, L. (2018). Optimal Data Entry Designs in Mobile Web Surveys for Older Adults. In Zhou, J. and Salvendy, G., editors, *Human Aspects of IT for the Aged Population. Acceptance, Communication and Participation*, volume 10926, pages 335–354. Springer International Publishing, Cham.

O'Reilly-Shah, V. N. (2017). Factors influencing healthcare provider respondent fatigue answering a globally administered in-app survey. *PeerJ*, 5:e3785.

Pereira, I. M., Bonfim, D., Peres, H. H. C., Góes, R. F., and Gaidzinski, R. R. (2017). Tecnologia móvel para coleta de dados de pesquisas em saúde.

Piau, A., Crissey, R., Brechemier, D., Balardy, L., and Nourhashemi, F. (2019). A smartphone Chatbot application to optimize monitoring of older patients with cancer. *International Journal of Medical Informatics*, 128:18–23.

Rajan, J. V., Moura, J., Gourley, G., Kiso, K., Sizilio, A., Cortez, A. M., Riley, L. W., Veras, M. A., and Sarkar, U. (2016). Understanding the barriers to successful adoption and use of a mobile health information system in a community health center in São Paulo, Brazil: a cohort study. *BMC Medical Informatics and Decision Making*, 16(1):146.

Ramsey, A. T., Wetherell, J. L., Depp, C., Dixon, D., and Lenze, E. (2016). Feasibility and Acceptability of Smartphone Assessment in Older Adults with Cognitive and Emotional Difficulties. *Journal of Technology in Human Services*, 34(2):209–223.

Reenskaug, T. and Coplien, J. O. (2009). The dci architecture: A new vision of object-oriented programming. `https://www.artima.com/articles/dci_vision.html`. Visitado em Junho/2019.

Rescorla, E. (2000). Http over tls. RFC 2818, RFC Editor. `http://www.rfc-editor.org/rfc/rfc2818.txt`.

Revilla, M., Toninelli, D., Ochoa, C., and Loewe, G. (2016). Do online access panels need to adapt surveys for mobile devices? *Internet Research*, 26(5):1209–1227.

Schick-Makaroff, K. and Molzahn, A. (2015). Strategies to use tablet computers for collection of electronic patient-reported outcomes. *Health and Quality of Life Outcomes*, 13(1):2.

Schobel, J., Pryss, R., and Reichert, M. (2015). Using Smart Mobile Devices for Collecting Structured Data in Clinical Trials: Results from a Large-Scale Case Study. In *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*, pages 13–18, Sao Carlos, Brazil. IEEE.

Shafranovich, Y. (2005). Common format and mime type for comma-separated values (csv) files. RFC 4180, RFC Editor. `http://www.rfc-editor.org/rfc/rfc4180.txt`.

Silveira, C. E., Andrade, A. V., de Assis, L. P., Medaglia, J., and Ribeiro, F. M. (2014). Sistema móvel para realização de pesquisas de demanda turística - a interdisciplinaridade entre turismo e sistemas de informação gerando resultados aplicados.

Thrul, J., Bühler, A., and Ferguson, S. G. (2015). An Internet-Based Ecological Momentary Assessment Study Relying on Participants' Own Mobile Phones: Insights from a Study with Young Adult Smokers. *European Addiction Research*, 21(1):1–5.

Toepoel, V. and Funke, F. (2018). Sliders, visual analogue scales, or buttons: Influence of formats and scales in mobile and desktop surveys. *Mathematical Population Studies*, 25(2):112–122.

Valdes Angues, R., Suits, A., Palmer, V. S., Okot, C., Okot, R. A., Atonywalo, C., Gazda, S. K., Kitara, D. L., Lantum, M., and Spencer, P. S. (2018). A real-time medical cartography of epidemic disease (Nodding syndrome) using village-based lay mHealth reporters. *PLOS Neglected Tropical Diseases*, 12(6):e0006588.

Yu, D. X., Parmanto, B., Dicianno, B. E., Watzlaf, V. J., and Seelman, K. D. (2017). Accessibility needs and challenges of a mHealth system for patients with dexterity impairments. *Disability and Rehabilitation: Assistive Technology*, 12(1):56–64.

Zhang, J., Joshi, R., Sun, J., Rosenthal, S. R., Tong, M., Li, C., Rampatige, R., Mooney, M., Lopez, A., and Yan, L. L. (2016). A feasibility study on using smartphones to conduct short-version verbal autopsies in rural China. *Population Health Metrics*, 14(1):31.

Zijlstra, T., Wijgergangs, K., and Hoogendoorn-Lanser, S. (2018). Traditional and mobile devices in computer assisted web-interviews. *Transportation Research Procedia*, 32:184–194.

APÊNDICE B – Código-fonte

Arquivo: app/lib/main.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_hooks/flutter_hooks.dart';
import 'package:gaia/screens/login/login_screen.dart';

void main() {
  runApp(App());
}

const _primaryColor = Color(0xFFDA5C47);
const _accentColor = Color(0xFFB0D57E);

class App extends HookWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Gaia',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.lightGreen,
        primaryColor: _primaryColor,
        accentColor: _accentColor,
      ),
      home: LoginScreen(),
    );
  }
}
```

Arquivo: app/lib/api/utils.dart

```
import 'dart:convert';

import 'package:firebase_auth/firebase_auth.dart';
```

```
import 'package:http/http.dart' as http;

Future<String> token() async {
  FirebaseUser user = await FirebaseAuth.instance.currentUser();
  final tokenResult = await user.getIdToken();

  return tokenResult.token;
}

Uri toUri(dynamic url) {
  if (url is String) {
    return Uri.parse(url);
  }

  if (url is Uri) {
    return url;
  }

  throw ArgumentError(
    'url must be String or Uri, found ${url.runtimeType.toString()}',
  );
}

Future<Uri> addToken(Uri uri) async {
  final query = Map.of(uri.queryParameters);

  query['token'] = await token();

  return uri.replace(queryParameters: query);
}

Future<Map<String, dynamic>> getWithAuth(dynamic url) async {
  final uri = await addToken(toUri(url));

  final response = await http.get(uri);

  if (!_ok(response)) {
    throw RequestFailed(response.statusCode);
  }
```

```
  return json.decode(response.body) as Map<String, dynamic>;
}

Future<Map<String, dynamic>> postWithAuth(
    dynamic url, Map<String, dynamic> body) async {
  final uri = await addToken(toUri(url));

  final jsonBody = json.encode(body);

  final response = await http.post(
    uri,
    headers: {
      'Content-Type': 'application/json',
    },
    body: jsonBody,
  );

  if (!_ok(response)) {
    throw RequestFailed(response.statusCode);
  }

  return json.decode(response.body) as Map<String, dynamic>;
}

bool _ok(http.Response resp) {
  return 200 <= resp.statusCode && resp.statusCode < 300;
}

class RequestFailed extends Error {
  final int statusCode;

  RequestFailed(this.statusCode);

  @override
  String toString() {
    return "Request failed <${statusCode}>";
  }
}
```

Arquivo: app/lib/api/endpoints.dart

```
final ENDPOINT = Uri.http('10.0.2.2:5000', '');

class Endpoints {
  static Uri surveyRoot() {
    return ENDPOINT.replace(path: '/surveys/');
  }

  static Uri surveyList() {
    return ENDPOINT.replace(path: '/surveys/list');
  }

  static Uri surveyDetail(int surveyId) {
    return ENDPOINT.replace(path: '/surveys/${surveyId}');
  }
}
```

Arquivo: app/lib/api/surveys.dart

```
import 'package:gaia/api/endpoints.dart';
import 'package:gaia/api/utils.dart';
import 'package:gaia/models/models.dart';

Future<List<Survey>> fetchSurveyList() async {
  final body = await getWithAuth(Endpoints.surveyList());

  List<dynamic> content = body['content'] as List<dynamic>;

  return content
      .map((s) => Survey.fromJson(s as Map<String, dynamic>))
      .toList();
}

Future<Survey> createSurvey(String title) async {
  final body = await postWithAuth(Endpoints.surveyRoot(), {'title': title});
```

```
  return Survey.fromJson(body);
}
```

Arquivo: app/lib/screens/routes.dart

```
import 'package:flutter/material.dart';
import 'package:gaia/models/models.dart';
import 'package:gaia/screens/answer/answer_screen.dart';
import 'package:gaia/screens/login/login_screen.dart';
import 'package:gaia/screens/question_create/question_create_screen.dart';
import 'package:gaia/screens/survey_create/survey_create_screen.dart';
import 'package:gaia/screens/survey_detail/survey_detail_screen.dart';
import 'package:gaia/screens/survey_list/survey_list_screen.dart';
import 'package:gaia/screens/sync/sync_screen.dart';

class Routes {
  static MaterialPageRoute login() {
    return MaterialPageRoute(
      builder: (BuildContext context) => LoginScreen(),
    );
  }

  static MaterialPageRoute sync() {
    return MaterialPageRoute(
      builder: (BuildContext context) => SyncScreen(),
    );
  }

  static MaterialPageRoute surveyList() {
    return MaterialPageRoute(
      builder: (BuildContext context) => SurveyListScreen(),
    );
  }

  static MaterialPageRoute surveyDetail(Survey survey) {
    return MaterialPageRoute(
      builder: (BuildContext context) => SurveyDetailScreen(survey: survey),
```

```
    );
  }

  static MaterialPageRoute surveyCreate() {
    return MaterialPageRoute(
      builder: (BuildContext context) => SurveyCreateScreen(),
    );
  }

  static MaterialPageRoute questionCreate() {
    return MaterialPageRoute(
      builder: (BuildContext context) => QuestionCreateScreen(),
    );
  }

  static MaterialPageRoute answer(Survey survey) {
    return MaterialPageRoute(
      builder: (BuildContext context) => AnswerScreen(survey: survey),
    );
  }
}
```

Arquivo: app/lib/screens/login/login_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_auth_buttons/flutter_auth_buttons.dart';
import 'package:flutter_hooks/flutter_hooks.dart';
import 'package:gaia/components/presentational/circular_background.dart';
import 'package:gaia/components/presentational/gaia_wordmark.dart';
import 'package:gaia/screens/routes.dart';

class LoginScreen extends HookWidget {
  @override
  Widget build(BuildContext context) {
    final loading = useState(false);
    final theme = Theme.of(context);

    return Scaffold(
```

```
      body: Container(
        child: Center(
          child: SizedBox.expand(
            child: Container(
              color: theme.primaryColor,
              child: loading.value
                  ? CircularProgressIndicator()
                  : Column(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: <Widget>[
                        CircularBackground(
                          child: Padding(
                            padding: const EdgeInsets.only(bottom: 8),
                            child: Image.asset(
                              'assets/images/ginger-cat/track-statistics.png',
                              scale: 0.1,
                            ),
                          ),
                        ),
                        GaiaWordmark(size: 72),
                        SizedBox(height: 8),
                        GoogleSignInButton(
                          text: 'Entrar com Google',
                          onPressed: () async {
                            await Navigator.pushReplacement(
                                context, Routes.surveyList());
                          },
                        )
                      ],
                    ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Arquivo: app/lib/screens/answer/answer_screen.dart

```
import 'dart:math';

import 'package:flutter/material.dart';
import 'package:gaia/components/presentational/ghost.dart';
import 'package:gaia/models/models.dart';
import 'package:gaia/screens/answer/answer_type_layouts.dart';

class AnswerScreen extends StatefulWidget {
  final Survey survey;

  const AnswerScreen({
    Key key,
    this.survey,
  }) : super(key: key);

  @override
  _AnswerScreenState createState() => _AnswerScreenState();
}

class _AnswerScreenState extends State<AnswerScreen> {
  final answers = <String, dynamic>{};

  int currentIndex = 0;

  SurveyQuestion get currentQuestion => widget.survey.questions[currentIndex];

  int get surveyLength => widget.survey.questions.length;

  dynamic get currentAnswer => answers[currentQuestion.uuid];

  void onAnswerChanged(dynamic answer) {
    print(answer);
    setState(() {
      answers[currentQuestion.uuid] = answer;
    });
  }
```

```
void onPreviousPressed() {
  setState(() {
    currentIndex = max(currentIndex - 1, 0);
  });
}

void onNextPressed() {
  setState(() {
    currentIndex = min(currentIndex + 1, surveyLength - 1);
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.survey.title),
    ),
    body: Column(
      children: <Widget>[
        Expanded(
          child: SingleChildScrollView(
            child: Center(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.center,
                children: <Widget>[
                  QuestionHeader(question: currentQuestion),
                  SizedBox(height: 16),
                  buildTypeSpecific(currentQuestion),
                ],
              ),
            ),
          ),
        ),
        buildBottomBar(),
      ],
    ),
  );
}
```

```
Widget buildTypeSpecific(SurveyQuestion question) {
  final answer = answers[currentQuestion.uuid];

  switch (question.type) {
    case QuestionType.single:
      return AnswerSingleChoiceQuestion(
        question: question,
        answer: answer as String,
        onAnswerChanged: onAnswerChanged,
      );

    case QuestionType.multiple:
      return AnswerMultipleChoiceQuestion(
        question: question,
        answer: answer as List<String> ?? [],
        onAnswerChanged: onAnswerChanged,
      );

    case QuestionType.number:
      return AnswerNumberQuestion(
        question: question,
        answer: answer as double,
        onAnswerChanged: onAnswerChanged,
      );

    case QuestionType.text:
      return AnswerTextQuestion(
        question: question,
        answer: answer as String,
        onAnswerChanged: onAnswerChanged,
      );
  }

  throw ArgumentError();
}

Widget buildBottomBar() {
  final theme = Theme.of(context);
```

```
    return Material(
      elevation: 8,
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        children: <Widget>[
          SizedBox(width: 4),
          Expanded(
            child: Ghost(
              show: currentIndex != 0,
              child: FlatButton(
                child: Text('VOLTAR'),
                textColor: theme.primaryColor,
                onPressed: onPreviousPressed,
              ),
            ),
          ),
          if (currentIndex < (surveyLength - 1))
            Expanded(
              child: FlatButton(
                child: Text('AVANÇAR'),
                textColor: theme.primaryColor,
                onPressed: onNextPressed,
              ),
            )
          else
            Expanded(
              child: FlatButton(
                child: Text('FINALIZAR'),
                textColor: theme.primaryColor,
                onPressed: () {},
              ),
            ),
          SizedBox(width: 4),
        ],
      ),
    );
  }
}
```

```
class QuestionHeader extends StatelessWidget {
  final SurveyQuestion question;

  const QuestionHeader({Key key, this.question}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.center,
      children: <Widget>[
        SizedBox(height: 24),
        Text(
          question.title,
          style: TextStyle(fontSize: 24),
          textAlign: TextAlign.center,
        ),
        if (question.description != null &&
            question.description.isNotEmpty) ...[
          SizedBox(height: 8),
          Text(
            question.description,
            style: TextStyle(fontSize: 16, color: Color(0xFF606060)),
            textAlign: TextAlign.center,
          ),
        ],
      ],
    );
  }
}
```

Arquivo: app/lib/screens/answer/answer_type_layouts.dart

```
import 'package:flutter/material.dart';
import 'package:gaia/models/models.dart';

typedef OnAnswerChanged = void Function(dynamic value);

class AnswerSingleChoiceQuestion extends StatelessWidget {
```

```
  final SurveyQuestion question;
  final String answer;
  final OnAnswerChanged onAnswerChanged;

  const AnswerSingleChoiceQuestion({
    Key key,
    this.question,
    this.answer,
    this.onAnswerChanged,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final options = question.extras['options'] as List<String>;

    return Column(
      children: <Widget>[
        for (final option in options) buildOption(context, option),
      ],
    );
  }

  Widget buildOption(BuildContext context, String option) {
    return RadioListTile(
      activeColor: Theme.of(context).primaryColor,
      controlAffinity: ListTileControlAffinity.leading,
      value: option,
      groupValue: answer,
      title: Text(option),
      onChanged: (v) {
        onAnswerChanged(v);
      },
    );
  }
}

class AnswerMultipleChoiceQuestion extends StatelessWidget {
  final SurveyQuestion question;
  final List<String> answer;
```

```
final OnAnswerChanged onAnswerChanged;

const AnswerMultipleChoiceQuestion({
  Key key,
  this.question,
  this.answer,
  this.onAnswerChanged,
}) : super(key: key);

@override
Widget build(BuildContext context) {
  final options = question.extras['options'] as List<String>;

  return Column(
    children: <Widget>[
      for (final option in options) buildOption(context, option),
    ],
  );
}

Widget buildOption(BuildContext context, String option) {
  return CheckboxListTile(
    activeColor: Theme.of(context).primaryColor,
    controlAffinity: ListTileControlAffinity.leading,
    value: answer.contains(option),
    title: Text(option),
    onChanged: (v) {
      final _answer = List.of(answer);

      if (v) {
        _answer.add(option);
      } else {
        _answer.remove(option);
      }

      onAnswerChanged(_answer);
    },
  );
}
```

```
}

class AnswerNumberQuestion extends StatelessWidget {
  final SurveyQuestion question;
  final double answer;
  final OnAnswerChanged onAnswerChanged;

  const AnswerNumberQuestion({
    Key key,
    this.question,
    this.answer,
    this.onAnswerChanged,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 32),
      child: TextField(
        decoration: InputDecoration(
          border: OutlineInputBorder(),
          hintText: 'Resposta',
        ),
        autofocus: true,
        style: TextStyle(fontSize: 20),
        minLines: 1,
        keyboardType: TextInputType.numberWithOptions(decimal: true),
        onChanged: (value) {
          onAnswerChanged(double.tryParse(value));
        },
      ),
    );
  }
}

class AnswerTextQuestion extends StatelessWidget {
  final SurveyQuestion question;
  final String answer;
  final OnAnswerChanged onAnswerChanged;
```

```
  const AnswerTextQuestion({
    Key key,
    this.question,
    this.answer,
    this.onAnswerChanged,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 32),
      child: TextField(
        decoration: InputDecoration(
          border: OutlineInputBorder(),
          hintText: 'Resposta',
        ),
        autofocus: true,
        style: TextStyle(fontSize: 20),
        onChanged: (value) {
          onAnswerChanged(value);
        },
      ),
    );
  }
}
```

Arquivo: app/lib/screens/survey_list/survey_list_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_hooks/flutter_hooks.dart';
import 'package:gaia/components/presentational/empty_list.dart';
import 'package:gaia/components/presentational/gaia_wordmark.dart';
import 'package:gaia/components/presentational/sync_button.dart';
import 'package:gaia/components/presentational/user_profile_button.dart';
import 'package:gaia/models/models.dart';
import 'package:gaia/screens/routes.dart';
import 'package:gaia/screens/survey_list/survey_list.dart';
```

```
class SurveyListScreen extends HookWidget {
  @override
  Widget build(BuildContext context) {
    final surveyList = useState(<Survey>[
      Survey(
        title: 'Pesquisa de Opinião',
        questions: [
          SurveyQuestion(
            uuid: 'aaaaaaaaaaaaaaa',
            title: 'Qual sua cor preferida?',
            description:
                'Lorem ipsum dolor sit amet, consectetur adipiscing elit. '
                'Fusce venenatis ut mauris a dignissim.',
            type: QuestionType.single,
            required: true,
            extras: {
              'options': [
                'Amarelo',
                'Azul',
                'Preto',
                'Verde',
                'Vermelho',
              ]
            },
          ),
          SurveyQuestion(
            uuid: 'aaaaaaaaaaaaaaab',
            title: 'Quais comidas você gosta?',
            description: '',
            type: QuestionType.multiple,
            required: true,
            extras: {
              'options': [
                'Pizza',
                'Sushi',
                'Macarrão',
                'Salada',
              ]
```

```
          },
        ),
        SurveyQuestion(
          uuid: 'aaaaaaaaaaaaaaabc',
          title: 'Qual sua idade?',
          type: QuestionType.number,
          required: true,
          extras: {},
        ),
        SurveyQuestion(
          uuid: 'aaaaaaaaaaaaaaabcd',
          title: 'Qual seu nome?',
          type: QuestionType.text,
          required: true,
          extras: {},
        ),
      ],
    ),
    Survey(
      title: 'Donec velit turpis',
      questions: List.generate(28, (i) => SurveyQuestion()),
    ),
    Survey(
      title: 'Consectetur adipiscing elit',
      questions: List.generate(54, (i) => SurveyQuestion()),
    ),
    Survey(
      title: 'Aliquam vel euismod purus',
      questions: List.generate(32, (i) => SurveyQuestion()),
    ),
  ]);

  return Scaffold(
    appBar: AppBar(
      title: GaiaWordmark(size: 28),
      actions: <Widget>[
        SyncButton(),
        UserProfileButton(),
      ],
```

```
      ),
      floatingActionButton: FloatingActionButton(
        tooltip: 'Create Survey',
        child: Icon(Icons.add),
        onPressed: () {
          Navigator.push(context, Routes.surveyCreate());
        },
      ),
      body: surveyList.value.isEmpty
          ? EmptyList(
              text: 'You have no surveys',
              paddingBottom: 80,
            )
          : SurveyList(
              data: surveyList.value,
              onSurveyPressed: (Survey survey) {
                Navigator.push(context, Routes.answer(survey));
              },
            ),
    );
  }
}
```

Arquivo: app/lib/screens/survey_list/survey_list.dart

```
import 'package:flutter/material.dart';
import 'package:gaia/components/presentational/identicon.dart';
import 'package:gaia/models/models.dart';

class SurveyList extends StatelessWidget {
  final List<Survey> data;
  final ValueChanged<Survey> onSurveyPressed;

  const SurveyList({
    Key key,
    @required this.data,
    @required this.onSurveyPressed,
  }) : super(key: key);
```

```
  Widget buildItem(Survey survey) {
    return ListTile(
      key: Key(survey.uuid),
      leading: Identicon(
        survey.title,
        size: 50,
      ),
      title: Text(survey.title),
      subtitle: Text("${survey.questions.length.toString()} perguntas"),
      onTap: () => onSurveyPressed(survey),
    );
  }

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      padding: EdgeInsets.only(bottom: 80),
      itemBuilder: (context, i) {
        return buildItem(data[i]);
      },
      itemCount: data?.length ?? 0,
    );
  }
}
```

Arquivo: app/lib/screens/survey_create/discard_dialog.dart

```
import 'package:flutter/material.dart';

class DiscardDialog extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: Text('Discard Changes?'),
      content: SingleChildScrollView(
        child: ListBody(
          children: <Widget>[
```

```
            Text('If you go back now, your draft will be discarded.'),
          ],
        ),
      ),
      actions: <Widget>[
        FlatButton(
          child: Text(
            'DISCARD',
            style: TextStyle(
              color: Theme.of(context).errorColor,
            ),
          ),
          onPressed: () {
            Navigator.pop(context, true);
          },
        ),
        FlatButton(
          child: Text(
            'STAY',
          ),
          onPressed: () {
            Navigator.pop(context, false);
          },
        ),
      ],
    );
  }
}
```

Arquivo: app/lib/screens/survey_create/question_list_item.dart

```
import 'package:flutter/material.dart';
import 'package:gaia/models/models.dart';

enum QuestionListAction {
  edit,
  move,
  delete,
```

```
}

class QuestionListItem extends StatelessWidget {
  final SurveyQuestion question;

  const QuestionListItem({
    Key key,
    this.question,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ListTile(
      key: Key(question.uuid),
      title: Text('${question.title}'),
      subtitle: Text('Lorem ipsum dolor sit amet'),
      trailing: _buildPopupMenu(),
      onTap: () {},
    );
  }

  Widget _buildPopupMenu() {
    return PopupMenuButton<QuestionListAction>(
      itemBuilder: (context) {
        return [
          PopupMenuItem(
            child: Text('Edit'),
            value: QuestionListAction.edit,
          ),
          PopupMenuItem(
            child: Text('Move'),
            value: QuestionListAction.move,
          ),
          PopupMenuItem(
            child: Text('Delete'),
            value: QuestionListAction.delete,
          ),
        ];
      },
```

```
    );
  }
}
```

Arquivo: app/lib/screens/survey_create/identicon_text_field.dart

```
import 'package:flutter/material.dart';
import 'package:gaia/components/presentational/identicon.dart';

class IdenticonTextField extends StatelessWidget {
  final String title;
  final ValueChanged<String> onChanged;

  IdenticonTextField({
    Key key,
    this.title,
    this.onChanged,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(16),
      child: Row(
        mainAxisSize: MainAxisSize.min,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: <Widget>[
          Identicon(
            title,
            size: 64,
          ),
          SizedBox(width: 16),
          Expanded(
            child: TextField(
              decoration: InputDecoration(
                border: OutlineInputBorder(),
                hintText: "Pesquisa de Opinião",
                labelText: "Título",
```

```
              ),
              style: Theme.of(context).textTheme.subhead.copyWith(fontSize: 18),
              onChanged: onChanged,
            ),
          ),
        ],
      ),
    );
  }
}
```

Arquivo: app/lib/screens/survey_create/survey_create_screen.dart

```
import 'package:flutter/material.dart';
import 'package:gaia/models/models.dart';
import 'package:gaia/screens/routes.dart';
import 'package:gaia/screens/survey_create/discard_dialog.dart';
import 'package:gaia/screens/survey_create/identicon_text_field.dart';
import 'package:gaia/screens/survey_create/question_list_item.dart';

class SurveyCreateScreen extends StatefulWidget {
  @override
  _SurveyCreateScreenState createState() => _SurveyCreateScreenState();
}

class _SurveyCreateScreenState extends State<SurveyCreateScreen> {
  Survey survey = Survey(
    title: '',
    questions: [
      SurveyQuestion(
        title: 'Aaaaaaaaaaaaaaaaaaaa',
      ),
      SurveyQuestion(
        title: 'Aaaaaaaaaaaaaaaaaaaa',
      ),
      SurveyQuestion(
        title: 'Aaaaaaaaaaaaaaaaaaaa',
      ),
```

```
    SurveyQuestion(
      title: 'Aaaaaaaaaaaaaaaaaaaa',
    ),
    SurveyQuestion(
      title: 'Aaaaaaaaaaaaaaaaaaaa',
    ),
    SurveyQuestion(
      title: 'Aaaaaaaaaaaaaaaaaaaa',
    ),
    SurveyQuestion(
      title: 'Aaaaaaaaaaaaaaaaaaaa',
    ),
    SurveyQuestion(
      title: 'Aaaaaaaaaaaaaaaaaaaa',
    ),
    SurveyQuestion(
      title: 'Aaaaaaaaaaaaaaaaaaaa',
    ),
    SurveyQuestion(
      title: 'Aaaaaaaaaaaaaaaaaaaa',
    ),
    SurveyQuestion(
      title: 'Aaaaaaaaaaaaaaaaaaaa',
    ),
    SurveyQuestion(
      title: 'Aaaaaaaaaaaaaaaaaaaa',
    ),
  ],
);

@override
Widget build(BuildContext context) {
  return WillPopScope(
    child: Scaffold(
      appBar: AppBar(
        title: Text('Novo questionário'),
        actions: <Widget>[
          IconButton(
            icon: Icon(
```

```
              Icons.check,
              color: Colors.white,
            ),
            onPressed: () {
              Navigator.pop(context);
            },
          )
        ],
      ),
      floatingActionButton: FloatingActionButton(
        tooltip: 'Adicionar pergunta',
        child: Icon(Icons.add),
        onPressed: () {
          Navigator.push(context, Routes.questionCreate());
        },
      ),
      body: buildBody(),
    ),
    onWillPop: () async {
      final quit = await showDialog<bool>(
        context: context,
        builder: (context) => DiscardDialog(),
      );

      return quit ?? false;
    },
  );
}

Widget buildBody() {
  return CustomScrollView(
    slivers: <Widget>[
      SliverList(
        delegate: SliverChildListDelegate([
          Padding(
            padding: const EdgeInsets.only(top: 16, left: 16, right: 16),
            child: Text(
              'Informações gerais',
              style: TextStyle(fontWeight: FontWeight.w500, fontSize: 20),
```

```
        ),
      ),
      IdenticonTextField(
        title: survey.title,
        onChanged: (value) {
          setState(() => survey.title = value);
        },
      ),
      Padding(
        padding: const EdgeInsets.only(
          top: 8,
          bottom: 8,
          left: 16,
          right: 16,
        ),
        child: Text(
          'Perguntas',
          style: TextStyle(fontWeight: FontWeight.w500, fontSize: 20),
        ),
      ),
    ]),
  ),
  SliverList(
    delegate: SliverChildBuilderDelegate(
      (context, index) {
        final question = survey.questions[index];

        return QuestionListItem(
          question: question,
        );
      },
      childCount: survey.questions.length,
    ),
  ),
  SliverList(
    delegate: SliverChildListDelegate([
      SizedBox(height: 72),
    ]),
  ),
```

```
      ],
    );
  }
}
```

Arquivo: app/lib/screens/sync/sync_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_hooks/flutter_hooks.dart';

class SyncScreen extends HookWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Sincronização'),
      ),
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Image.asset('assets/images/ginger-cat/uploading.png'),
          SizedBox(height: 16),
          CircularProgressIndicator(),
          SizedBox(height: 16),
          Text('Sincronizando...', style: TextStyle(fontSize: 24)),
          SizedBox(height: 32),
        ],
      ),
    );
  }
}
```

Arquivo: app/lib/screens/survey_detail/survey_detail_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_hooks/flutter_hooks.dart';
import 'package:gaia/models/models.dart';
```

```
import 'package:gaia/utils/hooks/misc.dart';

class SurveyDetailScreen extends HookWidget {
  final Survey survey;

  SurveyDetailScreen({@required this.survey});

  @override
  Widget build(BuildContext context) {
    final theme = useTheme();

    return Scaffold(
      appBar: AppBar(
        title: Text('Survey'),
      ),
      body: SizedBox.expand(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: <Widget>[
            Align(
              alignment: Alignment.centerLeft,
              child: Padding(
                padding: const EdgeInsets.all(16),
                child: Text(
                  survey.title,
                  style: theme.textTheme.display1.copyWith(color: Colors.black87),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Arquivo: app/lib/screens/question_create/question_type_layouts.dart

```
import 'package:flutter/material.dart';

class CreateSingleChoiceQuestion extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}

class CreateMultipleChoiceQuestion extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}

class CreateNumberQuestion extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}

class CreateTextQuestion extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}

class CreateScaleQuestion extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

Arquivo: app/lib/screens/question_create/question_create_screen.dart

```
import 'package:flutter/material.dart';
import 'package:gaia/components/presentational/collapsing_radio_group.dart';
import 'package:gaia/models/models.dart';
import 'package:gaia/screens/question_create/question_type_layouts.dart';
import 'package:gaia/utils/texts.dart';

class QuestionCreateScreen extends StatefulWidget {
  @override
  _QuestionCreateScreenState createState() => _QuestionCreateScreenState();
}

class _QuestionCreateScreenState extends State<QuestionCreateScreen> {
  SurveyQuestion question = SurveyQuestion(
    required: true,
  );

  @override
  Widget build(BuildContext context) {
    final theme = Theme.of(context);

    return Scaffold(
      appBar: AppBar(
        title: Text('Nova pergunta'),
      ),
      body: SingleChildScrollView(
        child: Column(
          children: <Widget>[
            SizedBox(height: 16),
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 16),
              child: TextField(
                decoration: InputDecoration(
                  border: OutlineInputBorder(),
                  labelText: 'Pergunta',
                  hintText: 'Qual sua cor favorita?',
                ),
                style: theme.textTheme.subhead.copyWith(fontSize: 20),
              ),
```

```
          ),
          SizedBox(height: 16),
          _buildSection('Tipo de pergunta'),
          CollapsingRadioGroup<QuestionType>(
            onSelected: (type) => setState(() => question.type = type),
            items: [
              for (final type in QuestionType.values)
                RadioItem(
                  key: type,
                  label: questionTypeText(type),
                ),
            ],
          ),
          _buildSection('Outras opções'),
          CheckboxListTile(
            title: Text('Obrigatória'),
            activeColor: Theme.of(context).primaryColor,
            controlAffinity: ListTileControlAffinity.leading,
            value: question.required,
            onChanged: (required) => setState(() {
              question.required = required;
            }),
          ),
          ..._buildTypeSpecific(),
        ],
      ),
    ),
  );
}

Widget _buildSection(String title) {
  return Align(
    alignment: Alignment.centerLeft,
    child: Padding(
      padding: const EdgeInsets.symmetric(vertical: 8, horizontal: 16),
      child: Text(title, style: TextStyle(fontSize: 22)),
    ),
  );
}
```

```
  List<Widget> _buildTypeSpecific() {
    switch (question.type) {
      case QuestionType.multiple:
        return [
          _buildSection('Respostas'),
          CreateMultipleChoiceQuestion(),
        ];

      case QuestionType.single:
        return [
          _buildSection('Respostas'),
          CreateSingleChoiceQuestion(),
        ];

      case QuestionType.number:
        return [
          CreateNumberQuestion(),
        ];

      case QuestionType.text:
        return [
          CreateTextQuestion(),
        ];
    }

    return [];
  }
}
```

Arquivo: app/lib/components/behavioral/snapshot_builder.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_hooks/flutter_hooks.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';

Widget defaultLoading(BuildContext context) {
  return Center(child: CircularProgressIndicator());
```

```
}

Widget defaultError(BuildContext context, Object error) {
  final theme = Theme.of(context);

  return Center(
    child: Column(
      mainAxisSize: MainAxisSize.max,
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Icon(
          FontAwesomeIcons.times,
          size: 48,
          color: theme.errorColor,
        ),
        SizedBox(height: 16),
        Text(
          "Error",
          style: theme.textTheme.headline,
        ),
      ],
    ),
  );
}

class SnapshotBuilder<T> extends HookWidget {
  final AsyncSnapshot<T> snapshot;
  final WidgetBuilder loading;
  final Widget Function(BuildContext context, T data) success;
  final Widget Function(BuildContext context, Object error) error;

  SnapshotBuilder({
    @required this.snapshot,
    @required this.success,
    this.loading = defaultLoading,
    this.error = defaultError,
  });

  @override
```

```
  Widget build(BuildContext context) {
    assert(snapshot != null && success != null);

    if (snapshot.hasData) {
      return success(context, snapshot.data);
    }

    if (snapshot.hasError) {
      return error(context, snapshot.error);
    }

    return loading(context);
  }
}
```

Arquivo: app/lib/components/presentational/radio_button.dart

```
import 'package:flutter/material.dart';

class RadioButton<T> extends StatelessWidget {
  final bool selected;
  final String label;
  final void Function(bool selected, T value) onSelected;
  final T value;

  const RadioButton({
    Key key,
    @required this.selected,
    @required this.label,
    @required this.onSelected,
    @required this.value,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return InkWell(
      onTap: () => onSelected(!selected, value),
      child: Row(
```

```
        children: <Widget>[
          IgnorePointer(
            ignoring: true,
            child: Radio(
              onChanged: (value) {},
              groupValue: true,
              value: selected,
            ),
          ),
          SizedBox(width: 4),
          Text(
            label,
            style: TextStyle(fontSize: 18),
          ),
        ],
      ),
    );
  }
}
```

Arquivo: app/lib/components/presentational/user_profile_button.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_hooks/flutter_hooks.dart';
import 'package:gaia/screens/routes.dart';
import 'package:gaia/utils/hooks/misc.dart';

class UserProfileButton extends HookWidget {
  @override
  Widget build(BuildContext context) {
    final user = useCurrentUser();

    if (user == null) {
      return Container(width: 0, height: 0);
    }

    return IconButton(
      icon: ClipRRect(
```

```
        borderRadius: BorderRadius.circular(16),
        child: Image.network(
          user.providerData.first.photoUrl,
        ),
      ),
      onPressed: () {
        Navigator.pushReplacement(context, Routes.login());
      },
    );
  }
}
```

Arquivo: app/lib/components/presentational/ghost.dart

```
import 'package:flutter/material.dart';

class Ghost extends StatelessWidget {
  final bool show;
  final Widget child;
  final Duration duration;

  const Ghost({
    Key key,
    @required this.show,
    @required this.child,
    this.duration = const Duration(milliseconds: 150),
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return IgnorePointer(
      ignoring: !show,
      child: AnimatedOpacity(
        opacity: show ? 1 : 0,
        duration: duration,
        child: child,
      ),
    );
```

```
  }
}
```

Arquivo: app/lib/components/presentational/sync_button.dart

```
import 'package:flutter/material.dart';
import 'package:gaia/screens/routes.dart';

class SyncButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return FlatButton.icon(
      icon: Icon(
        Icons.sync,
        color: Colors.white,
      ),
      label: Text(
        'SINCRONIZAR',
        style: TextStyle(color: Colors.white),
      ),
      onPressed: () {
        Navigator.push(context, Routes.sync());
      },
    );
  }
}
```

Arquivo: app/lib/components/presentational/identicon.dart

```
import 'package:flutter/widgets.dart';
import 'package:flutter_hooks/flutter_hooks.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:jdenticon_dart/jdenticon_dart.dart';

class Identicon extends HookWidget {
  final String value;
  final int size;
```

```
  Identicon(
    this.value, {
    @required this.size,
  }) : assert(size != null);

  @override
  Widget build(BuildContext context) {
    String svg = useMemoized(
      () => Jdenticon.toSvg(value.toString(), size: size),
      [value],
    );

    return SvgPicture.string(
      svg,
      fit: BoxFit.contain,
      height: size.toDouble(),
      width: size.toDouble(),
    );
  }
}
```

Arquivo: app/lib/components/presentational/circular_background.dart

```
import 'package:flutter/material.dart';

class CircularBackground extends StatelessWidget {
  final Widget child;

  const CircularBackground({Key key, this.child}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: ShapeDecoration(
        color: Colors.white,
        shape: CircleBorder(),
      ),
```

```
      child: child,
    );
  }
}
```

Arquivo: app/lib/components/presentational/empty_list.dart

```
import 'package:flutter/material.dart';

class EmptyList extends StatelessWidget {
  final String text;
  final double paddingBottom;

  const EmptyList({Key key, this.text, this.paddingBottom}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Image.asset('assets/images/ginger-cat/page-not-found.png'),
        Text(text, style: TextStyle(fontSize: 24)),
        SizedBox(height: paddingBottom)
      ],
    );
  }
}
```

Arquivo: app/lib/components/presentational/collapsing_radio_group.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_hooks/flutter_hooks.dart';
import 'package:gaia/components/presentational/radio_button.dart';

@immutable
class RadioItem<T> {
  final String label;
```

```
  final T key;

  RadioItem({
    this.key,
    this.label,
  });
}

class CollapsingRadioGroup<T> extends HookWidget {
  final List<RadioItem<T>> items;
  final ValueChanged<T> onSelected;

  CollapsingRadioGroup({
    Key key,
    @required this.items,
    @required this.onSelected,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final selected = useState<T>(null);

    final setSelected = (bool s, T v) {
      selected.value = s ? v : null;
      onSelected(selected.value);
    };

    return Column(
      mainAxisSize: MainAxisSize.min,
      children: <Widget>[
        for (var item in items)
          _Collapsible(
            show: selected.value == null || selected.value == item.key,
            child: RadioButton<T>(
              label: item.label,
              selected: item.key == selected.value,
              value: item.key,
              onSelected: setSelected,
            ),
```

```
          )
      ],
    );
  }
}

class _Collapsible extends HookWidget {
  final bool show;
  final Widget child;

  const _Collapsible({
    Key key,
    this.show,
    this.child,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final ticker = useSingleTickerProvider();

    return AnimatedOpacity(
      opacity: show ? 1 : 0,
      duration: const Duration(milliseconds: 100),
      child: AnimatedSize(
        duration: const Duration(milliseconds: 100),
        child: Container(
          height: show ? null : 0,
          child: child,
        ),
        vsync: ticker,
      ),
    );
  }
}
```

Arquivo: app/lib/components/presentational/gaia_wordmark.dart

```
import 'package:flutter/material.dart';
```

```
class GaiaWordmark extends StatelessWidget {
  final double size;

  const GaiaWordmark({Key key, this.size}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Text(
      'Sage',
      style: TextStyle(
        fontFamily: 'Mali SemiBold',
        fontSize: size,
        color: Colors.white,
      ),
    );
  }
}
```

Arquivo: app/lib/business/auth.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';

enum AuthError {
  cancelled,
  userDisabled,
  generic,
}

Future<AuthCredential> getGoogleCredential() async {
  final googleSignIn = GoogleSignIn();
  GoogleSignInAccount googleUser;
  try {
    googleUser = await googleSignIn.signIn();
  } catch (e) {
    throw AuthError.generic;
  }
```

```
  if (googleUser == null) {
    throw AuthError.cancelled;
  }

  final googleAuth = await googleUser.authentication;

  return GoogleAuthProvider.getCredential(
    accessToken: googleAuth.accessToken,
    idToken: googleAuth.idToken,
  );
}
```

Arquivo: app/lib/utils/crypto.dart

```
import 'dart:convert';

import 'package:encrypt/encrypt.dart';
import 'package:gaia/utils/collections.dart';
import 'package:pointycastle/pointycastle.dart';

// Warning: this was extracted from PointyCastle
// Must be REALLY sure before changing
final _kMaxInputSize = 450;

String _unsafeEncrypt(Encrypter encrypter, List<int> bytes) {
  final encrypted = encrypter.encryptBytes(bytes);
  return encrypted.base64;
}

List<String> encrypt(String publicKey, String data) {
  final pubKey = RSAKeyParser().parse(publicKey) as RSAPublicKey;

  final encrypter = Encrypter(RSA(
    publicKey: pubKey,
    encoding: RSAEncoding.OAEP,
  ));
```

```
  final dataPartitions = partition(utf8.encode(data), _kMaxInputSize);

  return dataPartitions.map((part) => _unsafeEncrypt(encrypter, part)).toList();
}
```

Arquivo: app/lib/utils/collections.dart

```
import 'dart:math';

Iterable<List<T>> partition<T>(List<T> input, int maxSize) sync* {
  final partitions = (input.length / maxSize).ceil();

  for (int i = 0; i < partitions; i++) {
    final end = min((i + 1) * maxSize, input.length);

    yield input.sublist(i * maxSize, end);
  }
}

main() {
  final a = [1, 2, 3, 4, 5, 6, 7, 8, 9];

  print(partition(a, 5).toList());
}
```

Arquivo: app/lib/utils/dimens.dart

```
import 'dart:math';

import 'package:flutter/material.dart';

final _base = Size(360, 592);

bool isTablet(BuildContext context) {
  return MediaQuery.of(context).size.shortestSide >= 600;
}
```

```
double test(double size, BuildContext context) {
  final screenSize = MediaQuery.of(context).size;

  final ratioHeight = screenSize.longestSide / _base.longestSide;
  final ratioWidth = screenSize.shortestSide / _base.longestSide;

  return size * max(ratioWidth, ratioHeight);
}
```

Arquivo: app/lib/utils/math.dart

```
T clamp<T extends num>(T value, T min, T max) {
  if (value < min) {
    return min;
  }

  if (value > max) {
    return max;
  }

  return value;
}
```

Arquivo: app/lib/utils/texts.dart

```
import 'package:gaia/models/models.dart';

String questionTypeText(QuestionType type) {
  switch (type) {
    case QuestionType.multiple:
      return 'Resposta múltipla';
    case QuestionType.single:
      return 'Resposta única';
    case QuestionType.number:
      return 'Numérica';
    case QuestionType.text:
      return 'Texto';
```

```
  }
  return null;
}
```

Arquivo: app/lib/utils/hooks/key.dart

```
import 'package:flutter/widgets.dart';
import 'package:flutter_hooks/flutter_hooks.dart';

GlobalKey<T> useGlobalKey<T extends State<StatefulWidget>>({
  String text,
  List keys,
}) {
  return Hook.use(_GlobalKeyHook(
    text: text,
    keys: keys,
  ));
}

class _GlobalKeyHook<T extends State<StatefulWidget>>
    extends Hook<GlobalKey<T>> {
  final String text;

  const _GlobalKeyHook({
    this.text,
    List keys,
  }) : super(keys: keys);

  @override
  _GlobalKeyHookState<T> createState() => _GlobalKeyHookState<T>();
}

class _GlobalKeyHookState<T extends State<StatefulWidget>>
    extends HookState<GlobalKey<T>, _GlobalKeyHook<T>> {

  GlobalKey<T> _key;

  @override
```

```
  GlobalKey<T> build(BuildContext context) {
    return _key ??= GlobalKey<T>();
  }
}
```

Arquivo: app/lib/utils/hooks/text_input.dart

```
import 'package:flutter/widgets.dart';
import 'package:flutter_hooks/flutter_hooks.dart';

TextEditingController useTextEditingController({
  String text,
  List keys,
}) {
  return Hook.use(_TextEditingControllerHook(
    text: text,
    keys: keys,
  ));
}

class _TextEditingControllerHook extends Hook<TextEditingController> {
  final String text;

  const _TextEditingControllerHook({
    this.text,
    List keys,
  }) : super(keys: keys);

  @override
  _TextEditingControllerHookState createState() =>
      _TextEditingControllerHookState();
}

class _TextEditingControllerHookState
    extends HookState<TextEditingController, _TextEditingControllerHook> {
  TextEditingController _textEditingController;

  @override
```

```
  TextEditingController build(BuildContext context) {
    return _textEditingController ??= TextEditingController(
      text: hook.text,
    );
  }

  @override
  void dispose() {
    super.dispose();
    _textEditingController.dispose();
  }
}
```

Arquivo: app/lib/utils/hooks/misc.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_hooks/flutter_hooks.dart';

ThemeData useTheme() {
  final context = useContext();
  return Theme.of(context);
}

void useAsyncEffect(Future<dynamic> Function() effect, [List keys]) {
  useEffect(() {
    effect();
  }, keys);
}

FirebaseUser useCurrentUser() {
  final future = useMemoized(() => FirebaseAuth.instance.currentUser(), []);

  final snapshot = useFuture(future);

  if (snapshot.hasData) {
    return snapshot.data;
  }
```

```
  return null;
}

T useArguments<T>() {
  return useMemoized(() {
    final context = useContext();

    final arguments = ModalRoute.of(context).settings.arguments;

    if (arguments is T) {
      return arguments;
    }

    return null;
  });
}
```

Arquivo: app/lib/models/models.dart

```
import 'package:json_annotation/json_annotation.dart';

part 'models.g.dart';

enum QuestionType {
  single,
  multiple,
  number,
  text,
}

@JsonSerializable()
class User {
  final String uid;

  const User({this.uid});

  factory User.fromJson(Map<String, dynamic> json) => _$UserFromJson(json);
```

```
  Map<String, dynamic> toJson() => _$UserToJson(this);
}

@JsonSerializable()
class Survey {
  String uuid;
  String title;
  User owner;
  List<SurveyQuestion> questions;
  int answers;

  Survey({
    this.uuid,
    this.title,
    this.owner,
    this.questions,
    this.answers,
  });

  factory Survey.fromJson(Map<String, dynamic> json) => _$SurveyFromJson(json);

  Map<String, dynamic> toJson() => _$SurveyToJson(this);
}

@JsonSerializable()
class SurveyQuestion {
  String uuid;
  QuestionType type;
  String title;
  String description;
  bool required;

  final Map<String, dynamic> extras;

  SurveyQuestion({
    this.uuid,
    this.type,
    this.title,
```

```
    this.description,
    this.required,
    this.extras,
  });

  factory SurveyQuestion.fromJson(Map<String, dynamic> json) =>
      _$SurveyQuestionFromJson(json);

  Map<String, dynamic> toJson() => _$SurveyQuestionToJson(this);
}
```

Arquivo: app/lib/models/models.g.dart

```
// GENERATED CODE - DO NOT MODIFY BY HAND

part of 'models.dart';

// **************************************************************************
// JsonSerializableGenerator
// **************************************************************************

User _$UserFromJson(Map<String, dynamic> json) {
  return User(
    uid: json['uid'] as String,
  );
}

Map<String, dynamic> _$UserToJson(User instance) => <String, dynamic>{
      'uid': instance.uid,
    };

Survey _$SurveyFromJson(Map<String, dynamic> json) {
  return Survey(
    uuid: json['uuid'] as String,
    title: json['title'] as String,
    owner: json['owner'] == null
        ? null
        : User.fromJson(json['owner'] as Map<String, dynamic>),
```

```
    questions: (json['questions'] as List)
        ?.map((e) => e == null
            ? null
            : SurveyQuestion.fromJson(e as Map<String, dynamic>))
        ?.toList(),
    answers: json['answers'] as int,
  );
}

Map<String, dynamic> _$SurveyToJson(Survey instance) => <String, dynamic>{
      'uuid': instance.uuid,
      'title': instance.title,
      'owner': instance.owner,
      'questions': instance.questions,
      'answers': instance.answers,
    };

SurveyQuestion _$SurveyQuestionFromJson(Map<String, dynamic> json) {
  return SurveyQuestion(
    uuid: json['uuid'] as String,
    type: _$enumDecodeNullable(_$QuestionTypeEnumMap, json['type']),
    title: json['title'] as String,
    description: json['description'] as String,
    required: json['required'] as bool,
    extras: json['extras'] as Map<String, dynamic>,
  );
}

Map<String, dynamic> _$SurveyQuestionToJson(SurveyQuestion instance) =>
    <String, dynamic>{
      'uuid': instance.uuid,
      'type': _$QuestionTypeEnumMap[instance.type],
      'title': instance.title,
      'description': instance.description,
      'required': instance.required,
      'extras': instance.extras,
    };

T _$enumDecode<T>(
```

```
  Map<T, dynamic> enumValues,
  dynamic source, {
  T unknownValue,
}) {
  if (source == null) {
    throw ArgumentError('A value must be provided. Supported values: '
        '${enumValues.values.join(', ')}');
  }

  final value = enumValues.entries
      .singleWhere((e) => e.value == source, orElse: () => null)
      ?.key;

  if (value == null && unknownValue == null) {
    throw ArgumentError('`$source` is not one of the supported values: '
        '${enumValues.values.join(', ')}');
  }
  return value ?? unknownValue;
}

T _$enumDecodeNullable<T>(
  Map<T, dynamic> enumValues,
  dynamic source, {
  T unknownValue,
}) {
  if (source == null) {
    return null;
  }
  return _$enumDecode<T>(enumValues, source, unknownValue: unknownValue);
}

const _$QuestionTypeEnumMap = {
  QuestionType.single: 'single',
  QuestionType.multiple: 'multiple',
  QuestionType.number: 'number',
  QuestionType.text: 'text',
};
```

Arquivo: server/sage/main.py

```
from firebase_admin import credentials, initialize_app
from flask import Flask, jsonify
from pony.flask import Pony

from gaia.config import PORT, FIREBASE_CERTIFICATE

app = Flask(__name__)

Pony(app)

initialize_app(
    credentials.Certificate(FIREBASE_CERTIFICATE)
)

if app.env != 'development':
    @app.errorhandler(Exception)
    def error_handler(error):
        from gaia.utils.exceptions import GaiaException
        if isinstance(error, GaiaException):
            return jsonify({
                'code': error.status_code,
                'type': type(error).__name__,
                'message': error.message,
            }), error.status_code

        print(type(error).__name__, error.args)

        return jsonify({
            'code': 500,
            'type': 'Internal Server Error',
            'message': error.args
        }), 500

from gaia.routes import blueprints

for (bp, prefix) in blueprints:
    app.register_blueprint(bp, url_prefix=prefix)
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=PORT)
```

Arquivo: server/sage/config/__init__.py

```
import base64
import json
from os import environ
from pathlib import Path

DEVELOPMENT = environ.get('ENVIRONMENT') != 'production'

if DEVELOPMENT:
    DATABASE = [
        'sqlite',
        str(Path(__file__).parent.parent.parent / 'db.sqlite3'),
        True,
    ]

else:
    DATABASE = [
        'postgres',
        environ['DATABASE_URL'],
    ]

_firebase_cert = environ.get('FIREBASE_ADMIN_KEY')
if _firebase_cert is None:
    _firebase_cert_path = Path(__file__).parent / 'firebase-admin.json'
    with open(_firebase_cert_path) as f:
        FIREBASE_CERTIFICATE = json.load(f)

else:
    _firebase_cert_b64_dec = base64.b64decode(_firebase_cert)
    FIREBASE_CERTIFICATE = json.loads(_firebase_cert_b64_dec)

PORT = int(environ.get('PORT', 5000))
```

```
Arquivo: server/sage/business/emails.py

from gaia.models.db import Survey


def compile_answers_and_send_email(survey: Survey):
    pass


Arquivo: server/sage/business/surveys.py

from gaia.models.db import Survey, Question
from gaia.utils.exceptions import Forbidden


def create_survey(survey_data, owner):
    survey = Survey(
        owner=owner,
        title=survey_data['title'],
    )

    for question_data in survey_data['questions']:
        Question(
            survey=survey,
            type=question_data['type'],
            title=question_data['title'],
            description=question_data['description'],
            required=question_data['required'],
            extras=question_data['extras'],
        )

    return survey


def edit_survey(survey_data, user):
    survey = Survey[survey_data['id']]

    if survey.owner != user:
```

```
        raise Forbidden('user is not the owner of this survey')

    return survey
```

Arquivo: server/sage/routes/users.py

```
from datetime import datetime

from flask import Blueprint, jsonify

from gaia.models.db import User
from gaia.utils.crypto import RSAKey
from .utils import verify_user, auth_required

bp = Blueprint('users', __name__)


@bp.route('/')
def root():
    user = verify_user()

    return jsonify({
        'teste': f'{datetime.now().isoformat()}',
        'oi': user.uid,
    })


@bp.route('/login')
@auth_required
def user_login(user: User):
    public_pem = RSAKey.from_pem(user.private_key).to_public_pem()

    return jsonify({
        'public_key': public_pem,
    })
```

Arquivo: server/sage/routes/surveys.py

```
from flask import Blueprint, jsonify, request
from pony.orm import select

from gaia.business.surveys import create_survey
from gaia.models.db import Survey
from gaia.models.schemas import validate_survey
from gaia.utils.exceptions import NotFound
from .utils import auth_required

bp = Blueprint('surveys', __name__)


@bp.route('/')
@auth_required
def survey_list(user):
    survey_uuids = select(s.uuid for s in Survey if s.owner == user)

    return jsonify({
        'content': [{'uuid': uuid} for uuid in survey_uuids]
    })


@bp.route('/', methods=['POST'])
@auth_required
def survey_create(user):
    survey_data = validate_survey(request.get_json())

    survey = create_survey(survey_data, user)

    return jsonify(survey.as_dict())


@bp.route('/<string:suuid>')
@auth_required
def survey_detail(user, suuid):
    survey = Survey.get(uuid=suuid)

    if survey is None or survey.owner != user:
```

```
        raise NotFound(f'survey with uuid {suuid} not found')

    return jsonify(survey.as_dict())
```

Arquivo: server/sage/routes/utils.py

```
from functools import wraps

from firebase_admin.auth import verify_id_token, CertificateFetchError, \
    RevokedIdTokenError, ExpiredIdTokenError, InvalidIdTokenError
from flask import request
from schema import SchemaError

from gaia.models.db import User
from gaia.utils.crypto import RSAKey
from gaia.utils.exceptions import BadRequest, Unauthorized


def verify_user() -> User:
    token = request.args.get('token')

    if token is None:
        return User[1]
        # raise BadRequest('Missing token')  # TODO

    try:
        data = verify_id_token(token, check_revoked=False)
    except (
            InvalidIdTokenError,
            ExpiredIdTokenError,
            RevokedIdTokenError,
            CertificateFetchError,
    ) as e:
        print(e)
        raise Unauthorized()

    uid = data['uid']
```

```
    user = User.get(uid=uid)

    if user is not None:
        return user

    return User(
        uid=uid,
        private_key=RSAKey.generate().to_private_pem(),
    )


def auth_required(decorated):
    @wraps(decorated)
    def decorator(*args, **kwargs):
        user = verify_user()
        return decorated(*args, **kwargs, user=user)

    return decorator
```

Arquivo: server/sage/routes/__init__.py

```
from .answers import bp as answers_bp
from .surveys import bp as surveys_bp
from .users import bp as users_bp

blueprints = [
    (answers_bp, '/answers'),
    (surveys_bp, '/surveys'),
    (users_bp, '/users'),
]
```

Arquivo: server/sage/routes/answers.py

```
from flask import Blueprint

from .utils import auth_required
```

```
bp = Blueprint('answers', __name__)


@bp.route('/', methods=['POST'])
@auth_required
def answer_create(user, sid):
    pass


Arquivo: server/sage/utils/crypto.py

import base64

from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import padding, rsa

b64ct = 'MR3Tjd6jKrB4+mjuHNEK72XkgtUd6x2l1+C1d6GJAgycJOTPFZucG7c8hq47vOlsgTmqZK0weNukW0E


class RSAKey:
    def __init__(self, private_key):
        self._pk = private_key

    @classmethod
    def generate(cls) -> 'RSAKey':
        private_key = rsa.generate_private_key(
            public_exponent=65537,
            key_size=4096,
            backend=default_backend(),
        )

        return cls(private_key)

    @classmethod
    def from_pem(cls, pem: str):
        private_key = serialization.load_pem_private_key(
            pem.encode('utf-8'),
            password=None,
```

```
            backend=default_backend(),
        )

        return cls(private_key)

    def to_private_pem(self) -> str:
        return self._pk.private_bytes(
            encoding=serialization.Encoding.PEM,
            format=serialization.PrivateFormat.TraditionalOpenSSL,
            encryption_algorithm=serialization.NoEncryption()
        ).decode('utf-8')

    def to_public_pem(self) -> str:
        public_key = self._pk.public_key()
        return public_key.public_bytes(
            encoding=serialization.Encoding.PEM,
            format=serialization.PublicFormat.SubjectPublicKeyInfo
        ).decode('utf-8')

    def decrypt(self, b64ciphertext_or_list):
        if isinstance(b64ciphertext_or_list, str):
            b64ciphertext_or_list = [b64ciphertext_or_list]

        return ''.join(
            self.decrypt_single(b64ciphertext) for b64ciphertext in b64ciphertext_o

    def decrypt_single(self, b64ciphertext):
        ciphertext = base64.b64decode(b64ciphertext)

        decrypted_bytes = self._pk.decrypt(
            ciphertext,
            padding.OAEP(
                mgf=padding.MGF1(algorithm=hashes.SHA1()),
                algorithm=hashes.SHA1(),
                label=None
            ),
        )

        return decrypted_bytes.decode('utf-8')
```

```
if __name__ == '__main__':
    a = 'hvPaBp0aBwGPKhsquMnM3g+fYpDCM6tecCYIhtJDIaqFy+/TUwyIOM7u2+vwODhRqlVFwQeTD/8v+Mt

    b = [
        'MDth5DAZXM1LypOwnsaQTmQ85qBLl0THPx7Uv06ts8GQvkLH0BFVPPxgnG5503+AgbfsbPtEUGgZzaK
        'xLYrIX0lc3z/5XNURdo+d3gmqhvWJa/w8P1nKmkAvZG3c9q6EfMwb+YZX0QduMLhNPY0CqMbC3Gf4Hv
        'oda4nmWq2R+/OVGtZylEDzJW+Q6Lowa7bZV7MqEIlH2jTjBX2nt4mzYI9WFmskDlevAdyPFIl2xMXx5
    ]

    with open('priv.pem') as f:
        key = RSAKey.from_pem(f.read())

    dec = key.decrypt(b)

    print(dec)
```

Arquivo: server/sage/utils/exceptions.py

```
from abc import ABC


class GaiaException(Exception, ABC):
    status_code = None
    default_message = None

    def __init__(self, message=None):
        self.message = message if message is not None else self.default_message

        super().__init__(self.message)


class BadRequest(GaiaException):
    status_code = 400
    default_message = 'Bad Request'
```

```
class Unauthorized(GaiaException):
    status_code = 401
    default_message = 'Unauthorized'


class Forbidden(GaiaException):
    status_code = 403
    default_message = 'Forbidden'


class NotFound(GaiaException):
    status_code = 404
    default_message = 'Not Found'


class InternalServerError(GaiaException):
    status_code = 500
    default_message = 'Internal Server Error'
```

Arquivo: server/sage/models/schemas.py

```
from schema import Schema, Or, Optional, SchemaError

from gaia.models.utils import QuestionType
from gaia.utils.exceptions import BadRequest

question_schema = Schema({
    'type': Or(*QuestionType.ALL),
    'title': str,
    'required': bool,
    Optional('description', default=''): str,
    Optional('extras', default=dict): {str: object},
})

survey_schema = Schema({
    Optional('uuid', default=None): Or(str, None),
    'title': str,
    'questions': [question_schema],
```

```
})

question_extra_schemas = {
    QuestionType.SINGLE: Schema({
        'options': [str],
    }),

    QuestionType.MULTI: Schema({
        'options': [str],
    }),

    QuestionType.NUMERIC: Schema({
        Optional('min', default=None): int,
        Optional('max', default=None): int,
    }),

    QuestionType.TEXT: Schema({
        Optional('min_length', default=None): int,
        Optional('max_length', default=None): int,
    }),
}


def validate_with(schema, data):
    try:
        return schema.validate(data)
    except SchemaError as e:
        print(str(e))
        raise BadRequest('schema validation failed')


def validate_survey(data):
    validate_with(survey_schema, data)

    extra_schema = question_extra_schemas[data['type']]

    validate_with(extra_schema, data['extra'])
```

Arquivo: server/sage/models/utils.py

```
from random import choice
from string import ascii_letters, digits


class DbEnum:
    ALL = []

    @classmethod
    def validate(cls, item):
        return item in cls.ALL


class QuestionType(DbEnum):
    SINGLE = 'single'
    MULTI = 'multi'
    NUMERIC = 'numeric'
    TEXT = 'text'

    ALL = [
        SINGLE,
        MULTI,
        NUMERIC,
        TEXT,
    ]


_ID_CHARS = ascii_letters + digits


def unique_id():
    return ''.join(choice(_ID_CHARS) for _ in range(20))
```

Arquivo: server/sage/models/__init__.py

Arquivo: server/sage/models/db.py

```
import uuid
from datetime import datetime

from pony.orm import *

from gaia.config import DATABASE
from gaia.utils.crypto import RSAKey
from .utils import QuestionType

db = Database()


class User(db.Entity):
    uid = Required(str, unique=True)
    active = Required(bool, default=lambda: True)

    private_key = Required(str)

    surveys = Set(lambda: Survey)


class Survey(db.Entity):
    uuid = Required(uuid.UUID, default=uuid.uuid4, index=True, unique=True)
    title = Required(str)
    owner = Required(User)

    questions = Set(lambda: Question)
    answers = Set(lambda: Answer)

    def as_dict(self):
        d = self.to_dict(related_objects=True)
        del d['owner']
        del d['id']
        d['answers'] = self.answers.count()
        d['questions'] = [q.as_dict() for q in self.questions]
        return d
```

```
class Question(db.Entity):
    uuid = Required(uuid.UUID, default=uuid.uuid4, index=True, unique=True)
    survey = Required(Survey)

    type = Required(str, py_check=QuestionType.validate)
    title = Required(str)
    description = Optional(str)
    required = Required(bool)

    extras = Required(Json, default=dict)

    answers = Set(lambda: QuestionAnswer)

    def as_dict(self):
        return self.to_dict(exclude='id answers survey')


class Answer(db.Entity):
    survey = Required(Survey)

    uuid = Required(uuid.UUID, index=True, unique=True)

    created_at = Required(datetime)
    uploaded_at = Required(datetime)

    question_answers = Set(lambda: QuestionAnswer)


class QuestionAnswer(db.Entity):
    answer = Required(Answer)
    question = Required(Question)
    extras = Required(Json)


db.bind(*DATABASE)
db.generate_mapping(create_tables=True)
```

```
def _ns():
    with db_session:
        if count(u for u in User) == 0:
            user = User(
                uid='testuser',
                private_key=RSAKey.generate().to_private_pem(),
            )

            survey = {
                'title': 'Questionário de Preferências',
                'questions': [
                    {
                        'type': 'single',
                        'title': 'Qual sua cor preferida?',
                        'description': 'Lorem ipsum dolor sit amet, consectetur adipisci
                                       'Fusce venenatis ut mauris a dignissim.',
                        'required': True,
                        'extras': {
                            'options': [
                                'Amarelo',
                                'Azul',
                                'Preto',
                                'Verde',
                                'Vermelho',
                            ]
                        }
                    },
                    {
                        'type': 'multi',
                        'title': 'Quais comidas você gosta?',
                        'description': '',
                        'required': True,
                        'extras': {
                            'options': [
                                'Pizza',
                                'Sushi',
                                'Feijoada',
                                'Churrasco',
                            ]
```

```
                    }
                },
                {
                    'type': 'numeric',
                    'title': 'Quantos anos você tem?',
                    'description': 'Em anos',
                    'required': True,
                    'extras': {
                        'min': 0,
                        'max': 130,
                    }
                },
                {
                    'type': 'text',
                    'title': 'Qual seu nome?',
                    'description': 'Nome completo',
                    'required': True,
                    'extras': {
                        'min_length': 10,
                        'max_length': 25,
                    }
                },
            ]
        }

        from gaia.business.surveys import create_survey
        create_survey(survey, user)


_ns()
```