INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DO RIO GRANDE DO SUL
CAMPUS RESTINGA

APLICATIVO DE COLETA E VISUALIZAÇÃO DE DADOS ESTATÍSTICOS DE INTERESSE PÚBLICO BASEADO NOS PRINCÍPIOS DE ARQUITETURA LIMPA

CRISTIAN WEBER

Porto Alegre
2023

CRISTIAN WEBER

APLICATIVO DE COLETA E VISUALIZAÇÃO DE DADOS ESTATÍSTICOS DE INTERESSE PÚBLICO BASEADO NOS PRINCÍPIOS DE ARQUITETURA LIMPA

Trabalho de Conclusão de Curso apresentado, junto ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal Educação, Ciência e Tecnologia do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Jean Carlo Hamerski

Porto Alegre
2023

CRISTIAN WEBER

APLICATIVO DE COLETA E VISUALIZAÇÃO DE DADOS ESTATÍSTICOS DE INTERESSE PÚBLICO BASEADO NOS PRINCÍPIOS DE ARQUITETURA LIMPA

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Jean Carlo Hamerski

Aprovado em janeiro de 2023.

Documento assinado digitalmente
gov.br
JEAN CARLO HAMERSKI
Data: 06/03/2023 08:46:31-0300
Verifique em https://verificador.iti.br

Jean Carlo Hamerski - IFRS

Documento assinado digitalmente
gov.br
EDUARDA RODRIGUES MONTEIRO
Data: 14/03/2023 10:51:27-0300
Verifique em https://validar.iti.gov.br

Eduarda Rodrigues Monteiro – IFRS

Documento assinado digitalmente
gov.br
ROBEN CASTAGNA LUNARDI
Data: 13/03/2023 08:57:37-0300
Verifique em https://verificador.iti.br

Roben Castagna Lunardi – IFRS

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL
Reitor: Prof. Júlio Xandro Heck
Pró-Reitora de Ensino: Prof. Lucas Coradini
Diretor-geral do *Campus* Restinga: Prof. Rudinei Müler
Coordenador do CST em Análise e Desenvolvimento de Sistemas: Prof. Iuri Albandes Cunha Gomes
Bibliotecária-chefe do *Campus* Restinga: Paula Porto Pedone

AGRADECIMENTOS

Primeiramente agradeço a Deus por iluminar esta caminhada.

Agradeço ao IFRS – Campus Restinga pela oportunidade de realizar meus estudos nesta magnífica instituição de ensino. Também agradeço ao corpo docente pelo ensino, apoio e orientação nos estudos.

Deixo um agradecimento especial ao professor orientador Sr. Jean Carlo Hamerski pela brilhante orientação, pelo incentivo e dedicação.

Agradeço à professora Eduarda Rodrigues Monteiro e ao professor Roben Castagna Lunardi por sua dedicação e colaboração na banca avaliadora.

Gratidão à minha esposa Fernanda Flores e ao meu filho Bernardo Flores Weber pelo incentivo, compreensão e paciência durante o período de estudo.

Aos demais familiares, amigos e colegas de serviço agradeço pelo apoio, paciência e incentivo na busca do conhecer através do estudo.

"Por vezes sentimos que aquilo que fazemos não é senão uma gota de água no mar. Mas o mar seria menor se lhe faltasse uma gota".

(Madre Teresa de Calcuta)

RESUMO

Vivemos na era digital onde é permitido às pessoas terem acesso às informações públicas, por força do art. 1º, art. 2º e art. 3º da Lei nº 12.527 (Brasil, 2011), e aos meios tecnológicos necessários para a sua visualização. É desejável que a informação pública deva ser de conhecimento da sociedade e a ela apresentada de forma textual, sonora ou gráfica. Com a popularidade dos smartphones e seus aplicativos, é possível que as pessoas tenham acesso aos dados estatísticos das informações públicas, não só para o seu conhecimento, mas como direito democrático. Este trabalho de conclusão de curso tem como objetivo projetar e desenvolver um aplicativo para dispositivos móveis com a finalidade de coletar informações estatísticas disponibilizadas pelos órgãos governamentais, analisá-las e apresentá-las para o usuário através de gráficos. Para o desenvolvimento do aplicativo, foi utilizado o framework Flutter, que apresenta as vantagens de ser um kit de desenvolvimento gratuito e de código aberto, que permite desenvolver aplicativos nativos para Android e iOS, além de permitir que a mesma estrutura de código, com pequenas adaptações, sejam compiladas para Web e Desktop. Por fim, na concepção do aplicativo quanto à arquitetura de software, foram utilizados os princípios da Arquitetura Limpa, visando agilidade na implementação e manutenção do código. Como fonte de dados, foi utilizada a API de dados agregados do Instituto Brasileiro de Geografia e Estatística (IBGE) que disponibiliza os dados das pesquisas e censos realizados por este órgão governamental. Ao final do trabalho de conclusão de curso, foi possível gerar uma versão do aplicativo considerada um produto minimamente viável que coleta as informações estatísticas utilizando a API do IBGE e permite que o usuário personalize o dado específico que deseje visualizar na forma de gráfico, podendo por fim salvar ou compartilhar os gráficos gerados nas redes sociais.

Palavras-chave: *Flutter; Dart; Dados Públicos; Arquitetura Limpa; IBGE.*

ABSTRACT

We live in the digital era where people are allowed to have access to public information, pursuant art. 1st, art. 2nd and art. 3rd of Law nº 12,527 (Brazil, 2011), and to the technological means necessary for its visualization. It is desirable that public information should be known by society and presented to it in textual, sound or graphic form. With the popularity of smartphones and their applications, it is possible for people to have access to statistical data from public information, not only for their knowledge, but as a democratic right. This course conclusion work aims to design and develop an application for mobile devices in order to collect statistical information provided by government agencies, analyze them and present them to the user through graphs. For the development of the application, the Flutter framework was used, which has the advantages of being a free and open source development kit, which allows the development of native applications for Android and iOS, in addition to allowing the same code structure, with small adaptations, are compiled for Web and Desktop. Finally, in the design of the application regarding the software architecture, the principles of Clean Architecture were used, aiming at agility in the implementation and maintenance of the code. As a data source, the Brazilian Institute of Geography and Statistics (IBGE) aggregate data API was used, which provides data from surveys and censuses carried out by this government agency. At the end of the course conclusion work, it was possible to generate a version of the application considered a minimally viable product that collects statistical information using the IBGE API and allows the user to customize the specific data that he wants to visualize in the form of a graph, being able to order to save or share the generated graphics on social networks.

Keywords: *Flutter; dart; Public Data; Clean Architecture; IBGE.*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Vivemos em um mundo que está em constante evolução. Atualmente atingimos a era digital ou era da informação que surgiu no final do século XX, predecessora da era industrial, resultado da evolução dos meios de comunicação através dos avanços tecnológicos da informática e da Internet (PENA, 2021).

O resultado deste avanço tecnológico pode ser observado tanto em equipamentos quanto em informação, pois os recursos computacionais que estão em constante evolução garantem uma maior capacidade de processamento e armazenamento de informações. Como consequência, temos o surgimento do *big data*, terminologia utilizada para imensos volumes de dados, os quais também podemos dizer que são fontes de conhecimento desde que sejam tratados. Segundo Eric Schmidt, ex-CEO do Google (MundoGEO, 2013), a cada dois dias atuais a Internet produz a mesma quantidade de informação que foi produzida desde o início da civilização até o ano de 2003. Em 2012 foram criados aproximadamente 2,5 bilhões de Gigabytes de dados por dia (MundoGEO, 2013). Já a publicação realizada em maio de 2020 pela Global DataSphere da International Data Corporation (IDC, 2020) previa que naquele ano seriam movimentados mais de 59 zetabytes (ZB) de dados no mundo, efeito da pandemia do COVID-19 que potencializou o uso da Internet tanto para trabalho quanto para entretenimento.

Diante da grande quantidade de fontes de informações geradas e disponíveis, principalmente por órgãos governamentais, resultante de pesquisas que podem ser acessadas, estudadas e utilizadas em benefício da sociedade em geral, tem-se a necessidade de permitir o acesso público a estas fontes de dados de maneira mais intuitiva e facilitada do que já provido pelos meios existentes. O acesso público aos dados é um marco evolutivo que atende os anseios da sociedade e, aliada à evolução tecnológica, traz benefícios e facilidades, visto que na atualidade os recursos tecnológicos permitem a conexão e acesso à informação de forma segura, rápida e de qualquer lugar.

O presente trabalho de conclusão de curso tem por objetivo viabilizar aos usuários o acesso a um conjunto de dados estatísticos que apresentam a realidade histórica da sociedade brasileira. Tal objetivo é alcançado através de um aplicativo desenvolvido para dispositivos móveis que permite o acesso aos dados estatísticos disponibilizados por órgãos governamentais e a visualização destes dados na forma de gráficos na tela do dispositivo móvel. Os gráficos ainda podem ser salvos no dispositivo móvel ou compartilhados nas redes sociais.

Para o projeto do aplicativo, foram utilizados os princípios de Arquitetura Limpa (MARTIN, 2020), padrão de arquitetura de *software* proposto por Robert C. Martin, que visa uma arquitetura com característica de independência de frameworks, interface de usuário, banco de dados ou qualquer outro agente externo. Além disso, a Arquitetura Limpa busca facilitar a manutenção do código, com baixo acoplamento entre as camadas.

Para o desenvolvimento do aplicativo, foi utilizado o framework Flutter em conjunto com a linguagem de programação Dart. O framework Flutter foi desenvolvido pela Google, é um SDK de código aberto (MORAIS, 2020). Por ser multiplataforma, permite o desenvolvimento de aplicativos tanto para Android quanto para iOS através de elementos de interação chamados widgets, onde é possível criar elementos estruturais como botões e menus, elementos de estilo tipo fonte e cores, configurar o layout de margens e espaçamento e elementos de design. O grande diferencial do Flutter é não utilizar os componentes widgets fornecidos pelo sistema operacional, pois utiliza seus próprios widgets ou personaliza novos. As vantagens do uso do Flutter são enormes (NEGRI, 2020), destacando-se principalmente o suporte oficial da Google, desenvolvimento rápido e correção de bugs, desenvolvimento ágil, código aberto e gratuito, grande compatibilidade e programação simples.

Ao final do presente trabalho de conclusão de curso, chegou-se a uma prova de conceito (POC) que permite ao usuário acessar as funcionalidades principais de visualização de dados estatísticos de interesse público na forma de gráficos e compartilhá-los.

1.1 Justificativa

De forma geral, a pesquisa é uma fonte geradora de conhecimento, através de fatos e estatísticas coletadas, sendo um recurso que demonstra novas tendências ou necessidades dentro de um determinado contexto, tendo como objetivo fundamental a busca de respostas para um problema proposto. O processo de pesquisa deve seguir métodos e regras que busquem garantir a fidelidade da realidade pesquisada, assim como a confiabilidade.

A informação é um dado valioso e estratégico que, ao ser analisado de forma correta, passa a ser uma ferramenta poderosa no auxílio da tomada de uma decisão, muitas vezes sendo determinante para o direcionamento ou definição de um objetivo, pois os dados representativos da informação são valores abstraídos da realidade, ou seja, são os registros científicos que validam uma pesquisa.

Diante da enorme quantidade de informações disponíveis, faz-se necessário identificar o que realmente é informação autêntica e confiável e não autêntica. A disponibilidade de dados em diferentes formas, seja textual, sonora ou gráfica, pode ter muita desinformação se

não corresponder à verdade, podendo ser considerada uma *fake news*, classificação designada para uma notícia dita como falsa. Atualmente, toda informação classificada de *fake news* passou a ser um problema, tanto para a sociedade brasileira quanto para a sociedade mundial. Toda evolução geram novas soluções, mas também podem gerar novos problemas, que é o caso do uso da tecnologia de forma errada.

O surgimento de um novo problema, causado pelo mau uso da tecnologia ao difundir *fake news*, por exemplo, acaba por demandar a criação de novos mecanismos para controlar e evitar tal problema. Mediante a nova realidade, a sociedade necessita de orientação, proteção e conhecimento sobre o fato. Um exemplo é a iniciativa tomada pelos representantes do Conselho Nacional de Justiça (CNJ), das associações da magistratura e dos tribunais superiores e da imprensa, onde foi lançado em 2019 o Painel de Checagem de *Fake News,* com o objetivo de alertar e conscientizar a população dos perigos de compartilhar informações falsas.

As leis também devem acompanhar a evolução tecnológica, a fim de regular o que é permitido e não permitido, referente às *fake news*, houve a necessidade de criar mecanismos oficiais para que haja responsabilidade e transparência no conteúdo disponibilizado, principalmente na Internet e redes sociais, com a aprovação no Senado Federal do Projeto de Lei PL 2630/2020, em 13 de maio de 2020, que:

> “estabelece normas relativas à transparência de redes sociais e de serviços de mensagens privadas, sobretudo no tocante à responsabilidade dos provedores pelo combate à desinformação e pelo aumento da transparência na Internet, à transparência em relação a conteúdos patrocinados e à atuação do poder público, bem como estabelece sanções para o descumprimento da lei.”, (Brasil, 2020).

O presente trabalho de conclusão de curso apresenta um modelo de aplicativo mobile nativo para Android e iOS, a fim de levar o conhecimento às pessoas que procuram se beneficiar do uso da tecnologia e constantemente buscam conhecer os dados e fatos da realidade, seja para o simples saber, checagem de dados estatísticos disponibilizados em textos da Internet, ou para auxiliar na tomada de decisões.

1.2 Objetivo Geral

Projetar e desenvolver de um aplicativo para dispositivos móveis que permite acessar e apresentar os dados estatísticos de forma gráfica ao usuário, utilizando-se para o projeto os preceitos de Arquitetura Limpa e fazendo-se uso das vantagens do framework Flutter.

1.3 Objetivos Específicos

Os objetivos específicos são:

- Levantar as fontes de dados públicas que serão utilizadas pelo aplicativo;
- Projetar a solução conceitual do aplicativo;
- Implementar parte ou o todo da solução conceitual em Flutter;
- Utilizar preceitos da Arquitetura Limpa no projeto e desenvolvimento do aplicativo.

1.4 Organização do Trabalho

O Capítulo 2 apresenta três aplicações similares à proposta deste trabalho, que realizam a consulta em uma base de dados e apresentam ao usuário os dados estatísticos de forma gráfica. O Capítulo 3 apresenta as tecnologias utilizadas na implementação da solução. O Capítulo 4 apresenta a solução conceitual, com diagramas e protótipos de tela da aplicação proposta. No Capítulo 5, é apresentada a implementação da solução, detalhando a fonte dos dados, descrevendo as particularidades das tecnologias utilizadas no desenvolvimento, e descrevendo a arquitetura e o desenvolvimento do aplicativo. Por fim, no Capítulo 6, teremos a conclusão do presente trabalho, onde são apresentadas as considerações finais, relacionando os pontos positivos e negativos e as futuras melhorias para a aplicação proposta.

2. TRABALHOS CORRELATOS

Esta seção apresenta, para fins de similaridade, alguns aplicativos que possam contribuir para este trabalho de alguma forma. Os aplicativos apresentados possuem a finalidade de consultar dados públicos estatísticos e, preferencialmente, apresentar as informações de forma gráfica ao usuário. Atualmente, empresas e o Poder Público prestam informações e disponibilizam serviços aos cidadãos através do uso da tecnologia. De acordo com Filho (2019, p. 10), as instituições governamentais tiveram que se adequar à modernização tecnológica para poder se enquadrar à nova realidade do mundo digital. Tal adequação tem como objetivo facilitar e fornecer um excelente serviço, com transparência, responsabilidade e eficiência (FILHO, 2019, p. 13). As empresas também aderiram à tecnologia da informação, propiciando vários benefícios tais como: melhor gerenciamento, eficiência na execução dos processos, capacidade competitiva, segurança dos dados e poder de comunicação. Esses benefícios tecnológicos, através da prestação de serviços, do entretenimento ou da disponibilização de informações, também são estendidos aos usuários que conseguem acessá-los de qualquer lugar e com poucos cliques, tendo o conteúdo desejado na palma da mão.

Aplicativos com o mesmo fim do que o apresentado no presente trabalho de curso foram analisados para fins de comparação. Esses aplicativos são disponibilizados gratuitamente ao público em geral e desenvolvidos para plataformas móveis como smartphones e tablets. Foram selecionados três aplicativos: i) IBGE; ii) InfoMoney; iii) Anatel Serviço Móvel. Para a seleção dos aplicativos, foram seguidos os seguintes critérios:

- aplicativos para Sistema Operacional Android;
- disponibilizados na loja oficial Google Play Store;
- fornecem dados públicos ou de utilidade pública.

A seguir é apresentado cada um dos aplicativos analisados.

2.1 Aplicativo IBGE

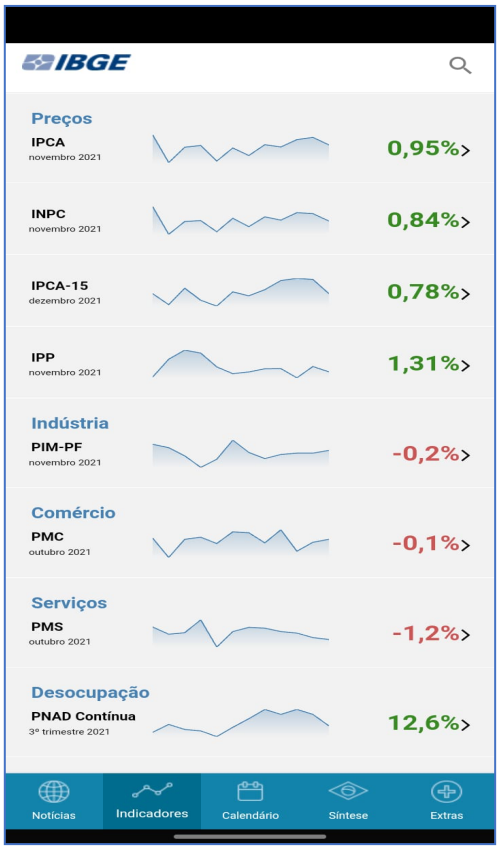
O primeiro aplicativo analisado foi o IBGE (STORE, 2022a), do Instituto Brasileiro de Geografia e Estatística, disponibilizado para os cidadãos brasileiros. “O objetivo do aplicativo é a disseminação das informações de indicadores econômicos, dados censitários, índices de preços e notícias das atividades da instituição.”

O aplicativo possui quatro funcionalidades, “Notícias”, “Indicadores”, “Calendário” e

“Síntese”, com destaque para as duas funcionalidades principais, “Indicadores” e “Síntese”.

A Figura 2.1 apresenta a tela de indicadores do IBGE. Os principais indicadores do País são apresentados em sete categorias: Preços, Indústria, Comércio, Serviços, Desocupação, PIB e Construção.

Figura 2.1- Tela indicadores do aplicativo IBGE



Fonte: Elaborado pelo autor (2022).

Os indicadores mostram a realidade da economia e são parâmetros utilizados para o cálculo de taxa de juros, inflação, aluguéis e outros índices (ADVFN, 2022).

A aplicação apresenta todos os índices na tela de indicadores, referente ao último mês analisado. Ao clicar em um determinado índice é apresentada nova tela com dados específicos do índice selecionado e com a possibilidade de visualizar os índices dos últimos doze meses, com distribuição mensal, acumulado no ano ou com variação acumulada em 12 meses. Uma desvantagem do aplicativo é não disponibilizar opção de visualização de dados de períodos anteriores a doze meses.

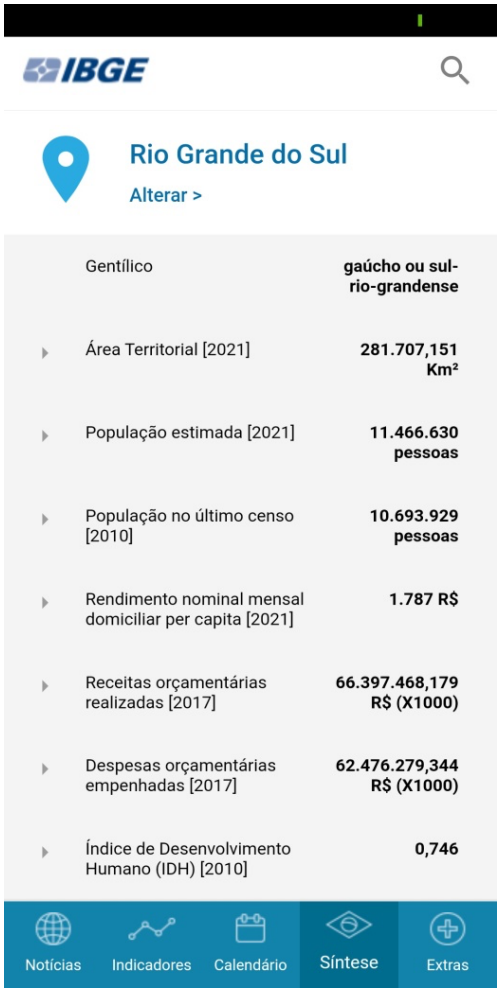
Na categoria de índices de Preço, podemos exemplificar o INPC – Índice Nacional de Preços ao Consumidor, que é um dos principais indicadores da economia, pois “mede a variação do custo de vida das famílias com chefes assalariados e com rendimento mensal compreendido entre 1 e 5 salários-mínimos mensais, representando 50% das famílias

brasileiras," (ADVFN, 2022).

Para utilizar o aplicativo, o usuário deve realizar a instalação a partir da plataforma do Google Play Store, não sendo necessário criar cadastro ou informar dados pessoais. As últimas atualizações da aplicação foram realizadas nos dias 9 de março de 2020, para corrigir problemas de conexão que ocorriam quando instalado Sistema Operacional Android de versão 7, 9 e 10; e 22 de abril de 2022 para atualização dos indicadores de síntese.

A Síntese, segunda funcionalidade a ser destacada, permite ao usuário selecionar e visualizar as informações do país, de determinado estado ou de determinada cidade. Para cidade, o aplicativo disponibiliza dados específicos como área total, população estimada, taxa de analfabetismo, taxa de escolarização, PIB per capita, taxa de mortalidade e população no último senso. É relevante destacar que na Síntese são apresentados dados quantitativos e percentuais, não havendo apresentação gráfica dos dados.

Figura 2.2- Tela síntese do aplicativo IBGE



Fonte:Elaborado pelo autor (2022).

2.2 Aplicativo InfoMoney

O aplicativo InfoMoney (STORE, 2022b), disponibilizado pela corretora de valores XP Investimentos Corretora de Câmbio, Títulos e Valores Mobiliários S.A., é um aplicativo disponibilizado gratuitamente que traz notícias e cotações para ajudar o usuário na tomada de decisões que envolvem aplicações financeiras (PLAY, 2022).

A aplicação possibilita, através de suas ferramentas, realizar consulta a cotações e criar alertas das empresas selecionadas.

Dentre as funcionalidades disponibilizadas ao usuário, pode-se citar:

- ver as últimas notícias relacionadas a investimentos, economia e política;
- acompanhar os índices, cotações e moedas;
- receber e visualizar alertas configurados;
- monitorar o ranking das ações, fundos e títulos públicos.

Através da funcionalidade “Mercados”, conforme é apresentado na Figura 2.2, é possível ter acesso em tempo real e visualizar graficamente os índices das bolsas de valores, as principais ações de valores em alta e em baixa, a cotação das moedas e visualizar os indicadores econômicos como o CDI, Poupança, Taxa SELIC, índice IPCA e IGP-M.

A Figura 2.3 apresenta o gráfico do Índice IBOVESPA, que apresenta dados corresponde a um ano. A ferramenta apresenta opção de filtragem, possibilitando ao usuário analisar seis opções de cotação. Abaixo do gráfico são apresentadas as cotações, valores de cotação de abertura, cotação mínima e máxima do período filtrado.

A aplicação está disponível na plataforma do Google Play Store, recebeu atualização em outubro de 2021 para adequação ao consentimento da LGPD. Avaliação recebida foi nota 3,1 sobre o total 5 estrelas, de um total de 2221 usuários. O número de instalações realizadas ultrapassa o número de 500 mil. Na avaliação dos usuários, há relatos de erro na página home, também ocorrido no período de análise, onde não foram apresentadas as notícias.

De modo geral, a aplicação é rápida na apresentação gráfica e na atualização ao alterar o período. A funcionalidade “Mercados” não apresentou erro, assim como a tela dos índices.

Figura 2.3- InfoMoney Índices

Fonte: Elaborado pelo autor (2022).

Figura 2.4- InfoMoney IBOVESPA

Fonte: Elaborado pelo autor (2022).

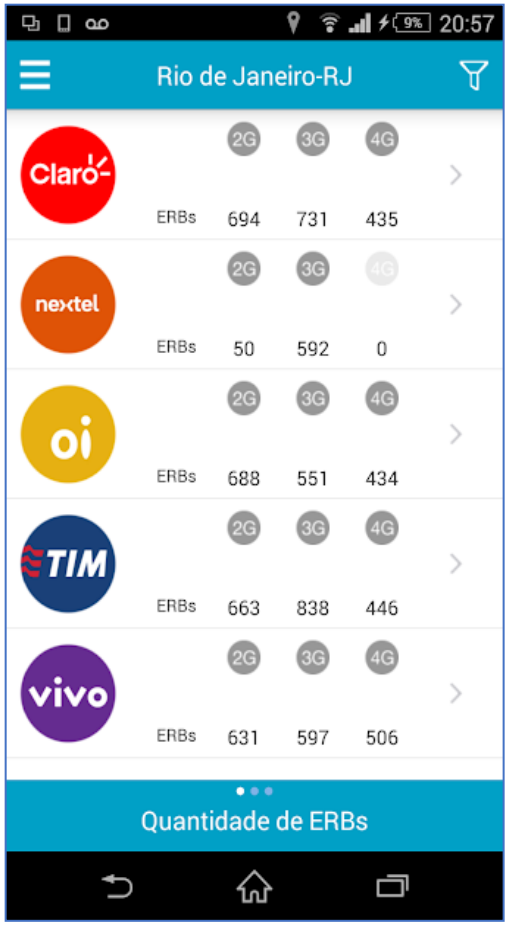
2.3 Aplicativo Anatel Serviço Móvel

O aplicativo Anatel Serviço Móvel, foi desenvolvido pela Anatel e disponibilizado em 2018, permitindo visualizar a distribuição das antenas de telefonia móvel, também conhecidas como Estação Rádio Base (ERB) e o ranking dos serviços de voz e dados.

As informações apresentadas são especificas por município, sendo necessário pesquisar e selecionar uma determinada cidade, dentre os 5570 municípios existentes no Brasil, conforme demonstra a Figura 2.4 (pesquisa da quantidade de ERBs na cidade do Rio de Janeiro por operadora e tecnologia).

A aplicação disponível na plataforma do Google Play Store recebeu atualização em abril de 2020. A avaliação recebida foi nota 2,6 sobre o total 5 estrelas, de um total de 10.588 usuários. O número de instalações realizadas ultrapassa o número de 500 mil. Problemas de funcionalidade foram relatados pelos usuários que utilizam Sistema Operacional Android e a versão mais atual da aplicação, pois há relatos de que a aplicação apresenta erros de localização de antenas e erro na informação do tipo de rede.

Figura 2.5- Quantidade de ERB por operadora

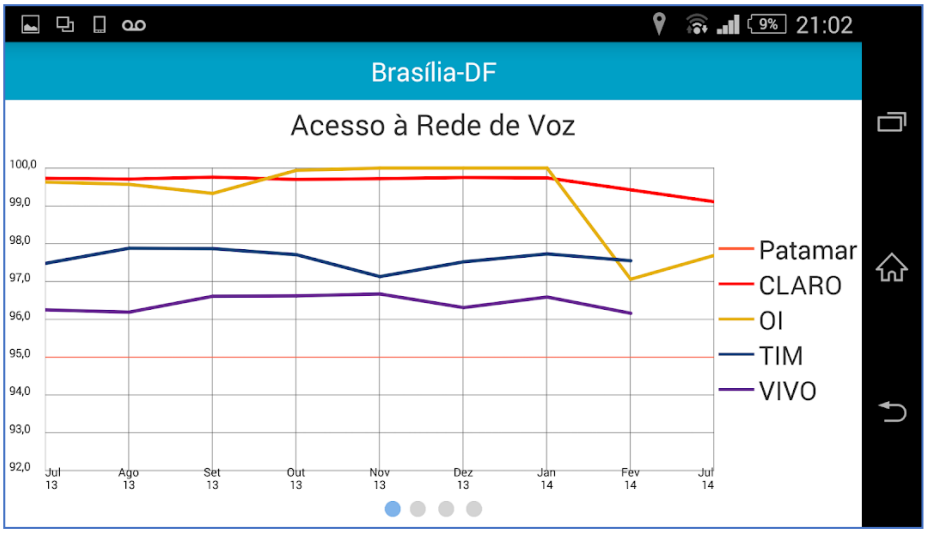


Fonte: Store (2022c).

No entanto, apesar dos erros relatados, a ferramenta é útil por apresentar os indicadores de qualidade referente a prestação do serviço de telefonia móvel, pois "a proposta da aplicação é disponibilizar a seus usuários dados sobre a telefonia celular, referente ao resultado de medidas de qualidade, número de acessos e queda das redes de dados e voz, e a última versão da aplicação inclui um novo módulo que permite aos usuários compartilharem suas experiências" (PLAY, 2022c), a partir dos registros informados pelos usuários sobre a falta de sinal ou queda de ligação e apresentar em um mapa de eventos.

Na Figura 2.5 é demonstrado o número de acessos a rede de voz por operadora na cidade de Brasília-DF.

Figura 2.6- Relação de acessos à rede de voz por operadora



Fonte: Store (2022d).

3. FUNDAMENTAÇÃO TEÓRICA

Neste tópico são apresentadas as tecnologias utilizadas para o desenvolvimento da solução proposta no trabalho de conclusão de curso. Inicialmente, a seção 3.1 apresenta o Flutter, um kit de desenvolvimento multiplataforma, o framework escolhido para o desenvolvimento do aplicativo, sendo abordado sobre a arquitetura Flutter, os Widgets e suas vantagens e desvantagens.

Na sequência, na seção 3.2, é apresentada a linguagem de programação Dart, utilizada no desenvolvimento da aplicação, pois permite criar aplicação nativa para Andoid quanto para iOS.

Ao término do capítulo temos a apresentação sobre Arquitetura de Software (Seção 3.3), com relação dos tipos de arquiteturas (subseção 3.3.1) e com ênfase a arquitetura limpa (subseção 3.3.2), finalizando com uma breve descrição na seção 3.4 sobre API.

3.1 Flutter

No ano de 2015, a empresa Google criou o Flutter, um framework ou kit de desenvolvimento utilizado para a construção de aplicações móveis, composto de um conjunto de classes, plug-ins, bibliotecas, interfaces e pacotes para auxiliar no desenvolvimento de software (FLUTTER, 2022). A proposta do framework Flutter é permitir um desenvolvimento rápido, com interfaces bonitas e com performance nativa, sem depender de API's e interpretadores de código do dispositivo, utilizando sua Virtual Machine (VM) que possibilita a compilação rápida do código.

O Flutter é baseado na linguagem de programação Dart (abordada na seção 3.2), sendo uma solução gratuita e de código aberto que permite o desenvolvimento de aplicativos multiplataforma, possibilitando que o aplicativo execute tanto no sistema operacional Android quanto no sistema operacional iOS (MORAIS, 2020), a partir de um único código fonte. Além de possibilitar a criação de aplicações Web e desktop.

Os recursos disponibilizados pelo framework Flutter permitem suporte aos itens de hardware, como câmera, GPS, rede, unidades de armazenamento, além de permitir acesso a vários tipos serviços, como armazenamento em nuvem, pagamento e autenticação.

3.1.1 Arquitetura do Flutter

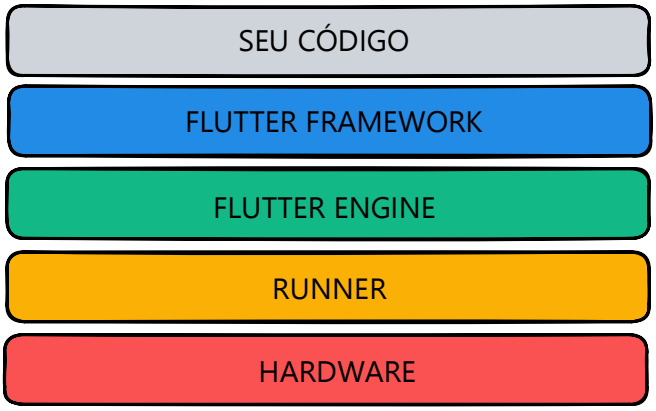
A arquitetura do Flutter é organização em camadas, permitindo a divisão do código em vários módulos. Cada camada é independente e responsável por uma determinada

funcionalidade da aplicação, do mais alto nível (interação com usuário) ao mais baixo nível (interação com os sistemas operacionais). A execução das funcionalidades ocorre através de recurso específico, como plug-ins, bibliotecas, APIs, pacotes e componentes.

A Figura 3.1 apresenta uma visão geral da arquitetura Flutter, com a distribuição de camadas. As características de cada camada da arquitetura são:

- Primeira camada - "Seu Código": código na linguagem Dart, contempla os packages e plug-ins, imagens, fontes, requisitos, casos de uso e outros ativos;
- Segunda camada - Flutter Framework: escrito em Dart, é responsável pela implementação visual, por layouts de tela e gerenciamento dos widgets (abordado na subseção 3.1.2), sendo o componente mais utilizado durante o desenvolvimento das aplicações. Utiliza o Skia, um motor gráfico, para a renderização 2D das telas;
- Terceira camada - Flutter Engine: escrito na linguagem C++, é responsável pela estrutura de execução e com função de receber o código do framework e compilar para execução no hardware;
- Quarta camada – Runner: camada de baixo nível, responsável pela interação com os sistemas operacionais, através do gerenciamento de Threads, Mensagens de Eventos e funcionalidades do SO, sendo escrito em uma determinada linguagem conforme a plataforma, Java e C++ para Android, Objective-C/Objective-C++ para iOS e macOS e C++ para Windows e Linux;
- Quinta camada – Hardware: tipo de processadores. Compilação do código de máquina.

Figura 3.1- Arquitetura Flutter



Fonte: FLUTTER(2022b).

3.1.2 Widgets

O framework Flutter, através do conceito de widgets, possibilita o desenvolvimento de

diversos componentes de forma personalizada. No Flutter, o widget é qualquer componente da interface de usuário, ou seja, tudo o que o usuário visualizar ou interagir é um widget. O grande diferencial do Flutter é possibilitar o desenvolvimento dos próprios widgets, ao invés de utilizar os componentes do sistema operacional.

Através dos componentes de widgets, são definidos os elementos estruturais, elementos de estilo e layouts de tela, contendo a seguinte composição:

- Estrutural: são os componentes ou objetos visuais de interface, como botões, menus, textos, *labels*, barra de rolagem, janelas e demais.
- Estilos: componentes para configurar estilo de texto, como tipo fontes, tamanho, formato e cor.
- Layout: componentes para organizar e configurar o layout, como margens, espaçamentos, coluna, linha, container e outros.
- Design específico por plataforma: conjunto de widgets, layouts e temas específico para um sistema operacional. Android (Material Components), iOS (Cupertino).

Segundo Morais (2020, p. 29), “no Flutter, o próprio aplicativo é um widget. O aplicativo é o widget de nível superior e sua interface é construída usando um ou mais filhos (children widgets), que novamente são construídos usando seus filhos widgets”. Portanto, a estrutura de uma aplicação Flutter é a combinação sequencial dos widgets pais e filhos, que formam os layouts, armazenando e posicionando os diversos componentes. Essa estrutura que representa a forma de organização dos widgets é denominada de *Widget Tree*, uma árvore hierárquica de widgets.

3.1.3 Vantagens e desvantagens do Flutter

Apesar de ser uma solução nova para o desenvolvimento de aplicações, grandes empresas estão migrando para o Flutter, por apresentar um menor custo no desenvolvimento, comparado a outras soluções. Para os desenvolvedores, o Flutter oferece alta produtividade e qualidade, vem sendo a nova aposta em substituição ao React Native, por apresentar melhor fluidez.

Algumas vantagens e desvantagens de utilizar o Flutter:

a) Vantagens:

- Código fonte único: desenvolvimento multiplataforma, permite criar aplicações a partir de um código base para diferentes sistemas operacionais;
- Agilidade: com os widgets próprios os aplicativos com uso do Flutter são mais rápidos do que aplicativos nativos;
- Desenvolvimento rápido: processo de desenvolvimento, correção e testes são otimizados com a tecnologia hot reload;
- Compatibilidade: a independência de recursos nativos do sistema permite que as aplicações em Flutter sejam executadas em diferentes versões de sistemas;
- Maior desempenho: no Flutter o código fonte da aplicação é transformado em código nativo, em tempo de compilação e não em tempo de execução, diferente do framework React Native.

b) Desvantagens:

- Tamanho do pacote gerado: um aplicativo gerado no Flutter pode ser até dez vezes maior se comparamos com Java nativo;
- Incompatibilidade 32 bits: impossibilidade de criar aplicativos para dispositivos 32 bits, como aparelhos anteriores ao iPhone 5s da iOS e computadores 32 bits;
- Limitação de bibliotecas: apesar de existir muitas bibliotecas, o número de plug-ins e pacotes é limitado, se comparado com o desenvolvimento nativo.

3.2 Dart

Dart é uma linguagem de script voltada à web, desenvolvida pelo Google em 2011, fortemente tipada, ou seja, os objetos e variáveis possuem um tipo específico. Possui sintaxe similar à linguagem C e ao Java, e surgiu com o propósito de substituir o JavaScript. Em 2013 foi lançada a primeira versão estável e em 2018 foi lançado o DART 2.0, otimizado para desenvolvimento de aplicações web e para dispositivos móveis (DART, 2020).

A linguagem Dart foi aprovada pela Associação Europeia de Fabricantes de Computadores – ECMA. Disponibiliza um grande repositório de ferramentas, dentre as quais a Dart Native que contém o DartVM, máquina virtual para executar o código Dart.

Segundo Andrade (2022), a máquina virtual DartVM pode compilar o código Dart em *ahead-of-time* (AOT) e *just-in-time* (JIT):

> Compilação *ahead-of-time* é quando o código é compilado diretamente para ARM nativo, o que possibilita a performance de uma aplicação nativa. *Just-in-time* compila o código diretamente no *device*, com a aplicação em execução, o que permite um retorno em tempo real da alteração e aumenta a velocidade do ciclo de desenvolvimento. Este ponto é chamado de *hot-reload*.

A linguagem de programação Dart teve popularidade a partir do framework Flutter que a utiliza como base. “Logo, o Dart funciona sem o Flutter, mas o Flutter não funciona sem o Dart”. (ALURA, 2022).

As principais vantagens da linguagem Dart são:

- concisa e fortemente tipada;
- alta performance;
- estabilidade e eficiência;
- ampla documentação;
- multiplataforma.

3.3 Arquitetura de Software

A arquitetura de software é a área da informática responsável pelo projeto do desenvolvimento de um sistema, pois define sobre os componentes de software, suas propriedades, seus relacionamentos e estrutura. Tarefa executada pelo especialista em arquitetura de software que, a partir do modelo e requisitos do projeto, define quais são as necessidades e a estrutura fundamentais do sistema.

Portanto, a tarefa de desenvolvimento de um software deve seguir um propósito e padrões, de modo que todos os envolvidos tenham funções específicas, que facilite o desenvolvimento, a implantação, a operação e a manutenção (MARTIN, 2020, p.194).

Além de acompanhar e supervisionar um projeto, a arquitetura de software é a base para a gestão de tarefas como: Plano de Projeto, Plano de Teste, Plano de *Deploy*, Desenho detalhado, Alocação de equipes e Release do produto.

No plano de projeto, dentre as várias atividades, será identificado o público ao qual se destina e o propósito do projeto a ser desenvolvido. A partir da identificação do propósito é desenvolvido o escopo do projeto que contém os requisitos técnicos, uma atividade complexa, na qual é definida a escolha dos algoritmos e estrutura de dados, seguindo padrões.

Segundo Valente (2020, p. 306), “Padrões arquiteturais propõem uma organização de mais alto nível para sistemas de software, incluindo seus principais módulos e as relações entre eles”. Os padrões ou tipos de arquitetura são descritos resumidamente na Seção 3.3.1. A Seção 3.3.2 apresenta a Arquitetura Limpa, que foi o padrão utilizado para o projeto do

aplicativo apresentado neste trabalho de conclusão de curso.

3.3.1 Tipos de Arquiteturas de Software

Os principais tipos de arquitetura de software são (VALENTE, 2020, p. 304; ARQUITETURA DE SOFTWARE, 2022):

I. Layers (camadas): arquitetura que organiza os módulos e componentes em camadas de funcionalidade, com relação de dependência entre elas.

   As principais camadas são:

   i. Camada de apresentação: contém o código de apresentação e controle;
   ii. Camada de negócios: contém o código de regras de negócio ou requisitos do sistema;
   iii. Camada de persistência: gerencia as seções de uso, mantendo os dados de diferentes fontes (arquivos, BD, rede...) e fornecendo quando solicitado;
   iv. Banco de dados: fonte de informação fora da aplicação;
   v. Assistentes e classes: recursos necessários para o funcionamento da aplicação.

II. Client-server (cliente-servidor): arquitetura de aplicação distribuída de tarefas entre o servidor que fornece um serviço ou recurso e o cliente que consome o serviço ou recurso.

III. Model-View-Controller (MVC): padrão arquitetural que divide a aplicação em três camadas, conforme a funcionalidade, a fim de ter um código mais limpo, reusável e de fácil manutenção. Cada parte ou camada possui sua responsabilidade que são:

   i. Modelo: camada que contém as classes e entidades responsáveis pelas regras de negócio da aplicação;
   ii. Visão: camada que contém o código de visualização, ou seja, responsável pela interface com o usuário;
   iii. Controle: camada responsável pela conexão ou ligação entre o código de visualização e as regras de negócio da aplicação.

IV. Microservices (microsserviços): padrão de arquitetura baseado em múltiplos serviços, componentes mínimos e independentes. Os serviços se comunicam entre si através do uso de APIs.

   Este padrão de arquitetura apresenta os seguintes benefícios:

   i. Agilidade: desenvolvimento rápido e eficiente com equipes independentes;
   ii. Código reutilizável: o reaproveitamento de funções permite que um serviço

possa ser usado em outro recurso;

iii. Escalabilidade flexível: a arquitetura permite que micro serviços sejam escritos em diferentes linguagens, conforme a necessidade específica do serviço.

V. Pipes-and-filters (PF): padrão de arquitetura que utiliza os componentes computacionais como filtros, que recebem o código (entrada), transformam conforme algoritmo e entregam o código processado (saída).

VI. Peer-to-Peer (P2P): padrão de arquitetura de rede de computadores onde cada nó, ou ponto, trabalha tanto como cliente e servidor, provendo o compartilhamento de um serviço, sem utilizar um servidor central.

VII. Service-Oriented Architecture (SOA): padrão arquitetural orientado a serviços, que possibilita a integração ou comunicação entre serviços através de protocolos de rede, como o SOAP – Protocolo Simples de Acesso a Objeto e JSON – JavaScript Object Notation. Através destes protocolos são realizadas solicitações que possibilitam ler ou alterar dados.

VIII. Publish-Subscribe (Pub/Sub): padrão arquitetural de evento através do envio de mensagem, a comunicação ocorre entre os publicadores (Publishers) que enviam as mensagens para os assinantes (subscribers). Padrão utilizado em implementação de sistemas distribuídos, a comunicação dos eventos pelos publicadores ocorre de forma assíncrona, onde os assinantes recebem notificação de um novo evento e reagem. A forma assíncrona de comunicação por evento ocorre sem a necessidade de acoplar o remetente ao destinatário.

3.3.2 Arquitetura Limpa

Mediante o crescente mercado de desenvolvimento de software e com os diversos padrões arquiteturais existentes, Robert C. Martin (2020) propõe um novo padrão arquitetural, utilizado na Engenharia de Software, que utiliza boas práticas para escrever bons códigos, detalhado em seu livro, intitulado “Arquitetura Limpa: O Guia do Artesão para Estrutura e Design de Software”. O autor relata que “o objetivo do livro é descrever arquiteturas e designers limpos e eficientes para que os desenvolvedores de software possam criar sistemas que tenham vidas longas e lucrativas.” (Martin, 2020, p. 44).

Para tal objetivo, assim como o princípio da arquitetura de software, o modelo de Arquitetura Limpa propõe padronizar e organizar o código, de modo que a implementação do sistema favoreça a reusabilidade, coesão, independência de tecnologia e testabilidade. A

reusabilidade permite aumento da produtividade, redução de tempo de desenvolvimento, facilidade de desenvolvimento e redução de custos. A coesão garante que cada classe deva ter uma única responsabilidade. A independência de tecnologia garante que o sistema não dependa exclusivamente de um recurso e possa migrar caso necessário sem alterar a regra de negócio. A testabilidade permite a identificação de falhas, a execução de teste em componente de software e a medição da probabilidade de falha de um componente.

Além do objetivo de padronizar e organizar o código, a Arquitetura Limpa, assim como os demais modelos de arquitetura, propõe a separação das preocupações, através da divisão do software em camadas, ou módulos, onde cada parte é responsável por resolver um problema.

Segundo Martin (2020, p. 275), as arquiteturas em camadas possibilitam gerar sistemas com as seguintes características:

- **Independência de framework:** a não dependência de recursos de biblioteca de software, pois utiliza o framework somente como ferramenta;
- **Independência de interface de usuário:** permite substituir a interface de usuário sem alteração nos demais módulos e sem alterar regras de negócio;
- **Independência de banco de dados:** possibilidade de troca da base de dados, pois as regras de negócio não estão ligadas à base de dados;
- **Independência de qualquer agente externo:** sem comunicação das regras de negócio com qualquer interface externa;
- **Testabilidade:** permite realizar teste das regras de negócio sem testar elementos externos ou banco de dados ou interface de usuário.

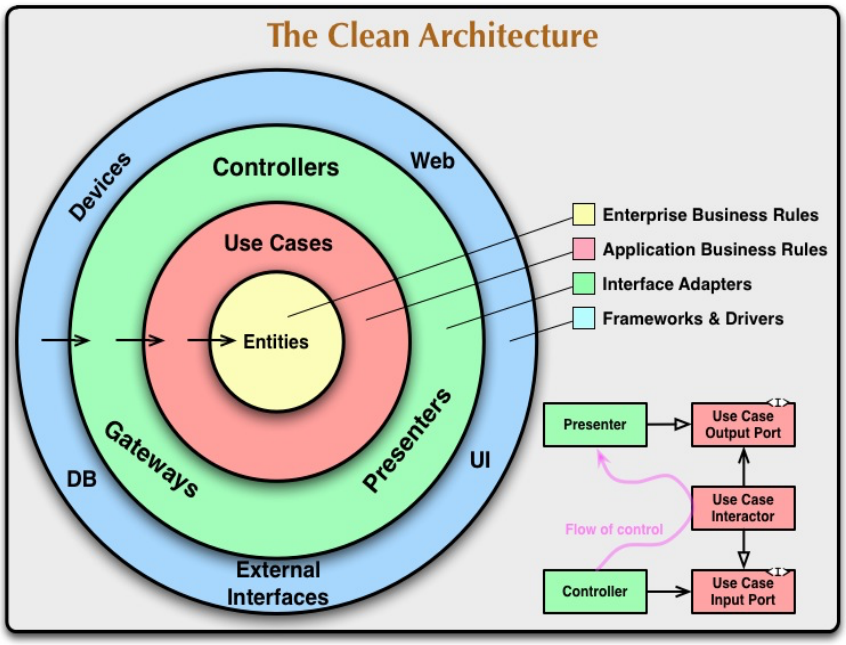
Para o funcionamento da Arquitetura Limpa, a regra da dependência do código fonte deve ser seguida, regra que consiste em direcionar as políticas de nível mais baixo no sentido das políticas de nível mais alto. Na Figura 3.2 temos a representação das diferentes áreas de software, proposta na Arquitetura Limpa, em camadas, sendo o nível mais baixo a camada mais externa onde temos o framework, que contém componentes como o banco de dados, interface de usuário, interface externa, e outros. O nível mais alto é a camada mais interna (centro) onde temos as entidades que são as políticas ou regras de negócio. Na regra de dependência, uma camada não conhece a outra, ou seja, ocorre o isolamento das camadas. Esse isolamento possibilita que os componentes sejam substituídos facilmente sem afetar o sistema, ou seja, sem ocasionar em mudanças no código fonte das outras camadas.

O modelo arquitetural proposto na Arquitetura Limpa segue a seguinte divisão em

camadas, sentido do nível mais alto para o nível mais baixo:

- Entities (Entidades): camada que reúne as regras de negócios do sistema todo. A entidade pode ser uma classe, objetos de negócio com métodos ou estruturas de dados e funções;
- Use Case (Casos de Uso): camada que reúne os casos de uso, que são as regras de negócio específicas de um sistema, pois representam as ações dentro do sistema e especifica as entradas e saídas;
- Interface Adapters (Adaptadores de Interface): nesta camada está contido o código que fornece a ligação entre os casos de uso e o mundo ou agente exterior, pois é responsável por prover código que fará conversão do formato de dados utilizado pelas entidades e casos de usos;
- Frameworks & Drivers (Frameworks e Drivers): camada que contém as ferramentas externas. São os agentes externos: bancos de dados, framework, interface do usuário a web.

Figura 3.2- Arquitetura Limpa



Fonte: The Clean Architecture (2012).

3.4 API

API é um acrônimo em inglês para *application protocol interface* (interface de programação de aplicação). Consiste em um conjunto de definições e protocolos que possibilitam realizar a integração entre sistemas, permitindo que os serviços dos diferentes

sistemas estabeleçam uma comunicação sem saber como foram implementados, pois são recursos que simplificam o desenvolvimento de sistemas (REDHAT, 2017).

A integração via API estabelece uma comunicação entre o sistema que fornece o serviço e o sistema que consome o serviço, pois o sistema solicitante requisita um serviço, o sistema solicitado recebe o pedido, realiza a operação necessária e devolve o resultado ao solicitante. Durante a integração, o sistema que recebe a solicitação tem todo controle e segurança da operação, condição estabelecida através das definições e dos protocolos.

Uma API é transparente ao usuário, pois somente via programação as aplicações acessam seus serviços disponibilizados, para usufruir de suas funções. A partir da integração entre sistemas, via API, é possível fornecer novas funcionalidades a um programa, ofertar novos serviços ou ampliar e melhorar os serviços existentes, mesmo que sejam plataformas distintas, possibilitando o uso em sites, aplicativos e sistemas operacionais.

O uso de APIs na integração de sistemas trouxe inúmeros benefícios nas empresas, como melhoria nos processos, automatização de processos burocráticos, disponibilização da integração sem a necessidade de desenvolver uma nova aplicação, segurança às transações evitando fraude, e fornecendo serviços de pagamento, emissão de notas fiscais, automação do marketing digital, consulta a documentos, entre outros vários benefícios.

O desenvolvimento de uma API é realizado para um determinado propósito, a fim de atender a necessidade de uma empresa. Dentre as finalidades, pode-se citar:

- API de sistema: finalidade de disponibilizar dados entre os sistemas da organização. Exemplo de uso: ERP, sistemas de clientes, sistemas de faturamento e bancos de dados próprios;
- APIs de processo: disponibilizam a integração a partir de um formato para expôr serviços internos, a fim de otimizar e agilizar a execução dos processos. Exemplo de uso: gestão de estoque a partir da rotina de vendas, gerar a análise de perfil de clientes;
- APIs de experiência: disponibilizam um contexto de negócios, a partir da obtenção dos dados e processos que foram obtidos via APIs de sistema e processo. Exemplo de uso: análise de perfil de usuários de aplicativos mobile, portais internos e dados de clientes.

Os tipos de API existentes pelo critério de acesso são:

- APIs públicas ou abertas: A API é disponibilizada para todos (desenvolvedores e usuários), seu código é aberto para uso livre. Uma empresa disponibiliza uma API de forma aberta para que seu serviço seja consumido;
- APIs privadas ou internas: A API é usada apenas internamente, uso exclusivo do criador ou empresa para a qual foi desenvolvida. Ao restringir o acesso somente interno, garante às empresas um maior controle, assim como maior produtividade ao reutilizar serviços;
- APIs de parceiros: A API é compartilhada com parceiros de negócios, e pessoas externas à empresa, com permissões exclusivas, com objetivo de promover a parceria comercial e estratégica dos envolvidos.

A comunicação via API pode utilizar como ferramenta para transporte da informação os seguintes modelos:

- JSON: acrônimo para de “Javascript Object Notation” ou simplesmente “Notação de objeto JavaScript”. Modelo para a transmissão de informações no formato de texto entre diferentes linguagens, formato de serialização de dados muito utilizado em web services. A principal característica do modelo JSON é a sua legibilidade, podendo facilmente ser lido por humanos, sem a necessidade de uma aplicação auxiliar;
- XML: “Extensible Markup Language” ou XML, é uma linguagem de marcação. Um conjunto de códigos para determinar a estrutura de dados, que facilita a troca de informação entre os sistemas.

4. SOLUÇÃO CONCEITUAL

O presente trabalho de conclusão de curso apresenta o projeto e desenvolvimento de um aplicativo para dispositivos móveis, utilizando o conceito de Arquitetura Limpa com a finalidade de arquitetar o sistema em camadas bem definidas, através do framework Flutter baseado na linguagem de programação Dart. O aplicativo tem como objetivo disponibilizar informações públicas a seus usuários por meio de uma interface intuitiva que permita uma fácil navegação para acesso aos dados estatísticos disponíveis, com posterior visualização na forma de gráficos.

A principal funcionalidade desejada é permitir que o usuário escolha um determinado assunto a ser pesquisado na fonte de dados estatísticos, e, a partir do assunto escolhido, o aplicativo realiza a consulta à fonte de dados, processa os dados recebidos e apresenta na tela um gráfico com base nos dados processados.

Para a solução desenvolvida neste trabalho de conclusão de curso, a fonte de dados escolhida a ser utilizada no projeto será à base de dados do Instituto Brasileiro de Geografia e Estatística (IBGE), órgão federal responsável por todos os dados estatísticos do país, que tem como sua missão institucional "retratar o Brasil com informações necessárias ao conhecimento de sua realidade e ao exercício da cidadania" (IBGE, 2022).

Na Seção 4.1 serão apresentadas maiores informações sobre o IBGE, a forma como os dados são disponibilizados, assim como a fonte de dados disponível e a tecnologia a ser utilizada para exploração pelo aplicativo. Na sequência, serão apresentadas as histórias de usuário, os requisitos funcionais e requisitos não funcionais do aplicativo (Seção 4.2) e o protótipo de telas (Seção 4.3).

4.1 Fonte de Dados do Aplicativo - IBGE

O IBGE foi criado em 1936, desde então realiza a atividade de pesquisa sobre vários assuntos, com extrema importância sobre a realidade da população brasileira, informações sobre economia, saúde, educação, cultura, produção industrial, produção agropecuária, emprego, entre outros dados relevantes. Está presente em todas as capitais, mais o Distrito Federal.

Pesquisas permanentes são realizadas pelo instituto, a partir das quais fornecem grande quantidade de informações e dados estatísticos, referente ao País, aos Estados e Municípios (CASTRO, 2018), gerando uma fonte de dados importante e uma referência para pesquisas, possibilitando inclusive a produção de mapas e gráficos.

Nesta subseção será apresentada uma breve descrição de algumas soluções que o IBGE disponibiliza para realizar pesquisa em suas bases de dados, estudo este que serviu como base para o desenvolvimento do aplicativo apresentado neste trabalho de conclusão e pode ser usado como referência por outros trabalhos acadêmicos que tenham este mesmo fim.

A consulta à base de dados do IBGE pode ser realizada diretamente na página eletrônica da instituição ou via API de serviço, que possibilita aos desenvolvedores configurar nas suas aplicações conexão à base de dados e obter retorno destes dados. Os dados obtidos via API retornam um conteúdo JSON, formato de representação e troca de informações.

Na página institucional do IBGE, uma série de serviços são disponibilizados, entre eles:

- Cidades
  - Endereço: https://cidades.ibge.gov.br
  - Possibilita a pesquisa sobre o país, estados e municípios.
  - Fornece indicadores, infográficos, tabelas, mapas e dados dos últimos sensos.
  - Indicadores: dados sobre população, economia, trabalho, educação, saúde, território e ambiente.
- Biblioteca
  - Endereço: https://biblioteca.ibge.gov.br
  - Possibilita a pesquisa do acervo da instituição.
  - Fornece livros, vídeos, fotos e periódicos da instituição.
- SIDRA: Sistema IBGE de Recuperação Automática
  - Endereço: https://sidra.ibge.gov.br/home/pnadcm
  - Dados fornecidos através de um banco de tabelas de estáticas.
  - Dados agregados: não identifica o informante.
  - Os dados fornecidos por este canal são apresentados em forma de quadros, gráficos e cartogramas.
  - Os dados fornecidos são indicadores econômicos, financeiros, sociais e políticos.
  - Dados das áreas da administração pública, agroindústria, economia, economia agrícola e estatística.
  - Detalhamento: dados fornecidos a nível municipal, com objetivo de mostrar a realidade municipal.
- Séries Históricas e Estatísticas

        - Endereço: https://seriesestatisticas.ibge.gov.br/
        - Possibilita a pesquisa de dados históricos e estatísticos.
        - Dados oficiais de natureza socioeconômica e demográfica.
        - Séries Históricas fornece indicadores econômicos relacionados ao mercado de capitais, disponibilizados pela CVM – Comissão Valores Mobiliários.
        - Séries Estatísticas fornece dados do país, estados e municípios, através de tabelas, cartogramas e pictogramas, referente a um conjunto de dados, época e local ou espécie. Dados referente a população, economia, saúde, emprego, educação e outros indicadores.
    - Estatísticas do Século XX
        - Endereço: https://seculoxx.ibge.gov.br
        - Dados de natureza econômica e populacional.
        - Indicadores sociais, políticos e culturais.
        - Resultado: dados retornados em planilha.
    - Diretório do IBGE
        - Endereço: ftp//geoftp.ibge.gov.br
        - Dados geográficos em diferentes formatos do território nacional.
        - Conteúdo: mapas rodoviários, mapas ferroviários, imagens do território, informações sobre posicionamento geodésico, organização do território.

Mediante a vasta disponibilidade de informações que a instituição disponibiliza, a fonte de dados a ser utilizada neste trabalho de conclusão será a SIDRA, através de suas tabelas. Os dados serão obtidos desta fonte de dados através da API Agregados[1], resultado de uma consulta após o usuário definir os parâmetros da pesquisa, sendo os dados consultados posteriormente apresentados graficamente ao usuário.

4.2 Histórias de usuário, requisitos funcionais e não funcionais

As histórias de usuário são descrições textuais resumidas e informais dos requisitos ou funcionalidades de um sistema, com base nas informações obtidas a partir do ponto de vista do usuário e representa o que ele faz ou necessita fazer como função de trabalho

[1] https://servicodados.ibge.gov.br/api/v3/agregados

(ALTASSIAN, 2022). Os requisitos funcionais e não funcionais servem para descrever o que o sistema deve fazer e como será feito; pois através dos requisitos funcionais são identificados os problemas e necessidades que o sistema deve atender, ou seja, representam as ações realizadas pelos usuários, enquanto os requisitos não funcionais atendem as propriedades e restrições de um sistema, pois descrevem como serão feito.

A seguir são apresentadas as histórias de usuário (subseção 4.2.1), os requisitos funcionais (subseção 4.2.2) e os requisitos não funcionais (subseção 4.2.3),

4.2.1 Histórias de usuário

Na Tabela 4.1 podemos visualizar as histórias de usuário projetadas para o aplicativo. As histórias de usuário estão organizadas por ordem de execução, iniciando com a história de cadastro de usuário. Para manter a integridade da pesquisa, o aplicativo deve garantir o acesso restrito mediante uso de senha. A partir da configuração de um usuário e senha, será possível ter acesso às funcionalidades do aplicativo. A história de usuário número 4 (US4) descreve a possibilidade de escolher a base de dados. Inicialmente será fixada a base de dados do IBGE como única fonte a ser pesquisada, mas com o avanço no desenvolvimento do aplicativo, outras fontes de dados podem ser adicionadas. A partir da história de usuário número 5 (US5) são descritas as funcionalidades de pesquisa dos dados, com opção de escolher o assunto e de realizar o compartilhamento da pesquisa realizada, pois o usuário poderá encaminhar o gráfico resultante da pesquisa para seus contatos ou salvá-la no seu dispositivo para uso posterior.

Todas as funcionalidades das histórias de usuário foram projetadas para serem executadas por um único ator.

Tabela 4.1- Descrição das Histórias de Usuário do sistema

Identificador	**História**
US 1	SENDO um usuário POSSO cadastrar usuário e senha de acesso no primeiro acesso ao aplicativo PARA QUE garanta meu acesso exclusivo à aplicação.
US 2	SENDO um usuário POSSO alterar a senha de acesso ao aplicativo PARA QUE eu mantenha uma senha confiável atualizada para acessar a aplicação.
US 3	SENDO um usuário POSSO remover meu usuário e os registros de pesquisas armazenadas. PARA QUE eu possa remover todos os dados do meu respectivo usuário.
US 4	SENDO um usuário POSSO selecionar de qual base de dados eu quero acessar dados estatísticos PARA QUE eu possa filtrar os dados estatísticos disponíveis para uma base de dados específica.

US 5	SENDO um usuário POSSO selecionar qual dado estatístico específico eu quero que seja exibido na forma de gráfico PARA QUE eu possa visualizar na forma de gráfico o respectivo dado estatístico.
US 6	SENDO um usuário POSSO selecionar de qual região eu quero acessar os dados estatísticos. PARA QUE eu possa filtrar os dados estatísticos para uma região específica.
US 7	SENDO um usuário POSSO selecionar de qual período eu quero que seja filtrado os dados que estão sendo exibidos na forma de gráfico. PARA QUE eu possa visualizar os dados exibidos na forma de gráfico em um período específico.
US 8	SENDO um usuário POSSO escolher se o atual dado estatístico que está sendo exibido na forma de gráfico será salvo no meu histórico. PARA QUE eu possa posteriormente resgatar o respectivo dado estatístico sem ter que fazer todos os filtros exigidos.
US 9	SENDO um usuário POSSO limpar todo o histórico de pesquisas realizadas. PARA QUE eu possa apagar todo o histórico das pesquisas realizadas.
US 10	SENDO um usuário POSSO apagar o histórico de um registro específico. PARA QUE eu possa apagar o histórico de uma determinada pesquisa armazenada.
US 11	SENDO um usuário POSSO compartilhar a imagem do gráfico que está sendo exibido. PARA QUE eu possa compartilhar o respectivo gráfico com quem eu desejar ou salvar no meu dispositivo.

4.2.2 Requisitos funcionais

Os recursos viáveis para o funcionamento adequado do aplicativo são descritos neste tópico, que tem por objetivo identificar os requisitos funcionais que são as funcionalidades ou serviços que a aplicação disponibilizará ao usuário, a fim de que atinja o objetivo proposto.

Na Tabela 4.2 são apresentados os requisitos funcionais identificados do aplicativo proposto no presente trabalho. Os requisitos funcionais RF001 e RF002 tratam sobre o gerenciamento do cadastro de usuário. O requisito funcional número RF003 trata sobre a seleção da base de dados. Os requisitos RF004, RF005 e RF006 tratam do serviço de busca de dados a partir da seleção de parâmetros de pesquisa. Nos requisitos funcionais RF007, RF008 e RF009 são tratadas as funções de histórico de pesquisa e de compartilhamento da pesquisa, pois o usuário pode armazenar as pesquisas realizadas para consultas futuras assim como efetuar o compartilhamento dos dados pesquisados.

.

Tabela 4.2- Descrição dos Requisitos Funcionais do Sistema

Id.	Nome	Prioridade	Status	Descrição	Responsável
RF001	Gerenciamento de Usuário e Senha	Importante	Identificado	O aplicativo deve disponibilizar funcionalidade que permitam realizar o	Usuário

				gerenciamento do cadastro de usuário e senha, sendo possível criar e alterar.	
RF002	Controle na exclusão de usuário e histórico	Importante	Identificado	O aplicativo deve disponibilizar funcionalidade que permita excluir o cadastro do usuário e todo histórico vinculado, quando existir.	Usuário
RF003	Seleção de Base de Dados	Essencial	Identificado	Quando usuário entrar na aplicação, ela deve apresentar tela que permita selecionar a fonte de dados a ser pesquisada.	Usuário
RF004	Pesquisa de Dados específicos	Essencial	Identificado	A partir da seleção da fonte de dados, o aplicativo deve apresentar ao usuário opção para a seleção de um determinado dado estatístico.	Usuário
RF005	Seleção de Região	Essencial	Identificado	A aplicação deve possibilitar ao usuário limitar a abrangência de uma pesquisa, através da seleção de um estado ou região.	Usuário
RF006	Especificação de Período da Pesquisa	Essencial	Identificado	Após a pesquisa realizada, ao usuário será permitido selecionar um determinado período para apresentação dos dados.	Usuário
RF007	Armazenar a pesquisa realizada em histórico.	Importante	Identificado	A aplicação possibilitará ao usuário armazenar a informação pesquisada em seu histórico de pesquisas, de modo que possibilite o resgate da informação para exibição.	Usuário
RF008	Gerenciamento do histórico de pesquisa.	Importante	Identificado	A aplicação possibilitará ao usuário gerenciar o histórico de suas pesquisas, permitindo a exclusão de uma pesquisa realizada ou limpar todo o seu histórico.	Usuário
RF009	Compartilhamento de Dados	Importante	Identificado	A aplicação deve dispor de recurso que permita compartilhar a informação estatística pesquisada.	Usuário

4.2.3 Requisitos não funcionais

Para a aplicação estar disponível ao usuário, é fundamental que o dispositivo móvel tenha acesso à Internet. O acesso à base de dados a ser pesquisada, assim como o compartilhamento da pesquisa realizada, depende da disponibilidade de conexão à Internet. No caso do dispositivo móvel não ter conexão à Internet, a aplicação deve identificar e apresentar informação de advertência ao usuário, não sendo possível utilizar as funcionalidades do aplicativo.

A segurança do aplicativo também é um requisito não funcional identificado, pois o acesso ao histórico de pesquisas realizadas deve ser restrito ao usuário detentor deste histórico, tanto para garantia da privacidade do usuário quanto para a integridade do dado pesquisado.

Na Tabela 4.3 são apresentados três requisitos não funcionais identificados.

Tabela 4.3- Descrição dos Requisitos Não Funcionais do Sistema

Id.	Nome	Prioridade	Status	Descrição	Responsável
RNF1	Segurança do Aplicativo	Essencial	Identificado	O aplicativo deve garantir a segurança dos dados, de forma que os dados pesquisados não sejam manipulados, assim mantendo a integridade da informação, bem como restringir as permissões de acesso às suas funcionalidades através do uso de senhas.	Analista
RNF2	Busca dos Dados	Essencial	Identificado	O dispositivo do aplicativo deve estar conectado à Internet para a busca das informações na fonte de dados.	Analista
RNF3	Compartilhamento do gráfico	Importante	Identificado	O compartilhamento da pesquisa para ser efetivado necessita conexão à Internet do dispositivo móvel.	Analista

4.3 Protótipos de telas

A Figura 4.2 apresenta o protótipo da tela de pesquisa para a fonte de dados de Séries Históricas e Estatísticas do IBGE. O usuário deve informar alguns parâmetros para a pesquisa ser efetivada. A tela, ao ser carregada, realizará uma consulta na fonte de dados, retornando a lista de temas e disponibilizará para seleção no campo tema. Demais campos não estarão selecionáveis enquanto usuário não efetuar a escolha de uma opção. Após o usuário realizar a seleção de tema, a aplicação realizará outra consulta na base de dados. Esta segunda consulta retorna os dados da categoria subtemas e o campo lista de subtema será habilitado, possibilitando ao usuário selecionar o segundo parâmetro da pesquisa. Este encadeamento de

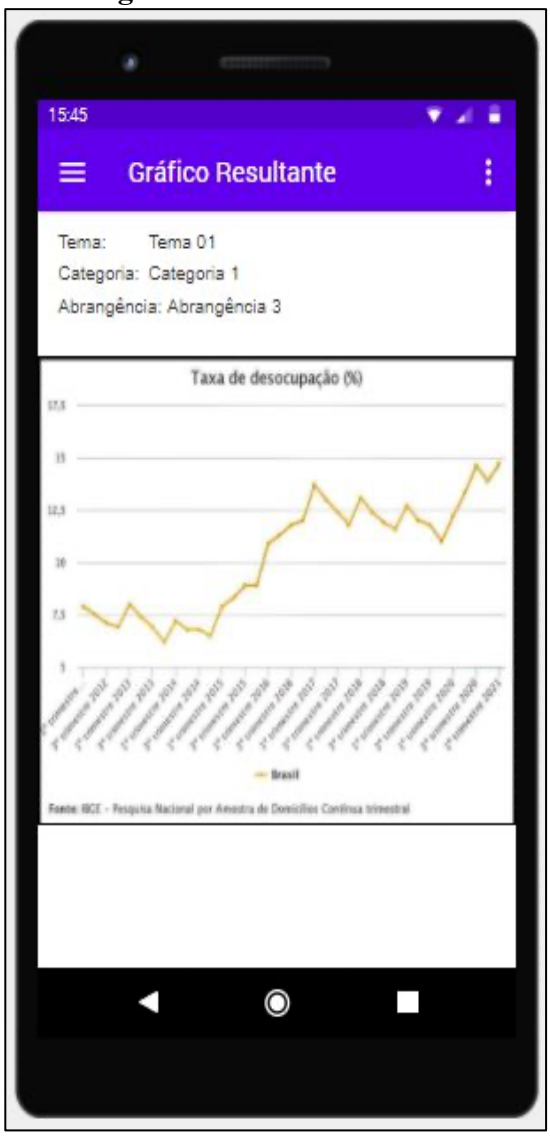
consulta dados ocorrerá até a seleção do último parâmetro. Quando todos os parâmetros forem preenchidos, a aplicação habilitará o botão “Gerar Gráfico”.

Quando o usuário clicar no botão gerar gráfico, a aplicação realizará a consulta aos dados utilizando todos os parâmetros selecionados. Os dados retornados na consulta serão apresentados em uma nova widget em forma gráfica (Figura 4.3).

Figura 4.1 – Tela Pesquisa de Série

Fonte: Autor

Figura 4.2 – Tela Resultado



Fonte: Autor

5. IMPLEMENTAÇÃO DA APLICAÇÃO

Este capítulo apresenta as informações, relativas ao desenvolvimento da aplicação proposta, em cinco seções.

Em cada seção será abordado um assunto específico, tendo a seguinte estrutura: na seção 5.1 será apresentado o ambiente e as tecnologias utilizadas; na seção 5.2 teremos os detalhes da fonte dos dados utilizada para a obtenção de informações; a arquitetura empregada na aplicação será detalhada na seção 5.3; a seção 5.4 apresentará o BLoC (Business Logic Object Components) e o Gerenciamento de Estados utilizados para tratar do dinamismo de construção das telas; por fim, na seção 5.5 são apresentadas as telas da solução mínima viável.

5.1 Tecnologias Utilizadas

Esta seção apresenta as tecnologias utilizadas no desenvolvimento da aplicação proposta. .

5.1.1 Ambiente de Desenvolvimento

A aplicação foi desenvolvida utilizando um notebook pessoal, com os seguintes recursos de *hardware* e *software*:

- **Sistema Operacional:** Windows 10
- **Tipo de Sistema:** Sistema operacional de 64 bits, processador baseado em x64;
- **Processador:** Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz;
- **Memória:** 12 GB, memória ddr3;
- **Disco Rígido:** SSD com 240 GB.

5.1.2 Ferramentas

Para a construção da aplicação, as ferramentas utilizadas foram obtidas e instaladas conforme orientação e documentação disponibilizadas na página oficial do Flutter. Os softwares utilizados foram:

- VS Code: Visual Studio Code, versão 1.71.0. Editor de código-fonte mais utilizado no mundo da programação. Desenvolvido pela Microsoft em 2015, é um editor de texto multiplataforma, está disponível para os Sistemas Operacionais Windows, Linux e Mac. Possui suporte a diversas linguagens de programação como: PHP, CSS, Phyton, C++, C#, SQL, JavaScript, Java, entre outras. Ferramenta open source, ou seja, tem código fonte aberto permitindo a sua

modificação e adaptação, conforme necessidade, além de possibilitar à criação de extensões para customizar o editor. Possui integração com o Git para o controle versionamento;

- Android Studio: Android Studio Bumblebee, 2021.1.1 Patch 1. Considerado uma IDE - Ambiente de Desenvolvimento Integrado, é uma ferramenta para desenvolver aplicativos para dispositivos móveis para Android, sendo distribuído gratuitamente pela Google, empresa dona do IDE. Na documentação oficial tem uma observação que diz: “o Flutter depende de uma instalação completa do Android Studio para fornecer suas dependências da plataforma Android”. (FLUTTER, 2022).
- Emulador Android: o Android Studio disponibiliza o recurso AVD Manager - Android Virtual Device Manager, um gerenciador virtual de dispositivo pelo qual é possível configurar um emulador que simula um dispositivo Android físico. Este recurso possibilita a execução e a realização dos testes de um aplicativo em desenvolvimento em diferentes configurações e versões de dispositivos Android.

5.1.3 Plug-ins ou Extensões

Extensões são ferramentas que disponibilizam recursos ao editor de código-fonte, auxiliando no desenvolvimento das aplicações, a fim de descomplicar, buscar agilidade e eficiência.

As duas principais extensões utilizadas no desenvolvimento da aplicação foram:

- Dart: Versão 3.48.4. Extensão de suporte à linguagem de programação Dart. Conforme descrito nos detalhes da extensão, é um recurso auxiliar para editar, refatorar, executar e recarregar aplicativos desenvolvidos para Flutter;
- Flutter: Versão 3.48.0. Conforme descrito nos detalhes da extensão, este plugin agrega suporte para editar, refatorar, executar e recarregar os aplicativos móveis com eficiência. Esta extensão ao ser instalada executará a instalação do plugin Dart automaticamente, pelo motivo de dependência.

5.1.4 Dependências ou Bibliotecas

Dependências ou bibliotecas “são pacotes que contém códigos que já executam algumas tarefas que você precisa, expondo para uso apenas a chamada da função e o resultado” (MEDIUM, 2022). Assim como as extensões, o objetivo é facilitar o desenvolvimento agregando funcionalidades. Pode-se citar como exemplos, o consumo de

serviço HTTP, captura de telas ou componentes, compartilhamento de informações ou imagens, acesso aos recursos nativos como sensores, câmera, banco de dados, entre outros.

Figura 5.1 - Dependências utilizadas

```
22
23  dependencies:
24    flutter:
25      sdk: flutter
26    http:
27    # Value equality
28    equatable: ^1.0.1
29    # Functional programming thingies
30    dartz: ^0.8.9
31    # Swap the connectivity package for this
32    data_connection_checker: ^0.3.4
33    # Local cache
34    shared_preferences: ^0.5.7+3
35    # Service locator
36    get_it: ^3.1.0
37    # Bloc for state management
38    flutter_bloc: ^3.0.0
39    dio:
40    charts_flutter: ^0.12.0 # source :: https://pub.dev/packages/charts_flutter
41    # https://pub.dev/packages/syncfusion_flutter_charts
42    syncfusion_flutter_charts: ^18.4.49
43    # social_share: ^2.1.1
44    # path_provider: ^2.0.1
45    # share_files_and_screenshot_widgets:
46
47    screenshot: ^0.2.0
48    share: ^0.6.5+4
```

Fonte: Autor

A Figura 5.1 exibe algumas dependências utilizadas na construção da aplicação, listadas a seguir:

- http: versão não informada (resolve automaticamente a versão a partir da compatibilidade com as demais dependências). Biblioteca que disponibiliza recursos para a realização de requisições HTTP. Fornece um conjunto de funções e classes que facilitam o consumo de serviços HTTP. Podemos verificar na linha 26 da Figura 5.1;
- flutter_bloc: versão 3.0.0. Biblioteca de gerenciamento do estado, responsável por notificar o Flutter para que realize uma determinada ação. Assunto será abordado na seção 5.4. Registro pode ser verificado na linha 38 da Figura 5.1;
- syncfusion_flutter_charts: versão 18.4.49. Biblioteca responsável pela criação do gráfico. Recebe os dados, processa e gera o gráfico. Conforme registro na linha 42 da Figura 5.1;
- screenshot: versão 0.2.0. Biblioteca que disponibiliza recurso de captura de tela ou

algum componente (widget) como imagem. Conforme registro na linha 47 da Figura 5.1;

- share: versão 0.6.5+4. Biblioteca com a funcionalidade de compartilhar conteúdo do aplicativo. Disponível para iOS e Android, fornece recurso simples de compartilhar uma mensagem, link ou arquivos locais. O compartilhamento é realizado utilizando a caixa de diálogo padrão da plataforma (sistema). Conforme registro na linha 48 da Figura 5.1;

O site "https://pub.dev/" é o repositório oficial de pacotes disponíveis para aplicativos Flutter e Dart. Além de disponibilizar os recursos, fornece aos desenvolvedores todas as informações necessárias como: instruções de instalação, compatibilidade de sistemas, exemplos de uso, versionamento, registro de alterações e pontuações de popularidade.

5.2 Fonte dos Dados

Conforme descrito no Capítulo 4, Seção 4.1.1, os dados pesquisados pela aplicação são originados das tabelas da SIDRA - Sistema IBGE de Recuperação Automática.

Através da SIDRA, que é uma "ferramenta para disponibilizar os dados das pesquisas e censos realizados pelo IBGE" (AGREGADOS, 2022), as informações são distribuídas em diversas tabelas. Cada tabela corresponde a um indicador sobre dados econômicos, financeiros, sociais e políticos que representam o aspecto de diversas áreas, sendo identificado como: da administração pública, agroindústria, economia, economia agrícola e estatística. Os dados fornecidos podem ser apresentados em níveis de detalhamento geográfico, onde é possível escolher a nível Brasil, nível regional, nível estadual ou municipal.

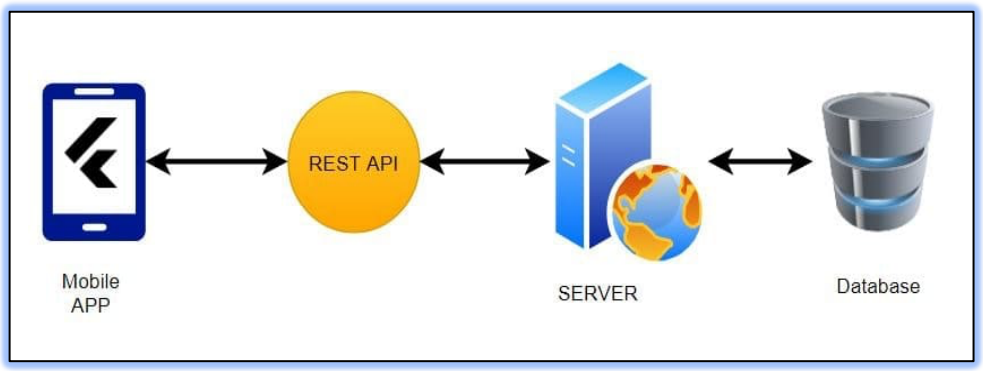
Portanto, tais informações censitárias resultam em mais uma fonte de dados com informações relevantes, pois, conforme apresentação no portal IBGEeduca, "todos os trabalhos realizados pelo IBGE são feitos pensando na nossa missão: retratar o Brasil com informações necessárias ao conhecimento de sua realidade e ao exercício da cidadania.", (IBGEeduca, 2022).

Para obter ou acessar os dados censitários em uma aplicação, conforme demonstrado através da Figura 5.2, é disponibilizada a *API* de dados agregados do IBGE, atualmente na versão 3.0.0. Cada tabela da SIDRA corresponde a um dado agregado da *API*. Portanto, a consulta via *API* retornará dados de um indicador SIDRA pesquisado.

Via *API* de dados agregados do IBGE, o acesso às informações ocorre através de requisições *get*, via *URL – Uniform Resource Locator*, utilizando o serviço de *httpClient*, ou

seja, a requisição é uma consulta realizada a um servidor através de um endereço eletrônico. O servidor recebe a consulta, processa a requisição e retorna os dados. A requisição solicitada, quando for bem sucedida, receberá de retorno um arquivo no formato *JSON*, estrutura descrita resumidamente no final da subseção 3.4. Na aplicação, este arquivo será devidamente processado, conforme planejado e programado.

Figura 5.2 - Requisição de dados via API



Fonte: REST API

A seguir será detalhado o acesso aos dados censitários através da API de dados agregados do IBGE. A seção 5.2.1 apresenta as opções de consulta aos dados por parâmetros, através de uma URL que conterá o endereço eletrônico de acesso. A seção 5.2.2 apresenta parâmetros adotados na aplicação.

5.2.1 Endereço eletrônico de acesso à API

A *API* de dados agregados do IBGE permite opções variadas de consultas, sendo possível realizar a consulta de todos os agregados, assim como de agregados por parâmetros. Ao todo são fornecidos cinco parâmetros, identificados na base de conhecimento como: período, assunto, classificação, periodicidade e nível territorial. No Anexo I é apresentado um resumo de todos os parâmetros e a respectiva *URL* de pesquisa para obter-se o retorno das informações, através de consulta da base de identificadores. Uma descrição resumida dos parâmetros de consulta à base de dados é apresentada a seguir:

- Agregado: A relação de agregados é obtida através do endereço eletrônico https://servicodados.ibge.gov.br/api/v3/agregados, pelo qual obtem-se o retorno via *API* de todos os agregados disponíveis.
- Parâmetro: Para retornar os agregados correspondentes a um determinado parâmetro, deve ser adicionado a *URL* descrita no parágrafo anterior um parâmetro específico. Vejamos dois exemplos, retorno dos agregados de assunto e nível geográfico:

- *Assunto:* para obter-se os agregados que contém o assunto 147, deve-se gerar a URL "https://servicodados.ibge.gov.br/api/v3/agregados?assunto=147"
- *Nível:* para obter-se os agregados associados ao nível geográfico N6, correspondente a município, deve-se gerar a URL "https://servicodados.ibge.gov.br/api/v3/agregados?nivel=N6".

Por agregado, a partir dos parâmetros, é possível realizar a pesquisa de informações específicas como: localidade, metadados, períodos e variáveis. A seguir é apresentado o detalhamento de cada uma dessas opções de pesquisa:

- Localidade: Possibilita consultar as localidades associadas a um agregado. Para consultar a localidade deve ser informado um ou mais parâmetros de nível geográfico. Quando informado mais de um parâmetro, deve-se utilizar o caractere *pipe* como delimitador. Exemplo de pesquisa das localidades do agregado 1705 (IPCA15 - Variação mensal, acumulada no ano, acumulada em 12 meses e peso mensal, para o índice geral, grupos, subgrupos, itens e subitens de produtos e serviços de fevereiro/2012 até janeiro/2020), cujo nível geográfico seja região metropolitana (N7) ou município (N6): https://servicodados.ibge.gov.br/api/v3/agregados/1705/localidades/N7|N6
- Metadados: Possibilita a pesquisa de todos os metadados de um determinado agregado. Metadados são variáveis específicas, referente a um agregado. O Anexo II apresenta a relação completa de metadados. Exemplo para obter-se os metadados do agregado 174 (Pessoas moradoras em domicílios particulares permanentes por situação e instalação sanitária): https://servicodados.ibge.gov.br/api/v3/agregados/174/metadados
- Períodos: Possibilita pesquisar os períodos associados a um determinado agregado. Exemplo para obter-se os períodos disponíveis para o agregado 1715 (Número de municípios, total e os com serviços de manejo de resíduos de construção e demolição, por forma de disposição no solo): https://servicodados.ibge.gov.br/api/v3/agregados/1715/periodos
- Variáveis: A API de agregados versão 3.0.0 possibilita pesquisar um conjunto de variáveis a partir de parâmetros. Duas formas estão disponíveis para essa pesquisa:

- Variáveis por agregado, períodos pesquisados e identificador da variável: Modo de pesquisa que obtém um conjunto de variáveis a partir das informações de agregado, período e variável. Possibilita informar mais de um parâmetro período e variável, utilizando o caracter *pipe*.
  Exemplo para obter os resultados referentes às variáveis do agregado 3847 "População Residente" (pesquisa: Indicadores de desenvolvimento Sustentável, assunto: Agenda 21 Local), período 2009, cujo variável seja 93 - População Residente, tendo a localidade por N3 - Unidade da Federação e estado seja o 23 - Ceará.
  https://servicodados.ibge.gov.br/api/v3/agregados/3847/periodos/2009/variaveis/93?localidades=N3[23]

- Variáveis por agregado e identificador da variável: Modo de pesquisa que obtém um conjunto de variáveis a partir das informações de variável. Neste modo, comparando como o modo descrito anteriormente, não utiliza o período. Também possibilita informar mais de uma variável, utilizando o separador caracter *pipe*._Exemplo para obter os resultados referentes às variáveis do agregado 1712 cujo produto produzido (226) seja abacaxi (4844) no Brasil (BR):
  https://servicodados.ibge.gov.br/api/v3/agregados/1712/variaveis?classificacao=226[4844]&localidades=BR

5.2.2 URL: parâmetros adotados na aplicação

No desenvolvimento da aplicação foi adotado o uso da pesquisa utilizando os parâmetros de variáveis por agregado, períodos, variável e localidade. Para versão da aplicação minimamente viável, o parâmetro período é o único que pode ter mais de um valor, separados por *pipe*. Os demais parâmetros devem ser simples, ou seja, valor único.

Os parâmetros utilizados em uma pesquisa gerada a partir da escolha por parte do usuário de um dado estatístico a ser exibido, utilizam seus respectivos identificadores a seguinte ordem de parâmetros na URL gerada:

1) Agregado: a variável "agregadoId" recebe o identificador do agregado;
2) Período: a variável "periodoId" recebe uma lista de períodos, pois o identificador do período será representado por seu próprio valor;
3) Variável: a variável "variavelId" recebe o identificador da variável;

4) Localidade: a localidade é composta por dois parâmetros. O primeiro representa o nível geográfico e o segundo parâmetro representa a localidade. Então, serão utilizadas duas variáveis. A primeira variável será a "nivelGeograficoId" e a segunda variável será a "localidadeId".

A Figura 5.3 exibe um exemplo da representatividade da URL diretamente no código fonte onde ela é repassada ao método que irá requisitar os dados ao servidor, utilizando na construção da URL os parâmetros agregadoId, periodoId, variavelId, nivelGeograficoId e localidadeId que são coletados a partir da escolha do usuário.

Figura 5.3 - Consulta agregados

```
response = await client.get(Uri.parse(
        'https://servicodados.ibge.gov.br/api/v3/agregados/$agregadoId/periodo
s/$periodoId/variaveis/$variavelId?localidades=$nivelGeograficoId[$localidadeI
d]'));
```

Fonte: Autor

5.3 Estrutura/Arquitetura do Aplicativo

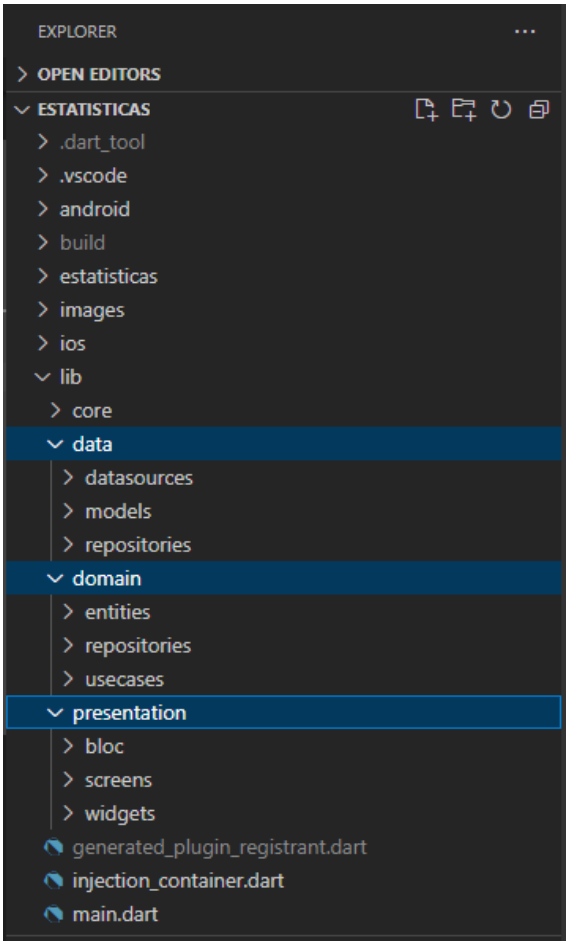
O modelo de arquitetura proposto na aplicação desenvolvida é o *Clean Architecture* (Arquitetura Limpa). O código está distribuído em três camadas. A Figura 5.4 ilustra a visão da estrutura geral de pastas, sendo utilizadas para a aplicação desenvolvida as camadas *Data*, *Domain* e *Presentation*. A ordem de comunicação, pelo fluxo de execução, seguindo a interação do usuário até a obtenção dos dados é: *Presentation*, *Domain* e *Data*. Cada camada possui uma determinada função:

- Presentation (Apresentação): camada responsável pela lógica de telas. Temos nesta camada a interface de usuário (*UI - User Interface*), através da subcamada *Screens*, responsável pela criação das interfaces visuais através dos Widgets, ou seja, monta as telas e componentes; e o *OutPut* que são controles, também conhecida por *ViewModel*, com objetivo de armazenar e gerenciar dados através de requisições de *APIs*, pois converte os inputs do usuário em um tipo esperado. As requisições utilizam Gerenciamento de Estado, recurso para obter os dados via BLoC (Bussines Lógic Component), localizados no subcamada Bloc;
- Domain (Domínio): esta camada separa a nossa aplicação com base nas regras de negócio da empresa, através das entidades (*entities*), e nas regras de negócio da aplicação, através dos casos de uso (*usecases*). Tem a responsabilidade de conectar e gerenciar o fluxo de dados das entidades, conforme propósito das regras de negócio, ou seja, é responsável somente pela execução da regra de negócio através do

repositório (*repositories*), não ocorrendo nenhuma implementação de outros objetos ou serviços;

- Data (Dados): camada responsável pela gerência sobre os dados, ou seja, tem responsabilidade sobre os dados, tais como o acesso, representação, mudança e tratamento. Através da implementação em *Datasources* com reponsabilidade de conexão à fonte de dados. Em *Models* tem a responsabilidade de tratar os dados externos para atender a estrutura de dados do domínio, pois devem controlar e atender a estrutura e formato das entidades. Já em *Repositories* (desta camada *Data*) tem-se a implementação da classe abstrata contida no *Repositories* da camada *Domain* que representa os Casos de Uso.

Figura 5.4 - Visão das camadas da aplicação

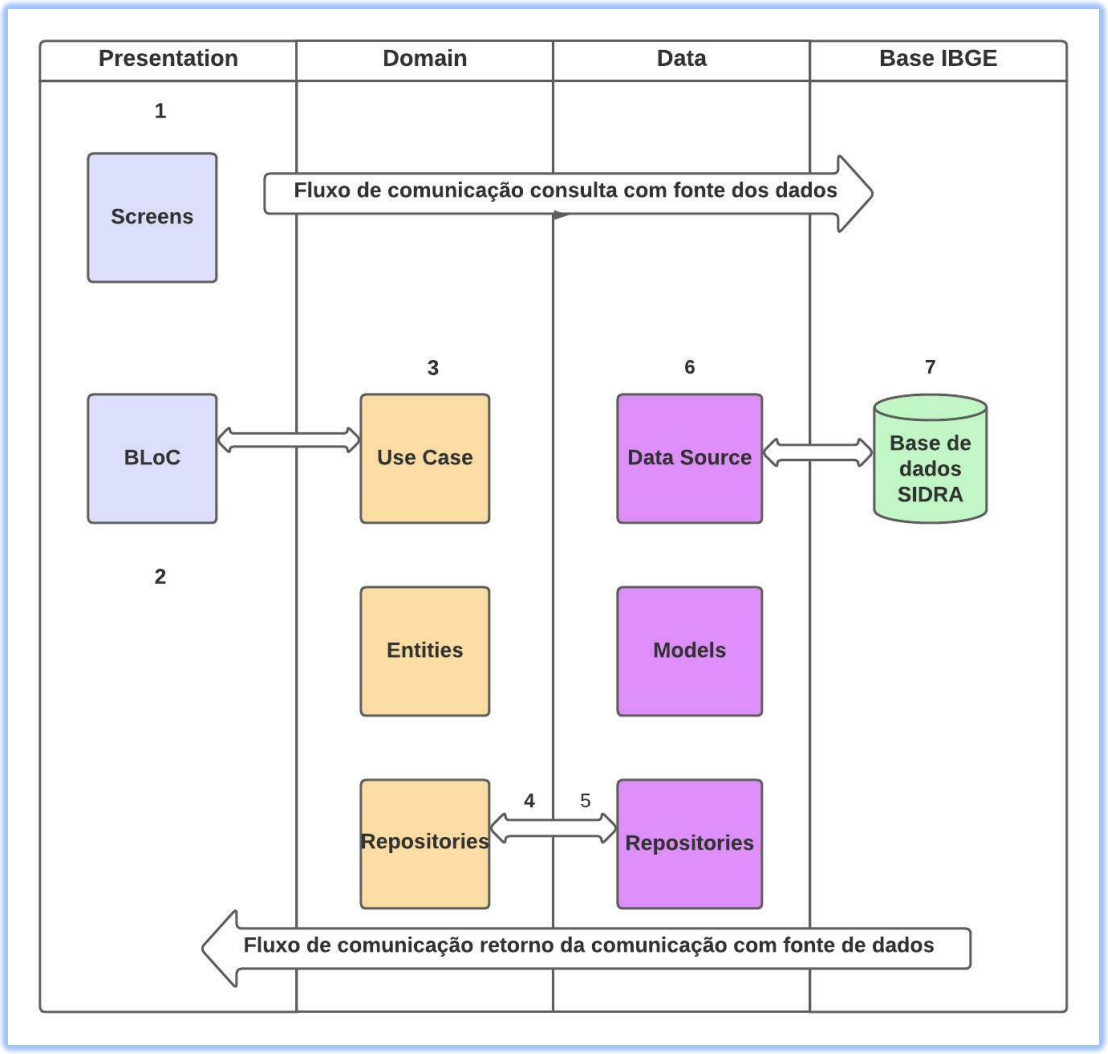


Fonte: Autor

A comunicação entre as camadas, utilizando o conceito do SOLID (que é um conjunto de princípios de desenvolvimento de software, detalhado no Anexo III), ocorre através de contratos intermediários. Um contrato é o acesso ao serviço que está implementado em uma subcamada ou em outra camada. A Figura 5.5 apresenta a visão geral da arquitetura do

aplicativo desenvolvido, organizado em camadas.

Figura 5.5 - Visão geral da Arquitetura



Fonte: Autor

Detalhando a Figura 5.5, o fluxo de comunicação entre as camadas *Presentation* e *Domain* ocorre através do padrão *BLoC* (Bussines Lógic Component) que será detalhado na seção 5.4. A comunicação via *BLoC* inicia a partir de um componente Widget (*Screens* – identificador 1), que dispara um evento. O evento corresponde a um Caso de Uso, ou seja, quando o evento (BLoC - identificador 2) for acionado por um componente, este acionará um Caso de Uso da camada *Domain* (identificador 3). Já a comunicação entre as camadas *Domain* e *Data* segue a caraterística de contrato, pois a camada *Domain* contém o contrato *Repositories* e na camada de *Data* é realizada a implementação de *Repositories*. Portanto, o contrato intermediário é uma assinatura do método sem implementação que contém o nome do método e seus parâmetros e corresponde a um caso de uso, sendo o *Repositories* a ligação ou o canal de comunicação entre as camadas *Domain* e *Data*, conforme identificadores 4 e 5.

Como resultado do evento iniciado, tem-se a consulta do dado pesquisado através de *Data Source* (identificador 6) que estabelecerá comunicação com a base SIDRA do IBGE (identificador 7). Quando a requisição à base de dados retornar, um fluxo contrário é gerado, até que o dado coletado na base de dados chegue no formato de entidade para a camada de apresentação. Esta, por sua vez, reage à mudança de estados gerada pela chegada dos dados à camada de aplicação.

5.4 *BLoC* e Gerenciamento de Estado

O *BLoC* foi um dos primeiros padrões a ser implementado no Flutter, tendo a função de representar as regras de negócio, a fim de gerenciar o estado da aplicação e a comunicação entre os componentes, através de notificações.

Portanto, monitorando a ocorrência dos eventos, a partir do estado de um componente acionado na interface de usuário, o *BLoC* executará um determinado caso de uso, pois é o componente responsável pela comunicação das camadas *Presentation* e *Domain*. O gerenciador, ao identificar alguma alteração, notifica a aplicação da ocorrência, alterando o estado somente daquele componente em função da mudança de estado. As notificações do estado, que permitem à aplicação atualizar as telas e seus componentes, são independentes do sistema utilizado, pois o código de visualização (tela - view) será responsável pelo vínculo com o sistema utilizado.

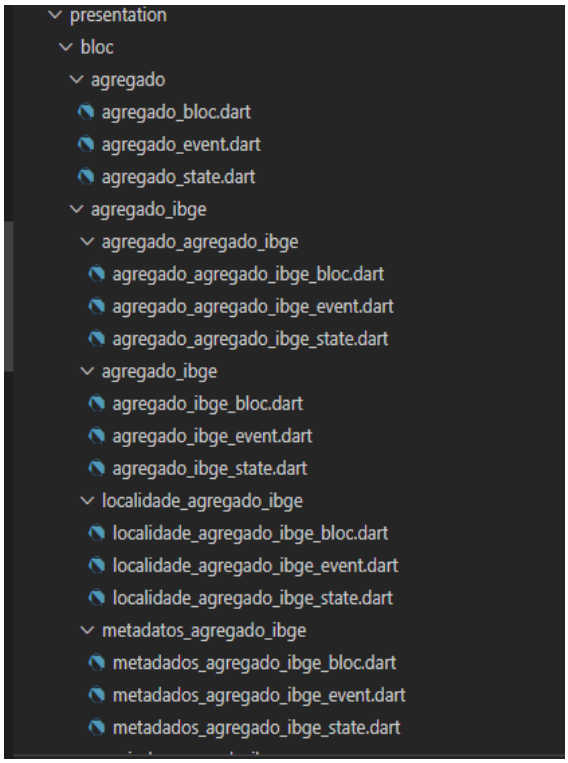
Na aplicação desenvolvida, o *BLoC* é utilizado em cada consulta de um parâmetro até a busca dos dados a serem apresentados ao usuário. A seguir é apresentado um trecho do código fonte do Bloc que gerencia os eventos e mudanças de estados gerados a partir de definição de uma localidade que o usuário deseja pesquisar. O código presente na camada de apresentação (*Presentation*), subcamada *Screens*, é acionado quando o usuário selecionar uma determinada localidade. O código faz o tratamento de mudança de estado gerada por esta funcionalidade, a partir do qual verifica o estado gerado, trata o erro caso a pesquisa tenha tido algum problema (*LocalidadeAgregadoError*), gera um texto “Processando” caso a requisição ainda não tenha retornado (*LocalidadeAgregadoWaiting*), ou senão continua com o processamento caso a pesquisa tenha sido efetuada com sucesso à base de dados do IBGE, atualizando os componentes correspondentes. Repare que toda a lógica do backend fica abstraída graças à separação entre as regras de negócio e a interface proporcionada pelo *BLoC*.

BlocBuilder<LocalidadeAgregadoIbgeBloc, LocalidadeAgregadoIgbeState>(

```
builder: (context5, state5) {
 if (state5 is LocalidadeAgregadoError) {
      return MessageDisplay( message: state5.message,);
     } else if (state5 is LocalidadeAgregadoWaiting) {
      return Text('processando');
     } else if (state5 is LocalidadeAgregadoLoaded) {
      return Column(.....
```

Conforme mencionado na subseção 5.1.4, para o desenvolvimento da aplicação foi utilizada a dependência "flutter_bloc", versão 3.0.0. A partir desta dependência é aplicada o gerenciamento de estado via *BLoC*. A Figura 5.6 apresenta uma visão geral dos BLoC existentes no aplicativo que possuem uma estrutura de arquivos conjunta, onde temos o *BLoC*, *Event* e *State*. Os arquivos com final "_bloc" são as classes que gerenciam as notificações correspondentes aos eventos de entrada e saída (inputs e outputs), gerados pela interação do usuário com a tela ou a partir de retornos de requisições de dados ao *backend*. Os arquivos com final "_state" são as classes que passam informações do estado da apresentação, criando um estado para cada situação em que se quer reagir. Por fim, os arquivos com final "_event" são as classes que contém as estruturas de dados manipuladas a partir de interações do usuário com a tela.

Figura 5.6 - Visão geral do BLoC



Fonte: Autor

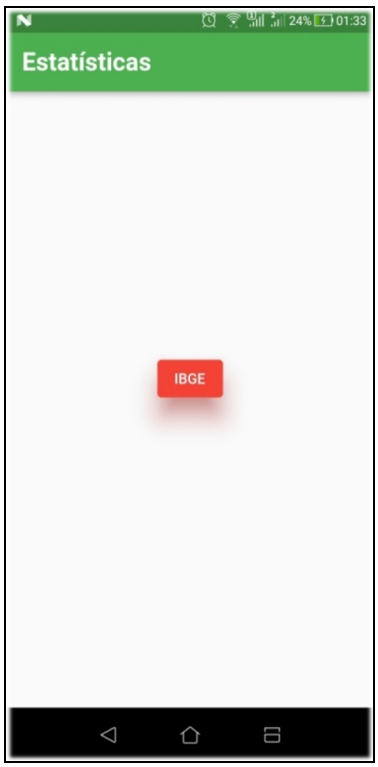
5.5 Aplicação

Nesta seção serão apresentas as telas da aplicação, que estão implementadas na camada de Apresentação (*Presentation*), subcamada *Screens*.

5.5.1 Tela Inicial

Ao iniciar a aplicação, a primeira tela (Figura 5.7) apresentada ao usuário é a tela de seleção da fonte de dados a ser pesquisada, no caso, a única implementada na versão minimamente viável do aplicativo que é a fonte de dados agregados IBGE.

Figura 5.7 - Tela Inicial



Fonte: Autor

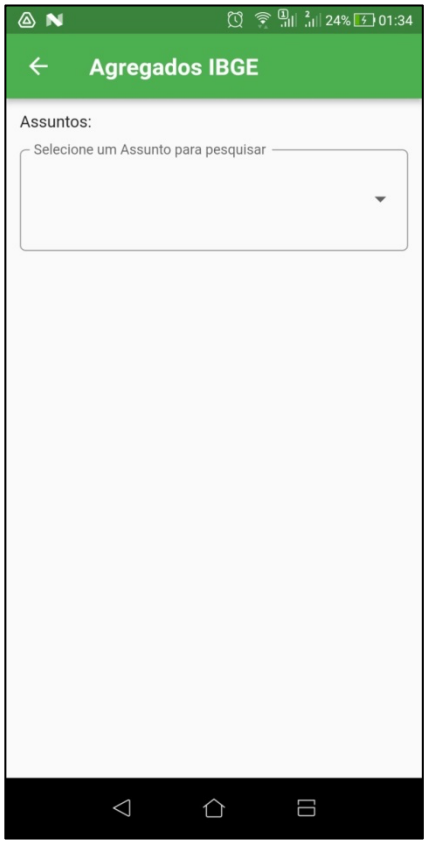
Ao clicar no botão identificado por “IBGE” será apresentada a segunda tela da aplicação.

5.5.2 Tela de seleção dos parâmetros Assunto e Pesquisa

Após a seleção da fonte de dados, através de um clique no botão localizado na tela inicial (Figura 5.7), será apresentada a segunda tela da aplicação (Figura 5.8).

Esta segunda tela é composta por vários componentes de seleção gerados a partir de um *Widget* chamado de *DropdownButton*, através dos quais são selecionados os parâmetros da pesquisa, pois permitem a interação com o usuário.

Figura 5.8- Assunto



Fonte: Autor

Figura 5.9 - Pesquisa

Fonte: Autor

Os componentes de seleção, à medida que são habilitados visualmente, são carregados com os dados de parâmetro correspondente. O primeiro parâmetro a ser habilitado é o assunto (Figura 5.8), corresponde a um agregado do IBGE, conforme mencionado na seção 5.2.1. Ao selecionar um determinado assunto, conforme demonstrado na Figura 5.8, será habilitado o parâmetro pesquisa (Figura 5.9), contendo informações associadas ao assunto.

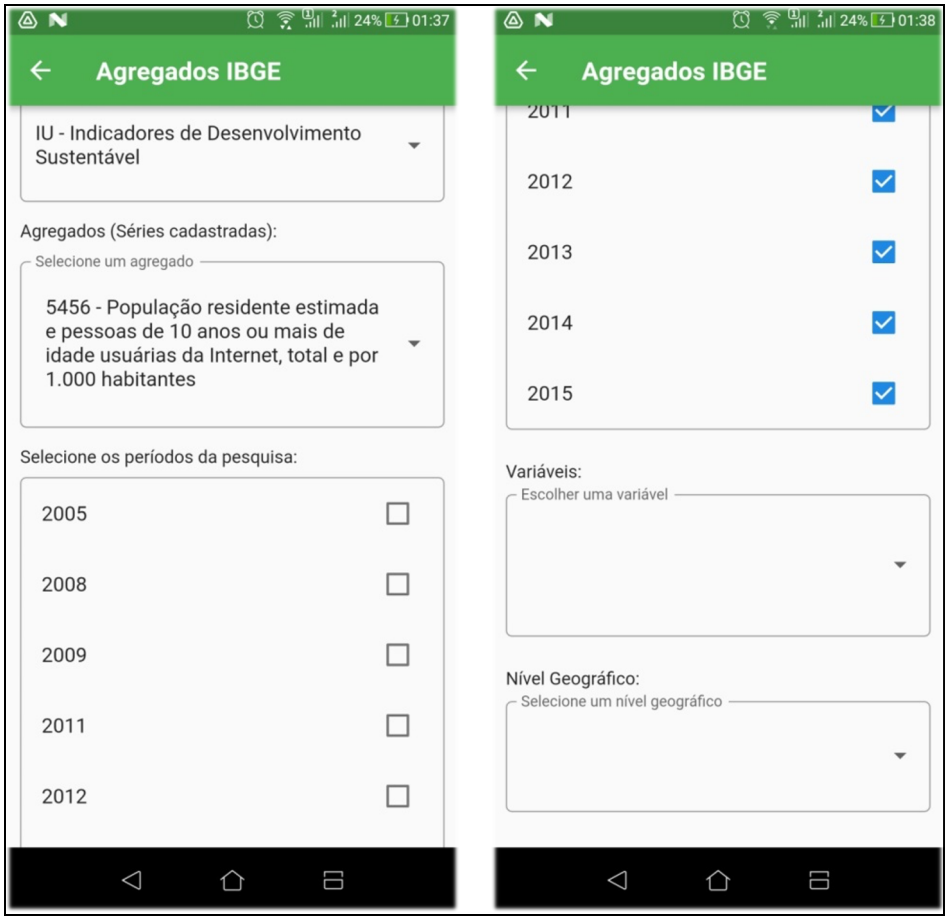
5.5.3 Funcionalidade de seleção dos parâmetros Agregados e Períodos

A partir da seleção dos dois parâmetros iniciais, assuntos e pesquisas, será apresentada a caixa de seleção de agregados relacionados com o assunto selecionado no primeiro parâmetro, conforme demonstrado na Figura 5.10.

Selecionando o terceiro parâmetro agregado, será apresentado o quarto parâmetro, uma caixa de múltipla de seleção de períodos, correspondente aos períodos associados ao assunto, através da qual é possível selecionar um, mais de um, ou todos os períodos.

Além da caixa de seleção de período, também será habilitada caixa de seção de variáveis e de nível geográfico, parâmetros finais necessários para compor a *URL* e realizar a pesquisa final.

Figura 5.10- Seleção Agregados e Seleção de Período



Fonte: Autor

5.5.4 Funcionalidade de Seleção dos parâmetros Variáveis, Nível Geográfico e Localidade

Os parâmetros finais, variável e nível geográfico, ao serem habilitados, carregam seus dados. Ao serem selecionados são utilizados para filtrar e apresentar o último parâmetro, a localidade relacionada ao nível geográfico.

Ao término da seleção de todos os parâmetros, será apresentado o botão “Gerar Gráfico” (Figura 5.11), no final da tela, através do qual, ao ser clicado, apresentará o gráfico resultante em uma nova tela, através da qual será possível compartilhar a informação gráfica utilizando os recursos disponibilizados pelo Sistema Android.

Figura 5.11 - Seleção de Variáveis, Nível Geográfico e Localidades.

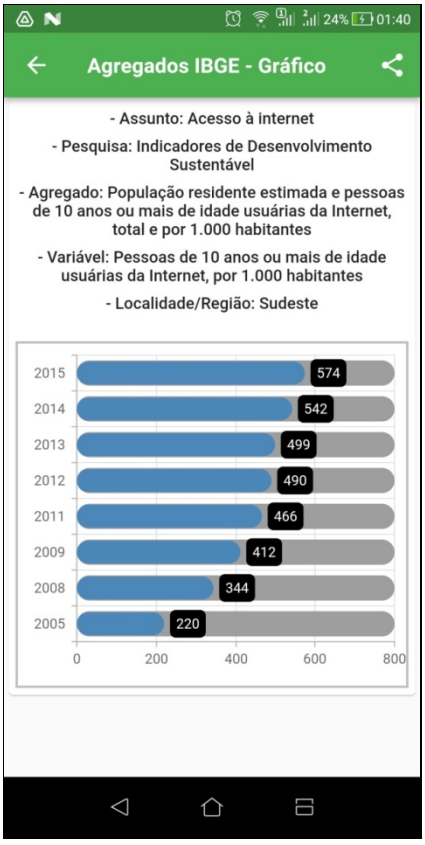
Fonte: Autor

5.5.5 Tela de Gráfico Resultante e Opções de Compartilhamento

A terceira e última tela da aplicação, demonstrada através da Figura 5.12, é o resultado da pesquisa realizada com sucesso, a partir da seleção de parâmetros, obtendo informações da base de dados Agregados do IBGE, processando-os e gerando os dados finais que são usados para a construção do gráfico.

A tela final apresenta na primeira parte ou parte superior as informações referentes aos parâmetros utilizados na composição da pesquisa. Na segunda parte ou parte inferior, é apresentado um gráfico de barras com os valores por período pesquisado. Na barra de título, no canto superior direito, temos o símbolo de compartilhamento, o qual utiliza recursos do sistema para a realização a tarefa de compartilhar a tela com os dados e o gráfico por e-mail, WhatsApp, Facebook, entre outras opções.

Figura 5.12- Tela Resultado Gráfico



Fonte: Autor

Figura 5.13- Tela Compartilhar

Fonte: Autor

6 CONCLUSÃO E TRABALHOS FUTUROS

A evolução tecnológica dos dispositivos móveis e da rede de dados para telefonia móvel, principalmente com o surgimento do sistema Android em 2003, criado com foco em sistemas para smartphone, tornou o smartphone popular e acessível a todas as pessoas.

Com a facilidade de acesso aos dispositivos e com a velocidade de conexão rápida, possibilitou a criação e a disponibilização de serviços de acesso às informações de modo ativa, ou seja, serviço de conexão permanente chamada de “OnLine”.

Aproveitando-se desta evolução tecnológica, este trabalho teve como objetivo desenvolver uma aplicação para dispositivos móveis que busca informações da base de dados do IBGE, uma fonte confiável, realiza o processamento das informações recebidas e gera gráficos que pode ser salvos ou compartilhados pelo usuário.

A referida aplicação, desenvolvida para o sistema operacional Android, foi implementada com base na Arquitetura Limpa, padrão de arquitetura de software com camadas bem definidas e utilizando BLoC para a gerência de estado a fim de separar o código da regra de negócio do código de tela.

Apesar da linguagem de programação Dart ser semelhante à linguagem C e a programação em Flutter não ser considerada difícil, pode-se concluir que, apesar das dificuldades encontradas, foi possível desenvolver a aplicação proposta através de uma POC – Prove de Conceito que alcançou o objetivo proposto.

Durante o desenvolvimento do trabalho, apesar de ser uma tecnologia nova, encontramos na Internet muitas informações, pois há a disponibilização de material pelo canal oficial do Flutter. A maior dificuldade encontrada na realização do trabalho está em relação à Arquitetura Limpa, pois o entendimento e a aplicação do conceito foi demorado. No decorrer do estudo fora muito pesquisado, vários vídeos foram assistidos, e todos os autores relatam que não é de fácil entendimento, sendo necessário muito estudo, prática e tempo até seu entendimento, sendo que a arquitetura limpa não tem um padrão fixo, pois deve ser planejada a cada caso.

Mas apesar das dificuldades encontradas com a Arquitetura Limpa, é um conceito que traz enormes vantagens no desenvolvimento de aplicações, atingindo o objetivo proposto por seu idealizador, Robert Cecil Martin (Uncle Bob), que é padronizar e organizar o código a fim de favorecer a sua reusabilidade.

Apesar de uma POC – Prova de Conceito ser alcançada, algumas histórias de usuários não foram implementadas, portanto, este trabalho deve continuar, podendo-se citar algumas

melhorias a serem implementadas e novas funcionalidades em trabalhos futuros: i) gerenciamento do histórico de pesquisa; ii) opção de selecionar outro tipo de gráfico: pizza, linha, área, etc.; iii) implementar novas fontes de dados; iv) salvar alguns dados localmente para que seja possível realizar a pesquisa mesmo quando não tiver conexão à Internet; v) gerenciamento da conta de usuário.

REFERÊNCIAS BIBLIOGRÁFICAS

ADVFN. **Indicadores Econômicos**. Disponível em: https://br.advfn.com/indicadores Acesso em: 07 jan. 2022.

AGREGADOS. **API de dados agregados do IBGE**. Disponível em: https://servicodados.ibge.gov.br/api/docs/agregados?versao=3#api-Variaveis-agregadosAgregadoPeriodosPeriodosVariaveisVariavelGet
Acesso em 10 jan. 2022.

AGREGADOS-METADADOS. **Metadados da API de dados agregados do IBGE**. Disponível em: https://servicodados.ibge.gov.br/api/docs/agregados?versao=3#api-Metadados Acesso em 20 jan. 2022.

ALURA. **Dart: primeiros passos com a linguagem.** Disponível em: https://www.alura.com.br/conteudo/dart-primeiro-passos?gclid=CjwKCAiAxJSPBhAoEiwAeO_fP9hj3VmVjmEBFAoXshPOCqtrsRnZntmrutiu4dhy8QPXN_28JeMBxBoC-WEQAvD_BwE
Acesso em: 16 jan. 2022.

ANDRADE, Kleber de Oliveira. **Introdução a Linguagem de Programação Dart.** 04 ago. 2019. Disponível em: https://medium.com/flutter-comunidade-br/introdução-a-linguagem-de-programação-dart-b098e4e2a41e.
Acesso em: 15 jan. 2022,

ARQUITETURA DE SOFTWARE. In: WIKIPÉDIA: a enciclopédia livre. 24 nov. 2016. Disponível em: https://pt.wikipedia.org/wiki/Arquitetura_de_software Acesso em: 17 nov. 2022

BRASIL**. Projeto de Lei 2630, Lei Brasileira de Liberdade, Responsabilidade e Transparência na Internet.** Disponível em: https://legis.senado.leg.br/sdleg-getter/documento?dm=8110634&disposition=inline Acesso em: 20, nov. 2021.

CASTRO, Alexandre. **Como Pesquisar Dados IBGE.** 05 mar. 2018. Disponível em: https://aredeurbana.com/2018/03/05/como-pesquisar-dados-no-ibge/
Acesso em: 18 dez. 2021.

DART. In: WIKIPÉDIA: a enciclopédia livre. 20 abr. 2013.
Disponível em: https://pt.wikipedia.org/wiki/Dart_(linguagem_de_programação) Acesso em: 12 jan. 2022.

FABRO, Clara. **O que é API e para que serve? Cinco perguntas e respostas.** 15 jun. 2020. Atualizado em: dez 2020. Disponível em: https://www.techtudo.com.br/listas/2020/06/o-que-e-api-e-para-que-serve-cinco-perguntas-e-respostas.ghtml Acesso em: 17 dez. 2021.

FERREIRA, Mauro. Coletando dados de Regiões e Estados com Python. 28 jan. 2021. Disponível em: https://pt.linkedin.com/pulse/coletando-dados-de-regi%C3%B5es-e-estados-com-python-mauro-ferreira. Acesso em: 17 dez. 2021.

FILHO, Wilson de Oliveira. **Aplicativos móveis na gestão pública: Interação**

Entre governo e cidadão. Orientador: José Roberto. 2019. 47 f. Monografia (Pós-Graduação Lato Sensu) – Universidade Candido Mendes / AVM, 2019.

FLUTTER. In: WIKIPÉDIA: a enciclopédia livre. 9 nov. 2018.
Disponível em: https://pt.wikipedia.org/wiki/Flutter Acesso em: 10 jan. 2022.

FLUTTER. **Camadas**.
Disponível em: https://flutterparainiciantes.com.br/arquitetura/. Acesso em: 23 dez. 2022.

GEOSPATIAL Big Data. **MundoGEO**, 2013. Disponível em:
https://mundogeo.com/2013/06/05/geospatial-big-data/ Acesso em: 21 nov. 2021.

IBGE, Instituto Brasileiro de Geografia e Estatística. In: WIKIPÉDIA: a enciclopédia livre.12 dez. 2018. Disponível em:
https://pt.wikipedia.org/w/index.php?title=Instituto_Brasileiro_de_Geografia_e_Estat%C3%ADstica&action=history. Acesso em: 10 dez. 2021.

IBGE, Instituto Brasileiro de Geografia e Estatística. **Missão Institucional**. Disponível em:
https://www.ibge.gov.br/acesso-informacao/institucional/o-ibge.html.
Acesso em: 28 dez. 2022.

IBGEeduca, **O IBGEeduca é o portal do IBGE**. Disponível em:
https://educa.ibge.gov.br/20592-sobre-o-ibge.html
Acesso em 15 set. 2022.

IDC's Global DataSphere Forecast Shows Continued Steady Growth in the Creation and Consumption of Data. **IDC**, 2020. Disponível em:
https://www.idc.com/getdoc.jsp?containerId=prUS46286020
Acesso em: 22 nov. 2021.

ÍRIS, Laboratório de Inovação e Dados do Governo do Ceará; AWS, Institute Social Good Brasil. **A Era dos Dados Para o Setor Público: Uma Nova Cultura Organizacional Analítica**. Ceara, 2021. Evento em: 22 set. 2021. Disponível em:
https://irislab.ce.gov.br/?smd_process_download=1&download_id=6257
Acesso em: 02 dez. 2021. Livro Digital.

MARTIN, Robert C.. **Arquitetura Limpa: O Guia do Artesão para Estrutura e Design de Software.** Rio de Janeiro: Alta Books, 2020. E-book Kindle.

MARTIN, Robert C. **The Clean Architecture**. 12 ago. 2012. Disponível em:
https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html. Acessado em: 20 jan. 2022. Citado na página 30.

MEDIUM**. Dependências Flutter e formas de hospedagem**. Disponível em:
https://medium.com/digitalproductsdev/depend%C3%AAncias-flutter-e-formas-de-hospedagem-56e5ffeebab7
Acesso em: 19 set. 2022.

MORAIS, Misael Elias de. **Flutter -Programação para Android & iOS.** Campina Grande, PB: Ed. do Autor, 2020. E-book Kindle.

ISBN 978-65-00-09999-7

NEGRI, Patrick. Saiba o que é flutter e quais são as suas vantagens. **Blog investigação filosófica.** Escrito em: 28 jul.2020. Atualizado em: 30 jul. 2021, Disponível em: https://www.iugu.com/blog/o-que-e-flutter. Acesso em: 10 dez. 2021.

PAINEL de checagem de Fake News. C**NJ - Conselho Nacional de Justiça**, 2019. Disponível em: https://www.cnj.jus.br/programas-e-acoes/painel-de-checagem-de-fake-news/. Acesso em: 22, nov. 2021.

PENA, Rodolfo F. Alves. **Era da Informação**. Mundo da Educação. Disponível em: https://mundoeducacao.uol.com.br/geografia/era-informacao.htm. Acesso em: 20 nov. 2021.

REDHAT. **O que é API?** 31 out. 2017.
Disponível em: https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces. Acesso em 23 jan. 2022.

REHKOPF, Max. Histórias de usuários com exemplos e um template. ALTASSIAN, 2022.
Disponível em: https://www.atlassian.com/br/agile/project-management/user-stories#:~:text=Uma%20hist%C3%B3ria%20de%20usu%C3%A1rio%20%C3%A9,do%20usu%C3%A1rio%20final%20ou%20cliente.
Acesso em: 28 dez. 2022.

REST API. **6 THINGS ABOUT HTTP REQUEST IN DART FOR BEGINNERS**
Disponível em: https://navoki.com/6-things-about-http-request-in-dart-for-beginners/. Acesso em: 01 out. 2022.

STORE, Google Play. **IBGE.** 2022. Disponível em: https://play.google.com/store/apps/details?id=br.gov.ibge&showAllReviews=true. Acesso em: 06 jan. 2022.

STORE, Google Play. **InfoMoney.** 2022. Disponível em: https://play.google.com/store/apps/details?id=com.cedrotech.infomoney Acesso em: 07 jan. 2022.

STORE, Google Play. **Anatel Serviços Móvel.** 2022. Disponível em: https://play.google.com/store/apps/details?id=br.com.mais2x.anatelsm
Acesso em: 05 jan. 2022.

STORE, Google Play. **Relação acesso a rede de voz por operadora.** 2022. Disponível em: https://play-lh.googleusercontent.com/jVTYDile8h2f034gExMRkpoEsCKfmE_s5BwSainBESzNh4lc7D7dW84xzHUV158KSZA=w1366-h600-rw
Acesso em: 05 jan. 2022.

VALENTE, Marco Tulio. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade.** Ed. do Autor, 2020. E-book Kindle. ISBN: 978-65-00-00077-1.

ANEXO I - Base de Identificadores

A API de dados agregados do IBGE disponibiliza recurso para a pesquisa de todos os identificadores. Através do endereço eletrônico https://servicodados.ibge.gov.br/api/docs/agregados?versao=3#api-acervo é possível visualizar a relação de identificadores, que são listados em duas colunas, a primeira corresponde ao código identificador e a segunda coluna à descrição.

Na presente data de elaboração do TCC, os identificadores existentes são: período, assunto, classificação, periodicidade, nível e variável.

Período: constam 883 períodos cadastrados. Podem identificar ano, semestre, trimestre, ou mês e ano. Vejamos alguns exemplos.

CÓDIGO	PERÍODO
1872	1872
1890	1890
1900	1900
1920	1920
201801	1º Semestre de 2018
201901	1º Semestre de 2019
202001	1º Semestre de 2020
202101	1º Semestre de 2021
198701	1º Trimestre de 1987
198801	1º Trimestre de 1988
198901	1º Trimestre de 1989
199001	1º Trimestre de 1990
199101	1º Trimestre de 1991
199201	1º Trimestre de 1992
2000	2000
2001	2001
2002	2002
2003	2003
2004	2004
201802	2º Semestre de 2018
201902	2º Semestre de 2019
202002	2º Semestre de 2020
202102	2º Semestre de 2021
200504	Abril de 2005
200604	Abril de 2006
200704	Abril de 2007
200804	Abril de 2008
200904	Abril de 2009
201004	Abril de 2010

201104	Abril de 2011
201204	Abril de 2012
201206	ABR MAI jun/12
201210	AGO SET out/12
201310	AGO SET out/13
201410	AGO SET out/14

Assunto: constam 240 assuntos distintos. Relação resumida da composição de código e respectivo assunto.

CÓDIGO	ASSUNTO
148	Abastecimento de água
110	Acesso a esgotamento sanitário
147	Acesso à internet
107	Acesso a serviço de coleta de lixo doméstico
146	Acesso a serviços de telefonia
109	Acesso a sistema de abastecimento de água
316	Acidentes
221	Acidentes, violência e segurança
128	Adequação de moradia
245	Agenda 21 Local
208	Aglomerado subnormal
88	Agricultura Familiar
78	Agroindústria rural
296	Água potável e saneamento
163	Alimentação
188	Alojamento e/ou alimentação
181	Antropometria
49	Aquisição
268	Doenças crônicas
20	Domicílios
27	Educação
302	Educação de qualidade
156	Emigração
91	Emissões de origem antrópica dos gases associados ao efeito estufa
173	Emprego
198	Empresarial não financeiro
12	Empresas
54	Empresas e outras organizações
298	Energia limpa e acessível
229	Equipamentos de ensino
304	Erradicação da pobreza
127	Escolaridade
215	Escolaridade materna e paterna
149	Esgotamento sanitário

157	Etnia
143	Existência de conselhos municipais
75	Extração vegetal
19	Famílias
18	Fecundidade
77	Floricultura
58	Fome zero e agricultura sustentável
235	Frequência escolar
42	Fundações e Associações sem fins lucrativos
145	Gasto público com proteção ao meio ambiente
144	Gastos com Pesquisa e Desenvolvimento PD
295	Geral
153	Gestão municipal de saneamento básico
134	Grau de endividamento
222	Higiene e saúde bucal
76	Horticultura
209	Hospedagem
273	Idosos
130	Igualdade de gênero
293	Imagem corporal
224	Imagem corporal, atitude em relação ao peso corporal e estado nutricional
122	Imunização contra doenças infecciosas infantis
234	Incidência de AIDS
115	Índice de Gini da distribuição do rendimento
200	Índice de Preços ao Produtor
82	Índices de preços
87	Indústria
303	Indústria, inovação e infraestrutura
195	Informação e comunicação
86	Informações básicas municipais

Classificação: constam 1391 classificações. Abaixo relação da composição de código e respectiva classificação.

CÓDIGO	CLASSIFICAÇÃO
588	Acessibilidade possível na maior parte das vias internas
957	Acesso à Internet por telefone móvel celular para uso pessoal
681	Acesso a televisão por assinatura
1192	Ações para evitar ou minimizar os danos causados pela seca
12236	Adequação da moradia
12542	Agente financeiro responsável pelo financiamento
1260	Agressor da única ocorrência ou a mais grave
12896	Agricultura familiar
59	Alfabetização
11563	Alfabetização da pessoa responsável pelo domicílio

147	Alguma vez o escolar já usou alguma droga, tais como: maconha, cocaína, crack, cola, loló, lança perfume, ecstasy ou outra
12682	Categorias de doenças
12920	Categorias de risco
9	Categorias de uso
492	Categorias dos estabelecimentos de hospedagem
43	Categorias profissionais
831	Classificação de montanha (Kapos)
12762	Classificação Nacional de Atividades Econômicas (CNAE 2.0)
11939	Classificação Nacional de Atividades Econômicas (CNAE)
12262	Cobertura de plano de saúde
1057	Cobertura de procedimentos
11982	Cobrança pelo serviço de abastecimento de água
1268	Com quem podia contar em momentos bons ou ruins
191	Combustível usado para cozinhar
828	Combustível utilizado na preparação de alimentos
12515	Composição da família
12655	Composição das aves em 31/12
12627	Composição do efetivo de bovinos nos estabelecimentos agropecuários com mais de 50 cabeças em 31/12
11550	Concentração de flúor na água
517	Conclusão de outro curso superior de graduação
883	Conclusão do curso de qualificação profissional
581	Conclusão do curso que o cônjuge do produtor frequentou
840	Conclusão do curso que o produtor frequentou
896	Conclusão do curso técnico de nível médio ou normal (magistério)
577	Condição da população em relação a idade
527	Condição de aposentada ou pensionista de instituto de previdência oficial no mês de referência
12393	Condição de aposentado no mês de referência
11559	Condição de aquisição do domicílio
344	Condição de atendimento
90	Condição de atividade
12251	Condição de atividade e condição de ocupação
12049	Condição de atividade e de ocupação na semana de referência
12244	Condição de atividade e posição na ocupação da pessoa responsável pela família
12243	Condição de atividade e seção de atividade da pessoa responsável pela família
12056	Condição de atividade na semana de referência
12495	Condição de atividade na semana de referência da pessoa de referência da família
12496	Condição de atividade na semana de referência das pessoas de referência dos domicílios

521	Condição de contribuição para instituto de previdência oficial em qualquer trabalho
526	Condição de contribuição para instituto de previdência oficial no trabalho principal
536	Condição de contribuição para instituto de previdência oficial no trabalho principal e em qualquer trabalho
12257	Condição de convivência
12494	Condição de economicamente ativa e de ocupada na semana de referência
426	Condição de estudante
675	Condição de estudante e rede de ensino
499	Condição de falar língua indígena no domicílio
385	Condição de indígena
561	Condição de morar com mãe e/ou pai
11284	Condição de ocupação
63	Condição de ocupação do domicílio
3704	Condição de presença

Periodicidade: constam 5 periodicidades disponíveis. Abaixo relação da composição de código e descrição da periodicidade.

CÓDIGO	PERIODICIDADE
P1	Anual
P8	Semestral
P9	Trimestral
P5	Mensal
P13	Trimestral móvel

Nível Geográfico: constam 46 Níveis Geográficos disponíveis:

CÓDIGO	NÍVEL GEOGRÁFICO
N15	Aglomeração Urbana
N17	Aglomerado Subnormal
N20	Área de Divulgação da Amostra para Aglomerados Subnormais
N1105	Área de influência PNSB
N18	Área de Ponderação
N23	Arranjo Populacional
N102	Bairro
N123	Bioma
N1	Brasil
N1100	Brasil, sem especificação de Unidade da Federação
N130	Capital/não capital de Unidade da Federação
N71	Categoria Metropolitana
N124	Corpo d'água

N10	Distrito
N1102	Estrangeiro
N2	Grande Região
N122	Grande Região PIMES
N1101	Ignorado
N8	Mesorregião Geográfica
N9	Microrregião Geográfica
N6	Município
N127	Núcleo de desertificação
N101	País do Mercosul, Bolívia e Chile
N128	Praia
N70	Recortes Metropolitanos
N25	Região Geográfica Imediata
N24	Região Geográfica Intermediária
N121	Região Hidrográfica
N14	Região Integrada de Desenvolvimento
N7	Região Metropolitana
N13	Região Metropolitana e Subdivisão
N132	Semiárido
N133	Semiárido de Unidade da Federação
N72	Subcategoria Metropolitana
N11	Subdistrito
N125	Terra Indígena por Unidade da Federação
N129	Território da Cidadania
N29	Território de Identidade
N1103	Total
N110	Total das áreas PME
N103	Total das áreas POF
N22	Total dos municípios das capitais
N21	Total dos municípios das capitais da Grande Região
N3	Unidade da Federação
N1104	Unidade da Federação, sem especificação de Município
N111	Unidade Federativa do Mercosul, Bolívia e Chile

Variável: constam 8967 variáveis disponíveis. Vejamos um resumo.

CÓDIGO	VARIÁVEL
6596	(+) Cessão de ativos não financeiros não produzidos (aquisições líquidas)
937	(+) Outras transferências correntes (líquidas recebidas do exterior)
935	(+) Rendas de propriedade (líquidas recebidas do exterior)
934	(+) Salários (líquidos recebidos do exterior)
942	(+) Transferências de capital (líquidas recebidas do exterior)
943	(=) Capacidade / necessidade líquida de financiamento
940	(=) Poupança bruta
936	(=) Renda nacional bruta
938	(=) Renda nacional disponível bruta

10477	Adições
2404	Agrotóxicos em linha de comercialização
451	Água e esgoto
10104	Ajuda Oficial ao Desenvolvimento (AOD)
10112	Alteração dos ecossistemas aquáticos ao longo do tempo
467	Altura
871	Aluguéis de imóveis
872	Aluguéis de máquinas, equipamentos e veículos
647	Aluguéis e arrendamento de veículos, máquinas e equipamentos
1840	Aluguéis, leasing e arrendamento de imóveis, máquinas, equipamentos e veículos
1567	Aluguel de imóveis
2269	Aluguel/locação de filmes
284	Animais abatidos
2235	Animais tosquiados
6839	Aquíferos transfronteiriços com acordo operacional
180	Aquisição (monetária) alimentar domiciliar per capita anual
1207	Aquisição alimentar domiciliar per capita anual
1314	Aquisições

ANEXO II - Relação de Metadados (AGREGADOS-METADADO, 2022)

- **URL**: página no portal da SIDRA;
- **Pesquisa**: pesquisa que pertence o agregado;
- **Assunto**: assunto informado pelo agregado;
- **Periodicidade**: frequência da coleta dos dados do agregado, identificado o período inicial e final. Conforme descrito em Anexo I, temos 883 períodos cadastrados. Podem identificar ano, semestre, trimestre, ou mês e ano;
- **Nível Territorial**: especifica o nível territorial abrangido pelo agregado, possuindo três níveis de classificação territorial:
    - **Administrativo**: "Quando o agregado abranger divisão político-administrativa do Brasil. Pode assumir os seguintes valores: N1 (Brasil), N2 (Região), N3 (Unidade da Federação), N8 (Mesorregião), N9 (Microrregião), N7 (Região metropolitana), N6 (Município), N10 (Distrito), N11 (Subdistrito), N102 (Bairro), N15 (Aglomeração urbana), N14 (Região Integrada de Desenvolvimento), N13 (Região metropolitana e subdivisão)";
    - **Especial**: "Quando o agregado abranger divisão de natureza especial. Pode assumir os seguintes valores: N17 (Aglomerado subnormal), N23 (Arranjo populacional), N101 (País do Mercosul, Bolívia e Chile), N104 (Argentina), N105 (Uruguai), N106 (Paraguai), N107 (Departamento - Paraguai), N108 (Departamento - Uruguai), N109 (Província), N111 (Unidade Federativa do Mercosul, Bolívia e Chile), N123 (Bioma), N124 (Corpo d'água), N125 (Terra indígena), N126 (Unidade de Conservação Ambiental), N127 (Núcleo de desertificação), N128 (Praia), N129 (Territórios da cidadania), N131 (Amazônia Legal), N132 (Semiárido), N133 (Semiárido de Unidade da Federação), N134 (Amazônia Legal de Unidade da Federação)";
    - **IBGE**: "Quando o agregado abranger divisão específica do IBGE. Pode assumir os seguintes valores: N18 (Área de Ponderação), N19 (Área de Ponderação Recalculada), N20 (Área de Divulgação da Amostra para Aglomerados Subnormais), N21 (Total dos municípios das capitais da Grande Região), N22 (Total dos municípios das capitais), N103 (Total das áreas - POF), N110 (Total das áreas - PME), N122 (Grande Região - PIMES), N130 (Capital / Não Capital

de Unidade da Federação), N1100 (Brasil, sem especificação de Unidade da Federação), N1101 (Ignorado), N1102 (Estrangeiro), N1103 (Total), N1104 (Unidade da Federação, sem especificação de Município), N1105 (Área de influência - PNSB)".

- **Variáveis**: especifica qual dimensão pode ser sumarizada ou não. E os valores possíveis são o nível territorial ou período. A sumarização, em relação ao período, especifica se a soma de todos os meses corresponde ao valor anual. E em relação ao nível territorial, especifica se a soma das unidades da federação corresponde ao valor Brasil;
- **Classificação**: agrupamento hierárquico das categorias. Lista da categoria macro para micro. Exemplo: regiões do Brasil - Brasil -> Grandes regiões -> Unidades da Federação (estados) -> Mesorregiões -> Microrregiões -> Municípios - ou logicamente.

ANEXO III - SOLID

SOLID – É um acrônimo ou a junção de cinco princípios de desenvolvimento de software.

S - Princípio da responsabilidade única (SRP)

O - Princípio do aberto\fechado (OCP)

L - Princípio da substituição de Liskov (LSP)

I – Princípio da separação de interfaces (ISP)

D – Princípio da inversão de dependência (DIP)

S - DART & SOLID - Princípio da responsabilidade única (SRP)

Uma classe deve ter um único motivo para mudar. E deve ter apenas uma responsabilidade por ator (ator, usuário, etc.).

Exemplo: uma determinada classe de conta corrente que trata da conta corrente e de salvar dados em uma base. Se esta classe possuir dados não pertinentes à conta corrente, está errado, pois não faz sentido à conta corrente ter dados tipo de usuário, senha e do DB para conexão ao banco. Aqui fere o princípio da responsabilidade única. Neste caso, a classe deve conhecer somente o domínio dela que são os dados da conta.

O - DART & SOLID - Princípio do Aberto\Fechado (OCP)

Afirma que as nossas classes devem ser suscetíveis a extensão, porém, elas não devem ser modificadas.

Devem ser abertas para extensão e fechadas para modificação.

Isso quer dizer que não vamos pegar uma classe que já está em produção, testada e validada, para fazer uma modificação, por exemplo, para adicionar um IF ou testar um caso que vai acontecer.

Exemplo: temos uma classe pagamento onde faz pagamento de boleto e imposto. Mas agora tem que adicionar o pagar cartão. Teria que alterar a classe.

Para não ferir o princípio, devemos criar a classe abstrata pagamento para tratar somente o pagamento.

Faz a extensão para pagamento de boleto e imposto. Ao criar nova exigência, tipo pagamento de cartão, cria a nova classe de pagar cartão e estende o pagamento.

L - DAT & SOLID - Princípio da substituição de Liskov (LSP)

Esse princípio foi cunhado pela Barbara Liskov. É um princípio matemático e este

princípio diz que seus subtipos devem ser substituíveis por seus tipos base.

Na prática, quando tivermos uma classe conta e tiver especializações desta classe, as especializações devem ser capazes de se alternar sem que quebrem o sistema.

Exemplo: temos uma classe conta que sabe depositar, transferir e realizar empréstimo. Temos duas classes, conta corrente e conta poupança que implementam a classe conta. Na conta poupança temos duas exceções, transferir e realizar empréstimo não faz. Então a classe conta poupança que implementa à classe conta, devolvem duas exceções, uma para transferir e outra para realizar empréstimo.

I - DART & SOLID - Princípio da segregação de interfaces (ISP)

Princípio diz que ter muitas interfaces específicas é melhor do que ter uma interface geral.

Exemplo: uma classe que tem três métodos. Do princípio anterior temos a classe conta que traz três métodos, depositar, transferir e realizar empréstimo.

Então, seguindo o exemplo do princípio de Liskov, quando implementa a conta corrente tudo bem. Mas, quando implementa a conta poupança gera duas exceções.

Porque realizar empréstimo e transferência, se conta poupança não tem.

Como está estendendo o contrato é obrigado a implementar os três métodos.

Toda vez que implementa um contrato é obrigado a implementar as operações dele.

A solução é quebrar o contrato que tem três operações em vários contratos de modo que cada tenha uma operação.

E cada classe, tipo conta poupança, implementa somente a funcionalidade que ela precisa. Vai implementar ou estender somente a funcionalidade que precisa.

D - DART & SOLID - Princípio da inversão de dependências (DIP)

Princípio da inversão de dependência diz que temos que depender de abstrações e não de classe concreta.

Isso quer dizer, quando tenho uma classe que faz uma determinada função, ela não deve ser utilizada em outra classe.

Solução é transformar a classe em abstrata e quando ela for utilizada em outra classe deve ser implementada por esta outra classe.

Ao ser implementada pela classe x, esta classe x terá que informar a classe concreta.