UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES
CAMPUS PATOS DE MINAS

WEGLISON RAMOS PEREIRA

# DESENVOLVIMENTO DE *SOFTWARE* PARA GERENCIAMENTO DE INFORMAÇÕES DA SECRETARIA DE SAÚDE DE PATOS DE MINAS E PROPOSTA DE UMA FERRAMENTA PARA ANÁLISE DE DADOS

PATOS DE MINAS - MG
2022

WEGLISON RAMOS PEREIRA

# DESENVOLVIMENTO DE *SOFTWARE* PARA GERENCIAMENTO DE INFORMAÇÕES DA SECRETARIA DE SAÚDE DE PATOS DE MINAS E PROPOSTA DE UMA FERRAMENTA PARA ANÁLISE DE DADOS

Trabalho apresentado à banca examinadora como requisito parcial de avaliação da disciplina de TCC2 da graduação em Engenharia Eletrônica e de Telecomunicações, da Faculdade de Engenharia Elétrica, da Universidade Federal de Uberlândia, Campus Patos de Minas.

Orientador: Prof.ª Dr.ª Eliana Pantaleão.

Patos de Minas, 18 de agosto de 2022

Banca Examinadora:

---

Prof.ª Dr.ª Eliana Pantaleão – FACOM/UFU (Orientadora)

---

Prof. Dr. Laurence Rodrigues do Amaral – FACOM/UFU (Membro 1)

---

Prof. Dr. Pedro Luiz Lima Bertarini – FEELT/UFU (Membro 2)

# AGRADECIMENTOS

Agradeço a Deus por me guiar, por me abençoar com saúde e sabedoria nos momentos mais difíceis. Agradeço também aos meus familiares que sempre me apoiaram em especial meus pais Euripedes Ramos Pereira e Ednilza de Sousa, meus irmãos e minha tia Maristela Pereira da Silva que me acolheu em sua casa e me deu toda a estrutura para me ambientar a cidade e me adaptar aos novos desafios.

Aos professores e amigos que com suas competências e amizades contribuíram muito no decorrer da minha vida acadêmica, sejam com conselhos, correções, conversas ou festas. Agradeço especialmente a minha orientadora Prof.ª Dr.ª Eliana Pantaleão, pelas ideias, dedicação e tempo empenhado para o desenvolvimento deste trabalho. Agradeço também ao Prof. Dr. Pedro Luiz Lima Bertarini e ao Prof. Dr. Laurence Rodrigues do Amaral pelos ensinamentos e por terem aceitado o convite de participarem da minha banca.

A todo o corpo docente, técnicos de laboratório e administrativos da Universidade Federal de Uberlândia em especial a UFU Campus Patos de Minas. Gostaria de fazer um agradecimento especial as técnicas e assistentes sociais Cilesia Pereira e Carolina Poswar que me auxiliaram grandiosamente em um momento conturbado e de angustia que vivenciei durante a graduação.

# RESUMO

Atualmente tem-se buscado modernizar e automatizar quase todas as formas de serviço. O governo, por exemplo, tem como objetivo digitalizar todos os serviços públicos até o ano de 2022 [1]. Existem diversos benefícios com a modernização, como por exemplo, redução da burocracia, corte de gastos relacionados a materiais de papelaria entre outros. O presente trabalho consiste em estudar e desenvolver um *software* para digitalização e gerenciamento de informações da Secretaria de Saúde de Patos de Minas (SSPM). A ideia da construção do sistema de digitalização surgiu a partir de um projeto de extensão entre a Universidade Federal de Uberlândia e a Secretaria de Saúde de Patos de Minas voltado a atender demandas na área da saúde. As informações a serem digitalizadas são referentes a questionários arquivados em arquivos de papel pela SSPM. O sistema desenvolvido possibilita tanto a digitalização como a busca e edição de informações previamente armazenadas. Além do desenvolvimento do programa de digitalização é proposta uma sugestão de ferramenta de análise de dados. Por meio da análise dos dados torna-se possível a extração de *insights* importantes relacionados ao controle de doenças endêmicas no município. Essa metodologia de extração de informações de um conjunto de dados caracteriza-se como descoberta de conhecimento em bancos de dados - do inglês *Knowledge Discovery in Databases* (KDD) - e vem sendo empregada por grandes organizações auxiliando os gestores quanto à tomada de decisões e ajustes no processo. Por fim, utilizando um conjunto de dados fictícios, é apresentado um *dashboard* com diferentes formas de gráficos, uma forma de exemplificar um modelo de análise de dados possível de ser aplicada ao conjunto de dados armazenados.

**Palavras-chave:** Desenvolvimento de *software*. Java. Banco de Dados. SQL. *Business Intelligence.* Análise de Dados. *Power BI*.

# ABSTRACT

Currently, efforts are being made to modernize and automate almost all forms of service. The government, for example, aims to digitize all public services by the year 2022 [1]. There are several benefits with modernization, such as reducing bureaucracy, cutting expenses related to stationery materials, among others. The present work consists of studying and developing software for digitizing and managing information from the Patos de Minas Health Department (SSPM). The idea of building the digitization system came from an extension project between the Federal University of Uberlândia and the Health Department of Patos de Minas aimed at meeting demands in the health area. The information to be digitized refers to questionnaires filed in paper files by the SSPM. The developed system makes it possible to digitize as well as search and edit previously stored information. In addition to the development of the digitization program, a suggestion for a data analysis tool is proposed. Through data analysis, it becomes possible to extract important insights related to the control of endemic diseases in the municipality. This methodology for extracting information from a dataset is characterized as knowledge discovery in databases - Knowledge Discovery in Databases (KDD) - and has been used by large organizations helping managers to make decisions and make adjustments. in the process. Finally, using a fictitious dataset, a dashboard is presented with different forms of graphs, a way of exemplifying a data analysis model that can be applied to the stored dataset.



**Keywords:** *Software* development. Java. Database. SQL. Business Intelligence. Data analysis. Power BI.

# LISTA DE FIGURAS

# LISTA DE ABREVIAÇÕES E SIGLAS

| | | |
|---|---|---|
| **API** | Interface de Programação de Aplicações | *Application Programming Interface* |
| **AWT** | Toolkit de janelas abstratas | *Abstract windows toolkit* |
| **BD** | Banco de Dados | *Database* |
| **BI** | Inteligência de Negócios | *Business Intelligence* |
| **DAO** | Objeto de Acesso aos Dados | *Data Access Object* |
| **DAX** | Expressões de Análises de Dados | *Data Analysis Expression* |
| **DDL** | Linguagem de Definição de Dados | *Data Definition Language* |
| **DML** | Linguagem de Manipulação de Dados | *Data Manipulation Language* |
| **DM** | Mineração de dados | *Data mining* |
| **GUI** | Interface Gráfica do Usuário | *Graphical User Interface* |
| **IDE** | Ambiente Integrado de Desenvolvimento | *Integrated Development Environment* |
| **JRE** | Ambiente de execução Java | *Java Runtime environment* |
| **JVM** | Máquina Virtual Java | *Java Virtual Machine* |
| **KDD** | Descoberta de Conhecimento de Dados | *Data Knowledge Discovery* |
| **MVC** | Modelo Controle e Visão | *Model Vision and Control* |
| **SGBD** | Sistema Gerenciador de Banco de Dados | *Database Management System* |
| **SQL** | Linguagem de Consulta Estruturada | *Structured Query Language* |
| **SSPM** | Secretaria de Saúde de Patos de Minas | *Health Department of Patos de Minas* |
| **SUS** | Sistema Único de Saúde | *Health Unic System* |
| **TI** | Tecnologia da Informação | *Information Technology* |
| **UVZ** | Unidade de Vigilância de Zoonoses | *Zoonosis Surveillance Unit* |

# Sumário

# 1 INTRODUÇÃO

Uma maneira inteligente de se evitar a perda de tempo na busca por documentos físicos é a utilização de um sistema computacional que digitalize e simplifique o processo de armazenamento e busca dos arquivos. A digitalização se difere de um documento digital, pois, no segundo caso, ele origina-se no meio eletrônico e tem a sua garantia graças ao uso de um certificado digital. Por outro lado, a digitalização de documentos consiste na conversão de arquivos analógicos para o meio digital. Essa conversão pode ser realizada por meio de fotocópia ou sistema computacional que represente fielmente o documento analógico [2].

Tendo em vista a necessidade de solução de problemas complexos em grandes organizações, muitas vezes faz-se necessário o desenvolvimento de um *software* que disponha de funcionalidades para a resolução do problema. O processo de desenvolvimento de *software* consiste em estudar, projetar e implementar um sistema computacional, isto é, transformar a necessidade de um utilizador ou de uma organização em um produto de *software*. Um *software* é caracterizado como um produto virtual, que se baseia essencialmente em um conjunto de códigos e instruções que visam solucionar um problema e podem ser escritos em uma ou várias linguagens de programação.

## 1.1 Tema do Projeto

Nesse trabalho de conclusão de curso é proposto o desenvolvimento de um *software* para gerenciamento de informações de domicílios. Tais informações são registradas em arquivos de papel e fazem parte do acervo da Secretaria de Saúde de Patos de Minas (SSPM). A função basilar do sistema consiste em digitalizar as informações para um banco de dados em busca de maior segurança e integridade dos arquivos.

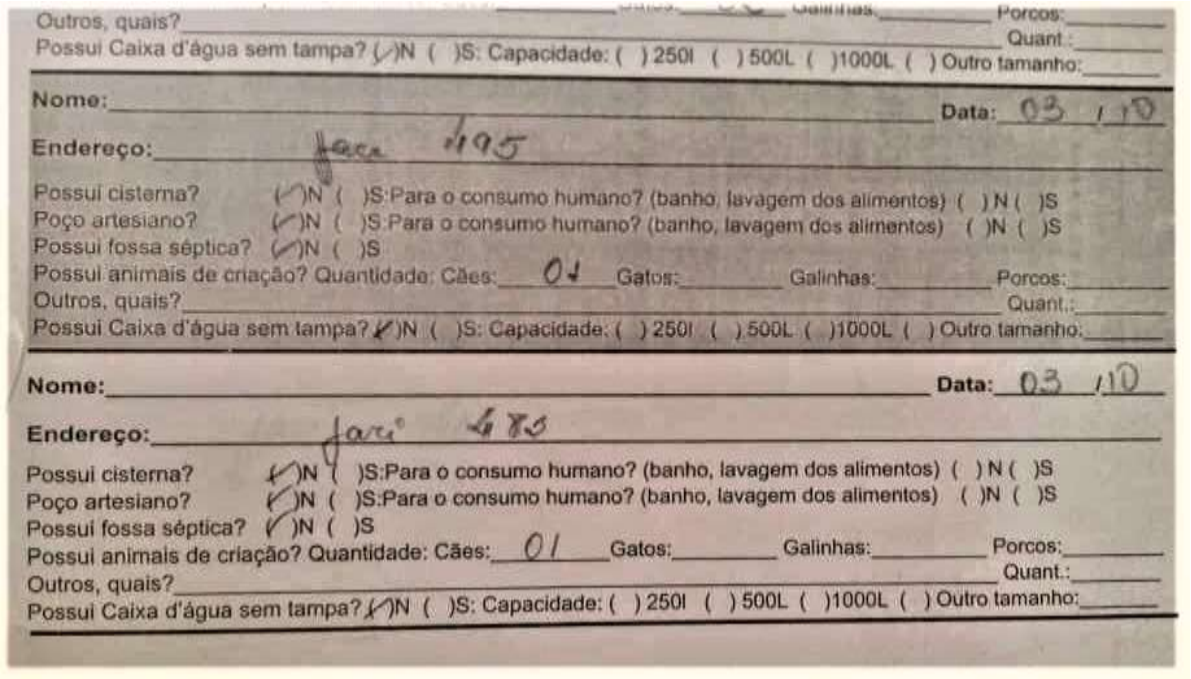
Além da digitalização, será sugerida uma ferramenta de análise de dados. Por meio das ferramentas de análise de dados tornam-se possíveis a manipulação de dados e a conseguinte extração de *insights* importantes sobre a organização.

## 1.2 Problematização

Um problema encontrado pela SSPM é a utilização de papel para registro de um grande volume de informações de domicílios como pode ser visto na Figura 1.1. Uma alternativa encontrada para a resolução desse problema corresponde à digitalização das informações para um banco de dados tendo em vista que tal sistema é econômico, seguro e longevo. Portanto

torna-se necessário o desenvolvimento de um sistema computacional que realize a digitalização dos dados cadastrados para um do banco de dados. Existem algumas ferramentas que proporcionam essa funcionalidade, sendo que uma delas é a linguagem de programação Java. Essa linguagem possui tanto ferramentas de desenvolvimento da interface gráfica do usuário (GUI), quanto a conexão com o banco de dados. Por disponibilizar essas ferramentas e ser uma linguagem com vasta documentação disponível na Internet, optou-se por utilizá-la para o desenvolvimento do projeto.

Figura 1.1: Ficha em Papel de Informações dos Domicílios


Outros, quais?_____ Porcos:_____
Possui Caixa d'água sem tampa? (✓)N ( )S: Capacidade: ( ) 250l ( ) 500L ( )1000L ( ) Outro tamanho:_____ Quant.:_____

Nome:_____ Data: 03 /10

Endereço: Faca 495

Possui cisterna? (✓)N ( )S:Para o consumo humano? (banho, lavagem dos alimentos) ( ) N ( )S
Poço artesiano? (✓)N ( )S:Para o consumo humano? (banho, lavagem dos alimentos) ( )N ( )S
Possui fossa séptica? (✓)N ( )S
Possui animais de criação? Quantidade: Cães: 01 Gatos:_____ Galinhas:_____ Porcos:_____
Outros, quais?_____ Quant.:_____
Possui Caixa d'água sem tampa? (✓)N ( )S: Capacidade: ( ) 250l ( ) 500L ( )1000L ( ) Outro tamanho:_____

Nome:_____ Data: 03 /10

Endereço: Faca 485

Possui cisterna? (✓)N ( )S:Para o consumo humano? (banho, lavagem dos alimentos) ( ) N ( )S
Poço artesiano? (✓)N ( )S:Para o consumo humano? (banho, lavagem dos alimentos) ( )N ( )S
Possui fossa séptica? (✓)N ( )S
Possui animais de criação? Quantidade: Cães: 01 Gatos:_____ Galinhas:_____ Porcos:_____
Outros, quais?_____ Quant.:_____
Possui Caixa d'água sem tampa? (✓)N ( )S: Capacidade: ( ) 250l ( ) 500L ( )1000L ( ) Outro tamanho:_____

Fonte: O Autor.

## 1.3 Objetivos

### 1.3.1 Objetivos Gerais

Considerando-se o problema do registro de informações em arquivos de papel pela SSPM, o presente trabalho tem por objetivo o desenvolvimento de um *software* que permita a digitalização e gerenciamento das informações. O sistema disponibilizará uma interface gráfica simples similar ao modelo existente em folha de papel. Além disso, o sistema também desempenhará as funções de busca e edição dos dados cadastrados.

Posteriormente, pretende-se sugerir uma ferramenta de análise de dados que viabilize a busca de informações comportamentais de microrregiões do município. Tais informações

podem auxiliar a SSPM quanto às providências relacionadas às políticas públicas de conscientização e controle de doenças endêmicas.

### 1.3.2 Objetivos Específicos

Os questionários realizados pela SSPM levantam informações domiciliares como o modelo de saneamento da residência, criação de animais, existência de caixa d'água entre outras questões. Essas informações são caracterizadas pelo endereço e data da realização do questionário, como pode ser visualizado na Figura 1.1, não constando nome ou identificação de nenhum habitante no formulário. O sistema de gerenciamento se baseará em três funcionalidades principais: o cadastro de um questionário, a busca por um questionário e a edição em cadastros já armazenados. Alguns exemplos específicos de funções que serão realizadas pelo sistema são a busca do questionário por bairro, data ou espécie de animal que existem em um bairro.

A partir das informações armazenadas no banco de dados, torna-se possível estudá-las em busca de parâmetros relevantes como, por exemplo, a relação entre uma doença e a quantidade de casas que possuem caixa d'água descoberta. Essa análise pode ser realizada por comparações entre tabelas ou *dashboards* descritivos. Um *software* que possibilita tais análises é o *Microsoft Power BI* que proporciona integração com vários modelos de banco de dados. Por meio dessa ferramenta computacional, torna-se possível importar os dados de uma base local ou em servidores na nuvem, além de criar tabelas, mapas e gráficos. Diante disso, além do desenvolvimento do sistema de gerenciamento, será proposta uma plataforma de *Business Intelligence* (BI) que possibilite a análise descritiva dos dados.

## 1.4 Considerações Finais

Por meio do desenvolvimento de um sistema computacional, procura-se automatizar o arquivamento de informações da SSPM tendo em vista que o modelo atual utiliza papel para o registro dessas informações. No capítulo subsequente, é realizado um estudo teórico sobre os principais conceitos necessários para desenvolvimento do presente trabalho.

# 2 REFERENCIAL TEÓRICO

## 2.1 Controle de Zoonoses

Nos últimos anos, o Ministério da Saúde sistematizou a aplicação de recursos para respaldar os municípios na elaboração e na implantação de unidades de controle de zoonoses integradas ao Sistema Único de Saúde (SUS). Essas unidades estão situadas especialmente em capitais, regiões metropolitanas, municípios sedes de regionais de saúde, municípios de fronteira e em alguns municípios mais populosos, sendo designadas atualmente de Unidades de Vigilância de Zoonoses (UVZ) [3].

A contenção de zoonoses de relevância para a saúde pública compreende doenças de transmissão vetorial, alguns dos animais que são vetores de doenças podem ser visualizados na Figura 2.1. As doenças fiscalizadas pelo programa nacional de vigilância e controle de zoonoses do Ministério da Saúde incluem raiva, leishmaniose, peste, leptospirose, febre maculosa brasileira, hantavirose, doença de Chagas, febre amarela, chikungunya, dengue malária e febre do Nilo Ocidental. As zoonoses de relevância regional ou local são as seguintes: toxoplasmose, esporotricose, ancilostomíase, toxocaríase, histoplasmose, criptococose, complexo equinococose entre outras [4].

Figura 2.1: Animais que Possibilitam a Transmissão Vetorial.

Fonte: [5].

Qualquer ação, estratégia de vigilância, prevenção e controle de zoonoses de importância para a saúde pública têm de ser antecedidas por levantamento do contexto de influência na

saúde pública, por meio de avaliação da capacidade de disseminação, da gravidade e considerando a população exposta e as espécies de animais envolvidas [4].

### 2.1.1 Centro de Controle de Zoonoses de Patos de Minas

O centro de controle de zoonoses é o departamento da SSPM encarregado de prevenir, inspecionar doenças e desenvolver sistemas de vigilância sanitária e epidemiológica no município. O órgão efetua cirurgias em cães como, por exemplo, esterilização, ovariohisterectomia nas fêmeas, orquiectomia nos machos, sendo essas algumas formas de se evitar a procriação indesejada. O centro conta também com os serviços de controle da raiva, contenção de roedores, animais peçonhentos e sinantrópicos, laboratório de controle de zoonoses, entomologia, controle de vetores, incentivo da saúde humana e educação sobre vigilância responsável para zoonoses [6].

## 2.2 Desenvolvimento de Software

O processo de desenvolvimento de *software* pode ser pensado como uma estrutura de passos organizados que são usados para projetar, desenvolver, testar e ajustar um *software*. Existem alguns padrões distintos para se desenvolver um sistema computacional, porém em quase todos os processos é comum deparar-se com os seguintes passos: levantamento de requisitos, análise de requisitos, projeto, implementação, testes e implantação [7].

### 2.2.1 Levantamento de Requisitos

Nesta etapa o objetivo consiste em compreender o problema, dando aos desenvolvedores e usuários o mesmo entendimento do que deve ser construído para solução do problema. Em consonância, busca-se conhecer e privilegiar as necessidades dos futuros utilizadores do *software* [7].

### 2.2.2 Análise de Requisitos

Essa etapa é conhecida também como especificação de requisitos, nela é realizado um estudo detalhado dos dados levantados na atividade de levantamento de requisitos. A partir disso, são construídos modelos com a finalidade de retratar o sistema de *software* a ser desenvolvido [7]. Também é realizada a análise da linguagem de programação, do ambiente de

desenvolvimento e do sistema gerenciador de banco de dados a serem utilizados no desenvolvimento do sistema.

Em um estudo junto ao departamento da secretária de saúde responsável pela aplicação dos questionários levantou-se quais as principais funcionalidades que a aplicação necessitária prover. Verificou-se que a mesma deveria possuir alguns menus de fácil acesso, sendo eles: cadastro de um novo questionário, busca de questionários por endereço e edição em cadastros já existentes. A partir desses menus existiriam alguns submenus com funcionalidades mais especificas que auxiliariam os servidores no dia a dia.

A partir das características levantadas idealizou-se a arquitetura de software que possibilitasse tais funcionalidades, a partir disso projetou-se como poderia ser a implementação de classes e métodos.

### 2.2.3 Projeto

A etapa de projeto é considerada uma das mais importantes durante o desenvolvimento de *software*. O sistema é projetado a partir da descrição levantada na fase de análise de requisitos [7]. O projeto possui duas vertentes principais que são: projeto da arquitetura (ou projeto de alto nível), e projeto detalhado (ou projeto de baixo nível). Para um sistema desenvolvido utilizando o paradigma de orientação a objetos o planejamento da arquitetura é habitualmente desempenhado por um arquiteto de *software.* O planejamento da arquitetura visa agrupar as classes de objetos associados em subsistemas e também fazer um levantamento dos recursos de hardware disponíveis. Já no projeto detalhado são modeladas as relações entre os objetos que compõem o modulo com o objetivo de realizar as funcionalidades do sistema. Além disso, nesse momento é planejada a interface gráfica do usuário e o projeto de banco de dados [7].

### 2.2.4 Implementação

A etapa de implementação consiste na codificação do sistema. Para o paradigma de desenvolvimento orientado a objetos, a codificação se dá estabelecendo as classes de objetos do sistema em questão [7].

### 2.2.5 Testes

Para verificar a possibilidade de ocorrência de erros no sistema é importante a realização de testes a fim de se validar o produto de s*oftware*. Por meio dos testes, é possível obter um

relatório com possíveis falhas e possíveis soluções. Ao final dessa etapa, são realizadas as correções e adaptações necessárias para a implantação do sistema [7].

### 2.2.6 Implantação

A etapa de implantação corresponde à instalação do sistema para o usuário final. Caso exista um sistema anterior que será substituído pelo novo sistema, é necessário efetuar a sincronização da base de dados para o correto funcionamento da aplicação [7].

## 2.3 Paradigmas de Linguagens de Programação

Linguagens naturais facilitam a comunicação entre pessoas, de forma similar às linguagens naturais, as linguagens de programação possibilitam a comunicação entre seres humanos e computadores, contudo possuem um campo mais reduzido de expressões [8].

"Uma linguagem de programação é uma ferramenta utilizada pelo profissional de computação para escrever programas, isto é, conjuntos de instruções a serem seguidas pelo computador para realizar um determinado processo" [9]. A sintaxe de uma linguagem refere ao que constitui um programa estruturalmente preciso e é definida por uma gramática livre de contexto. O vocabulário de uma linguagem de programação estabelece um conjunto de regras para definir variáveis e funções. O significado de um programa é definido por sua semântica, ou seja, quando um programa é executado, o efeito de cada comando sobre os valores das variáveis é dado pela semântica [9].

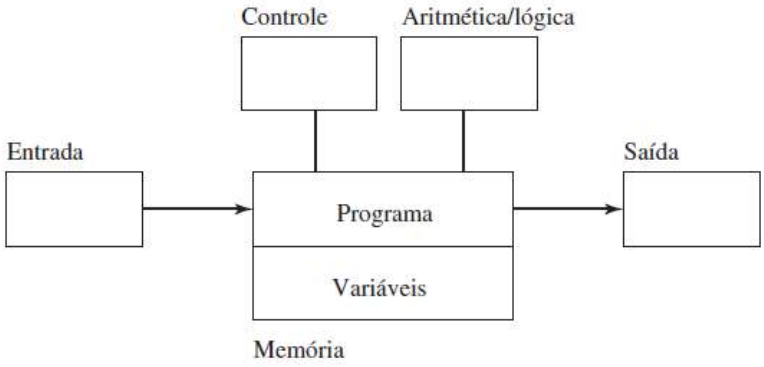
Paradigma de programação é um conceito importante no estudo das linguagens de programação. Pode-se pensar em paradigma de programação como um padrão de resolução de problemas que se associa a um determinado gênero de programas e linguagens. Quatro paradigmas de programação diferentes e fundamentais que podem ser destacadas são: programação imperativa, programação orientada a objetos, programação funcional e programação lógica [8].

### 2.3.1 Programação imperativa

Este paradigma de programação é considerado o mais antigo, possui como base o modelo computacional clássico de Von Neumann - Eckert [9]. Para esse modelo, as variáveis e o programa são armazenados juntos, o programa consiste em uma série de comandos para

execução de cálculos, atribuição de valores a variáveis, obtenção de entradas e produção de saídas ou interrupções como mostrado na Figura 2.2.

Figura 2.2: Modelo de Programação Imperativa.




Fonte: [9].

A abstração de procedimentos é uma característica para a programação imperativa, da mesma maneira as atribuições, os laços, as sequências, os comandos condicionais e a manipulação de exceções. Algumas linguagens de programação baseadas nesse paradigma são as seguintes: Cobol, Fortran, C, Ada e Perl [9].

### 2.3.2 Programação Orientada a Objetos

A programação orientada a objetos baseia-se em um modelo no qual um programa é uma coleção de objetos independentes que se relacionam entre si por meio de trocas de mensagens. Devido a essa lógica, a transferência de mensagens possibilita que objetos de dados atuem de forma ativa ao invés de passiva. Essa característica ajuda a compreender e distinguir melhor a programação orientada a objetos da imperativa. A classificação de objetos, a herança e a troca de mensagens são características fundamentais da programação orientada a objetos. Algumas linguagens comumente utilizadas que se baseiam no paradigma orientado a objetos são as seguintes: Smalltalk, C++, Java e C# [9].

### 2.3.3 Programação Funcional

A programação funcional molda um problema computacional por um agrupamento de funções matemáticas, cada uma com um espaço de entrada (domínio) e resultado. Essa característica diferencia a programação funcional das linguagens que utilizam comando de

atribuição. Alguns exemplos de linguagens funcionais são Clojure (um dialeto de Lisp para a JVM), Haskell e Swift [9].

### 2.3.4 Programação Lógica

O paradigma de programação lógica possibilita um programador modelar um problema atribuindo qual resultado esperado, em vez de como ele deve ser obtido. Comumente essas linguagens são definidas como linguagens baseadas em regras. A principal linguagem de programação lógica é a Prolog [9].

## 2.4 Arquitetura de *Software*

Existem diferentes definições de arquitetura de *software*. Uma dessas definições afirma que a arquitetura de *software* contempla o projeto em mais alto nível, ou seja, o objetivo deixa de ser a organização e interface de classes individuais e passa a ser em unidades de maior dimensão, como por exemplo pacotes, componentes, módulos, subsistemas, camadas e serviços [10].

Nas palavras de Ralph Johnson [10]. “A arquitetura é sobre coisas importantes. Seja lá o que for”. Essa descrição de arquitetura de *software* pode ser expressa pela frase de Ralph Johnson, que destaca que arquitetura de *software* abrange as decisões de projeto mais significativas em um sistema. Essas decisões são tão relevantes que, uma vez definidas e aplicadas, torna-se muito complexo a modificação e ajustes futuros. Consequentemente, a última definição de arquitetura descrita por Ralph Johnson é mais expressiva que a anterior, pois considera que arquitetura não é unicamente um conjunto de módulos, mas um conjunto de decisões [10].

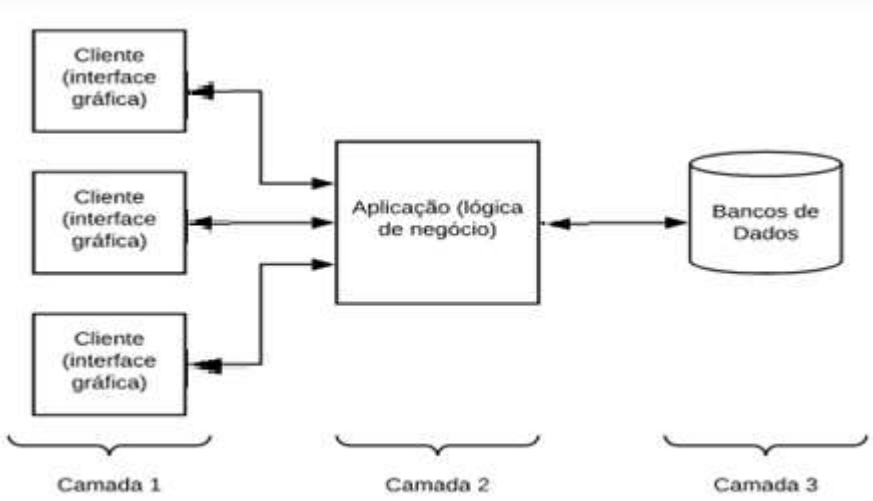
Existem alguns padrões conhecidos de arquitetura de *software*, como arquitetura em três camadas, padrão MVC (*Model View Controller*), microsserviços e padrão DAO (*Data Access Object*). Esses padrões são descritos nas seções seguintes.

### 2.4.1 Arquitetura em Três Camadas

A arquitetura em três camadas tem como propósito promover a segmentação das funcionalidades utilizando camadas para o desacoplamento da lógica de apresentação, da lógica de negócio e da lógica de acesso aos dados como mostrado na Figura 2.3. A separação em três

camadas propicia ao sistema mais flexibilidade, possibilitando que elas sejam desenvolvidas e editadas de forma independente [11].

Figura 2.3: Representação de Arquitetura de Projeto em Três Camadas.




Fonte: [10].

A camada de apresentação é quem possibilita toda interação com o usuário, é responsável tanto pela exibição de informações como da coleta e processamento de entradas e eventos de interfaces, tais como cliques em botões, marcação de texto, sobreposição de painéis, entre outros. A camada de apresentação pode ser uma aplicação *desktop* em um sistema operacional com interface gráfica, como também uma aplicação *web* [10].

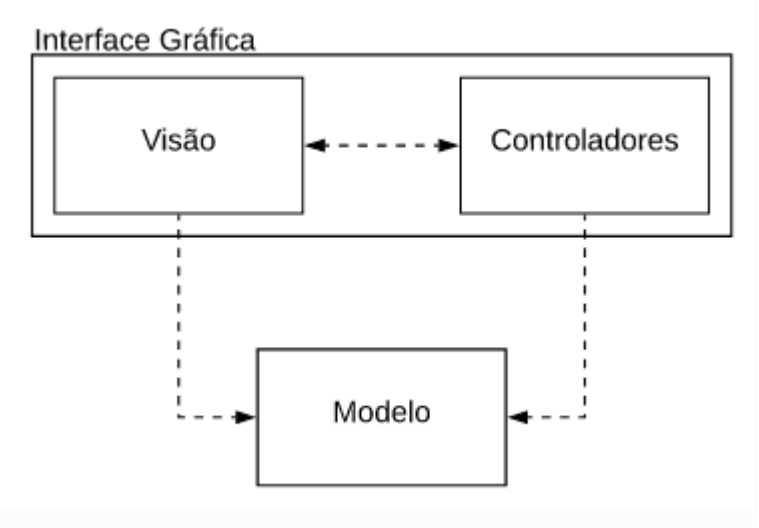
A camada de aplicação é responsável pelas regras de negócio, esta camada faz a comunicação entre a camada de apresentação e a camada de acesso a dados.

A camada de acesso a dados é a camada que contém os objetos que são responsáveis pela persistência dos dados da aplicação.

### 2.4.2 Padrão MVC: Modelo, Controle e Visão

O padrão MVC é bastante difundido nos processos de desenvolvimento de *software* devido à arquitetura possibilitar a divisão do projeto em camadas que se comunicam mutuamente. Cada uma dessas camadas, o *Model*, o *Controller* e a *View* mostradas na Figura 2.4, executam estritamente as funcionalidades que lhe foram definidas. Quando um evento é disparado na interface gráfica, como um clique em um botão, a interface irá se comunicar com o controlador que, por sua vez se comunica com os objetos do negócio [11].

Figura 2.4: Representação do Padrão MVC e Suas Interações.




Fonte: [10]

A camada de visão é responsável pelas classes de apresentação da interface gráfica, abrangendo janelas, botões, menus, barras de rolagem entre outros [10].

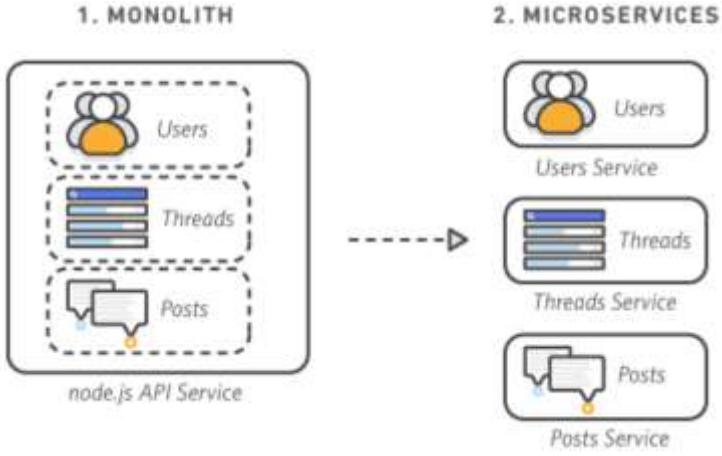
A camada de controladores é responsável por processar e interpretar eventos gerados por dispositivos de entrada, como mouse e teclado. A partir do disparo de um evento, os controladores têm a função de solicitar uma alteração no estado do modelo ou da visão [10].

A camada de modelo armazena os dados processados pela aplicação. Sendo assim, classes de modelo não têm qualquer compreensão ou submissão em relação às classes de visão e controladores. Classes de modelo possuem métodos que modificam dados e estados dos objetos de domínio [10].

### 2.4.3 Microsserviços

Microsserviços é uma arquitetura moderna de desenvolvimento de *software* em que a aplicação como um todo consiste em instâncias de serviços independentes. O conjunto de serviços independentes se comunicam por meio de aplicações de programação de interfaces (APIs) específicas e dessa forma deixam de se caracterizar em uma estrutura monolítica como mostrado na Figura 2.5. Como são executados de maneiras distintas, cada serviço pode ser atualizado, implementado em uma linguagem diferente e escalado para satisfazer as demandas de funções específicas de uma aplicação [12].

Figura 2.5: Divisão de uma Aplicação Monolítica em Microsserviços.



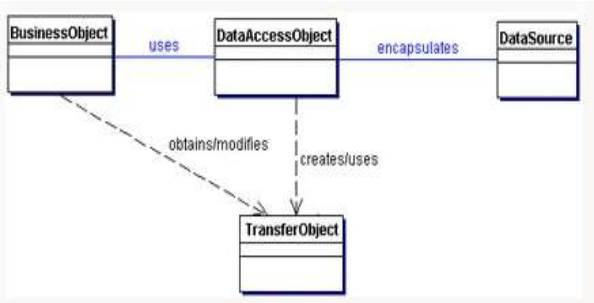

Fonte: [12]

Devido aos microsserviços atuarem de forma independente, eles possuem a característica de proporcionarem o desenvolvimento em tecnologias diferentes. Com isso o desenvolvimento da aplicação não se prende a linguagens de programação, *frameworks* e bancos de dados. Ao se analisar o desenvolvimento de um microsserviço de cadastro de clientes em um sistema de comércio eletrônico, por exemplo, pode-se empregar a linguagem Java com um banco de dados relacional. Para a mesma aplicação, pode-se desenvolver um microsserviço de recomendação de produtos similares em *Python* com um banco de dados não relacional. A arquitetura em microsserviço é considerada complexa, porém possibilita maior liberdade em aplicações de grande porte [10].

### 2.4.4 Padrão DAO: Objeto de Acesso aos Dados

O padrão DAO é um padrão de projeto que omite e agrupa os mecanismos de acesso a dados escondendo os detalhes da execução das consultas ao banco de dados da origem dos dados como mostrado diagrama de classes da Figura 2.6. Nesse padrão, a classe *DataAccessObject* encapsula o acesso aos dados, a classe *DataSource* é a origem dos dados e a classe *BusinessObject* contém a lógica de negócio e usa o objeto *DataAccessObject* para realizar alterações nos dados persistidos [13].

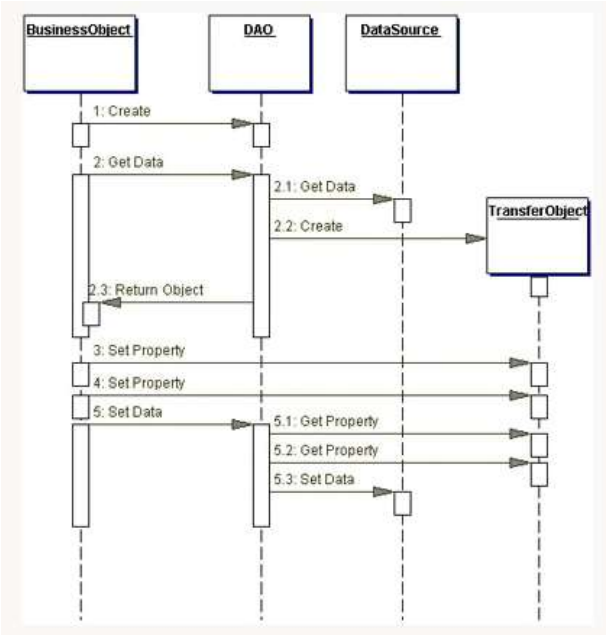
Figura 2.6: Representação de Um Diagrama de Classe Para o Padrão DAO.




Fonte: [14]

O objetivo do padrão é tornar o código mais organizado, fácil de manter e reutilizável onde uma única classe fica responsável por fazer a interface entre os objetos de negócio e a fonte de dados como mostrado na Figura 2.7.

Figura 2.7: Representação de um Diagrama de Sequência DAO.




Fonte: [14].

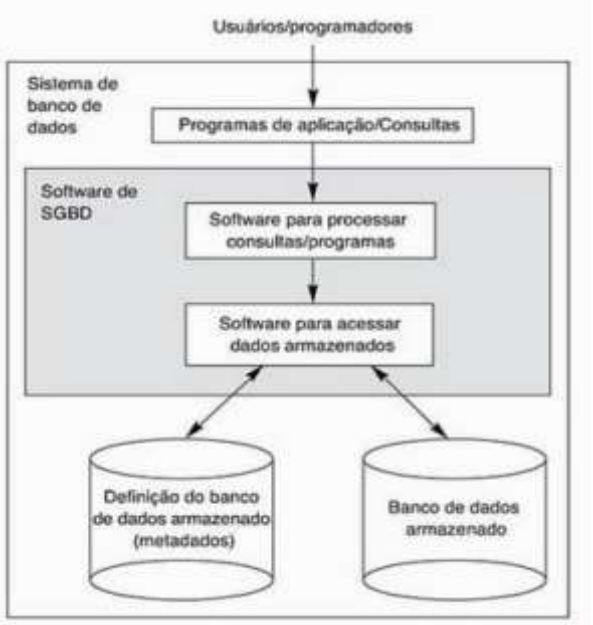
## 2.5 Banco de Dados

O banco de dados é uma coleção de dados relacionados. O banco de dados possui um fornecedor de informações do qual o dado é derivado, é interligado com eventos no mundo real e com um público que está ativamente interessado no conteúdo. Para um banco de dados ser considerado como preciso e confiável, ele necessita ser um reflexo verdadeiro do modelo que representa [15].

### 2.5.1 Sistema Gerenciador de Banco de Dados (SGBD)

Um sistema gerenciador de banco de dados (SGBD) é um conjunto de ferramentas que possibilitam desenvolvedores criar e manter banco de dados. O SGBD é um *software* de uso abrangente que facilita o processo de definição, construção, manipulação e compartilhamento de banco de dados. Outras funções exercidas pelos SGBD são a manutenção e a proteção dos dados por longos períodos. Para simplificar o entendimento, na Figura 2.8 ilustrado um sistema de banco de dados como sendo a união do banco de dados com o *software* de SGBD [15].

Figura 2.8: Diagrama Simplificado de um Sistema de Banco de Dados.




Fonte: [15].

### 2.5.2 Banco de Dados Relacional

No modelo relacional é comum a utilização de tabelas para representar os dados. Cada linha na tabela é um registro com uma identificação exclusiva. As colunas da tabela contêm atributos dos dados de cada registro, normalmente, tem um valor para cada atributo, facilitando o estabelecimento das relações entre os pontos de dados [16]. O modelo relacional estrutura-se por uma coleção de relações. Quando se considera uma relação como uma tabela de valores, cada linha da tabela representa uma coleção de valores de dados relacionados, esses dados relacionados são definidos como tuplas como mostrado na Figura 2.9. Uma linha ou tupla corresponde a uma entidade que acontece de fato na vida real [15].

Figura 2.9: Exemplo de Uma Tabela Relacional.


Fonte: Adaptado de [15]

### 2.5.3 SQL

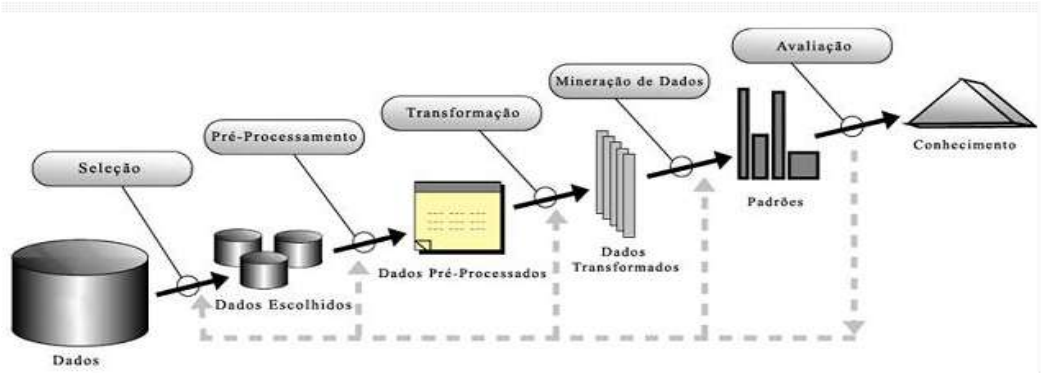
O nome SQL é uma referência a *Structured Query Language* ou linguagem de consulta estruturada. Originalmente a SQL era chamada de SEQUEL e foi desenvolvida pela *IBM Research* para servir de interface para um sistema de banco de dados relacional experimental. Hoje em dia a SQL é uma linguagem padrão para SGBDs relacionais comerciais. A SQL é uma linguagem de banco de dados bem ampla, ela possui instruções para definição de dados, consultas e atualizações. Portanto ela é considerada uma linguagem de definição de dados (DDL) e também uma linguagem de manipulação de dados (DML). Além dessa característica, a SQL tem as funcionalidades de definir visões sobre o banco de dados, especificar segurança e autorização, definir restrições de integridade e realizar controle de transações. Essa linguagem também possui regras que possibilitam embutir instruções SQL em linguagens de programação como Java, COBOL ou C/C++, essas linguagens usam APIs específicas para envio de instruções SQL ao SGBD [15].

## 2.6 Análise de Dados

Ao se deparar com grandes organizações que utilizam recursos de Tecnologia da Informação (TI), percebe-se que estas possuem uma grande quantidade de dados armazenados. Uma solução que tem sido muito debatida nos ambientes tecnológicos é o conceito de *Big Data* que utiliza essa grande quantidade de dados armazenados para levantar *insights* que auxiliem o dia a dia das empresas e estratégias de negócio. Esse aspecto não se limita às empresas de negócios, mas abre um leque para outras organizações como centros de pesquisa e órgãos públicos que colhem e armazenam grande quantidade de dados. Estes dados, sendo examinados, proporcionam informações que contribuem com a rotina das instituições, aprimorando a oferta

de seus serviços. Devido a essa nova demanda surge o processo de descoberta de conhecimento em base de dados demonstrado na Figura 2.10. Esse conceito advém do inglês *Knowledge Discovery in Database* (KDD), cuja intenção é a partir da base de dados absorver conhecimento. Por meio deste conhecimento extraído é possível tomar decisões de acordo com o resultado obtido. Durante o processo existem várias fases dentre elas a mais significativa é a etapa de mineração e análise de dados, do inglês *Data Mining* (DM) que objetiva identificar padrões de dados e extração de características [17].

Figura 2.10: Diagrama do Processo de Análise de Dados.




Fonte: [Fayyad. 1996]

Existem diferentes tipos de análises de dados como pode ser visto na Figura 2.11, cada tipo é voltado a uma proposta específica. Os métodos mais comuns de análises de dados são descritos a seguir.

Figura 2.11: Tipos de Análises de Dados.


Fonte: [18]

### 2.6.1 Análise Descritiva

A análise descritiva extrai os dados brutos e por meio da agregação ou da mineração de dados fornece *insights* valiosos sobre o passado. No entanto essas descobertas simplesmente sinalizam que algo está errado ou certo sem explicar o porquê. Por esse motivo funções de planejamento de demanda mais robustas não se contentam apenas com análises descritivas e preferem combiná-las com outros tipos de análises de dados [19].

### 2.6.2 Análise Diagnóstica

Os dados armazenados podem começar a ser comparados com outros dados para responder a pergunta de por que algo aconteceu no passado. Este é o processo de coleta e interpretação de diferentes conjuntos de dados para identificar anomalias, detectar padrões e determinar relacionamentos. Algumas abordagens que usam análises de diagnóstico incluem alertas, detalhamento, descoberta de dados, mineração de dados e correlações [20].

### 2.6.3 Análise Preditiva

A análise preditiva, de modo geral, é uma categoria de análise de dados que usa variáveis descritivas e preditivas do passado para analisar e identificar a probabilidade de um resultado futuro desconhecido. Ela reúne uma série de metodologias de mineração de dados, métodos de previsão, modelos preditivos e técnicas analíticas para analisar dados atuais, avaliar riscos e oportunidades e capturar relacionamentos e fazer previsões sobre o futuro. Neste estágio o questionamento não é mais o que aconteceu, mas a razão por que aconteceu e o que pode acontecer no futuro [20].

### 2.6.4 Análise Prescritiva

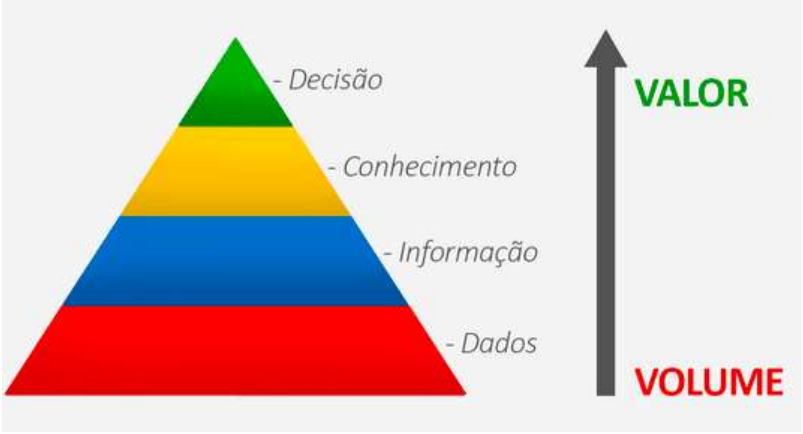
A análise prescritiva é uma combinação de dados, modelos matemáticos e várias regras de negócios para inferir ações para influenciar os resultados desejados futuros. Alguns estudiosos se referem a essas regras como modelagem de demanda, mas também pode incluir simulação, maximização de probabilidade e otimização. Uma análise prescritiva normalmente não envolve apenas uma resposta individual, mas é, na verdade, uma série de outras ações [20].

### 2.6.5 *Business Intelligence*

*Business Intelligence*, ou simplesmente BI é um conceito proposto no fim da década de 1980 pelo *Gartner Group,* instituto de pesquisa e análise do setor de tecnologia da informação. Atualmente BI é um conjunto de processos utilizados para extrair inteligência a partir de dados. Esse processo é exemplificado utilizando a pirâmide da Figura 2.12 e envolve o uso de tecnologias para coletar, armazenar, analisar e compartilhar as informações que são bases para a gestão de um negócio [21].

O escopo de BI abrange mineração de dados, análise de processos, análise de desempenho e análise descritiva. O BI explora todos os dados produzidos por uma empresa e possibilita a apresentação de relatórios de fácil compreensão, medidas de desempenho e tendências que guiam as decisões de gerenciamento [22]. BI é o processo de transformar dados em informação e através da descoberta, transformar a informação em conhecimento [23].

Figura 2.12: Pirâmide que Retrata o Propósito de *Business Intelligence*.




Fonte: [21].

## 2.7 Considerações finais

Em um processo de desenvolvimento de *software,* é necessário um planejamento minucioso em busca do maior êxito possível. Neste capítulo, foram descritos alguns conceitos importantes que serão empregados no decorrer do desenvolvimento do trabalho. Foram levantados alguns temas como as etapas que enfatizam a arquitetura de *software*, banco de dados e análise de dados. No capítulo seguinte são abordadas as principais tecnologias utilizadas bem como os métodos empregados para realização do trabalho.

# 3 MATERIAIS, TECNOLOGIAS E MÉTODOS

## 3.1 Materiais e Tecnologias

A seguir, as ferramentas computacionais utilizadas para o desenvolvimento do trabalho são descritas, assim como as principais tecnologias empregadas e como elas são aplicadas.

### 3.1.1 Linguagem de Programação Java

A linguagem Java foi desenvolvida na década de 1990 e baseia-se no paradigma de programação orientada a objetos. Inicialmente foi desenvolvida pela empresa americana *Sun Microsystems* e posteriormente foi vendida para a *Oracle* mantenedora atual da linguagem como destacado na logomarca mostrada na Figura 3.1. A característica mais relevante dessa linguagem é ser multiplataforma, ou seja, uma aplicação pode ser desenvolvida somente uma vez e posteriormente ser executada em quaisquer plataformas que suportem a linguagem.

Diferentemente das linguagens convencionais, a linguagem Java não realiza a compilação para código nativo da arquitetura do sistema operacional, ela é compilada para um *bytecode* que será posteriormente executado por uma máquina virtual [24].

Figura 3.1: Logotipo da Linguagem Java.


Fonte: [25].

A partir da linguagem Java são desenvolvidos diversos modelos de *softwares* como jogos *online*, páginas na internet, aplicativos *mobile*, *firmware* de equipamentos, *softwares* para equipamentos entre outros.

#### 3.1.1.1 Java Runtime Environment

O *Java Runtime Enviroment* (JRE) é um conjunto de aplicativos composto por bibliotecas e pela Máquina virtual Java (JVM). Esse conjunto de aplicativos possui funções que possibilitam a execução de aplicativos baseados na linguagem Java em computadores e dispositivos móveis. O JRE exerce a função de interpretador sendo que ele carrega o *bytecode* da aplicação compilada em Java, interpreta o código para a arquitetura computacional e depois executa a aplicação [24].

### 3.1.2 Ambiente de Desenvolvimento Integrado – NetBeans

O *NetBeans* é um ambiente de desenvolvimento originalmente desenvolvido pela *Sun Microsystem*. Possui as funcionalidades de edição de interface gráfica de usuário, edição de código-fonte, controle de versão, bem como suporte para aplicativos da Web. O *NetBeans* é utilizado para desenvolver principalmente com Java, mas também suporta outras linguagens, entre elas PHP, C / C ++ e HTML5. O *NetBeans* é o ambiente de desenvolvimento oficial para o Java 8, possui código-fonte aberto e é gratuito [26].

### 3.1.3 Interface Gráfica do Usuário -IDE NetBeans

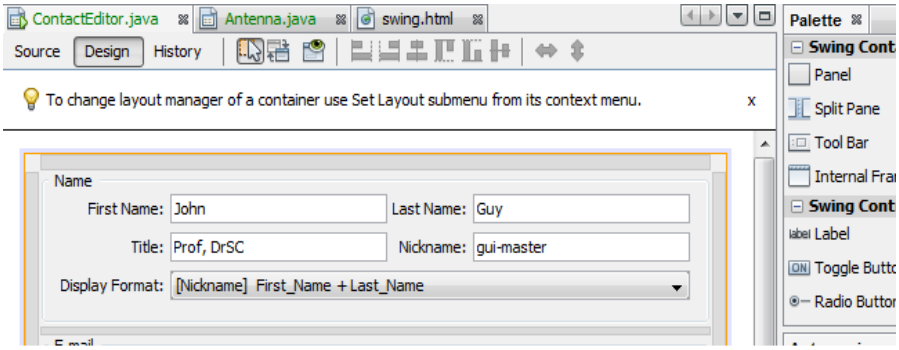
A interface gráfica com o usuário (GUI) se refere aos componentes gráficos da aplicação, todo o *design* de *layout* é projetado nessa etapa do desenvolvimento. A ferramenta de interface gráfica do IDE *NetBeans* possibilita a construção de GUIs de aparência profissional mesmo para um desenvolvedor com pouca experiência em gerenciadores de *layout*. É possível criar objetos por meio da *Application Programming Interface* (API) Java *Swing* que disponibiliza uma coleção de elementos gráficos que podem ser utilizados na plataforma Java. O *Swing* é compatível com o *Abstract Window Toolkit* (AWT) mas trabalha de forma oposta a essa biblioteca. A API *Swing,* diferente do AWT, não delega a tarefa de renderização ao sistema operacional, ele renderiza os elementos por conta própria [27]. Alguns dos principais componentes do Swing do IDE *NetBeans* são descritos a seguir.

#### 3.1.3.1 *JFrame*

Possibilita a criação da janela do programa com barra de título, ícones e botões de comando como mostrado na Figura 3.2. Entre os principais métodos, destacam-se: *pack()* que ajusta a janela para o tamanho dos componentes, *setSize(int, int)* que determina a largura e altura da janela, *setLocation(int, int)* que define a localização da janela na tela em relação aos eixos

coordenados (x,y), *setBounds(int, int, int, int)* que define localização e o tamanho e *setVisible(boolean)* que ativa ou desativa a visualização da janela [27].

Figura 3.2: Representação de um *JFrame* da Biblioteca Java Swing.




Fonte: [27].

### 3.1.3.2 *JPanel*

Cria um modelo básico de container para inserção de componentes. Entre os principais métodos desse componente destacam-se o *add(Component, int),* que adiciona o componente e a sua posição e o *setLayout(LayoutManagaer),* que altera o tipo de layout [27].

### 3.1.3.3 *JButton*

Cria um botão destinado a executar uma ação, muitas vezes a ação consiste no disparo de um evento que aciona um método em um objeto. Entre os principais métodos temos o *setText(String)* que altera o texto do botão e *setIcon(Icon)* que altera o ícone do botão [27].

### 3.1.3.4 Código Fonte

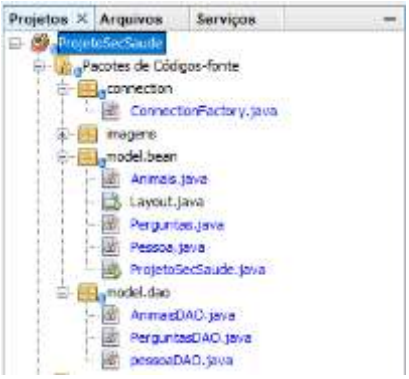
Uma funcionalidade interessante no conjunto de ferramentas da GUI do *NetBeans* é a geração de código automática para os componentes usados no layout, como por exemplo os painéis, campos de texto e botões. Dessa forma o trabalho do desenvolvedor se torna mais prático e há uma certa redução do esforço com codificação.

### 3.1.3.5 Projeto e Bibliotecas

Ao se iniciar um novo projeto no IDE *NetBeans*, automaticamente são criados pacotes de código fonte, sendo que esses pacotes podem conter uma ou várias classes e bibliotecas. Cada pacote possui classes para uma determinada finalidade. Dentro de cada pacote podem existir

classes que correspondem a interface gráfica, imagens, objetos e classes de conexões com banco de dados como pode ser visualizado na Figura 3.3.

Figura 3.3: Pacotes e classes IDE *NetBeans*.




Fonte: O Autor.

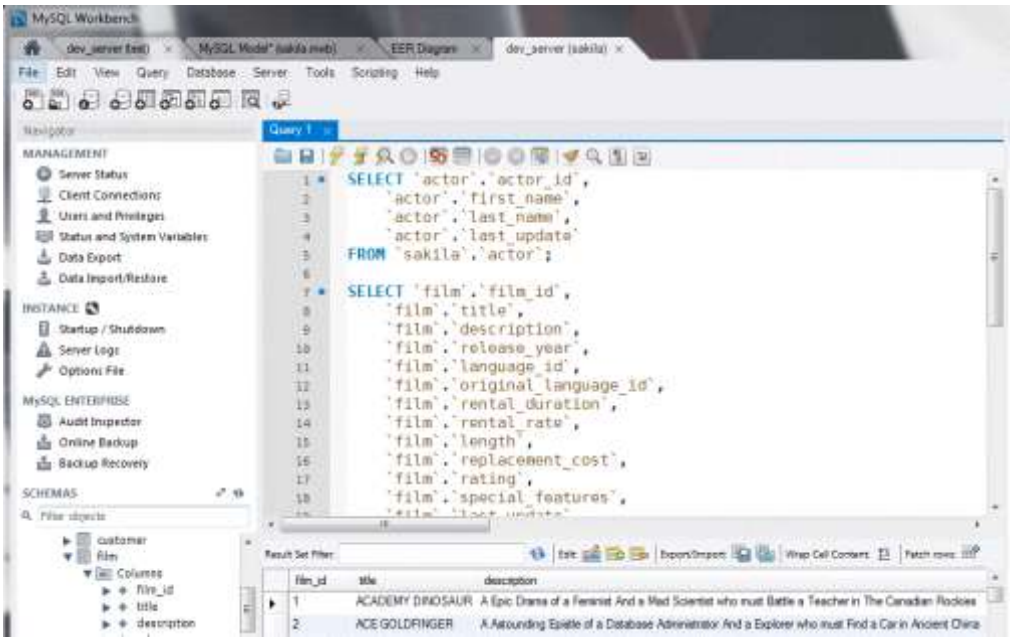
### 3.1.4 SGBD MySQL

O MySQL é um sistema de gerenciamento de banco de dados relacional de código aberto mantido pela Oracle e baseado em linguagem de consulta estruturada (SQL). Originalmente o sistema foi criado em 1995 pela empresa sueca MySQL AB, posteriormente o programa foi adquirido pela Sun Microsystems em 2008. Por fim, a Oracle assumiu o controle do SGBD em 2010. Os desenvolvedores podem usar uma versão disponibilizada gratuitamente, enquanto que as empresas devem obter uma licença comercial [28].

Resumidamente, o sistema de banco de dados MySQL consiste em uma arquitetura cliente-servidor. O servidor é o responsável por armazenar os dados, responder às requisições, controlar a consistência dos dados entre outras funcionalidades. O cliente se comunica com o servidor por meio de instruções e consultas SQL. A versão para desenvolvedores do MySQL é chamada de edição da comunidade - do inglês *community edition* - e possui o servidor e uma interface gráfica. O servidor deve ser devidamente instalado e configurado para receber conexões dos clientes. No MySQL o principal cliente é a interface gráfica fornecida pela Oracle, denominada MySQL Workbench. [29].

#### 3.1.4.1 MySQL Workbench

Figura 3.4: Interface Gráfica do MySQL Workbench




Fonte: [30]

O MySQL Workbench é uma interface gráfica unificada para gerenciamento de bancos de dados e pode ser vista na Figura 3.4. Essa ferramenta fornece algumas funcionalidades como: modelagem de dados, diagramação e criação de tabelas, desenvolvimento de consultas e instruções SQL, configuração de servidor, políticas de usuários, backup e muito mais. A aplicação MySQL Workbench está disponível em Windows, Linux e Mac OS [30].

### 3.1.5 Power BI

O *Microsoft Power BI* é a ferramenta de *Business Intelligence* da *Microsoft*. Por meio dessa ferramenta, é possível estudar, tornar coerentes e visuais os dados que estão armazenados em formatos e fontes diferentes. Essa ferramenta é uma coleção de serviços de *software*, aplicativos e conectores que trabalham juntos para transformar fontes de dados não relacionadas em informações coerentes, visualmente interpretativas e interativas. Além disso, ela também possibilita o fácil acesso e compartilhamento dessas informações [31]. A Figura 3.5 retrata algumas fontes de dados que podem ser inseridas no *Power BI*.

Figura 3.5: Fontes de dados para o Power BI.

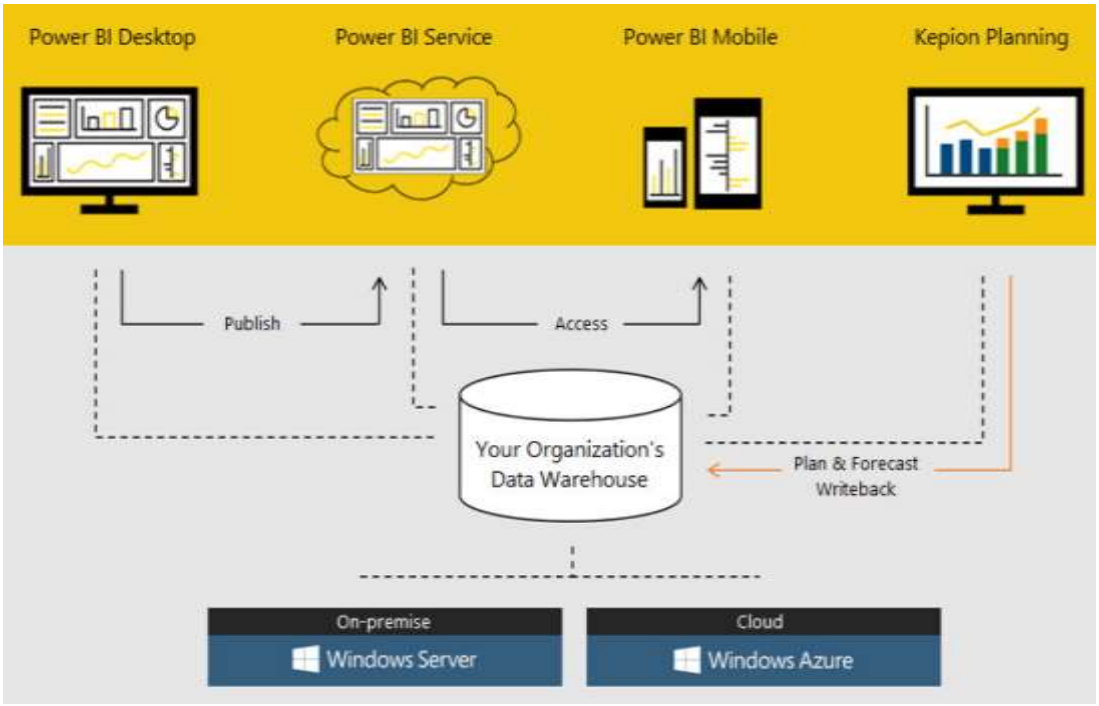

Fonte: [31].

A ferramenta de BI da *Microsoft* foi lançada com o *slogan* de projeto crescente e foi disponibilizada inicialmente em julho de 2011 junto com o SQL *Server*. Posteriormente em setembro de 2013, a organização divulgou uma nova versão denominada *Power BI* para *Office* 365. No início, o sistema se baseava apenas em recursos do Excel como *Power Query*, *Power Pivot* e *Power View*. A *Microsoft* acrescentou novos recursos ao longo do tempo, incluindo opções de segurança e conectividade de dados corporativos. O *Power BI* foi lançado como produto independente em julho de 2015 e funciona baseando-se em conexões entre várias bases de dados. Ele pode se conectar com uma planilha do Excel ou com um banco de dados em nuvem ou local. Os dados obtidos de fontes em nuvem são atualizados automaticamente. Já para planilhas ou bancos de dados armazenados localmente é necessário atualizar ou configurar manualmente um agendamento de atualização caso necessitem de painéis atualizados diariamente [32].

### 3.1.5.1 Componentes do Power BI

O *Power BI* possui uma gama de aplicativos que podem ser utilizados no computador pessoal ou em dispositivos móveis. O *Power BI Desktop* é a versão local, o *Power BI Service* é a versão em nuvem e o *Mobile Power BI* roda em dispositivos móveis, a Figura 3.6 mostra uma representação para cada uma dessas versões. No Power BI existem algumas ferramentas que merecem destaque especial visto que ajudam a criar e compartilhar relatórios de dados de forma eficiente [32].

Figura 3.6: *Power BI* em Diferentes Plataformas.




Fonte: [33].

- Power *Query*: ferramenta de transformação e filtragem de dados;
- Power Pivot: ferramenta de modelagem de dados;
- Power View: ferramenta de visualização de dados;
- Power Q & A**:** mecanismo de perguntas e respostas em linguagem natural;
- DAX (*Data Analysis Expressions*): As fórmulas DAX incluem funções, operadores e valores para realizar cálculos avançados e consultas em dados nas tabelas e colunas relacionadas nos modelos de dados tabulares;

## 3.2 Métodos

A seguir serão descritos os métodos utilizados para realização do trabalho, que envolvem o desenvolvimento do software proposto e a sugestão de ferramenta análise de dados.

### 3.2.1 Desenvolvimento do *Software* de Digitalização

Para o desenvolvimento do *software* de digitalização, os conceitos descritos nas Seções 2.3, 2.4, 3.1.1, 3.1.2 e 3.1.3 foram de suma importância. Aspectos do paradigma de programação orientada a objetos bem como conceitos de arquitetura de *software* foram os

pilares para o desenvolvimento da solução proposta. A opção por realizar todo o desenvolvimento da aplicação em Java baseia-se na familiaridade do desenvolvedor com a linguagem e por essa ela disponibilizar ferramentas para o desenvolvimento tanto do *back-end* (gerenciamento e processamento dos dados) como do *front-end* (interface gráfica do sistema). O IDE utilizado é o *NetBeans* que é o ambiente oficial para a linguagem Java 8.

O intuito principal foi desenvolver uma aplicação computacional que representasse fielmente o questionário original, ou seja, os mesmos campos e perguntas que constam no modelo em papel da Figura 1.1 foram transpostos para a versão virtual do questionário.

Por meio das características da linguagem Java foi possível a implementação das classes de objetos presentes no questionário de cadastro como animais, perguntas e pessoas além de a versão virtual possibilitar ainda algumas inserções de novas funcionalidades, como busca e edição das informações cadastradas.

#### 3.2.1.1 Criação da Interface Gráfica do *Software*

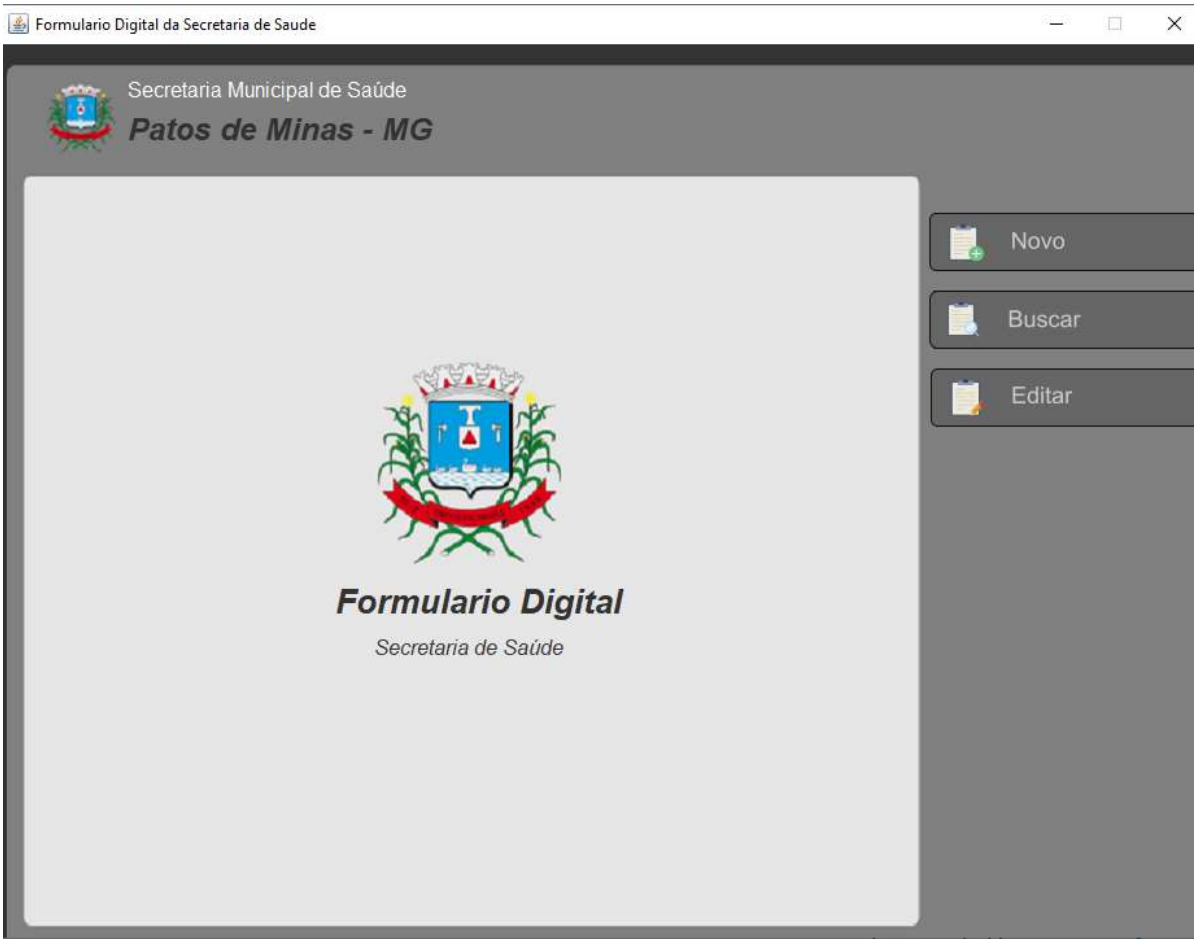
Utilizando o IDE *NetBeans* foi possível projetar e desenvolver os *layouts* usando as facilidades da API *Swing*. Essa API possibilita criar um esboço gráfico do sistema onde vários componentes de uma paleta de opções mostrada na Figura 3.2 podem ser arrastados e editados para atender as características pretendidas pelo desenvolvedor. Além das facilidades de criação e edição dos objetos gráficos, o IDE *NetBeans* gera o código padrão para os componentes, sendo necessário apenas customizar os parâmetros pertinentes para deixar o objeto gráfico como pretendido.

A seguir serão descritas as principais interfaces gráficas desenvolvidas para o *software*, e o funcionamento básico de cada uma delas.

##### 3.2.1.1.1 Tela Inicial do *Software*

Em destaque no *layout* da tela inicial, tem-se um painel com uma imagem representando a bandeira da cidade de Patos de Minas, bem como uma frase referindo-se ao sistema como sendo um formulário digital como pode ser visto na Figura 3.7. Do lado direito superior há um painel com alguns botões onde podem ser selecionadas as funções de “NOVO”, “BUSCAR” ou “EDITAR” um questionário.

Figura 3.7: Tela inicial do *software* de digitalização.




Fonte: O Autor

Todos as telas baseiam-se em *jPanels* ou painéis sobrepostos, onde a cada ação do usuário, um ou vários painéis são colocados em destaque e os demais são deixados em segundo plano. Toda essa transposição entre painéis é realizada por meio do método booleano *setVisible* como mostrado na Figura 3.8. Sobre os painéis podem ser anexados os objetos necessários como botões, campos de textos, tabelas entre outros componentes e sempre que o método *setVisible* é definido como *true* os objetos presentes sobre o painel são mostrados ao usuário.

Figura 3.8: Iniciando os Painéis.

Layout.java | pessoaDAO.java | ConnectionFactory.java

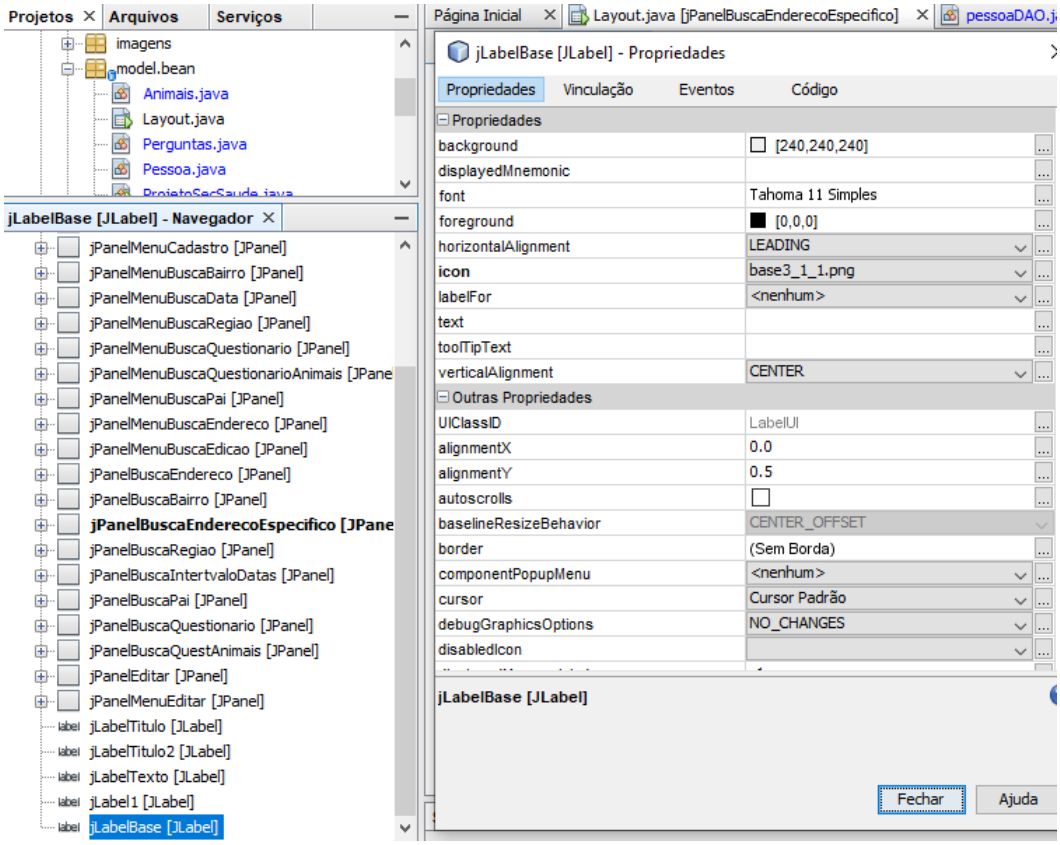
Projeto | Histórico

```
ArrayList<Perguntas> listPerg = new ArrayList();

public Layout() {
    initComponents();
    jPanelCadastro.setVisible(false);
    jPanelMenuCadastro.setVisible(false);
    jPanelBuscaEndereco.setVisible(false);
    jPanelMenuBuscaEndereco.setVisible(false);
    jPanelBuscaPai.setVisible(false);
```

Fonte: O Autor.

Para o caso da tela inicial o único painel que não é definido como *setVisible(false)* é o do próprio *jFrame* e dessa forma a imagem anexada em um componente chamado *jLabelBase,* que pode ser visualizado na Figura 3.9, se sobrepõe aos demais painéis aparecendo na tela inicial da aplicação.

Figura 3.9: Sobreposição de Painéis.




Fonte: O Autor.

#### 3.2.1.1.2 Tela de Cadastro

A tela de cadastro funciona de forma similar à tela inicial, contudo ela é acionada por meio do clique no botão “NOVO” presente na tela inicial da Figura 3.7. Ao ser pressionado, o botão dispara um evento *mouseClicked.* Esse evento é importante para o desenvolvimento de praticamente todas as telas do *software* e é definido como um método *void,* ou seja, não retorna nenhum tipo de resultado, porém ele aplica algumas alterações e torna visíveis ou não os painéis por meio do método *setVisible* que pode receber como parâmetro verdadeiro ou falso.

Figura 3.10: Código de Acionamento dos Painéis de Novo Cadastro.

```
private void jLabelnovoMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    jPanelCadastro.setVisible(true);
    jPanelMenu.setVisible(false);
    jPanelMenuCadastro.setVisible(true);
    jPanelBuscaPai.setVisible(false);
    jPanelBuscaEndereco.setVisible(false);
    jPanelBuscaEnderecoEspecifico.setVisible(false);
    jPanelBuscaIntertvaloDatas.setVisible(false);
```

Fonte: O Autor

Dessa forma para a tela de cadastro o método *setVisible* é definido como *true* apenas para o *jPanelCadastro* e *jPanelMenuCadastro* como pode ser visto na Figura 3.10 e assim apenas esses dois painéis são apresentados ao utilizador do *software.*

Figura 3.11: Tela de Cadastros


Fonte: O Autor.

Na tela de cadastro podem ser inseridas as informações correspondentes ao questionário a ser digitalizado. Após o termino do preenchimento das informações o utilizador pode optar por “VOLTAR”, “CANCELAR” ou “SALVAR” o cadastro. Caso ele acione o botão “SALVAR”, novamente é disparado um evento do tipo *mouseClicked*, com isso são realizadas algumas validações, como por exemplo a verificação se já existe um cadastro armazenado no banco de dados para o endereço informado (método verificador a ser explicado em seções subsequentes) e também tratamentos para possíveis exceções para campos de datas ou inteiros. Caso exista

um cadastro para um endereço digitado não é criado um novo com as mesmas informações, mas sim atualizado o *array* de questionários e animais atrelados ao endereço. Após todas as validações, o questionário é inserido nas tabelas do banco de dados.

Outra possibilidade é o usuário clicar nos botões de “CANCELAR” ou “VOLTAR”, esses botões novamente disparam um evento *mouseClicked*, porém os dados digitados pelo usuário nos campos de texto são apagados e novamente o método *setVisible(true)* aciona ou o painel da tela inicial para o clique no botão “CANCELAR” ou o painel de cadastro para o clique no botão “VOLTAR”.

#### 3.2.1.1.3 Tela de Busca

A tela de busca serve para o usuário pesquisar se existe algum questionário já cadastrado no banco de dados. Essa busca de questionário pode ser feita por filtro de endereço, data, região ou animal. O funcionamento da tela de busca segue o mesmo padrão das telas anteriores. Na tela inicial da Figura 3.7 possui um botão “BUSCAR”, caso o utilizador clique nesse botão é disparado um evento *mouseClicked*, que por sua vez ativa o painel de busca como pode ser visualizado na Figura 3.12.

Figura 3.12: Código de Acionamento dos Painéis de Busca.

```
private void jLabelBuscarMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    jPanelMenu.setVisible(false);
    jPanelMenuBuscaEndereco.setVisible(false);
    jPanelBuscaPai.setVisible(true);
    jPanelBuscaEndereco.setVisible(false);
    jPanelCadastro.setVisible(false);
    jPanelBuscaEnderecoEspecifico.setVisible(false);
    jPanelBuscaIntertvaloDatas.setVisible(false);
    jPanelMenuBuscaData.setVisible(false);
```

Fonte: O Autor.

Na tela de busca o usuário pode escolher algum dos filtros mencionados anteriormente selecionando os botões disponíveis mostrados na Figura 3.13. A partir do clique em um dos botões de filtro um evento do tipo *void ActionPerformed* é disparado e a lógica de sobreposição de painéis é novamente usada, dessa vez ativando um painel específico para cada tipo de busca, como mostrado na Figura 3.14 para o caso de busca por bairro.

Figura 3.13: Tela de Buscas


Fonte: O Autor.

Figura 3.14: Código de Acionamento dos Painéis de Busca por Bairro.

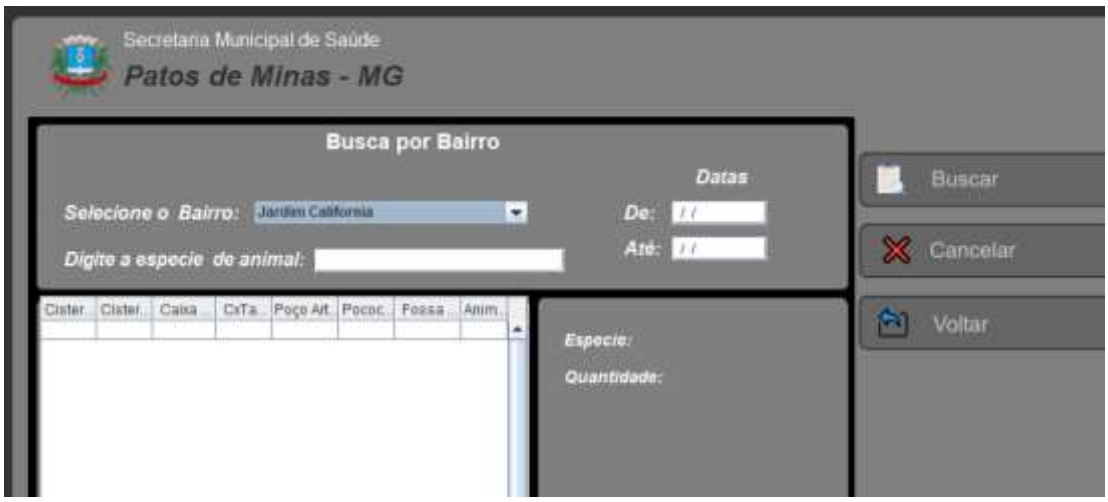
```
4198    private void jButtonBuscaBairroActionPerformed(java.awt.event.ActionEvent evt) {
4199        // TODO add your handling code here:
4200        jPanelMenu.setVisible(false);
4201        jPanelMenuBuscaEndereco.setVisible(false);
4202        jPanelBuscaPai.setVisible(false);
4203        jPanelBuscaEndereco.setVisible(false);
4204        jPanelCadastro.setVisible(false);
4205        jPanelBuscaEnderecoEspecifico.setVisible(false);
4206        jPanelBuscaIntertvaloDatas.setVisible(false);
4207        jPanelMenuBuscaData.setVisible(false);
4208        jPanelMenuBuscaEdicao.setVisible(false);
4209        jPanelMenuEditar.setVisible(false);
4210        jPanelEditar.setVisible(false);
4211        jPanelMenuBuscaPai.setVisible(false);
4212        jPanelBuscaBairro.setVisible(true);
4213        jPanelMenuBuscaBairro.setVisible(true);
```

Fonte: O Autor.

Tomando como exemplo uma busca por bairro, a partir do clique no botão “BAIRRO” o painel de busca por bairro é ativado e os demais anulados. O usuário pode selecionar qual bairro e espécie de animal ele deseja filtrar como pode ser visto na Figura 3.15. Ao clicar no botão “BUSCAR” algumas validações são feitas, como por exemplo se existem questionários para o bairro selecionado, ou se existe a espécie de animal informada. Existem também alguns tratamentos de possíveis exceções entre outras validações. Quando há um retorno nulo o usuário é informado de que não existem questionários com aquela espécie de animal para o bairro

informado. Porém se houver questionários com a espécie de animal cadastrados no banco de dados o usuário recebe uma contagem de quantos animais daquela espécie existem no bairro.

Figura 3.15: Tela de Busca por Bairro e Espécie de Animal.




Fonte: O Autor.

#### 3.2.1.1.4 Tela de Edição

Outra tela presente na versão virtual do questionário é a tela de edição. Essa funcionalidade de edição é essencial já que podem haver erros de digitação em questionários cadastrados pelo usuário. A tela de edição é acionada de maneira análoga às demais, sendo que na tela inicial há um botão chamado “EDITAR”. Esse botão ao ser acionado dispara um método *mouseClicked* que ativa o painel de edição como mostrado na Figura 3.16.

Figura 3.16: Código de Acionamento dos Painéis de Tela de Edição.

```
private void jLabelEditarMouseClicked(java.awt.event.MouseEvent evt)
    jPanelMenu.setVisible(false);
    jPanelMenuBuscaEndereco.setVisible(false);
    jPanelBuscaPai.setVisible(false);
    jPanelBuscaEndereco.setVisible(false);
    jPanelCadastro.setVisible(false);
    jPanelBuscaEnderecoEspecifico.setVisible(false);
    jPanelBuscaIntertvaloDatas.setVisible(false);
    jPanelMenuBuscaData.setVisible(false);
    jPanelMenuBuscaEdicao.setVisible(true);
    jPanelMenuBuscaRegiao.setVisible(false);
    jPanelMenuBuscaPai.setVisible(false);
    jPanelBuscaBairro.setVisible(false);
    jPanelMenuEditar.setVisible(false);
    jPanelEditar.setVisible(true);
```

Fonte: O Autor.

Na tela de edição é necessário informar qual o endereço correto e a data do questionário e a partir desse preenchimento o usuário pode clicar em “BUSCAR”, “VOLTAR” ou “CANCELAR” como mostrado na Figura 3.17. Se a opção selecionada for “BUSCAR”, novamente um evento *mouseClicked* é disparado e algumas validações são realizadas. Também é realizado um tratamento em relação a possíveis exceções na conversão de data. Posteriormente é efetuada a validação no banco de dados referente a existência ou não de questionários cadastrados para o endereço e data informados. Para o caso de retorno nulo, um *pop-up* com a informação de que o endereço informado não possui cadastros é apresentado para o usuário. Caso contrário, se houver um questionário cadastrado, os dados são preenchidos na tela como um cadastro a ser salvo, bastando o mesmo verificar onde é preciso realizar a alteração.

Figura 3.17: Tela de Edição.


Fonte: O Autor.

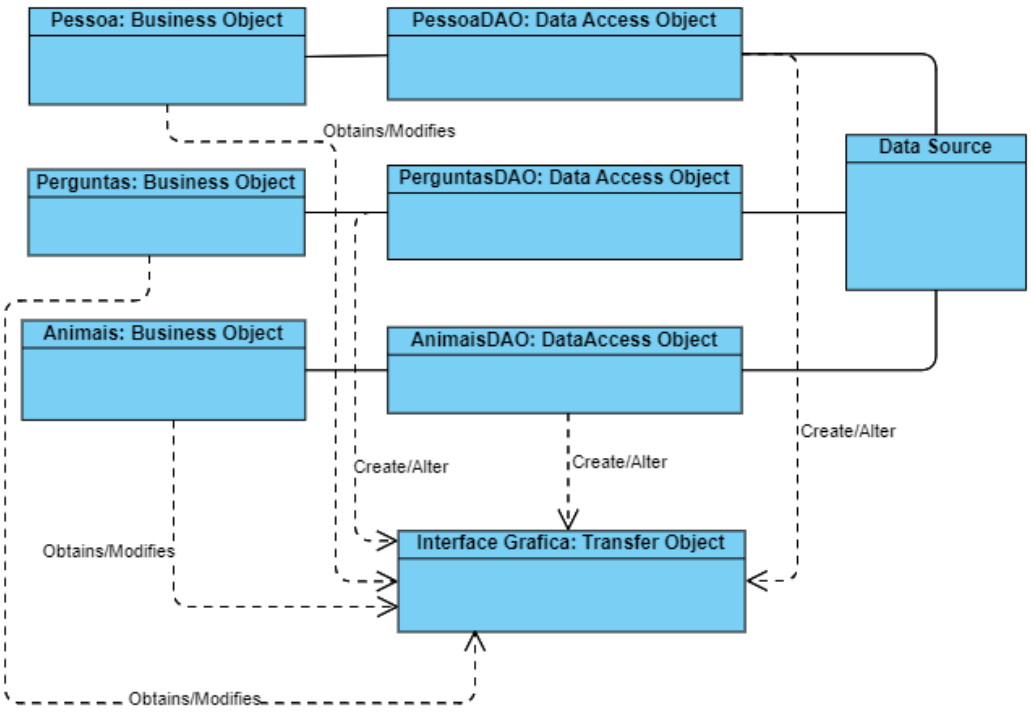
Após realizadas as alterações no questionário o usuário pode salvar o mesmo clicando no botão “SALVAR”. Esse botão dispara um evento *mouseClicked* que salva as alterações no

banco de dados de forma análoga ao evento de cadastro, mudando apenas a *query* no banco de dados, sendo enviado um comando de atualização.

### 3.2.1.2 Classes de Objetos e Classes de Acesso aos Dados (DAO)

Na sequência serão descritas as classes responsáveis por criar os objetos, encapsulá-los e persisti-los no banco de dados. Para o desenvolvimento da aplicação utilizou-se o padrão DAO descrito na seção 2.4.4. Esse padrão se destaca por implementar uma lógica de classes específicas para acesso aos dados persistidos. Dessa forma os botões presentes nas telas de cadastro, busca ou edição disparam eventos que acionam métodos das classes DAO, esses métodos recebem e encapsulam os objetos (pessoas, animais e perguntas) e enviam as instruções SQL para o banco de dados. As respostas das requisições feitas ao banco de dados são processadas e apresentadas de forma transparente, sem a necessidade de os eventos disparados nas telas de interface gráfica implementarem um código de acesso diretamente no banco de dados, ficando essa tarefa a cargo das classes de acesso aos dados (DAO). O diagrama de classes para o sistema desenvolvido pode ser visualizado na Figura 3.18.

Figura 3.18: Diagrama de Classes.




Fonte: O Autor.

#### 3.2.1.2.1 Classes de Objetos

Para o desenvolvimento do *software* de digitalização faz-se necessário converter as informações presentes no formulário em papel para o ambiente virtual. O formulário original da Figura 1.1 contém questionamentos sobre espécies de animais e formas de saneamento básico para os endereços. Dessa forma foram idealizadas algumas classes em Java que representassem os objetos do questionário original. As classes de objetos são descritas a seguir.

##### 3.2.1.2.1.1 Classe Animal

A classe Animal contém os atributos de espécie, quantidade e data como mostrado na Figura 3.19. Esses atributos são necessários pois no formulário é preciso descrever a espécie e a quantidade de animais, além disso as informações sobre a presença de animais no endereço estão vinculadas a uma data em que foi feita a entrevista com o morador.

Figura 3.19: Classe Animal.

```
public class Animais {
    private String especie;
    private int quantidade;
    private Date data;

    public Animais() {...2 linhas }
    public Animais(String especie, int quantidade ,Date data) {...5 linhas }
    public String getEspecie() {...3 linhas }
    public void setEspecie(String especie) {...3 linhas }
    public int getQuantidade() {...3 linhas }
    public void setQuantidade(int quantidade) {...3 linhas }
    public Date getData() {...3 linhas }
    public void setData(Date data) {...3 linhas }
    @Override
    public String toString() {...3 linhas }
}
```

Fonte: O Autor.

##### 3.2.1.2.1.2 Classe Pergunta

A classe Pergunta contém os atributos de capacidade, data do questionário e oito atributos booleanos como pode ser visualizado na Figura 3.20. De maneira similar aos questionamentos sobre animais, uma pergunta também está atrelada a data que o questionário é realizado.

Figura 3.20: Classe Pergunta.

```
public class Perguntas {

    private boolean cisterna , cisternaconsumo , cxdagua , tampada , pcartesiano , pococonsumo , fseptica , animais;
    private int capacidade;
    private Date data;

    public Perguntas() {...2 linhas }
    public Perguntas(boolean cisterna, boolean cisternaconsumo, boolean cxdagua, boolean tampada, boolean pcartesiano,
    public boolean isCisterna() {...3 linhas }
    public void setCisterna(boolean cisterna) {...3 linhas }
    public boolean isCisternaconsumo() {...3 linhas }
    public void setCisternaconsumo(boolean cisternaconsumo) {...3 linhas }
    public boolean isCxdagua() {...3 linhas }
    public void setCxdagua(boolean cxdagua) {...3 linhas }
    public boolean isTampada() {...3 linhas }
    public void setTampada(boolean tampada) {...3 linhas }
    public boolean isPcartesiano() {...3 linhas }
    public void setPcartesiano(boolean pcartesiano) {...3 linhas }
    public boolean isPococonsumo() {...3 linhas }
    public void setPococonsumo(boolean pococonsumo) {...3 linhas }
    public boolean isFseptica() {...3 linhas }
    public void setFseptica(boolean fseptica) {...3 linhas }
    public boolean isAnimais() {...3 linhas }
    public void setAnimais(boolean animais) {...3 linhas }
    public int getCapacidade() {...3 linhas }
    public void setCapacidade(int capacidade) {...3 linhas }
    public Date getData() {...3 linhas }
    public void setData(Date data) {...3 linhas }
    @Override
    public String toString() {...3 linhas }
```

Fonte: O Autor.

### 3.2.1.2.1.3 Classe Pessoa

A classe Pessoa que pode ser visualizada na Figura 3.21, na verdade representa o endereço, uma vez que os questionários não podem identificar o nome do morador. Nessa classe os atributos endereço (rua), bairro, cidade e região correspondem ao endereço. Existe também um atributo id que é utilizado para vinculação dos dados, por esse campo é possível vincular um endereço as perguntas e animais correspondentes por meio do conceito de chave primária e chave estrangeira no banco de dados. Outros dois atributos presentes nessa classe Pessoa são os *arraylists* de perguntas e animais que podem armazenar todas as perguntas e animais para o endereço, porém todas com datas diferentes.

Figura 3.21: Classe Pessoa.

```
public class Pessoa {
    private int id ;
    private String  endereco , bairro , cidade , regiao , numero;
    private ArrayList<Animais> listaAnimais = new ArrayList();
    private ArrayList<Perguntas> listaPerguntas = new ArrayList();
    public Pessoa() {...2 linhas }
    public Pessoa(String numero, String nome, String endereco, String bairro, String cidade, String regiao)
    public Pessoa(int id ,String numero, String nome, String endereco, String bairro, String cidade, String
    public int somaAnimais(Date data){...8 linhas }
    public int getId() {...3 linhas }
    public void setId(int id) {...3 linhas }
    public String getNumero() {...3 linhas }
    public void setNumero(String numero) {...3 linhas }
    public String getEndereco() {...3 linhas }
    public void setEndereco(String endereco) {...3 linhas }
    public String getBairro() {...3 linhas }
    public void setBairro(String bairro) {...3 linhas }
    public String getCidade() {...3 linhas }
    public void setCidade(String cidade) {...3 linhas }
    public String getData() {...3 linhas }
    public void setData(String data) {...3 linhas }
    public String getRegiao() {...3 linhas }
    public void setRegiao(String regiao) {...3 linhas }
    public ArrayList<Animais> getListaAnimais() {...3 linhas }
    public void setListaAnimais(ArrayList<Animais> listaAnimais) {...3 linhas }
    public ArrayList<Perguntas> getListaPerguntas() {...3 linhas }
    public void setListaPerguntas(ArrayList<Perguntas> listaPerguntas) {...3 linhas }
    @Override
    public String toString() {...3 linhas }
    }
```

Fonte: O Autor.

Nas seções anteriores foram descritas as classes de objetos juntamente com os atributos usados para a criação do questionário no formato digital. As seções seguintes abordam o padrão de projeto utilizado para encapsular os dados e persisti-los no banco de dados.

#### 3.2.1.2.2 Acesso aos Dados (DAO)

O padrão DAO é utilizado para abstrair e encapsular os acessos ao banco de dados. As classes DAO gerenciam as conexões com o banco de dados para inserir, obter e atualizar as informações necessárias. Para o questionário digital têm-se as tabelas do banco de dados que armazenam os dados dos objetos Animais, Perguntas e Pessoas. Para a inserção ou atualização desses dados nas tabelas foram idealizadas três classes, são elas: PessoaDAO, AnimaisDAO e PerguntasDAO. Cada classe DAO implementa os métodos de inserção, atualização e busca dos dados armazenados. A seguir serão descritos alguns dos principais métodos presentes em cada classe DAO.

#### 3.2.1.2.2.1 Classe PessoaDAO

Na classe PessoaDAO são implementados os métodos responsáveis por realizar o encapsulamento das informações de cada objeto Pessoa e enviá-las para as tabelas do banco de dados. As consultas ou alterações são implementadas por meio de instruções SQL presentes nos métodos da classe e que aplicam os comandos de criação, leitura, atualização e deleção (CRUD), sendo que na linguagem SQL esses comandos são definidos como INSERT, SELECT, UPDATE e DELETE. O método de inserção de objetos pessoas no banco de dados é definido como *insertPessoas* e é mostrado na Figura 3.22.

Figura 3.22: Método insertPessoas.

```
public int insertPessoas(Pessoa pessoa) {
    int id = 0;
    String sql = "INSERT INTO tbl_pessoas (numero,endereco,bairro,"
            + "cidade,regiao) VALUES (?,?,?,?,?,?)";
    PreparedStatement stmt = null;
    ResultSet rs = null;

    try {
        stmt = con.prepareStatement(sql, stmt.RETURN_GENERATED_KEYS);
        stmt.setString(1, pessoa.getNumero());
        stmt.setString(3, pessoa.getEndereco());
        stmt.setString(4, pessoa.getBairro());
        stmt.setString(5, pessoa.getCidade());
        stmt.setString(6, pessoa.getRegiao());
        stmt.executeUpdate();
        rs = stmt.getGeneratedKeys();
        if (rs.next()) {
            id = rs.getInt(1);
        }
        return id;
    } catch (SQLException ex) {
        System.err.println("Erro:" + ex);
    } finally {
        ConnectionFactory.closeConnetion(con, stmt);
    }
    return id;
}
```

Fonte: O Autor.

Outro método implementado na classe PessoaDAO é o método verificador. Esse método recebe como parâmetro um objeto pessoa como mostrado na Figura 3.23, posteriormente realiza uma validação por meio do banco de dados para verificar se há a existência de um registro com

as mesmas informações do objeto recebido como parâmetro. Caso o retorno do banco seja diferente de nulo é apresentada uma mensagem para o usuário informando que já possui um cadastro para o endereço e perguntando se a intenção é adicionar mais questionários para o mesmo endereço. Já para o caso de um retorno nulo do banco de dados o método de inserção é acionado e o cadastro é inserido na tabela de pessoas.

Figura 3.23: Método Verificador.

```
public Pessoa verificador(Pessoa pessoa) {
    String sql = "SELECT * FROM tbl_pessoas WHERE LOWER(numero) = LOWER(?) "
            + "AND LOWER(endereco) = LOWER(?) AND LOWER(bairro) = LOWER(?) "
            + "AND LOWER(cidade) = LOWER(?) AND LOWER(regiao) = LOWER(?)";
    PreparedStatement stmt = null;
    ResultSet rs = null;
    try {
        stmt = con.prepareStatement(sql);
        stmt.setString(1, pessoa.getNumero());
        stmt.setString(2, pessoa.getEndereco());
        stmt.setString(3, pessoa.getBairro());
        stmt.setString(4, pessoa.getCidade());
        stmt.setString(5, pessoa.getRegiao());
        rs = stmt.executeQuery();
        if (rs != null && rs.next()) {
            Pessoa p = new Pessoa();
            p.setId(Integer.parseInt(rs.getString("idpessoas")));
            p.setNumero((rs.getString("numero")));
            p.setEndereco(rs.getString("endereco"));
            p.setBairro(rs.getString("bairro"));
            p.setCidade(rs.getString("cidade"));
            p.setData(rs.getString("regiao"));
            return p;
        }
    } catch (SQLException ex) {
        System.err.println("Erro:" + ex);
    }
    return null;
}
```

Fonte: O Autor.

Existe ainda um método implementado na classe PessoaDAO que é bastante útil, definido como deletePessoa. Esse método recebe como parâmetro um objeto Pessoa a ser excluído do banco de dados. A partir dos campos do objeto Pessoa uma *query* é encapsulada e enviada para o banco de dados como pode ser visto na Figura 3.24. Caso exista um registro com os dados constantes na *query* o registro é então deletado do banco e retornado um atributo booleano *true.* Caso contrário, o retorno é um atributo *booleano false*

Figura 3.24: Método deletePessoa.

```
public boolean deletePessoa(Pessoa pessoa) {
    String sql = "DELETE FROM tbl_pessoas WHERE LOWER(numero) = LOWER(?) "
            + "AND LOWER(endereco) = LOWER(?) AND LOWER(bairro) = LOWER(?) "
            + "AND LOWER(cidade) = LOWER(?) AND LOWER(regiao) = LOWER(?)";
    PreparedStatement stmt = null;
    try {
        stmt = con.prepareStatement(sql);
        stmt.setString(1, pessoa.getNumero());
        stmt.setString(2, pessoa.getEndereco());
        stmt.setString(3, pessoa.getBairro());
        stmt.setString(4, pessoa.getCidade());
        stmt.setString(5, pessoa.getRegiao());
        stmt.executeUpdate();
        return true;
    } catch (SQLException ex) {
        System.err.println("Erro:" + ex);
        return false;
    } finally {
        ConnectionFactory.closeConnetion(con, stmt);
    }
}
```

Fonte: O Autor.

#### 3.2.1.2.2.2 Classe AnimaisDAO

A classe AnimaisDAO é utilizada para implementar os métodos de acesso ao banco de dados no que se refere a inserção ou alterações de registros de objetos Animal. Esses objetos ficam armazenados em uma lista presente no objeto Pessoa, dessa forma sempre que é necessário inserir ou alterar um registro de animais no banco de dados é preciso informar um objeto Pessoa que está atrelado ao objeto Animal.

Dentre os métodos que podem ser destacados para essa classe tem-se o de inserção e de soma de quantidade de animais por bairro. O método de inserção de animais recebe como parâmetro um objeto Pessoa, verifica a quantidade de objetos Animais presentes na lista de animais e adiciona todos esses objetos a tabela de animais do banco de dados como pode ser visto na Figura 3.25.

Figura 3.25: Método insertAnimais.

```
public void insertAnimais(Pessoa pessoa) {
    String sql = "INSERT INTO tbl_animais (idanimais,"
            + "especie,quantidade,dataanimais) VALUES (?,?,?,?)";

    PreparedStatement stmt = null;
    for (int i = 0; i < pessoa.getListaAnimais().size(); i++) {
        try {
            stmt = con.prepareStatement(sql);
            stmt.setInt(1, pessoa.getId());
            stmt.setString(2, pessoa.getListaAnimais().get(i).getEspecie());
            stmt.setInt(3, pessoa.getListaAnimais().get(i).getQuantidade());
            java.sql.Date data = new java.sql.Date(pessoa.getListaAnimais().get(i).getData().getTime());
            stmt.setDate(4, data);
            stmt.executeUpdate();
        } catch (SQLException ex) {
            System.err.println("Erro:" + ex);
        }
    }
}
```

Fonte: O Autor.

O método de soma de espécies de animais por bairro mostrado na Figura 3.26 é uma possibilidade de automatização que a digitalização dos questionários proporciona. Esse método busca no banco de dados um somatório de quantos animais de determinada espécie foram cadastrados para um bairro especifico. Esse resultado pode ajudar a entender melhor e a controlar o crescimento de determinada espécie em uma região.

Figura 3.26: Método *somaEspecieBairro*.

```
public int somaEspecieBairro(String bairro, Date dataIncial, Date dataFinal, String especie) {

    String sql = "SELECT sum(quantidade) FROM tbl_pessoas INNER JOIN tbl_animais ON "
            + "tbl_pessoas.idpessoas = tbl_animais.idanimais WHERE LOWER(especie) = LOWER(?) "
            + "AND bairro = ? AND dataanimais BETWEEN ? AND ?";
    PreparedStatement stmt = null;
    ResultSet rs = null;
    int soma = 0;
    try {
        stmt = con.prepareStatement(sql);
        stmt.setString(1, especie);
        stmt.setString(2, bairro);
        stmt.setDate(3, dataIncial);
        stmt.setDate(4, dataFinal);
        rs = stmt.executeQuery();
        while (rs.next()) {
            soma = rs.getInt("sum(quantidade)");
        }
    } catch (SQLException ex) {
        System.err.println("Erro:" + ex);
    } finally {
        ConnectionFactory.closeConnetion(con, stmt, rs);
    }
    return soma;
}
```

Fonte: O Autor

Para o cálculo da quantidade de uma espécie de animal por bairro o método recebe como parâmetro o bairro, o intervalo de datas e a espécie. Uma consulta SQL utiliza uma cláusula INNER JOIN que faz a união de todos os registros de objetos Pessoa que possuem animais atrelados. Essa cláusula é necessária pois possibilita a verificação de qual bairro o objeto animal está vinculado. Na mesma instrução SQL é realizado o cálculo da quantidade de animais cadastrados no banco de dados para a espécie e datas informados.

#### 3.2.1.2.2.3 Classe PerguntasDAO

A classe PerguntasDAO é utilizada para acesso aos registros armazenados no banco de dados referente as respostas dos questionários. O método utilizado para efetuar a inserção dos dados de perguntas na tabela do banco de dados recebe como parâmetro um objeto Pessoa que por sua vez possui uma lista de perguntas atreladas. Como mostrado na Figura 3.27, uma instrução SQL encapsula todas os objetos Perguntas presentes na lista de perguntas do objeto Pessoa e as insere no banco de dados.

Figura 3.27: Método *insertPerguntas*.

```
public void insertPerguntas(Pessoa pessoa) {
    String sql = "INSERT INTO tbl_perguntas (idperguntas,cisterna,cisternaconsumo,"
            + "cxdagua,tampada,pcartesiano,pococonsumo,fseptica,animais,capacidade,"
            + "datapergunta) VALUES (?,?,?,?,?,?,?,?,?,?,?)";
    PreparedStatement stmt = null;
    try {
        for (int i = 0; i < pessoa.getListaPerguntas().size(); i++) {
            stmt = con.prepareStatement(sql);
            stmt.setInt(1, pessoa.getId());
            if (pessoa.getListaPerguntas().get(i).isCisterna() == true) {
                stmt.setInt(2, 1);
            } else {
                stmt.setInt(2, 0);
            }
            if (pessoa.getListaPerguntas().get(i).isCisternaconsumo() == true) {
                stmt.setInt(3, 1);
            } else {
                stmt.setInt(3, 0);
            }
            if (pessoa.getListaPerguntas().get(i).isCxdagua() == true) {
                stmt.setInt(4, 1);
            } else {
                stmt.setInt(4, 0);
            }
            if (pessoa.getListaPerguntas().get(i).isTampada() == true) {
                stmt.setInt(5, 1);
            } else {
                stmt.setInt(5, 0);
            }
            if (pessoa.getListaPerguntas().get(i).isPcartesiano() == true) {
                stmt.setInt(6, 1);
            } else {
```

Fonte: O Autor.

A seguir serão abordados os procedimentos usados para a criação das conexões com o banco de dados e como foram criadas as tabelas para persistência dos dados.

## 3.2.2 Banco de Dados

### 3.2.2.1 Conexão Com o Banco de Dados

A linguagem Java possui alguns *drivers* que proporcionam integrações entre *softwares* baseados em Java com sistemas externos, um desses *drivers* é o conector JDBC MySQL. O conector JDBC é um arquivo Java com a extensão .jar que pode ser adicionado ao projeto dentro dos pacotes de classes. O *driver* JDBC utiliza uma URL para acessar o banco de dados. Essa URL consiste em uma *string* contendo informações sobre o endereço do servidor onde o banco de dados está sendo executado, a porta, nome de usuário, nome do banco de dados e senha.

Figura 3.28: Classe Para Conexão Com o Banco de Dados.

```
public class ConnectionFactory {
    public static Connection getConnection() {

        java.sql.Connection conexao;
        //classe do driver
        String driver = "com.mysql.jdbc.Driver";
        // informações sobre o banco
        String url = "jdbc:mysql://localhost:3306/bancosecsaude?"
                + "useTimezone=true&serverTimezone=UTC&SSL=false";
        String user = "root";
        String password = "41621ETE002";
        // estabelecendo a conexão com o banco de dados
        try {
            Class.forName(driver);
            conexao = DriverManager.getConnection(url, user, password);
            System.out.println("Conectou-se ao banco !");
            return conexao;
        } catch (java.lang.ClassNotFoundException e) {
            System.err.println("Driver não existe !");
            return null;
        } catch (java.sql.SQLException e) {
            System.err.println("Não conectou !");
            return null;
        }
    }
    public static void closeConnetion(Connection con) {...12 linhas }
    public static void closeConnetion(Connection con, PreparedStatement stmt)
    public static void closeConnetion(Connection con, PreparedStatement stmt,
}
```

Fonte: O Autor.

Quando se utiliza o *driver* JDBC para se conectar a um servidor de banco de dados que não esteja hospedado no próprio servidor da aplicação, a classe *DriverManager* é quem gerencia o estabelecimento da conexão. Para isso é preciso especificar para essa classe qual o *driver* que

será usado na conexão, no caso do banco de dados MySQL é preciso informar a classe do *driver* como sendo com.mysql.jdbc.Driver. Após o registro do *driver* na classe *DriverManager* uma conexão pode ser estabelecida com o banco de dados acionando o método *DriverManager.getConnection*.

Para o sistema gerenciador de cadastros de questionários é implementada uma classe que usa os conceitos descritos acima como pode ser visto na Figura 3.28. Uma classe chamada ConnectionFactory implementa um método estático que retorna uma conexão com o banco de dados. Sempre que as classes de objetos de acesso aos dados (DAO) precisam iniciar uma conexão com o banco esse método é utilizado. Esse método possui uma URL com as informações de conexão com o servidor MySQL e usa a classe *DriverManager* para estabelecer a conexão com o banco de dados.

#### 3.2.2.2 Tabelas Do Banco de Dados

Em sistemas de banco de dados relacionais como o MySQL, uma tabela pode ser definida como o conjunto de dados dispostos em linhas e colunas que se auto relacionam. Os dados das colunas da tabela representam os campos ou atributos do objeto, sendo que cada coluna possui um tipo especifico de dado, como por exemplo dados numéricos, alfanuméricos, datas e dados de localização. As linhas representam os registros ou tuplas e podem ter um número ilimitado de combinações. Os objetos inseridos ocupam as linhas da tabela e são caracterizados pelos campos ou colunas.

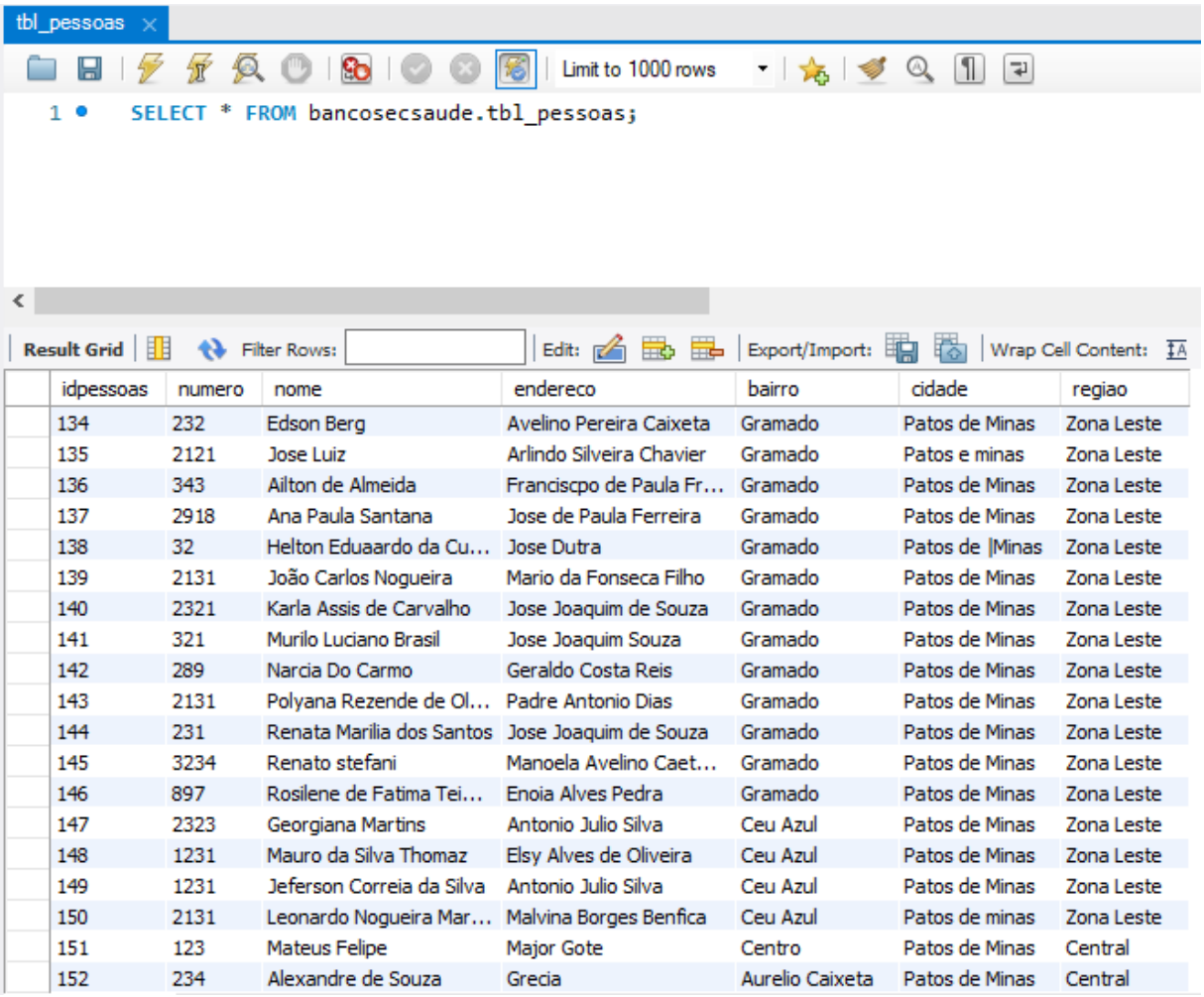
Para o questionário digital são implementadas as classes de objetos Pessoa, Animais e Perguntas. Sendo assim, faz se necessário a construção de tabelas no banco de dados para que os dados desses objetos possam ser persistidos e acessados. Dessa forma as seguintes tabelas são definidas:

- tabela de pessoas;
- tabela de animais;
- tabela de perguntas;

A tabela de pessoas pode ser vista na Figura 3.29 onde os dados persistidos, como por exemplo nome de pessoas, são dados fictícios. Essa tabela armazena os dados dos endereços das residências como número (varchar), nome (varchar), endereço (varchar), bairro (varchar), cidade (varchar) e região (varchar). A tabela de pessoas possui ainda a

coluna idpessoas (inteiro) que é usada como chave primária para vincular um endereço aos respectivos animais e perguntas.

Figura 3.29: Tabela de pessoas.

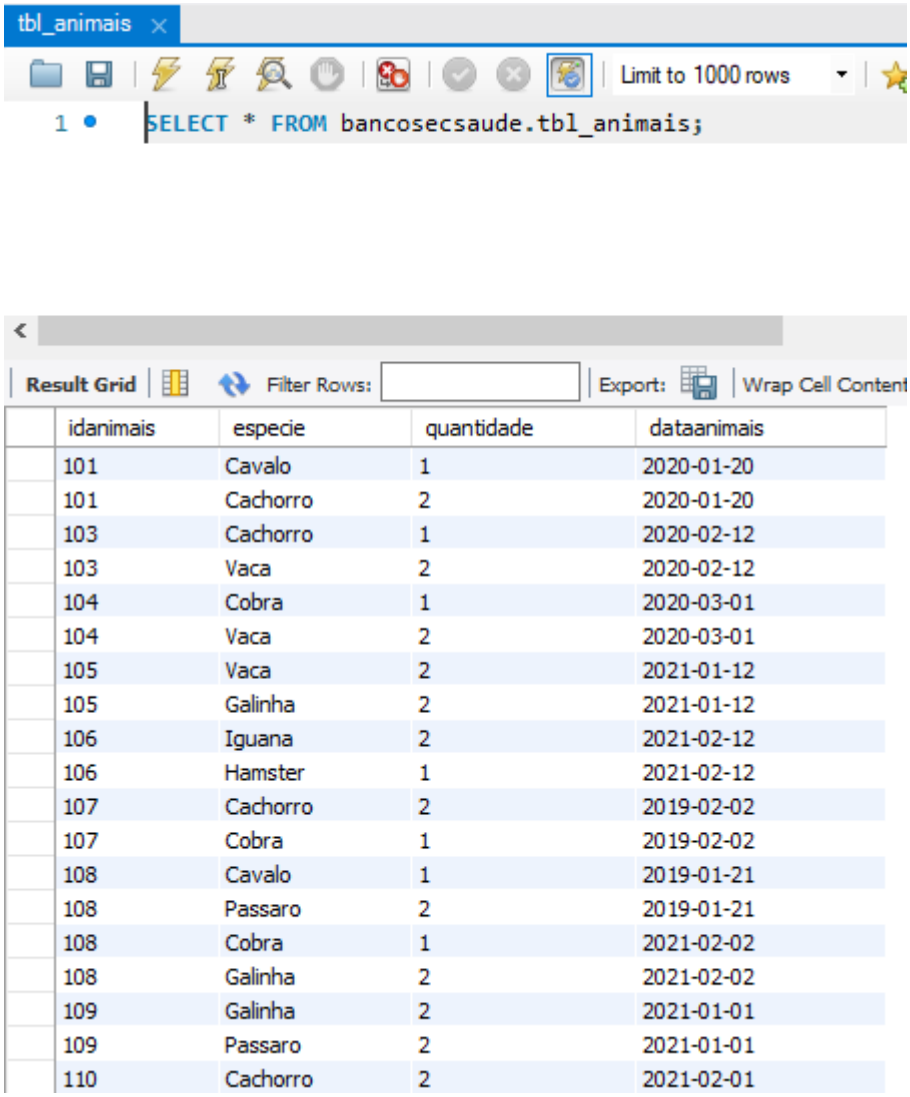



| idpessoas | numero | nome | endereco | bairro | cidade | regiao |
|---|---|---|---|---|---|---|
| 134 | 232 | Edson Berg | Avelino Pereira Caixeta | Gramado | Patos de Minas | Zona Leste |
| 135 | 2121 | Jose Luiz | Arlindo Silveira Chavier | Gramado | Patos e minas | Zona Leste |
| 136 | 343 | Ailton de Almeida | Franciscpo de Paula Fr... | Gramado | Patos de Minas | Zona Leste |
| 137 | 2918 | Ana Paula Santana | Jose de Paula Ferreira | Gramado | Patos de Minas | Zona Leste |
| 138 | 32 | Helton Eduaardo da Cu... | Jose Dutra | Gramado | Patos de \|Minas | Zona Leste |
| 139 | 2131 | João Carlos Nogueira | Mario da Fonseca Filho | Gramado | Patos de Minas | Zona Leste |
| 140 | 2321 | Karla Assis de Carvalho | Jose Joaquim de Souza | Gramado | Patos de Minas | Zona Leste |
| 141 | 321 | Murilo Luciano Brasil | Jose Joaquim Souza | Gramado | Patos de Minas | Zona Leste |
| 142 | 289 | Narcia Do Carmo | Geraldo Costa Reis | Gramado | Patos de Minas | Zona Leste |
| 143 | 2131 | Polyana Rezende de Ol... | Padre Antonio Dias | Gramado | Patos de Minas | Zona Leste |
| 144 | 231 | Renata Marilia dos Santos | Jose Joaquim de Souza | Gramado | Patos de Minas | Zona Leste |
| 145 | 3234 | Renato stefani | Manoela Avelino Caet... | Gramado | Patos de Minas | Zona Leste |
| 146 | 897 | Rosilene de Fatima Tei... | Enoia Alves Pedra | Gramado | Patos de Minas | Zona Leste |
| 147 | 2323 | Georgiana Martins | Antonio Julio Silva | Ceu Azul | Patos de Minas | Zona Leste |
| 148 | 1231 | Mauro da Silva Thomaz | Elsy Alves de Oliveira | Ceu Azul | Patos de Minas | Zona Leste |
| 149 | 1231 | Jeferson Correia da Silva | Antonio Julio Silva | Ceu Azul | Patos de Minas | Zona Leste |
| 150 | 2131 | Leonardo Nogueira Mar... | Malvina Borges Benfica | Ceu Azul | Patos de minas | Zona Leste |
| 151 | 123 | Mateus Felipe | Major Gote | Centro | Patos de Minas | Central |
| 152 | 234 | Alexandre de Souza | Grecia | Aurelio Caixeta | Patos de Minas | Central |

Fonte: O Autor.

A tabela de animais mostrada na Figura 3.30 serve para armazenar as informações dos animais para cada questionário. Essa tabela possui as colunas de espécie (varchar), quantidade (inteiro) e data (date), além de uma coluna idanimais (inteiro) que é uma chave estrangeira do idpessoas e serve para vinculação do animal ao respectivo endereço.

Figura 3.30: Tabela de animais.

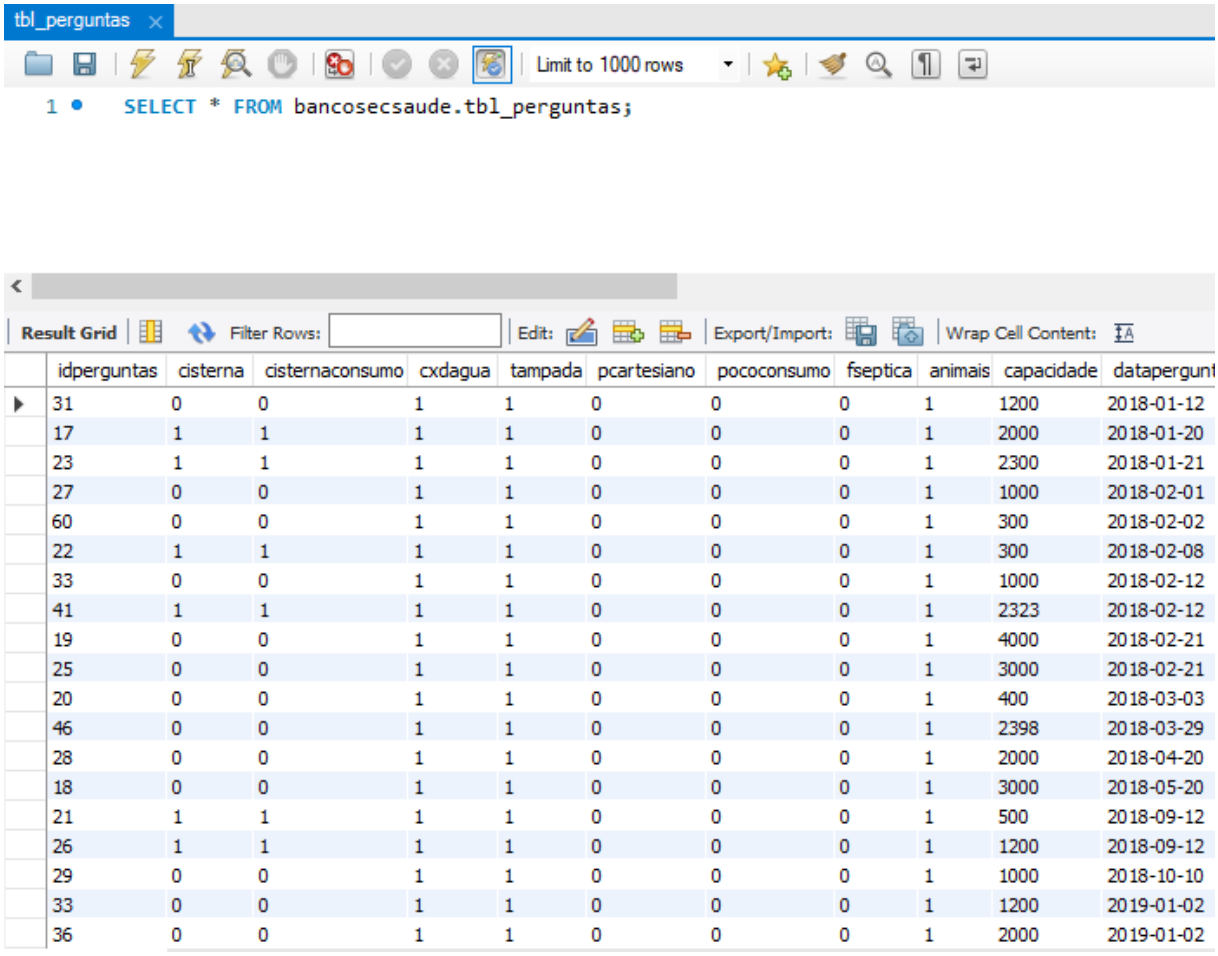



| idanimais | especie | quantidade | dataanimais |
|---|---|---|---|
| 101 | Cavalo | 1 | 2020-01-20 |
| 101 | Cachorro | 2 | 2020-01-20 |
| 103 | Cachorro | 1 | 2020-02-12 |
| 103 | Vaca | 2 | 2020-02-12 |
| 104 | Cobra | 1 | 2020-03-01 |
| 104 | Vaca | 2 | 2020-03-01 |
| 105 | Vaca | 2 | 2021-01-12 |
| 105 | Galinha | 2 | 2021-01-12 |
| 106 | Iguana | 2 | 2021-02-12 |
| 106 | Hamster | 1 | 2021-02-12 |
| 107 | Cachorro | 2 | 2019-02-02 |
| 107 | Cobra | 1 | 2019-02-02 |
| 108 | Cavalo | 1 | 2019-01-21 |
| 108 | Passaro | 2 | 2019-01-21 |
| 108 | Cobra | 1 | 2021-02-02 |
| 108 | Galinha | 2 | 2021-02-02 |
| 109 | Galinha | 2 | 2021-01-01 |
| 109 | Passaro | 2 | 2021-01-01 |
| 110 | Cachorro | 2 | 2021-02-01 |

Fonte: O Autor.

Existe também a tabela de perguntas que é usada para persistir as respostas do questionário. Essa tabela possui as colunas cisterna (tinyint), cisternaconsumo (tinyint), cxdagua (tinyint), tampada (tinyint), pcartesiano (tinyint), pococonsumo (tinyint), fseptica (tinyint), animais (tinyint), capacidade (inteiro), dataperguntas (date) e idperguntas (inteiro). A maioria dos dados da pergunta são do tipo booleano (verdadeiro ou falso), porém como pode ser visualizado na Figura 3.31, no banco de dados é atribuido o valor 1 quando a resposta for verdadeira e 0 quando a resposta for falsa.

Figura 3.31: Tabela de perguntas.




| idperguntas | cisterna | cisternaconsumo | cxdagua | tampada | pcartesiano | pococonsumo | fseptica | animais | capacidade | datapergunt |
|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1200 | 2018-01-12 |
| 17 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 2000 | 2018-01-20 |
| 23 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 2300 | 2018-01-21 |
| 27 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1000 | 2018-02-01 |
| 60 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 300 | 2018-02-02 |
| 22 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 300 | 2018-02-08 |
| 33 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1000 | 2018-02-12 |
| 41 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 2323 | 2018-02-12 |
| 19 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 4000 | 2018-02-21 |
| 25 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 3000 | 2018-02-21 |
| 20 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 400 | 2018-03-03 |
| 46 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 2398 | 2018-03-29 |
| 28 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 2000 | 2018-04-20 |
| 18 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 3000 | 2018-05-20 |
| 21 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 500 | 2018-09-12 |
| 26 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1200 | 2018-09-12 |
| 29 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1000 | 2018-10-10 |
| 33 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1200 | 2019-01-02 |
| 36 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 2000 | 2019-01-02 |

Fonte: O Autor.

## 3.3 Análise dos Dados

Na Seção 2.6 são descritas as principais características do conceito de análise de dados em que a partir de um conjunto de dados armazenados é possível obter conhecimento. A Figura 2.10 traz uma exemplificação das etapas necessárias para o processo de KDD, que simplificadamente é um conjunto de atividades que visa identificar nos dados novos padrões que sejam válidos, potencialmente úteis e interpretáveis. O *Business Inteligence* é uma forma de descoberta de conhecimento de maneira descritiva. Nessa metodologia os dados são tratados, processados e apresentados na maioria das vezes de forma visual, em formato de *dashboards*. Uma ferramenta gratuita que vem se difundido e facilitando a criação de *dashboard*s é o Power BI, uma ferramenta fácil de ser instalada e utilizada.

A seguir são descritas as metodologias necessárias para a criação de *dashboards* no *Power BI* a partir de um banco de dados local. Essa seção serve como uma proposta de ferramenta de

análise de dados para quem venha a utilizar o *software* de digitalização de questionários, visto que os dados armazenados podem ser tratados e apresentados de forma a extrair conhecimento.

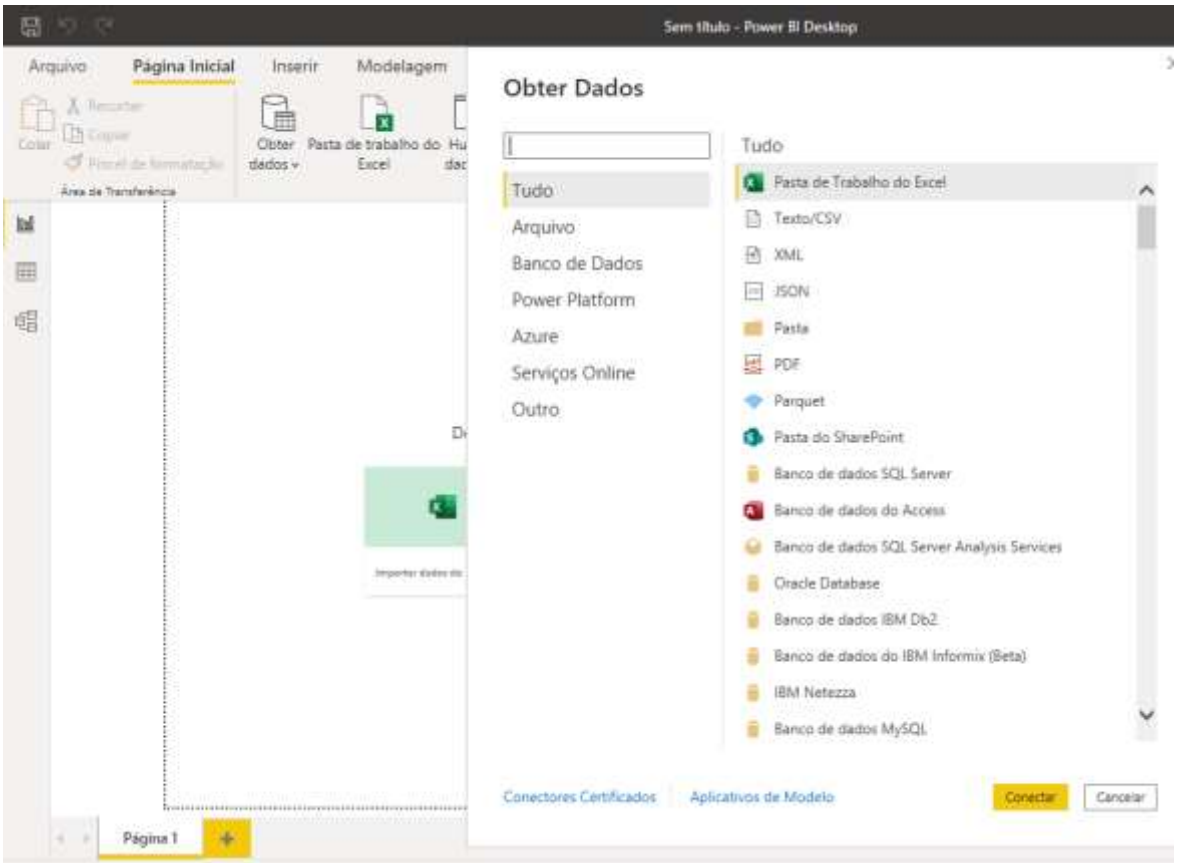
### 3.3.1 Criação de *Dashboards* Com Power BI

O *Power BI desktop* possui uma versão gratuita que pode ser instalada em computadores com sistema operacional *Windows*. Para fazer o *download* dessa versão basta ir ao *site* do *Microsoft Power BI* e baixar a versão que se adequa ao sistema operacional da máquina a ser usada, existem as versões de 32 e 64 bits.

#### 3.3.1.1 Obtenção dos Dados

*O Power BI* possibilita a conexão com vários tipos de fontes de dados, incluindo pastas de trabalho do Excel, arquivos CSV, banco de dados e serviços em nuvem. Para a obtenção dos dados é preciso selecionar o campo “Obter Dados” e optar por uma das fontes de dados disponíveis como mostrado na Figura 3.32.

Figura 3.32: Obter dados *Power BI.*

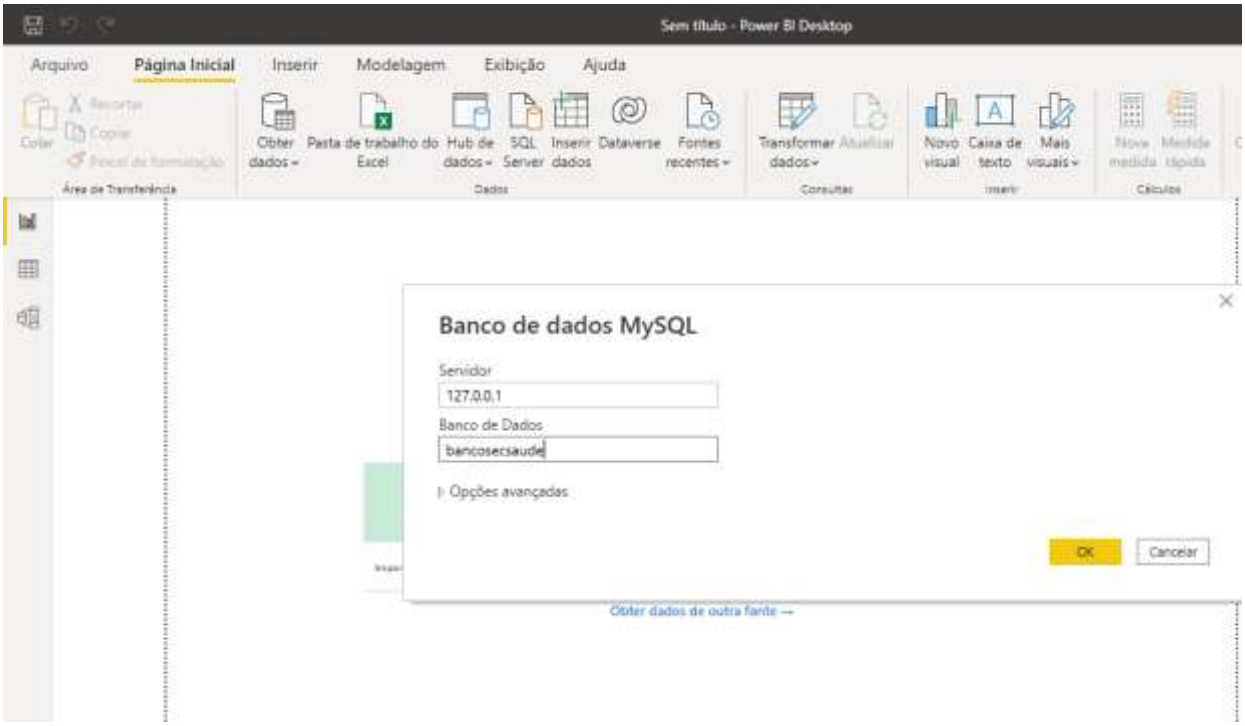



Fonte: O Autor.

Após selecionar o tipo de fonte de dados é preciso informar o local onde a mesma está armazenada. Quando a fonte de dados é um banco de dados, é necessário informar qual o

endereço do servidor e qual o nome do banco de dados. Dessa forma, para a obtenção de dados de um banco de dados MySQL é necessário selecionar a opção banco de dados MySQL nas opções de fontes de dados e indicar o endereço IP e o nome do banco de dados como exemplificado pela Figura 3.33.

Figura 3.33: Conectando *Power BI* ao Banco de Dados.

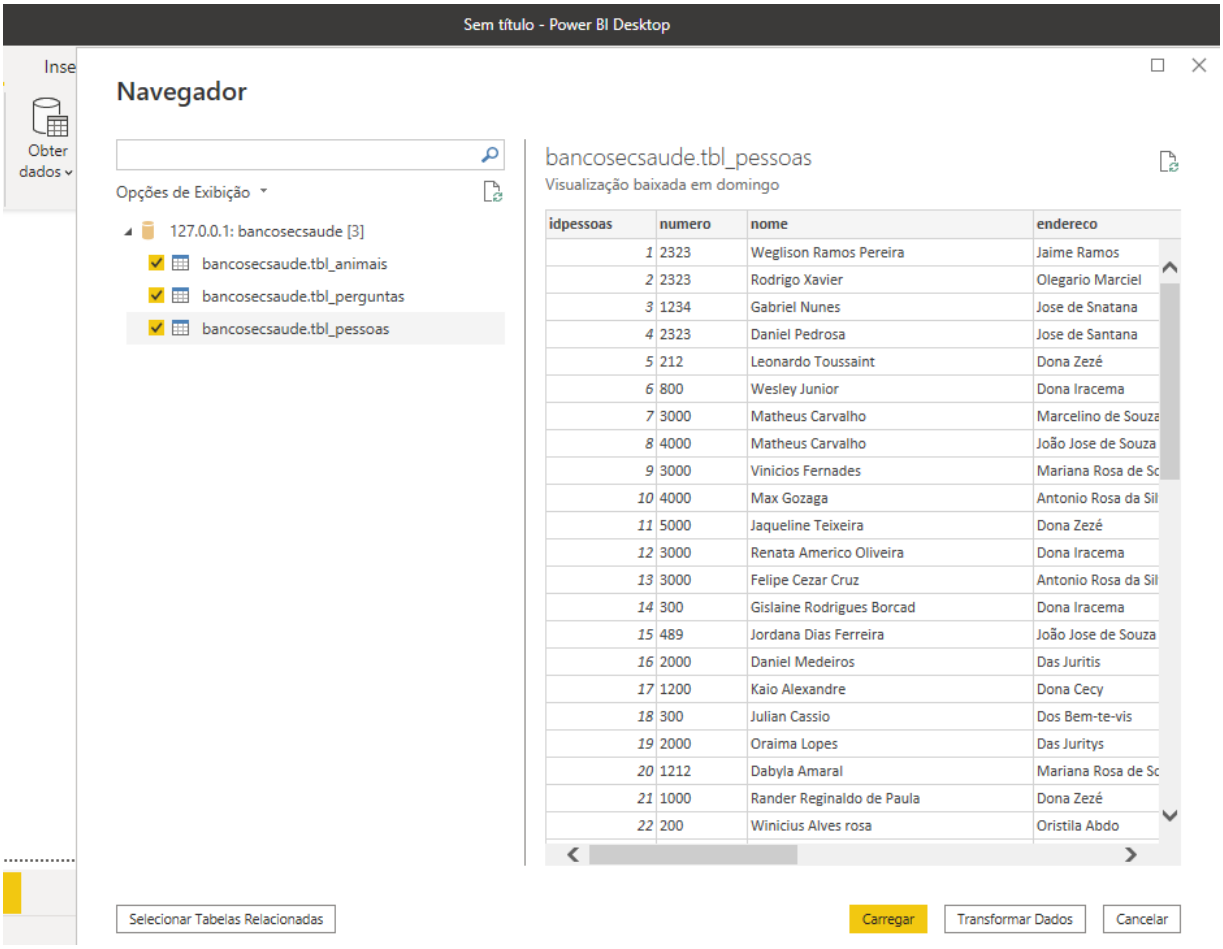



Fonte: O Autor.

Após a conexão com o banco de dados o *Power BI* apresenta uma tela onde existem as tabelas que eventualmente podem ser extraídas do banco. Para importar a tabela é necessário selecioná-la e escolher as opções de carregar ou transformar dados como mostrado na Figura 3.34, para essa figura os dados ilustrados são fictícios.

A opção carregar dados insere os dados para a área de trabalho do *Power BI* diretamente sem antes passá-los por uma ferramenta de tratamento e formatação dos dados. Quando a opção selecionada for a de transformar dados, as tabelas podem ser transformadas e editadas pela ferramenta Power *Query*. Essa metodologia é descrita na seção seguinte.

Figura 3.34: Selecionado as Tabelas do Banco de Dados.



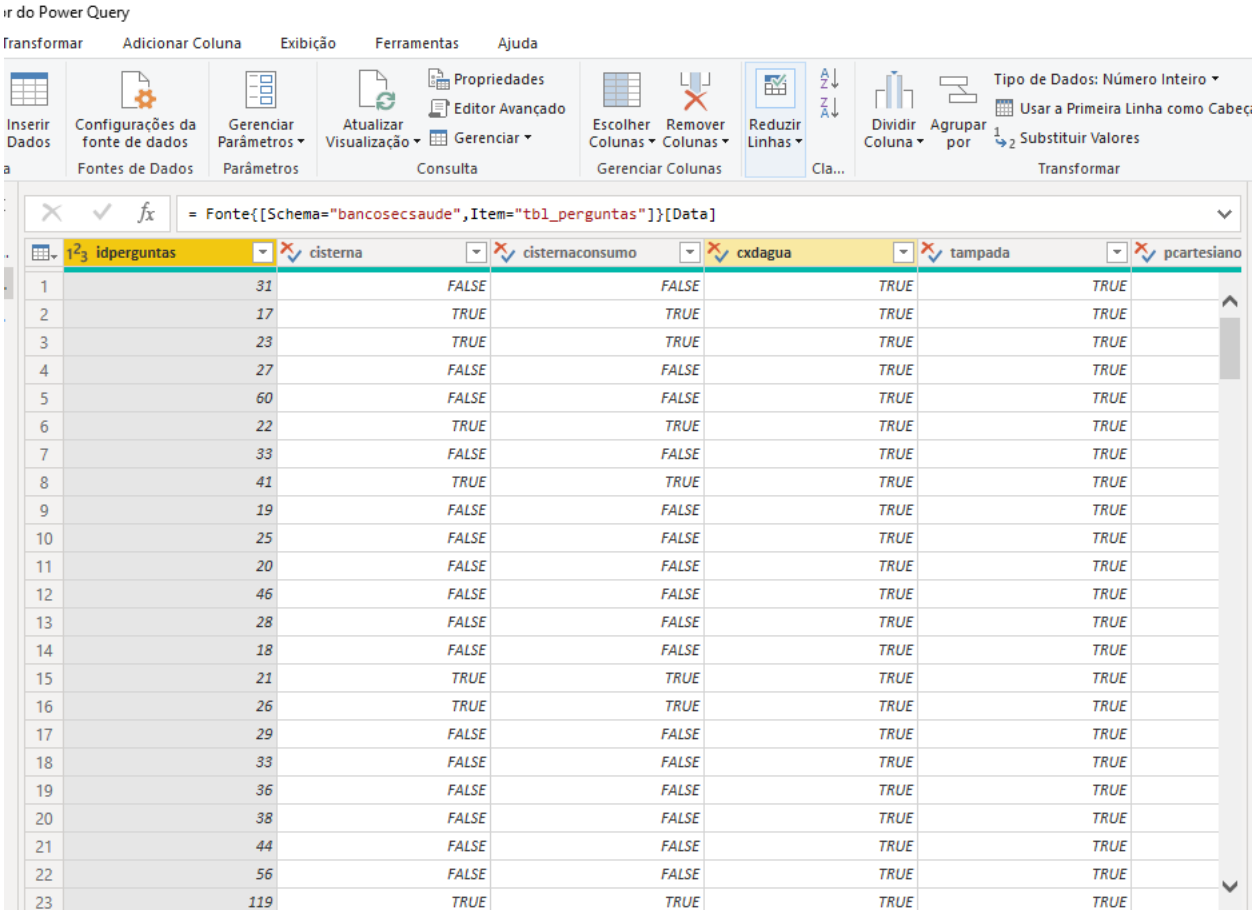

Fonte: O Autor.

### 3.3.1.2 Tratamento dos Dados

No processo de seleção das tabelas do banco de dados da Figura 3.34, se a opção selecionada for transformar dados o usuário é redirecionado para o *Power Query*. No *Power Query* existe a possibilidade de renomear tabelas ou colunas, substituir dados, criar colunas a partir de combinações de outras, inserir ou eliminar linhas, remover colunas e linhas em branco entre outas funcionalidades. A partir do clique no nome da tabela no lado superior esquerdo da tela do *Power Query* mostrado na Figura 3.35 é possível editar os dados da respectiva tabela. Após as edições e tratamentos dos dados é necessário aplicar e fechar, dessa forma os dados são carregados e enviados para a área de trabalho do *Power BI.*

Caso seja necessário realizar nova edição nos dados após já terem sido enviados para a área de trabalho do *Power BI* é preciso clicar em Página Inicial > Transformar Dados e retornar novamente para a interface do *Power Query* uma vez que não é possível editar os dados das tabelas diretamente no *Power BI.*

Figura 3.35: Editor do *Power Query*.




| | idperguntas | cisterna | cisternaconsumo | cxdagua | tampada | pcartesiano |
|---|---|---|---|---|---|---|
| 1 | 31 | FALSE | FALSE | TRUE | TRUE | |
| 2 | 17 | TRUE | TRUE | TRUE | TRUE | |
| 3 | 23 | TRUE | TRUE | TRUE | TRUE | |
| 4 | 27 | FALSE | FALSE | TRUE | TRUE | |
| 5 | 60 | FALSE | FALSE | TRUE | TRUE | |
| 6 | 22 | TRUE | TRUE | TRUE | TRUE | |
| 7 | 33 | FALSE | FALSE | TRUE | TRUE | |
| 8 | 41 | TRUE | TRUE | TRUE | TRUE | |
| 9 | 19 | FALSE | FALSE | TRUE | TRUE | |
| 10 | 25 | FALSE | FALSE | TRUE | TRUE | |
| 11 | 20 | FALSE | FALSE | TRUE | TRUE | |
| 12 | 46 | FALSE | FALSE | TRUE | TRUE | |
| 13 | 28 | FALSE | FALSE | TRUE | TRUE | |
| 14 | 18 | FALSE | FALSE | TRUE | TRUE | |
| 15 | 21 | TRUE | TRUE | TRUE | TRUE | |
| 16 | 26 | TRUE | TRUE | TRUE | TRUE | |
| 17 | 29 | FALSE | FALSE | TRUE | TRUE | |
| 18 | 33 | FALSE | FALSE | TRUE | TRUE | |
| 19 | 36 | FALSE | FALSE | TRUE | TRUE | |
| 20 | 38 | FALSE | FALSE | TRUE | TRUE | |
| 21 | 44 | FALSE | FALSE | TRUE | TRUE | |
| 22 | 56 | FALSE | FALSE | TRUE | TRUE | |
| 23 | 119 | TRUE | TRUE | TRUE | TRUE | |

Fonte: O Autor.

### 3.3.1.3 Criando o *Dashboard*

Após realizado o tratamento e correções pelo *Power Query* os dados estão prontos para serem usados na área de trabalho do *Power BI*. A criação dos *dashboards* pelo *Power BI* é bem simples, necessitando apenas escolher um formato de gráfico em uma paleta de opções e editar as informações de acordo com o desejado.

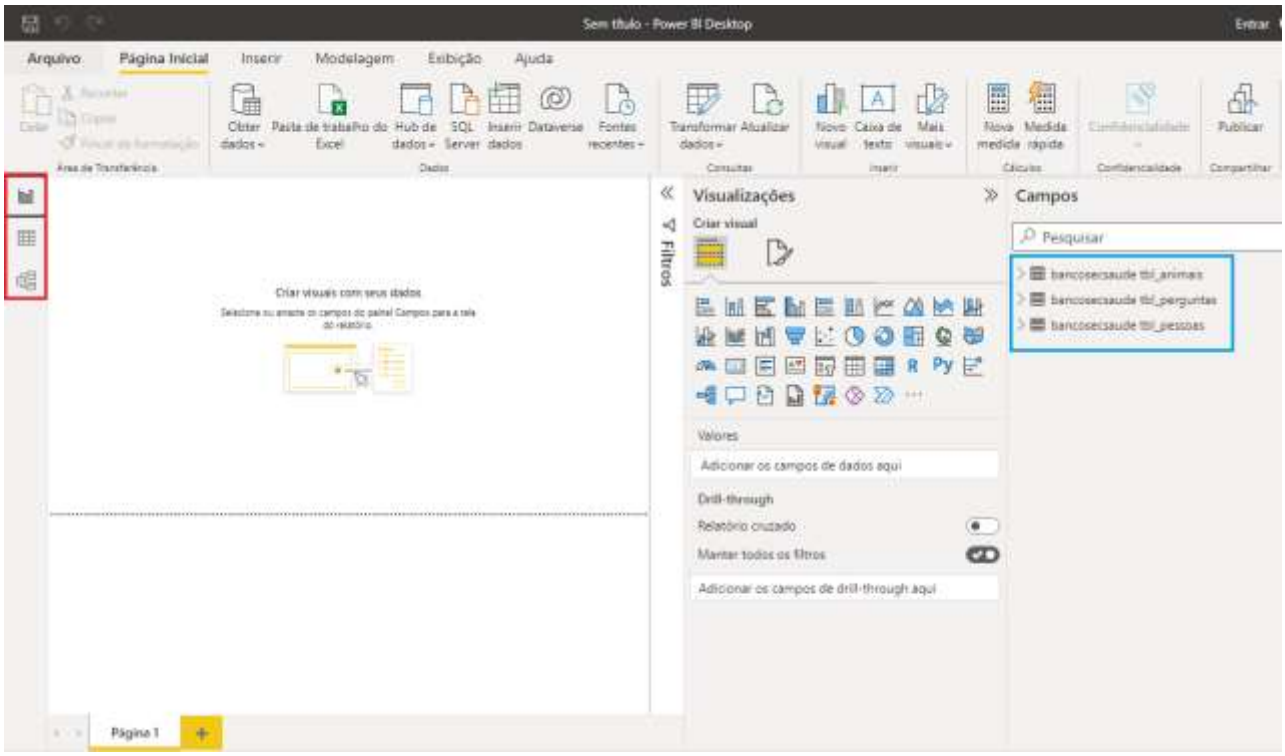
Na área de trabalho do *Power BI* os dados podem ser apresentados de três formas que podem ser selecionadas do lado esquerdo superior da área de trabalho:

- Em formato de relatório;
- Em formato de dados (tabelas);
- Em formato de relacionamentos;

A opção de relatório mostrada na Figura 3.36 é utilizada quando se quer criar *dashboards*, nela os dados advindos da fonte de dados são mostrados na parte direita (seleção azul na Figura 3.36) e podem ser anexados aos campos de valores dos gráficos. Os gráficos já possuem

algumas configurações básicas que podem ser editadas com a anexação das informações da fonte de dados nos campos de valores e dessa forma os gráficos e tornam visualmente interpretativos.

Figura 3.36: Área de trabalho *Power BI*– Formato Relatório.

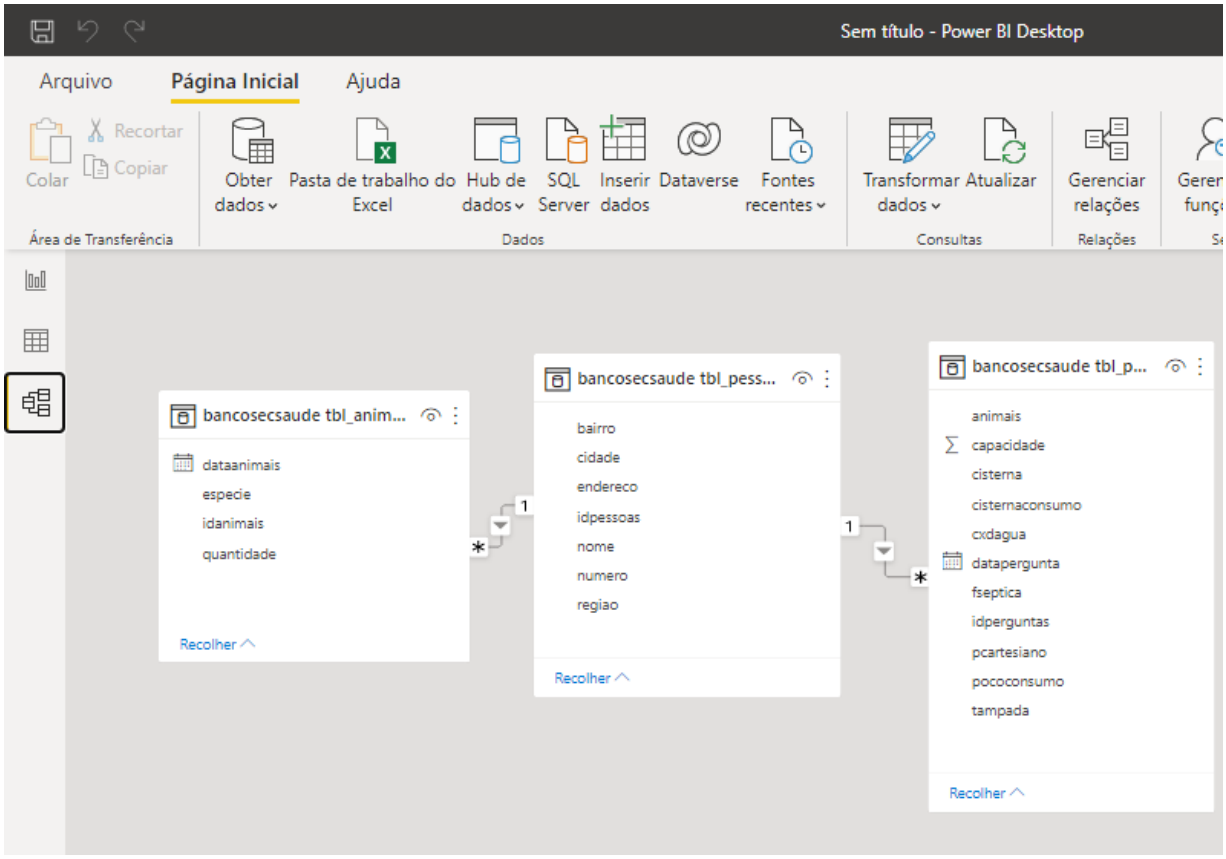



Fonte: O Autor.

A segunda possibilidade é utilizar a área de trabalho no formato dados (tabular), nela os dados são apresentados de forma semelhante a uma planilha do Excel. Essa funcionalidade é útil pois as vezes é preciso visualizar como está a formatação em uma coluna e com essa opção não é necessário abrir os dados em outra ferramenta como o *Power Query*.

Além das duas formas já descritas existe a possibilidade de visualizar os dados em forma de relacionamento como mostrado na Figura 3.37, onde as relações entre as tabelas são destacadas na forma de interligações. As relações são criadas a partir dos campos de chave primária e chave estrangeira presentes nas tabelas do banco de dados. É possível ver a cardinalidade entre as tabelas onde pode existir 1 registro na tabela pessoa que se associa a N registros nas tabelas de animais e perguntas. Essa ferramenta é similar a um diagrama de banco de dados e serve para modelar de forma eficaz os *dashboards*.

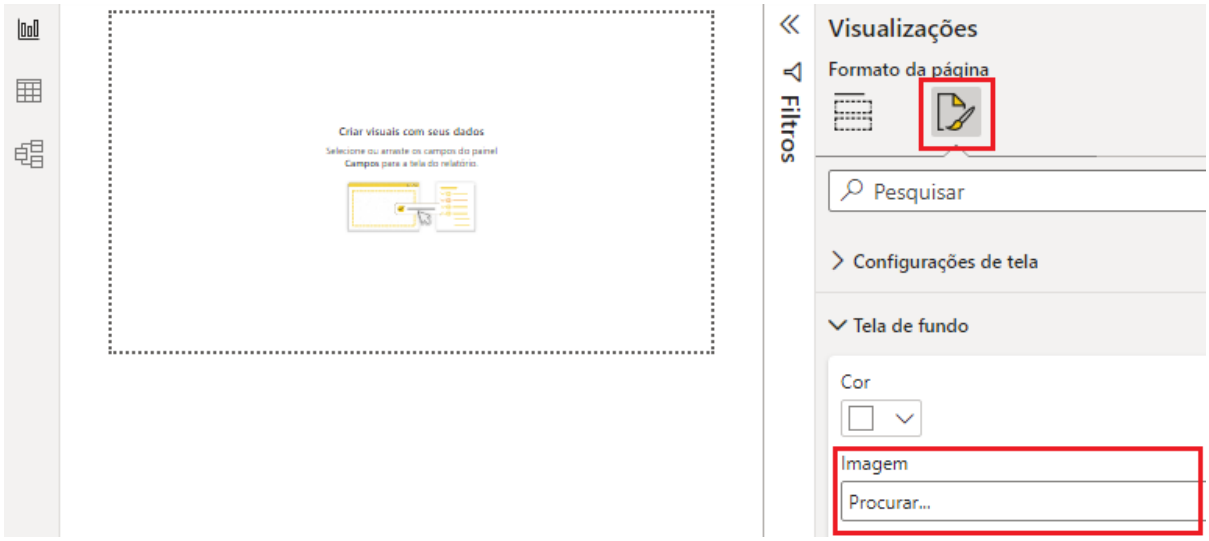
Figura 3.37: Área de trabalho Power BI – Formato Relacionamentos.




Fonte: O Autor.

Na opção de área de trabalho em formato de relatório é possível adicionar um plano de fundo a área de trabalho para que o fundo do *dashboard* fique com um aspecto mais atrativo visualmente. Essa opção pode ser aplicada utilizando a ferramenta de edição representada por um pincel de formatação mostrado na Figura 3.38.

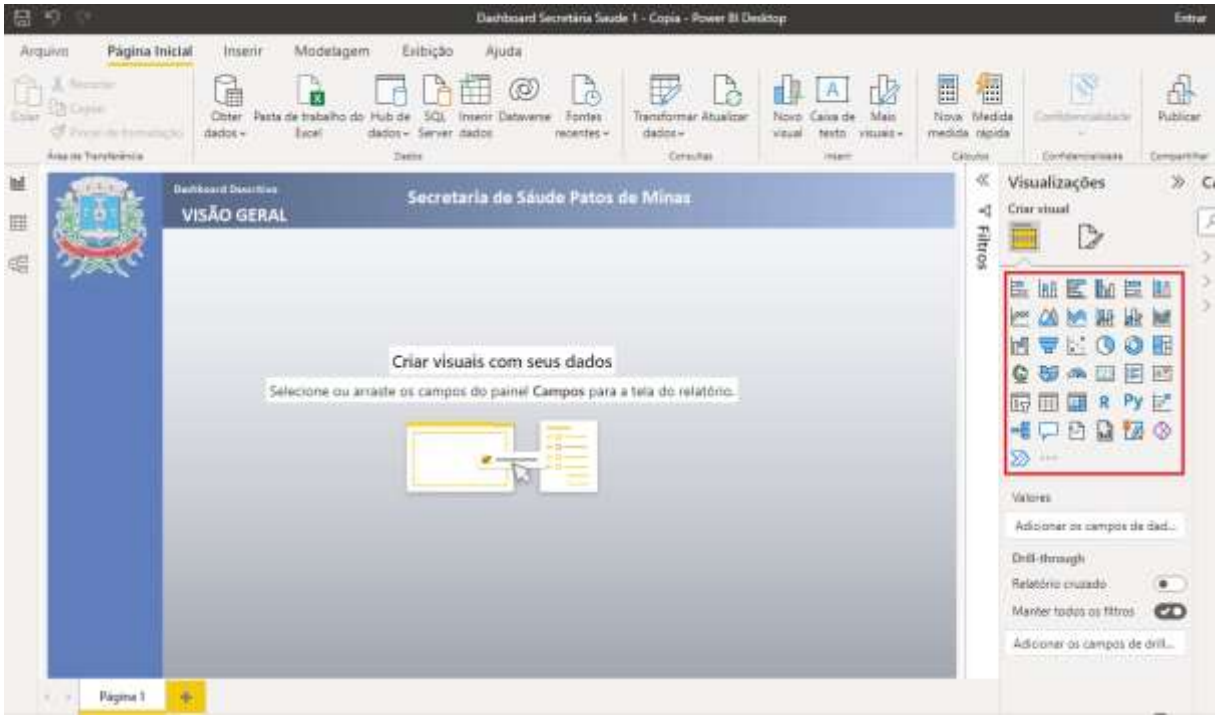
Figura 3.38: Adição de Plano de Fundo ao *Dashboard*.




Fonte: O Autor.

Para a criação do *dashboard* com os dados de questionários armazenados foi projetada uma imagem de plano de fundo que é mostrada na Figura 3.39. Essa imagem foi desenvolvida tendo em vista que o *dashboard* ganha um aspecto mais profissional com um *design* próprio e que represente os dados a serem exibidos.

Figura 3.39: Imagem de Fundo do *dashboard.*




Fonte: O Autor.

Os gráficos podem ser adicionados a área de trabalho selecionando-os na paleta de opções destacada em vermelho da Figura 3.39.

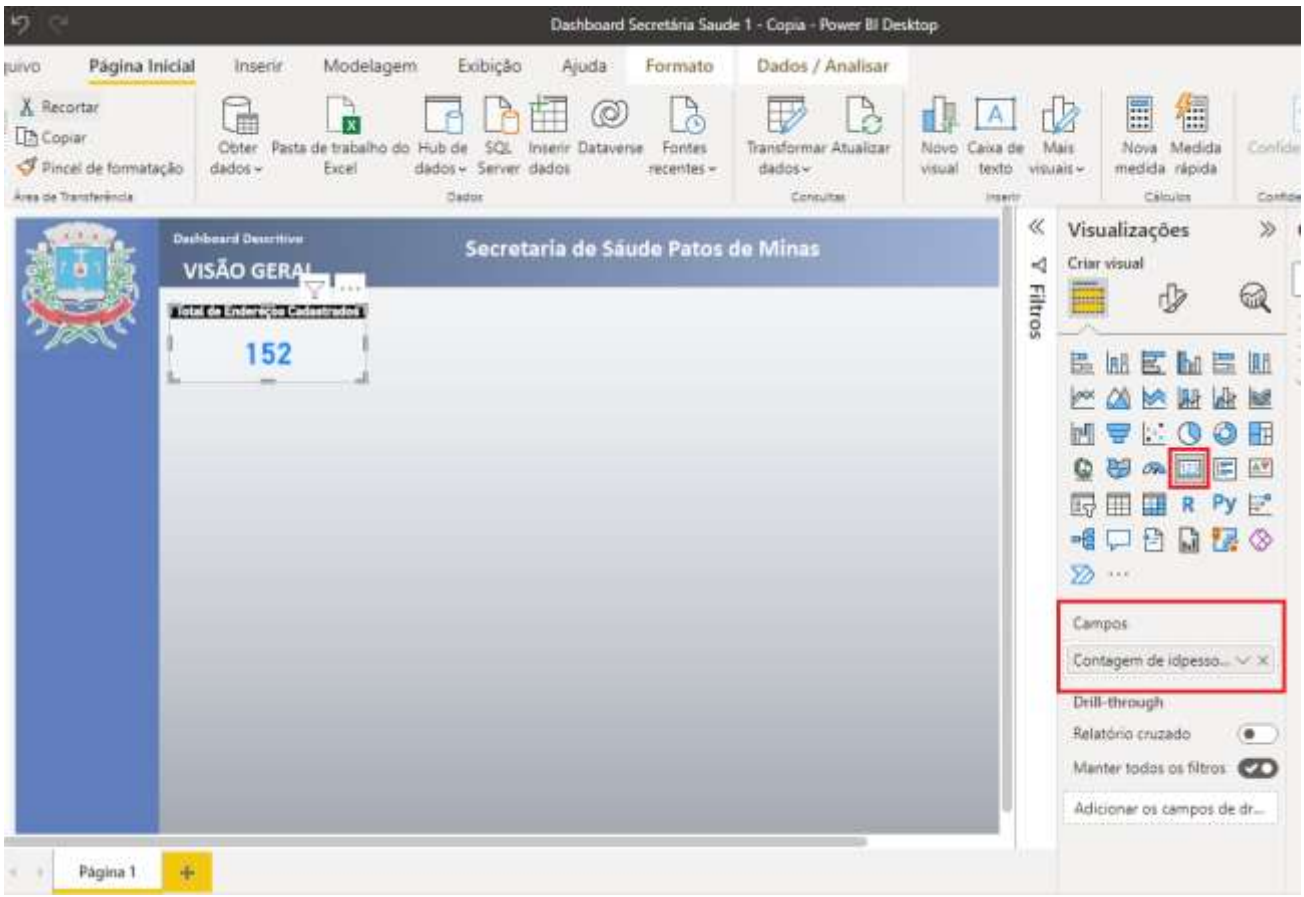
Existem vários formatos de gráficos disponíveis no *Power BI*, alguns dos mais utilizados são:

- Gráficos de área;
- Gráficos de barras e colunas;
- Cartões;
- Velocímetro;
- Gráfico de linha;
- Árvore de decomposição;
- Gráficos de rosca;

- Gráficos de funil;
- Gráfico de pizza;
- Mapas;
- Segmentação de dados.

Para a construção do *dashboard* são utilizados alguns desses gráficos elencados, o total de endereços cadastrados por exemplo é apresentado em um cartão que pode ser visualizado na Figura 3.40. O cartão possui uma característica de apresentação de dados quantitativos, o campo a ser configurado o valor a ser exibido no cartão fica exposto logo abaixo da paleta de opções de gráficos e é definido como campos.

Figura 3.40: Gráfico de cartão.




Fonte: O Autor.

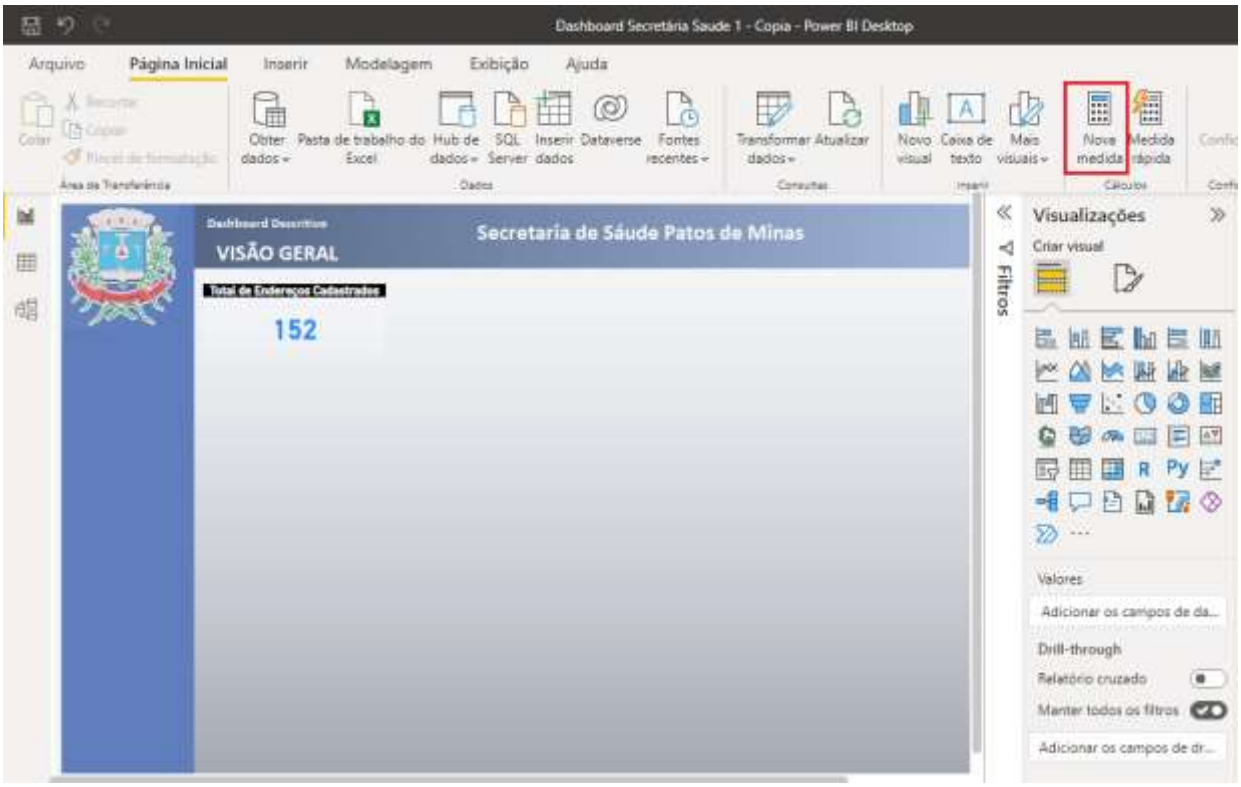
Outro cartão configurado para o *dashboard* recebe a quantidade total de questionários cadastrados, porém para realizar a contagem de questionários é preciso criar uma medida. Medidas no *Power BI* são similares as fórmulas do Excel e são implementadas por meio de fórmulas DAX.

#### *3.3.1.3.1* Criando Medidas No *Power BI*

*Data Analysis Expression* (DAX) é uma linguagem disponibilizada no *Power BI* para a criação de medidas e funções de maior complexidade. Essa linguagem é comum em serviços da *Microsoft* que realizam análise de dados como por exemplo o *Power BI*, *Power Pivot*, SQL *Server Analysis Services* e *Azure Analysis Services*.

As fórmulas DAX possibilitam a criação de medidas que são cálculos mais complexos e que não são encontrados por padrão nas opções disponíveis nos gráficos. Para criar uma medida no *Power BI* é necessário selecionar a opção de nova medida como mostrado na Figura 3.41.

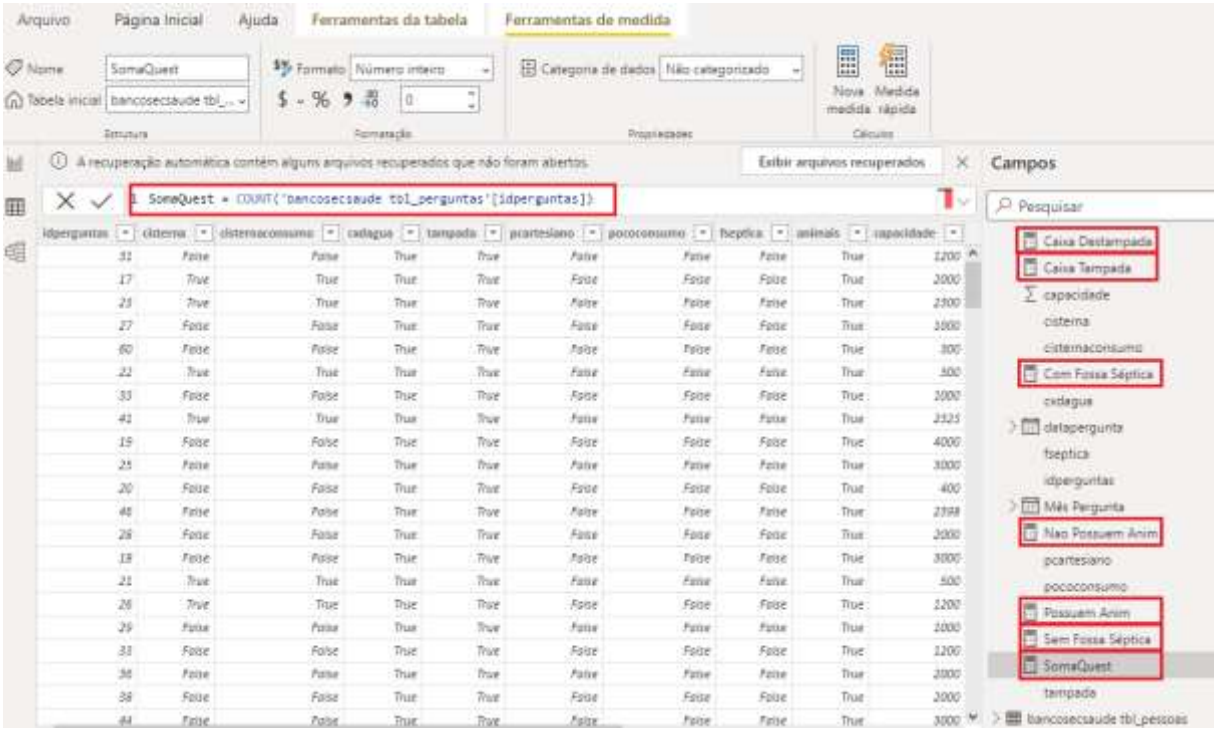
Figura 3.41: Criação de Medidas *Power BI.*




Fonte: O Autor.

Clicando em nova medida uma caixa de fórmula é aberta e podem ser inseridas as funções necessárias. Como a intenção é saber quantos questionários possuem cadastrados é necessário realizar um somatório de quantas perguntas existem. Dessa forma a função DAX que melhor executa esse somatório é a função COUNT mostrada na caixa de fórmulas da Figura 3.42**.** Essa função faz um somatório do número de índices de perguntas (idperguntas) da tabela, ao contrário de uma função SUM por exemplo, que realizaria um somatório dos valores escalares presentes no campo de índice.

Figura 3.42: Criando Medida




Fonte: O Autor.

Na Figura 3.42 todas as fórmulas selecionadas e que possuem ícones de calculadora são medidas criadas para serem utilizadas nos gráficos do *dashboard* de questionários. Essas medidas podem ser visualizadas com mais detalhes na Figura 3.43.

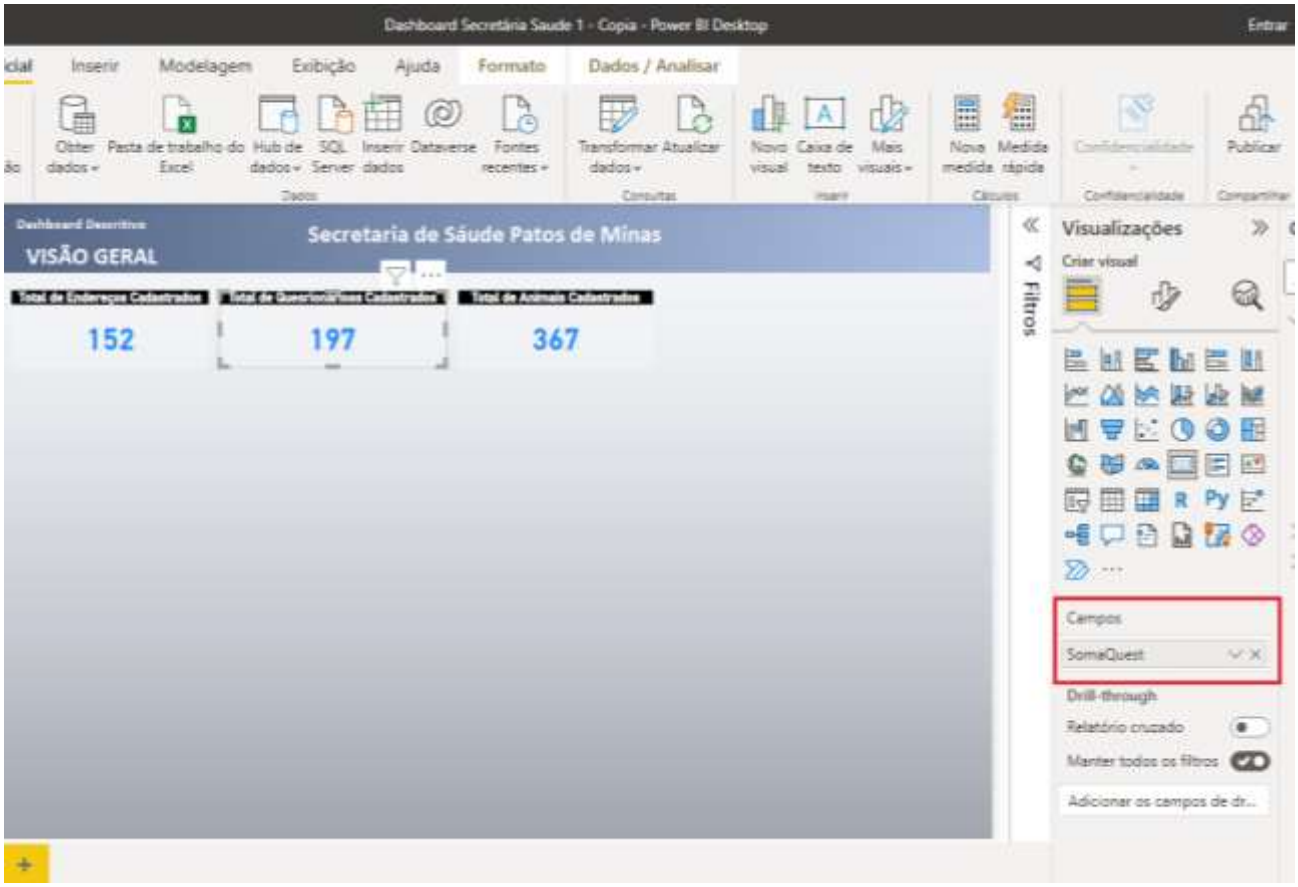
Figura 3.43: Medidas Criadas Usando Fórmulas DAX.

```
1 Caixa Destampada = COUNTAX(FILTER('bancosecsaude tbl_perguntas','bancosecsaude tbl_perguntas'[tampada]=FALSE()),
  'bancosecsaude tbl_perguntas'[tampada])

1 Caixa Tampada = COUNTAX(FILTER('bancosecsaude tbl_perguntas','bancosecsaude tbl_perguntas'[tampada]=TRUE()),'bancosecsaude
  tbl_perguntas'[tampada])

1 Com Fossa Séptica = COUNTAX(FILTER('bancosecsaude tbl_perguntas','bancosecsaude tbl_perguntas'[fseptica]=TRUE()),
  'bancosecsaude tbl_perguntas'[fseptica])

1 Nao Possuem Anim = COUNTAX(FILTER('bancosecsaude tbl_perguntas','bancosecsaude tbl_perguntas'[animais]=FALSE()),
  'bancosecsaude tbl_perguntas'[animais])

1 Possuem Anim = COUNTAX(FILTER('bancosecsaude tbl_perguntas','bancosecsaude tbl_perguntas'[animais]=TRUE()),'bancosecsaude
  tbl_perguntas'[animais])

1 Sem Fossa Séptica = COUNTAX(FILTER('bancosecsaude tbl_perguntas','bancosecsaude tbl_perguntas'[fseptica]=FALSE()),
  'bancosecsaude tbl_perguntas'[fseptica])

1 SomaQuest = COUNT('bancosecsaude tbl_perguntas'[idperguntas])
```

Fonte: O Autor.

Após a criação da medida que possibilita saber o número de questionários cadastrados foram inseridos mais dois cartões no *dashboard*, um preenchido com a medida “SomaQuest” e o outro preenchido com o somatório do campo de quantidade da tabela de animais como mostrado na Figura 3.44.

Figura 3.44: Inserção dos cartões no *dashboard*.




Fonte: O Autor.

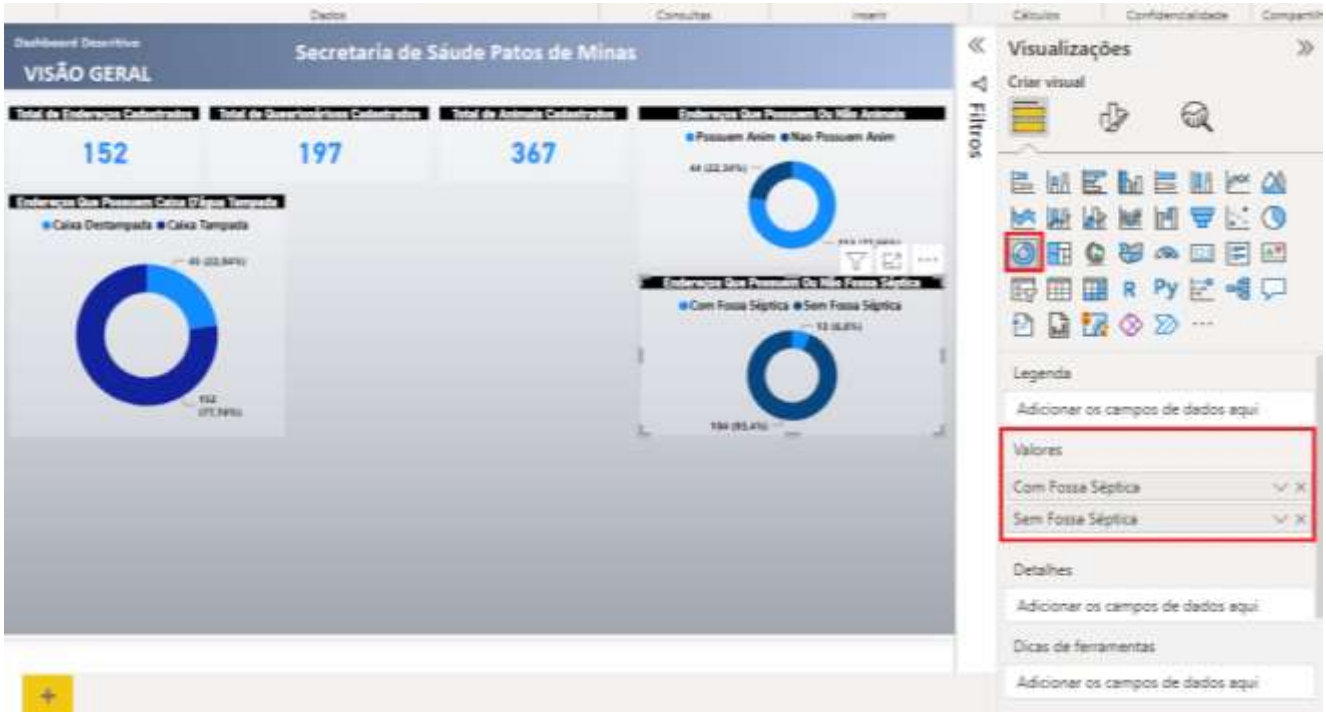
Posteriormente são inseridos os gráficos de rosca que podem ser vistos na Figura 3.45, esses gráficos são usados para representar uma porcentagem em relação ao total, proporciona uma comparação e é recomendado quando se quer comparar por exemplo a relação de endereços que possuem determinada característica em relação ao total.

Foram adicionados os seguintes gráficos de rosca ao *dashboard*:

- Relação de quantos endereços cadastrados que possuem caixa d'água tampada;
- Relação de quantos endereços que possuem fossa séptica;
- Relação de quantos endereços possuem animais;

Todos os gráficos de rosca necessitam que sejam implementadas medidas para o cálculo dos somatórios uma vez que os campos da tabela de perguntas para esses casos são dados booleanos e não seria possível obter um somatório a partir de dados booleanos, dessa forma são usadas fórmulas DAX para aplicar um filtro e devolver o resultado do somatório. As medidas criadas podem ser visualizadas na Figura 3.43.

Figura 3.45: Inserção dos Gráficos de Rosca ao *Dashboard*.




Fonte: O Autor.

Na sequência é adicionado o gráfico de colunas de barras empilhadas que é indicado para mostrar quantidades em relação a um certo dado. Por meio desse gráfico pode-se demonstrar quais são as espécies de animais mais frequentes nos domicílios

Para os gráficos do *Power BI* podem ser realizadas formatações gráficas, essas formatações são realizadas por meio da ferramenta de pincel de formatação destacada na Figura 3.46. Nessa ferramenta podem ser alterados cores do gráfico, texto do título, alinhamento, formatação textual, escala das colunas, efeitos entre outros. Dessa forma os gráficos podem ser trabalhados e formatados da forma que o usuário deseja, dando maior liberdade e autonomia na customização e criação.

Figura 3.46: Inserção do Gráfico de Colunas de Barras Empilhadas.


Fonte: O Autor.

Há também a inserção no *dashboard* de uma visão do tipo segmentação de dados. Essa visão serve como um filtro para os outros gráficos pois são sincronizados a depender da opção selecionada na visão de segmentação de dados.

É indicado o uso da visão de segmentação de dados quando se pretendente exibir os filtros mais importantes na tela do relatório para facilitar o acesso. Há também a possibilidade de configurar quais as visões que são afetadas pelas ações na tela de segmentação de dados, por padrão todas as visões ou gráficos são afetados pelas alterações na visão de segmentação.

Para o *dashboard* de questionários existem as relações entre as tabelas de pessoas, perguntas e animais, essas relações possibilitam que a segmentação de dados seja aplicada ao caso em estudo. Por meio das relações uma pessoa (endereço) e consequentemente um bairro estão vinculados a um questionário, dessa forma uma das possibilidades é usar o bairro como campo dados para a visão de segmentação de dados, assim os gráficos podem ser sincronizados por filtragem por bairro como mostrado na Figura 3.47. A partir dessa configuração, sempre que se clicar em um bairro o *dashboard* se atualiza para mostrar os dados referentes ao bairro selecionado.

Figura 3.47: Inserção de Quadro de Segmentação de dados.


Fonte: O Autor.

## 3.4 Considerações finais

O processo de desenvolvimento do sistema bem como da proposta de ferramenta de análise de dados é realizado a partir de uma parceria entre a Universidade Federal de Uberlândia e a Secretária de Saúde de Patos de Minas. A metodologia de desenvolvimento do sistema é dividida em três vertentes, a primeira delas consiste no desenvolvimento da ferramenta de gerenciamento e digitalização dos questionários, a segunda na criação do banco de dados MySQL e conexão com o *software* Java usando um *driver* JDBC e a terceira em uma sugestão de ferramenta de análise de dados utilizando a ferramenta *Microsoft Power BI*.

Não foi necessário o investimento de quantias financeiras em aquisições de equipamentos ou matéria prima para o desenvolvimento do trabalho visto que o produto desenvolvido é uma aplicação de *software* e todas as ferramentas necessárias encontram-se disponíveis em laboratórios da instituição (Palácio de Cristais) ou em máquinas pessoais.

No capítulo seguinte serão abordadas considerações acerca dos resultados obtidos por meio da realização do presente trabalho.

# 4 RESULTADOS E DISCUSSÕES

A seguir são realizadas discussões acerca do desenvolvimento do *software* de digitalização e sobre a criação do *dashboard* para análise de dados.

## 4.1 *Software* de Digitalização de Questionários

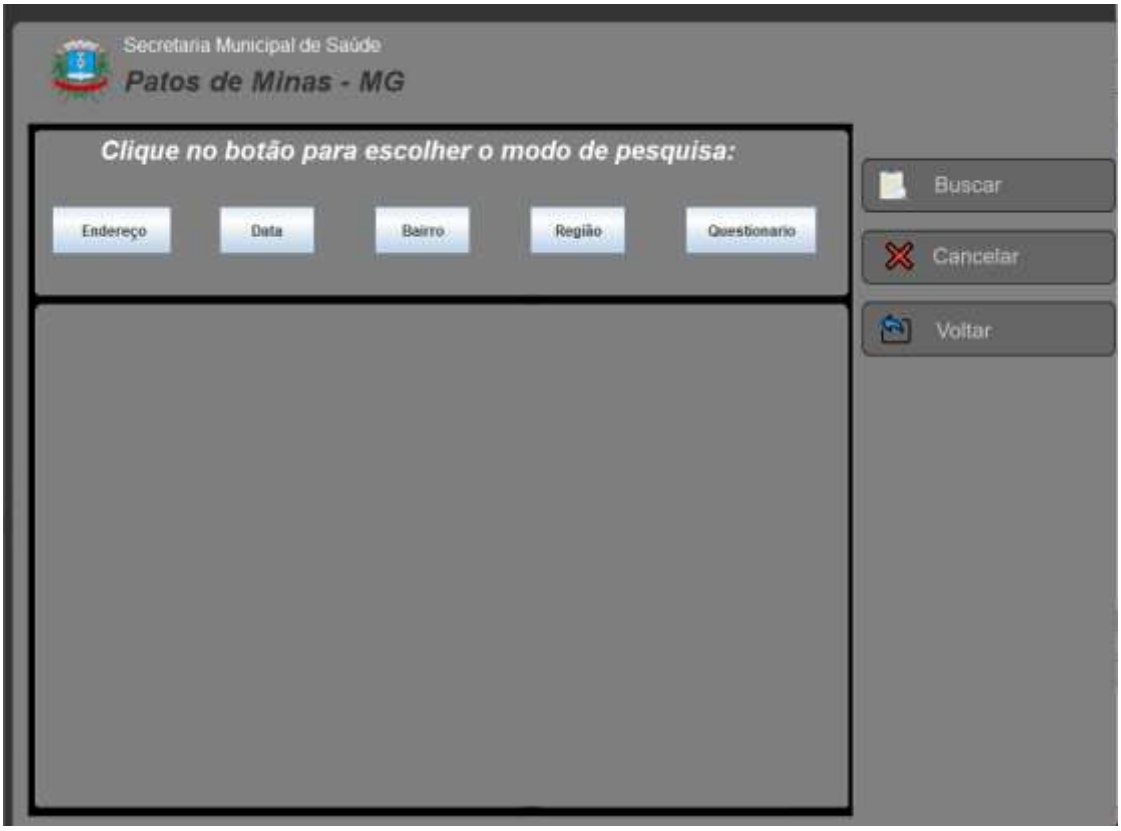
O *software* de digitalização de questionários da SSPM visa automatizar e agilizar o processo de arquivamento dos questionários que estão em formato de papel e é desenvolvido utilizando conceitos abordados durante a graduação. Para esse sistema são aplicadas várias teorias tanto para o desenvolvimento do *front-end* como do *back-end*. As interfaces gráficas são criadas por meio da API *swing* do Java, essa API disponibiliza os objetos gráficos que possibilitam a criação de um *layout* com aspecto profissional. O desenvolvimento da interface gráfica leva em consideração a criação de uma identidade visual que represente fielmente o questionário em papel com a mesma estrutura de campos a serem preenchidos. A Figura 4.1 mostra a tela de cadastros do sistema de digitalização de questionários, nela o usuário pode cadastrar os dados de endereço e respostas do questionário e inseri-las no banco de dados clicando na opção “SALVAR”.

Figura 4.1: Tela de Cadastros.


Fonte: O Autor.

Além do cadastro o usuário pode realizar uma busca por meio da funcionalidade de busca por questionários. Essa busca pode ser realizada de várias maneiras, por endereço, data, bairro, região e questionário. A Figura 4.2 mostra a tela de buscas, nela o utilizador pode selecionar um dos botões de acordo com a busca desejada. A partir da seleção do formato de busca é necessário informar alguns dados e clicar em “BUSCAR”.

Figura 4.2: Tela de Buscas Por Questionário.




Fonte: O Autor.

A funcionalidade de busca não permite que sejam realizadas edições no questionário, logo é desenvolvida a tela de edição em uma outra opção sendo que para a edição o usuário precisa informar o endereço e a data do questionário. A partir desses campos é feita uma consulta ao banco de dados e caso existam registros de questionários para os dados informados um questionário é apresentado na tela de edição. Após o processo de edição o utilizador do sistema pode novamente salvar as alterações no banco de dados.

Em relação ao desenvolvimento das funcionalidades da aplicação são utilizados conceitos de programação orientada a objetos aliados ao padrão de projetos DAO. As ações executadas na interface do *software* são instantaneamente replicadas para o *back-end* onde os métodos e funções são acionados e os resultados apresentados em tempo real. A utilização do padrão DAO

se mostra eficaz pois facilita o acesso aos dados. As classes DAO encapsulam os objetos de negócio e implementam métodos que fazem as consultas ao banco de dados. Esses métodos são acionados por eventos disparados a partir de uma ação na interface do sistema. Dessa forma existe uma lógica que desacopla as classes de interface do sistema das classes de acesso ao banco de dados.

O banco de dados utilizado pelo sistema é o MySQL, nesse banco são criadas as tabelas que armazenam os registros das informações. Todas as configurações no banco de dados são feitas por meio do sistema gerenciados de banco de dados *MySQL workbench*. Esse SGBD simplifica as operações de controle, gerenciamento e *backups* que se fossem realizados via linhas de comando seriam mais complexas.

As tabelas do banco possuem relacionamentos, sendo que na tabela de pessoas (endereços) existe um identificador como chave primária que se associa as tabelas de animais e de perguntas por meio de uma chave estrangeira. Essas relações são necessárias para o correto funcionamento do sistema uma vez que ao efetuar-se buscas no banco de dados, os registros de animais e questionários vinculados ao endereço são retornadas.

## 4.2 Analise Descritiva dos Dados

Como relatado nas Seções 2.6 e 2.6.1 a análise de dados pode ser realizada de forma descritiva, esse tipo de análise demonstra visões de comportamentos passados, serve para diagnosticar tendencias ou padrões. Dessa forma atitudes e correções podem ser tomadas em busca de melhoras nos processos.

O *dashboard* criado com os dados armazenados pelo sistema de digitalização de questionários se enquadra como uma forma de análise de dados descritiva. Pelo *dashboard* podem ser retiradas informações sobre as condições de saneamento básico em determinada região, a quantidade de domicílios que possuem animais, as espécies de animais mais comuns, a quantidade de domicílios que possuem caixa d'água sem tampa dentre outras. O *dashboard* desenvolvido pode ser visualizado na Figura 4.3. Cada gráfico do *dashboard* é configurado com o intuito de apresentar uma característica que agregue conhecimento a quem o analisa, são gráficos que possibilitam comparações, inferências e conclusões acerca dos dados armazenados.

Figura 4.3: *Dashboard* Final.

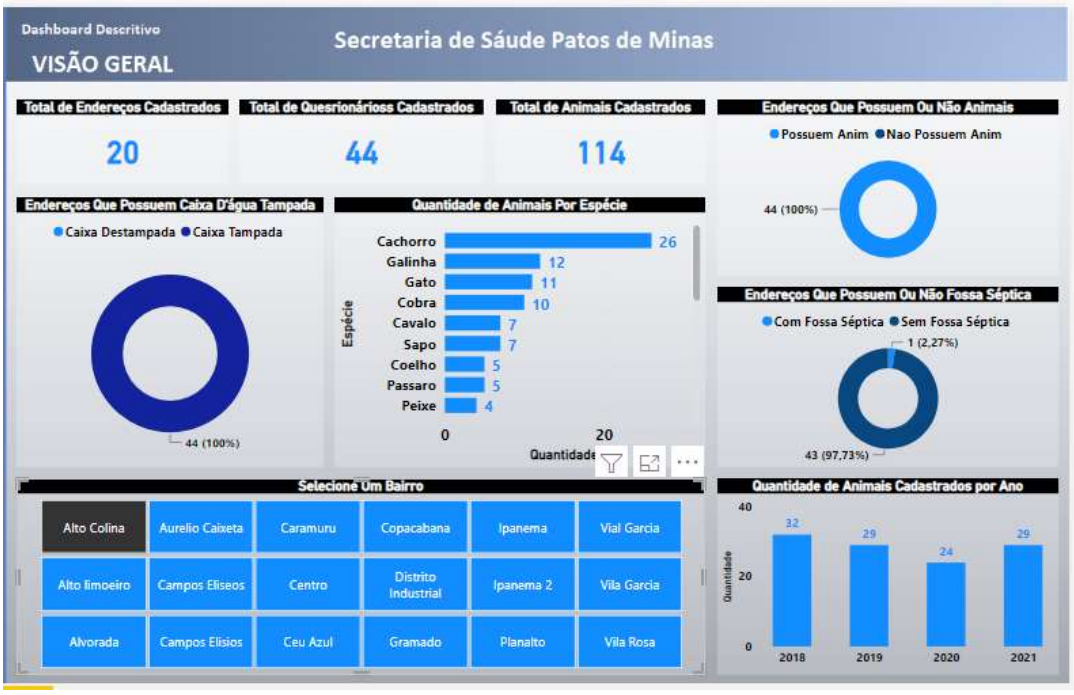

Fonte: O Autor.

Pelo *dashboard* são obtidas informações de quantos endereços e questionários existem armazenados bem como a quantidade de animais registrados, esses valores são apresentados em cartões que são visões com característica quantitativa. Os resultados são obtidos de forma simples se comparados a uma busca com o mesmo objetivo em dados armazenados em formato de papel. Outras possibilidades de obtenção de *insights* por meio do desenvolvimento do *dashboard* são as comparações. As comparações possuem características qualitativas e quantitativas, como por exemplo as relações da quantidade de endereços que possuem animais, quantidade de endereços que possuem caixa d'água tampada, quantidade de endereços que possuem fossa séptica entre outras. Todas essas comparações podem ser obtidas por meio dos gráficos de rosca.

Além dos cartões e gráficos de rosca o *dashboard* também traz alguns gráficos de colunas que são gráficos de fácil entendimento. Esses gráficos possuem uma característica quantitativa sobre uma variável. Outro componente gráfico que é implementado para o *dashboard* é a segmentação de dados, por meio desse componente torna-se possível a criação de um menu de bairros. Assim, a cada seleção de um bairro o dashboard sincroniza as informações e apresenta

apenas os dados relacionados ao bairro selecionado. A Figura 4.4 demonstra o resultado quando se seleciona um bairro no menu de segmentação de dados.

Figura 4.4: Seleção Por Bairro.




Fonte: O Autor.

Existe também a possibilidade de seleção de vários bairros como pode ser visualizado na Figura 4.5. Assim sendo, caso seja necessário visualizar os dados para uma região composta por vários bairros a ferramenta de segmentação de dados pode ser utilizada.

Figura 4.5: Seleção Múltipla de Bairros.


Fonte: O Autor.

# 5 CONCLUSÃO E TRABALHOS FUTUROS

Por meio do sistema desenvolvido objetivou-se modernizar o armazenamento de informações de questionários e reduzir os custos com material para o setor de endemias da Secretária de Saúde de Patos de Minas. As informações digitalizadas são essenciais visto que campanhas e ações voltadas ao controle de doenças endêmicas muitas vezes se baseiam em estudos das informações obtidas em anos anteriores. Nesse contexto, os dados podem fundamentar e direcionar medidas preventivas, como por exemplo doses de remédios a serem adquiridas em campanhas de vacinação animal e medidas para o controle de doenças vetoriais como dengue, malária, leishmanioses, doença de chagas e febre amarela. Além disso é demonstrado de forma prática como poderia ser desenvolvido um sistema de análise descritiva dos dados para as informações armazenadas.

Como sugestão para trabalhos futuros, poderia ser estudado uma forma de migrar o sistema de questionário para *web*. Dessa forma a aplicação poderia ser acessada de qualquer plataforma e os agentes poderiam cadastrar as informações no banco de dados já no momento da visita aos domicílios.

.

# REFERÊNCIAS


[1] Bordin, “Modernização dos serviços públicos aos cidadãos brasileiros,” Instituto de Tecnologia Educacional do Brasil, 04 Julho 2020. [Online]. Available: https://psalm.escreveronline.com.br/redacao/tema-modernizacao-dos-servicos-publicos-aos-cidadaos-brasileiros/. [Acesso em 10 NOVEMBRO 2020].

[2] “Por que fazer a digitalização de documentos,” [Online]. Available: https://www.totvs.com/blog/negocios/digitalizacao-de-documentos/. [Acesso em 16 Setembro 2020].

[3] M. d. Saúde, “Vigilância de Zoonoses (SVS),” [Online]. Available: https://www.saude.gov.br/saude-de-a-z/vigilancia-de-zoonoses-svs. [Acesso em 19 Setembro 2020].

[4] M. d. S. -. D. d. V. d. D. Transmissíveis, “Manual de Vigilâcia, Prevenção e Controle de Zoonozes.,” Ministério da Suade, 2016. [Online]. Available: https://bvsms.saude.gov.br/bvs/publicacoes/manual_vigilancia_prevencao_controle_zoonoses.pdf. [Acesso em 19 Setembro 2020].

[5] M. d. Saúde, “Ministério da Saúde libera recursos para vigilância das zoonoses e doenças de transmissão vetorial,” [Online]. Available: https://www.paho.org/bra/index.php?option=com_content&view=article&id=3092:ministerio-da-saude-libera-recursos-para-vigilancia-das-zoonoses-e-doencas-de-transmissao-vetorial&Itemid=463. [Acesso em 19 Setembro 2020].

[6] A. d. C. d. P. M. d. P. d. Minas., “Prefeitura fala sobre trabalho feito pelo Centro de Controle de Zoonoses em 2015,” [Online]. Available: https://patoshoje.com.br/noticia/prefeitura-fala-sobre-trabalho-feito-pelo-centro-de-controle-de-zoonoses-em-2015-28177.html. [Acesso em 17 Setembro 2020].

[7] H. L. PINTO, “Atividades básicas ao desenvolvimento de software.,” Devmedia, [Online]. Available: https://www.devmedia.com.br/atividades-basicas-ao-processo-de-desenvolvimento-de-software/5413. [Acesso em 05 Agosto 2020].

[8] M. A. V. d. Lima, “Linguagens de Programação - Visão Geral,” 2009. [Online]. Available: http://www.facom.ufu.br/~madriana/PF/Intro.pdf. [Acesso em 18 Setembro 2020].

[9] F. Varejão, em *Linguagens de Programção (Java, C, C++ eoutras): Conceitos e Técnicas*, Editora Campus, 2002.

[10] M. T. Valente, Engenharia de Software Moderna, Independente , 1ª Edição (1 junho 2020), 2020.

[11] H. MEDEIROS, “Introdução ao Padrão MVC,” [Online]. Available: https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308. [Acesso em 06 Setembro 2020].

[12] A. W. Services, “O que são Microsserviços,” 2020. [Online]. Available: https://aws.amazon.com/pt/microservices/. [Acesso em 15 Setembro 2020].

[13] J. C. Macoratti, “Apresentando o padrão DAO - Data Access Object,” [Online]. Available: http://www.macoratti.net/11/10/pp_dao1.htm. [Acesso em 05 Agosto 2020].

[14] Oracle, “Padrão De Projeto DAO,” [Online]. Available: https://www.oracle.com/java/technologies/dataaccessobject.html. [Acesso em 12 Setembro 2020].

[15] N. Elmasri, Sistemas de Banco de Dados, Pearson 6º Edição.

[16] O. Brasil, “Banco de Dados Relacional,” [Online]. Available: https://www.oracle.com/br/database/what-is-a-relational-database/. [Acesso em 06 Setembro 2020].

[17] R. J. O. d. Nascimento, “Mineração e Análise de Dados em SQL,” [Online]. Available: https://www.devmedia.com.br/mineracao-e-analise-de-dados-em-sql/29337. [Acesso em 25 Agosto 2020].

[18] N. Fallon, “Análise Preditiva ou Prescritiva? Sua empresa precisa de ambos,” Medium, [Online]. Available: https://medium.com/@edselferri/an%C3%A1lise-preditiva-ou-prescritiva-sua-empresa-precisa-de-ambos-63b49caf09cf. [Acesso em 30 07 2022].

[19] T. Maydon, “The 4 Types Of Data Analytics,” 17 Janeiro 2017. [Online]. Available: https://insights.principa.co.za/4-types-of-data-analytics-descriptive-diagnostic-predictive-prescriptive. [Acesso em 11 Novembro 2020].

[20] K. Ferreira, “Análise de dados: o que é e dicas fundamentais para obter sucesso na sua análise,” 19 Setembro 2019. [Online]. Available: https://rockcontent.com/br/blog/analise-de-dados/. [Acesso em 12 Novembro 2020].

[21] D. Godoi, “Business Intelligence – BI: Tudo o que você precisa sabe,” [Online]. Available: https://www.cetax.com.br/business-intelligence-tudo-o-que-voce-precisa-saber/. [Acesso em 20 Setembro 2020].

[22] Investopedia, “Business Intelligence,” [Online]. Available: https://www.investopedia.com/terms/b/business-intelligence-bi.asp. [Acesso em 18 Setembro 2020].

[23] G. Group, “Analytics and Business Intelligence,” [Online]. Available: https://www.gartner.com/en/information-technology/glossary/business-intelligence-bi. [Acesso em 12 Setembro 2020].

[24] G. C. e. Participações, “Faça downlod do Java e execute programas sem nenhuma complicação,” [Online]. Available: https://www.techtudo.com.br/tudo-sobre/java.html. [Acesso em 20 Setembro 2020].

[25] Oracle, “Oracle Java,” [Online]. Available: https://www.oracle.com/br/java/. [Acesso em 18 Setembro 2020].

[26] NetBeans, “NetBeans IDE Features,” [Online]. Available: https://netbeans.org/features/index_pt_BR.html. [Acesso em 18 Setembro 2020].

[27] H. Medeiros, “Java Swing: Conheça os componentes JTextField e JFormattedTextField,” [Online]. Available: https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981. [Acesso em 19 Setembro 2020].

[28] A. C. Góis, “MySQL: o que é e como usar o sistema,” Tecmundo, 03 09 2021. [Online]. Available: https://www.tecmundo.com.br/software/223924-mysql-usar-o-sistema.htm. [Acesso em 05 07 2022].

[29] P. Pisa, “O que é MySQL,” TechTudo, 2019. [Online]. Available: https://www.techtudo.com.br/noticias/2012/04/o-que-e-e-como-usar-o-mysql.ghtml. [Acesso em 05 07 2022].

[30] Oracle, “Mysql Workbench - Enhaced Data Migration,” Oracle, [Online]. Available: https://www.mysql.com/products/workbench/. [Acesso em 07 2022].

[31] M. Pereira, “Aprenda como o Power BI pode ser seu aliado para poderosas análises de dados,” [Online]. Available: https://www.voitto.com.br/blog/artigo/o-que-e-power-bi. [Acesso em 19 Setembro 2020].

[32] J. S. L. H. Margaret Rouse, “Microsoft Power BI,” techtarget.com, Dezembro 2018. [Online]. Available: https://searchcontentmanagement.techtarget.com/definition/Microsoft-Power-BI. [Acesso em 20 Novembro 2020].

[33] F. I. Ltd, “Visualize data using power bi dashboards and reports,” Fiverr., [Online]. Available: https://www.fiverr.com/hardik39945/visualize-data-using-power-bi-dashboards-and-reports. [Acesso em 25 Outubro 2020].

[34] J. C. Macoratti, “Padrões de Projeto : O modelo MVC - Model View Controller,” [Online]. Available: http://www.macoratti.net/vbn_mvc.htm. [Acesso em 06 Setembro 2020].

[35] A. C. D. Neto, “Bancos de Dados Relacionais,” [Online]. Available: https://www.devmedia.com.br/bancos-de-dados-relacionais/20401. [Acesso em 06 Setembro 2020].

[36] Microsoft, “O que é o serviço do Power BI,” Microsoft, 05 Setembro 2019. [Online]. Available: https://docs.microsoft.com/pt-br/power-bi/fundamentals/power-bi-service-overview. [Acesso em 20 Novembro 2020].

[37] “The 4 Types Of Data Analytics,” 17 Janeiro 2017. [Online]. [Acesso em 11 Novembro 2020].