UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEONARDO PERTILE MONTEIRO DE CASTRO

Sistema de Apoio ao Projeto OBSSAN

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Leandro Krug Wives

Porto Alegre
2014

AGRADECIMENTOS

Aos meus pais, por estarem sempre presentes me guiando nessa longa caminhada, proporcionando o melhor ambiente possível para a minha formação como pessoa e acadêmica.

Aos meus amigos, por compreenderem a ausência nos momentos de estudo, e sempre me proporcionarem os melhores momentos de descontração tão necessários.

Aos colegas de trabalho, que ajudaram muito no meu crescimento como profissional e tiveram muita compreensão com relação as minhas obrigações acadêmicas.

À Universidade Federal do Rio Grande do Sul, em especial ao Instituto de informática, pela estrutura e o ensino de excelência disponibilizados de forma gratuita.

Às duas integrantes do OBSSAN, Angélica Siqueira e Daniela Garcez, pela sua disponibilidade, boa vontade e auxílio na construção desse projeto.

Ao meu orientador, Prof. Leandro Krug Wives, por ter me apresentado o projeto, estar sempre disponível, compreender as minhas outras responsabilidades, e ter compartilhado seu conhecimento.

RESUMO

O objetivo deste trabalho é criar um sistema web para o Observatório Socioambiental em Segurança Alimentar e Nutricional, que auxilie na visualização e análise dos dados de produção alimentar por município do Rio Grande do Sul. O sistema tem a sua base de dados como um Data Warehouse utilizando a modelagem dimensional. Seu desenvolvimento foi feito através da utilização de um processo com princípios ágeis, tendo como sua estrutura básica um framework web e diversos componentes Javascript. Os dados do sistema são visualizados através de buscas com a aplicação de filtros, as quais têm como principal forma de apresentação um mapa com o contorno dos municípios demarcados utilizando a API Javascript disponibilizada pelo Google Maps. São apresentadas telas da versão atual do sistema, explicando seu uso, com o intuito de demonstrar a aplicação dos conceitos e técnologias utilizados.

Palavras-chave: Sistema web, Data Warehouse, OBSSAN, Google Maps API Javascript, dados georreferenciados.

Support System to the OBSSAN Project

ABSTRACT

The goal of this work is to create a web based system to the Social and Environmental Monitoring on Food and Nutrition Security (OBSSAN), that will help to visualize and analyze the food production data by city in the Rio Grande do Sul. The system has it's database as a Data Warehouse using the dimensional modeling. It's development was done through a process with agile principles, having in it's basic structure a web framework and several Javascript components. The system's data are visualized through searches with the application of filters, searches whose main presentation mode is a map with the outline of the cities demarcated using the Javascript API provided by Google Maps. The screens of the current system version are presented explaining it's use, in order to demonstrate the application of the concepts and technologies used.

Keywords: web based system, Data Warehouse, OBSSAN, Google Maps API Javascript, georeferenced data.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CONSEA	Conselho Estadual de Segurança Alimentar e Nutricional
CSS	Cascading Style Sheets
DW	Data Warehouse
DHAA	Direito Humano à Alimentação Adequada
HTML	HyperText Markup Language
IBGE	Instituto Brasileiro de Geografia e Estatística
MER	Modelo Entidade-Relacionamento
MVC	Model-View-Controller
OBSSAN	Observatório Socioambiental em Segurança Alimentar e Nutricional
PGDR	Programa de Pós-Graduação em Desenvolvimento Rural
PHP	PHP: Hypertext Preprocessor
PLANSAN	Plano Nacional de Segurança Alimentar e Nutricional
SAD	Sistema de apoio à decisão
SAN	Segurança Alimentar e Nutricional
SIDRA	Sistema IBGE de Recuperação Automática(SIDRA)
SISAN	Sistema Nacional de Segurança Alimentar e Nutricional
TCC	Trabalho de Conclusão de Curso
UFRGS	Universidade Federal do Rio Grande do Sul

SUMÁRIO

1 INTRODUÇÃO

Este trabalho descreve o desenvolvimento de um sistema web para os pesquisadores do Observatório Socioambiental em Segurança Alimentar e Nutricional (OBSSAN). A premissa básica foi criar um sistema que disponibilizasse uma forma de pesquisar informações, em um banco de dados modelado para armazenar as mesmas, de produção de alimentos dos municípios do Rio Grande do Sul, permitindo interação de forma visual, em mapas, a fim de facilitar sua análise.

O OBSSAN foi criado no ano de 2012, visando operacionalizar no estado do Rio Grande do Sul os indicadores propostos no Plano Nacional de Segurança Alimentar e Nutricional (PLANSAN) 2012/2015. Além disso, ele tem o objetivo de ser um espaço de estudo, pesquisa, e extensão, desenvolvendo o diálogo e um fluxo de informações a nível estadual e municipal entre o Conselho Estadual de Segurança Alimentar e Nutricional (CONSEA), academia, setores governamentais e sociedade civil (SIQUEIRA et al., 2014). O OBSSAN foi criado pelo Núcleo de Estudos e Pesquisas em Segurança Alimentar e Nutricional (NESAN), vinculado ao Programa de Pós-Graduação em Desenvolvimento Rural (PGDR) da UFRGS, em parceria com o CONSEA e o Departamento de Informática da UFRGS. A fim de facilitar o acesso às informações que compõem os indicadores propostos no PLANSAN, e assim cumprir seus objetivos, o OBSSAN propôs a criação de uma plataforma web com acesso público que poderá assim subsidiar ações de assistência técnica, gestão, controle social e produção de análises sobre Segurança Alimentar e Nutricional (SAN) no Rio Grande do Sul.

É importante salientar que a segurança alimentar e nutricional (SAN) consiste na realização do direito de todos ao acesso regular e permanente a alimentos de qualidade, em quantidade suficiente, sem comprometer o acesso a outras necessidades essenciais tendo como base práticas alimentares promotoras de saúde que respeitem a diversidade cultural e que sejam ambiental, cultural, econômica e socialmente sustentáveis (CONSEA, 2006). Com o intuito de monitorar a execução das políticas de SAN no Brasil, foi criado o já citado PLANSAN 2012/2015, institucionalizado pelo Decreto Federal 7.272/2010. Esse plano propõe sessenta indicadores divididos em sete dimensões, sendo somente a primeira dimensão, Produção de Alimentos, abordada neste trabalho. O sistema proposto deverá

centralizar o acesso as informações de produção de alimentos nos municípios do estado de Rio Grande do Sul, sendo parte integrante da plataforma web planejada pelo OBSSAN.

O objetivo principal deste trabalho é, portanto, desenvolver o sistema para apoiar o OBSSAN em suas tarefas relacionadas com a coleta e análise de informações, facilitando assim a criação e monitoramento de políticas de SAN.

O texto está estruturado da seguinte forma. O capítulo seguinte apresenta o referencial teórico, isto é, a metodologia e os conceitos que estão por trás do sistema desenvolvido. No capítulo 3 são a aplicados alguns dos conceitos apresentados, contendo o processo de desenvolvimento utilizado, requisitos e modelagem do sistema. O capítulo 4 expõe as tecnologias utilizadas para construção do sistema juntamente com alguns exemplos de seu uso. Já no capítulo 5 é apresentado o sistema desenvolvido juntamente com algumas formas de utilizá-lo.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta a metodologia de desenvolvimento seguida, as ferramentas utilizadas e os conceitos relacionados com o trabalho a fim de facilitar a sua compreensão.

2.1 Metodologia

Softwares aplicativos, ou sistemas, são programas de computador que auxiliam usuários a desempenhar tarefas. Esses sistemas são abstratos e intangíveis, não sendo governados por leis da física, propriedades de materiais, ou processos de fabricação (SOMMERVILLE, 2011). Com isso, eles facilmente se tornam muito complexos, difíceis de entender e de complicada modificação. Com o intuito de mitigar essas situações, são utilizados processos já estabelecidos para o seu desenvolvimento. Por se tratar de um sistema web, foi utilizada uma abordagem baseada em *Web Engineering*, que segundo Pressman e Lowe (2009) é definida como uma proposta de processo ágil, porém disciplinada de construção de aplicativos web de alta qualidade. Esse processo estabelece uma série de atividades básicas que deveriam ser aplicadas para construção da maioria dos projetos web. As atividades que fazem parte desse grupo são: Comunicação, Planejamento, Modelagem, Construção e Implantação, conforme a Figura 2.1.

Cada uma dessas atividades é composta de uma série de ações, tendo cada uma delas a possibilidade de ser composta por tarefas que levem a completar a ação. Como as tarefas podem ser escolhidas da forma que melhor se adaptem às necessidades do projeto, as ações e as atividades acabam, por consequência, sendo aplicáveis em projetos com diferentes características. Para uma adaptação mais eficiente do processo, é sugerido um enfoque maior na agilidade do projeto utilizando os princípios das metodologias ágeis. Como consequência disso, neste trabalho deu-se ênfase a alguns dos princípios e valores propostas no *Agile Manifesto* (BECK et al., 2001), sendo elas: priorização de software em funcionamento mais do que documentação abrangente e entrega frequente de software funcional com valor agregado.

O processo é organizado em diferentes incrementos, sendo em cada um deles entregue algum conteúdo ou funcionalidade para o usuário final. Como esses incrementos podem

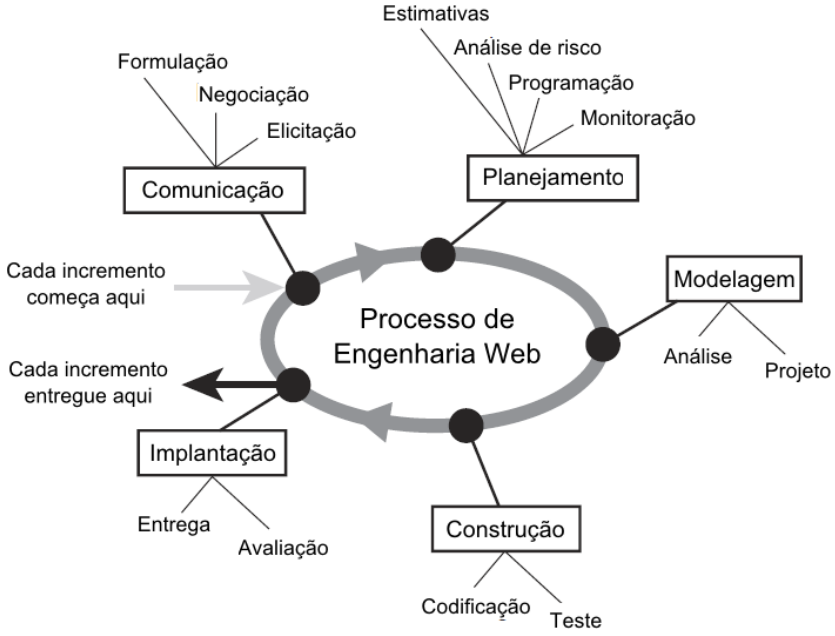


Figura 2.1: Exemplo de incremento do processo com atividades e ações
Fonte: Adaptado de PRESSMAN; LOWE 2009, p.26

incluir diferentes funcionalidades com complexidades variadas, cada um deles deve ter suas atividades, ações e tarefas adaptadas de forma a melhor atender as suas necessidades. Como exemplo da necessidade de adaptação, pode-se utilizar o primeiro incremento, no qual ocorre intensa atividade comunicação, planejamento e modelagem, em função de se estar iniciando o projeto. Já os incrementos seguintes devem ter, no geral, a construção como foco maior.

De qualquer forma, um dos primeiros passos consiste em identificar os requisitos do sistema, e documentá-los. Para tanto, optou-se por *User Stories*.

2.1.1 User Stories

A aquisição dos requisitos de um sistema é uma das partes mais importantes do processo de desenvolvimento. Segundo Sommerville (2004), requisitos são as descrições das funcionalidades que o sistema deve fornecer ao usuário de forma a ajudá-lo a resolver algum problema, juntamente com as suas restrições operacionais, ou seja, todos os motivos pelos quais o sistema está sendo criado e todas as demandas que deve atender. Em conformidade com a ideia ágil do projeto, decidiu-se utilizar o método de *user stories* para essa ação.

User story é uma pequena e simples descrição textual de uma nova funcionalidade feita, em geral, em cartões de papel, sendo essa descrição criada na perspectiva de uma pessoa que normalmente é um usuário ou cliente do sistema (COHN, 2009). O objetivo principal de sua utilização é trocar o foco da descrição textual das funcionalidades, para a discussão a respeito delas, tentando assim eliminar ao máximo os erros causados por má

interpretação ou ambiguidade na forma textual. Existem diversos formatos padrões em que elas podem ser escritas, entretanto para este trabalho foi definido o seguinte:

Como um <papel de usuário> eu quero <objetivo/razão>.

2.2 Arquitetura Web

Como descrito na introdução, o sistema será uma aplicação web, visando assim facilitar o seu acesso não só pelos pesquisadores como também pelo público em geral. Essa escolha foi feita pela simplicidade de se acessar aplicações desse tipo, bastando ter um computador com navegador web instalado e acesso à Internet.

Sommerville (2011) define que o projeto da arquitetura de um sistema deve se preocupar em como o sistema deve ser organizado e sua estrutura básica. Além disso, a arquitetura de software é importante por influenciar diretamente a performance, robustez e manutenibilidade do sistema (BOSCH 2000 apud SOMMERVILLE 2011, p.149), por isso, aplicações web normalmente seguem o padrão model-view-controller (MVC).

Na web, arquitetura física é baseada no padrão Cliente-Servidor. A estrutura da aplicação também é simples, sendo dividida em duas entidades, que geralmente ficam em computadores diferentes, o cliente e o servidor. A comunicação entre ambos é feita por rede através do protocolo *Hypertext Transfer Protocol* (HTTP). As subseções seguintes explicam os componentes físicos e lógicos normalmente encontrados em uma aplicação web (a saber: cliente, servidor e MVC).

2.2.1 Servidor

A função básica do servidor é aguardar requisições de um cliente, atendê-las quando ocorrerem, e enviar os dados referentes a requisição. No caso do sistema proposto neste trabalho, ele pode ser chamado de servidor web, sendo responsável por responder pedidos de clientes, tipicamente navegadores web, através de recursos como documentos *HyperText Markup Language* (HTML) (DEITEL; DEITEL, 2008).

Os documentos HTML, ou páginas web, informam a maneira de formatar e exibir o seu conteúdo, contudo possuem apenas conteúdo estático, ou seja, o que está definido nele é o que será visualizado pelo usuário, sem possibilidade de alteração. Levando em conta que o sistema é baseado em um banco de dados, e precisará mostrar informações diferentes para cada tipo de busca feita pelo cliente, a utilização de somente documentos HTML puros fica inviabilizada. Para solucionar isso, podem ser usadas linguagens de script como o *PHP: Hypertext Preprocessor* (PHP). O PHP (2014) é uma das linguagens de script, do lado servidor, de maior popularidade para criar documentos HTML dinamicamente (DEITEL; DEITEL, 2008), sendo possível, através dele, acessar bancos de dados que armazenem os dados do sistema.

Banco de dados é uma coleção organizada de dados, existindo diversas estratégias de se atingir essa organização. Um sistema de gerência de banco de dados (SGBD) disponibiliza mecanismos para organização, recuperação e modificação de dados, permitindo acesso e armazenamento dos dados sem preocupação com a organização física dos mesmos (DEITEL; DEITEL, 2008). Um dos tipos de bancos de dados mais utilizado é o relacional, no qual os dados são armazenados em tabelas que armazenam os dados em linhas compostas por colunas. A linguagem padrão utilizada para a manipulação dos dados nesse tipo de banco é a *Structured Query Language* (SQL).

2.2.2 Cliente

O cliente tem o papel básico de iniciar requisições para servidores, esperar a resposta, e recebê-la. Por ser uma aplicação web, o cliente é tipicamente um navegador web, através do qual o usuário tem acesso aos recursos da aplicação. Como o sistema encontra-se no servidor web, é necessário que se tenha o seu endereço na rede, ou *Uniform Resource Locator* (URL), para poder acessá-lo. A URL é utilizada pelo HTTP para identificar os recursos disponibilizados pelo servidor, permitindo que ambos interajam e troquem informações de maneira confiável e uniforme (DEITEL; DEITEL, 2008). Através desse processo são recebidos os documentos HTML da aplicação, os quais irão ser interpretados pelos navegador afim de exibir o seu conteúdo para o usuário.

Juntamente com o documento HTML é necessária a utilização do *Cascading Style Sheets* (CSS) para compor a interface final que será renderizada pelo navegador. CSS, segundo Deitel e Deitel (2008), é uma tecnologia que permite a especificação da apresentação dos elementos de uma página web, separado a estrutura da apresentação, facilitando assim a manutenção da mesma.

Além desses recursos, o cliente ainda conta com a utilização de JavaScript, uma linguagem de script interpretada dentro do navegador, a qual melhora a funcionalidade e aparência de páginas web (DEITEL; DEITEL, 2008). Isso ocorre em função da capacidade do JavaScript de alterar o documento HTML, após ele ter sido recebida do servidor e renderizada no navegador.

2.2.3 MVC

Toda aplicação web é baseada na arquitetura (física) Cliente-Servidor, recém descrita. No entanto, também é importante organizar a lógica de aplicação em diferentes camadas a fim de isolar seus componentes e minimizar o acoplamento. No que tange aplicações web, uma arquitetura (lógica) bastante utilizada é o padrão model-view-controller (MVC).

Apesar de sistemas de software serem únicos, e possuírem cada um características diferentes, quase sempre sua arquitetura pode ser classificada dentro de uma categoria que reflita as características básicas de seu domínio de aplicação. Esses domínios de aplicação geralmente possuem padrões de projeto de arquitetura que descrevem, de maneira

abstrata, boas práticas já utilizadas e testadas em sistemas, e ambientes diferentes, com sucesso. Pode-se então reutilizar esses padrões em novos sistemas, visando aproveitar as vantagens de ele já ter sido testado e utilizado com sucesso.

Um desses padrões é o de camadas, que segundo Sommerville (2011), parte do princípio no qual a separação e independência são fundamentais no projeto arquitetural, por permitirem mudanças localizadas. Cada uma das camadas possui funções específicas, fornecendo serviços para as demais. O MVC foi utilizado por ser um padrão de três camadas comumente utilizado em aplicações web, o qual facilita a implementação e melhora o reuso (PRESSMAN; LOWE, 2009). Isso ocorre graças a separação do sistema entre interface, navegação, e comportamento da aplicação, conforme pode ser compreendido na Figura 2.2.

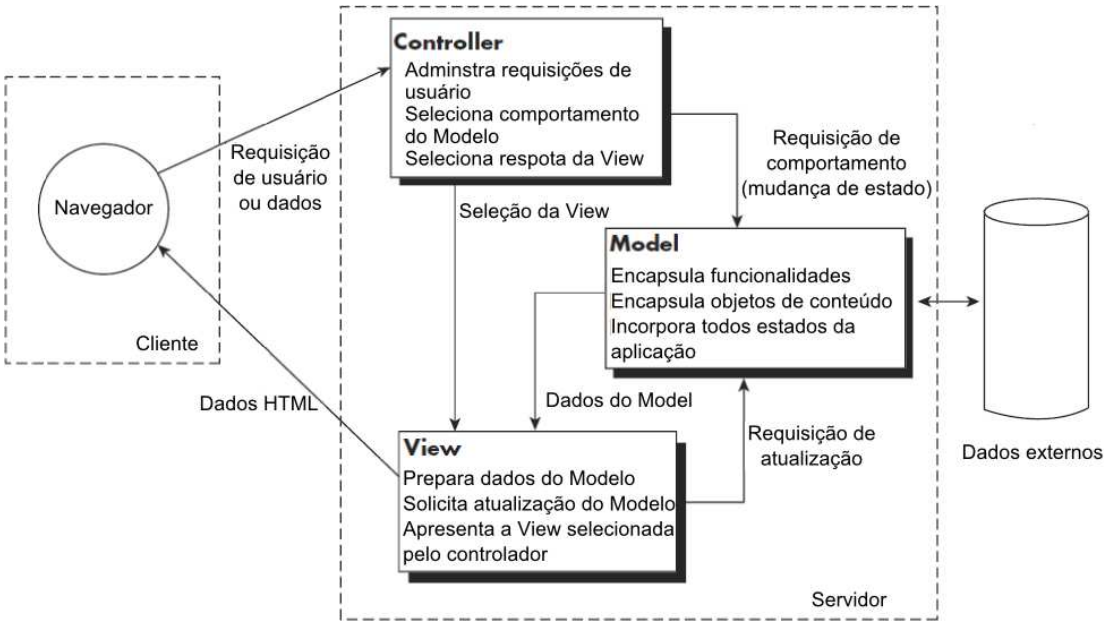


Figura 2.2: Arquitetura MVC
Fonte: Adaptado de PRESSMAN; LOWE 2009, p.285

Essas três camadas são definidas a seguir:

***Controller* (controlador):** Administra o acesso, a manipulação e o fluxo de dados entre o model e a view. Para isso, ele monitora a interação do usuário com a aplicação, baseando-se nisso para escolher quais dados pegar da *Model* e utilizá-los para construir ou atualizar a view.

***Model* (modelo):** Contém toda a lógica de processamento e conteúdo específico da aplicação, sendo feito a partir dele o acesso de fontes externas de dados, no caso o banco de dados MySQL.

***View* (visão):** Possui todas as funções específicas de interface, possibilitando a apresentação de todo o conteúdo gerado pelo *Model* para o usuário.

Um fluxo simples no padrão MVC se inicia com uma requisição do usuário, feita através do navegador, a qual é encaminhadas ao *controller* que seleciona qual a *view* adequada para atendê-la. O *controller* ainda identifica, a partir da requisição, qual ação tomar, solicitando ao *model* que implemente a ação ou retorne os dados que satisfaçam a requisição. Os dados retornados pelo *model* precisam, em geral, ser formatados e organizados para apresentação. Isso é feito pela *view* antes de a aplicação responder a requisição ao cliente, que irá visualizá-la no seu navegador.

2.2.4 Frameworks

Além de se fazer o reuso de padrões para arquitetura do sistema, é possível utilizar o reuso no desenvolvimento do sistema em si, ou seja, na sua programação. O reuso pode ser definido como uma estratégia na qual o processo de desenvolvimento é guiado pela reutilização de software já existente (SOMMERVILLE, 2011). As suas principais vantagens são semelhantes as já citadas no padrão MVC, pois ao se fazer uso de um software, ou componente, já testado e utilizado com sucesso em outras aplicações, diminui-se a chance de erros e aumenta-se a segurança do desenvolvedor, acelerando assim o processo de desenvolvimento.

Entretanto, existem também algumas desvantagens, principalmente ligadas a escolha do componente. Isso se deve ao fato de que é necessário compreender seu funcionamento, verificando as maneiras de adaptá-lo, além de analisar se é realmente adequado à situação. Frameworks de aplicação seguem esse conceito, pois, segundo Pressman (2009), são um conjunto de bibliotecas ou componentes utilizados para criar a estrutura básica de uma aplicação, ou seja, fazem com que o desenvolvedor não precise lidar com detalhes básicos e possa desenvolver diretamente o código vinculado aos requisitos do sistema. Além disso, sua implementação é feita normalmente utilizando padrões de projeto, incluindo MVC, e diversos outros discutidos por GAMMA et al. 1995 (SOMMERVILLE, 2011), o que torna sua utilização ainda mais segura. Para utilizar um framework, não se deve alterar seu código fonte, e sim criar classes concretas, que estendam as operações herdadas das classes abstratas, afim de realizar as tarefas desejadas.

Nos frameworks de aplicação web, segundo Sommerville (2011), normalmente existe suporte a um conjunto de recursos muito utilizados nesse tipo específico de aplicação, como gerenciamento de sessão, controle de acesso por autenticação, classes que auxiliem na montagem de páginas dinamicamente através de templates, classes que forneçam interfaces abstratas para acesso a diferentes bancos de dados.

2.3 Data Warehouse

Pelo fato de a ferramenta desenvolvida ter caráter analítico, onde as informações deverão ser consultadas, cruzadas e apresentadas de forma visual, em mapas ou gráficos,

optou-se por desenvolver um Data Warehouse (DW) ao invés de um banco de dados tradicional.

O conceito básico de banco de dados já está disponível há bastante tempo. Com a sua evolução, foram criados dois novos conceitos visando atender as diferentes necessidades de armazenamento de dados que surgiam: os bancos de dados operacionais e os bancos de dados informacionais ou analíticos. Os bancos de dados operacionais, segundo Inmon(1997), são os que trabalham com dados primitivos, que estão em pequena quantidade por processo, sendo exatos em relação ao momento do acesso, voltados para transações, e necessitando de alta disponibilidade. Um exemplo disso seria um banco de dados que alimentasse um sistema, o qual tivesse como principal objetivo criar, ler, atualizar e deletar dados referentes as operações cotidianas de uma empresa através de transações.

Já os bancos de dados informacionais, são os que utilizam dados derivados, que estão em grande quantidade por processo, representando valores já decorridos ou instantâneos, voltados para análise, e com disponibilidade atenuada (INMON, 1997). É nesse segundo conceito que o DW se encaixa. Nele é armazenada uma cópia dos dados das transações, ou seja, dos dados operacionais, estruturando-os de um forma específica para consultas e análises (KIMBALL, 1998). Um exemplo da diferença entre os dois conceitos pode ser visto na figura 2.3, na qual se tem a informação atual no operacional e o histórico no informacional.

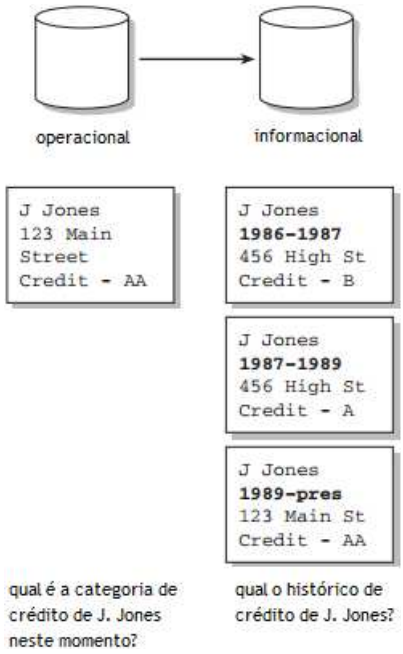


Figura 2.3: Exemplo de tipos de consultas nos dois tipos de base de dados
Fonte: Adaptada de (INMON, 1997, p.21)

Complementando a definição anterior, anterior "Um data warehouse é um conjunto de dados baseado em assuntos, integrado, não-volátil e variável em relação ao tempo, de

apoio a decisões gerenciais"(INMON, 1997, p.33). As características descritas por Inmon são definidas a seguir:

Baseado em assuntos: apresenta informações específicas, como movimento diário de um produto comercializado, tendo os dados originais vindo de estrutura operacional de processos de compra e venda.

Integrado: múltiplas fontes ou tipos de dados são condensadas com objetivo de retirar inconsistências.

Não-volátil: os dados são carregados em grande quantidade e geralmente não sofrem modificação após a finalização do processo.

Variável em relação ao tempo: os dados são uma série sofisticada de instantâneos capturados num determinado momento.

A partir dessas características, o DW apresenta um comportamento primário de geração de relatórios gerenciais, podendo consolidar grande quantidade de dados, com fontes diferentes, em informações relevantes. Essas informações tem o principal papel de ajudar na tomada de decisões, o que torna o DW um sistema de apoio à decisões (SAD). Introduzido o conceito, parte-se para a modelagem dos dados que farão parte do mesmo, de forma a que satisfaçam da melhor maneira as características apresentadas.

2.4 Modelagem de Dados

Para que um DW funcione da maneira esperada, a organização, ou modelo, dos dados é essencial. O modelo entidade-relacionamento (MER) é o mais comumente utilizado, principalmente, em bases de dados operacionais. Entretanto, como possui características muito diferentes, a base de dados informacional necessita de uma abordagem diferente, a modelagem dimensional. A seguir serão descritas as duas modelagens e os motivos principais que levam a utilização de uma em detrimento da outra.

2.4.1 Entidade-Relacionamento

O modelo entidade-relacionamento separa os dados em várias entidades distintas, sendo cada uma delas transformada em uma tabela do banco de dados, buscando-se remover qualquer redundância de dados que exista (KIMBALL, 1998). Para isso, utiliza-se geralmente a criação de novas tabelas, as quais irão armazenar os dados antes redundantes, e se relacionarão com a tabela original através de colunas de identificação, chamadas de chaves estrangeiras. Esse processo acaba gerando uma grande quantidade de tabelas com muitos relacionamentos entre si, tendo assim uma representação complexa por diagramas. Esse diagrama acaba sendo muito simétrico, ou seja, todas as tabelas parecem

iguais, não possibilitando avaliar concretamente sua importância dentro do modelo, e se armazenam valores ou descrições estáticas.

Para bases de dados operacionais, essa falta de redundância é a chave para um bom desempenho. Isso se justifica, pois quando uma transação precisa modificar um dado na base, ela atuará em apenas um ponto, e não em diversos. Todavia, para gerar relatórios gerenciais, principal objeitvo do DW, esse modelo acaba tendo problemas, tanto de desempenho, quanto de compreensão de sua estrutura. Os relatórios gerenciais tendem a utilizar uma grande quantidade de tabelas para serem gerados, sendo assim, são necessárias operações de *join*, junção em português, entre muitos registros de tabelas ao mesmo tempo, algo que os sistemas de gerência de banco de dados relacional não conseguem lidar bem. Visando atender as necessidades do DW, foi então proposto o modelo dimensional.

2.4.2 Dimensional

O modelo dimensional, ou também conhecido como *star schema*, esquema estrela em português, parte do mesmo princípio do entidade-relacionamento. Ele separa os dados em entidades que depois são transformadas em tabelas no banco de dados. Contudo, a maneira que os dados são agrupados é diferente. Existe uma tabela dominante com múltiplos relacionamentos conectando-a as demais tabelas secundárias, cada uma delas tendo apenas uma junção com a tabela central, tornado a representação por um diagrama muito assimétrica (KIMBALL, 1998). Essa organização é muito mais simples, e facilita a compreensão dos usuários finais. A tabela dominante é chamada de tabela de fatos, e as secundárias de tabelas de dimensão, conforme a figura 2.4.

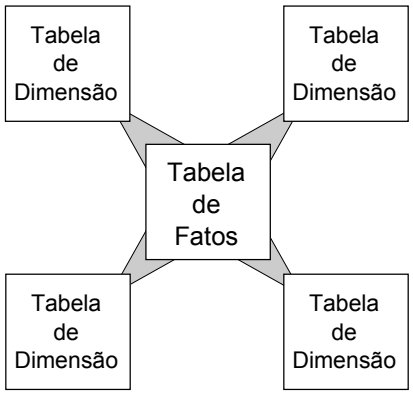


Figura 2.4: Esquema estrela de exemplo com quatro dimensões
Fonte: Figura criada pelo autor

Segundo Kimball (1998), a tabela de fatos armazena, na sua essência, dados de medições numéricas aditivas, sendo cada uma dessas informações obtidas através da intersecção com todas as dimensões. Esses fatos são chamados de fatos aditivos. Sendo assim, a chave primária da tabela de fatos é composta basicamente da chave primária de cada uma das tabelas dimensões ligadas a ela. O motivo principal de se utilizar dados numéri-

cos nessa tabela de fatos, é que em praticamente todas as consultas que forem realizadas nela, será utilizada uma soma para compactar o conjunto de resposta. Como essas tabelas costumam ter um grande número de registros, ao compactá-los será possível construir um conjunto de resposta muito mais facilmente.

Além dos fatos aditivos, ainda existem os fatos semi-aditivos e os não-aditivos. Os semi-aditivos só podem ser adicionados em algumas dimensões, já os não-aditivos não podem ser adicionados nunca. Dependendo da informação que o fato não-aditivo represente, pode-se utilizar contagens para que ocorra a compactação dos registros, assim como a que ocorre nos aditivos, caso contrário, cada um dos registros deverá compor o conjunto de resposta.

As tabelas dimensionais armazenam descrições textuais das informações que estão sendo modeladas no DW, sendo que cada uma dessas descrições ajuda a definir um componente da respectiva dimensão (KIMBALL, 1998). Um exemplo disso seria uma dimensão produto em um DW de vendas, no qual cada um dos registros dessa dimensão representaria um produto específico vendido.

Algumas vezes são utilizados dados numéricos nas dimensões, o que pode ocasionar certa dúvida sobre a classificação dos mesmos como fato ou atributo da dimensão. Uma das maneiras de solucionar isso é verificando se esse campo numérico é uma medição que pode variar continuamente em cada amostragem , o que o torna um fato, ou se é uma descrição praticamente constante de um item, o que o torna um atributo de dimensão. Um dado que poderia se encaixar nesse perfil seria o tamanho de um produto, visto que é expresso de forma numérica, mas tem um comportamento de uma descrição textual, ou seja, para uma mesma descrição de produto não varia. Os atributos das dimensões tem como função principal servir de fonte para restrições das consultas aos fatos ou como cabeçalhos de linha no conjunto de resposta das consultas.

Outro conceito relacionado a modelagem dimensional e a DW é o de *Online Analytical Processing* (OLAP), o qual é uma categoria de software com intuito da rápida exploração e análise de uma base de dados utilizando uma abordagem multidimensional com vários níveis de agregação (RIVEST; BÉDARD; MARCHAND, 2001). Ou seja, é uma interface intuitiva para que o usuário consiga criar relatórios de forma facilitada. Juntamente com ele, surgiu o conceito de *Spatial Online Analytical Processing* (SOLAP), que segundo Bédard (1997 apud RIVEST; BÉDARD; MARCHAND 2001), é uma plataforma visual construída para suportar rápidas e fáceis análises espaço-temporais, seguindo uma abordagem multidimensional, disponibilizando exibições cartográficas, tabulares e de diagramas.

3 VISÃO GERAL DA PROPOSTA E MODELAGEM

Neste capítulo será apresentado de que forma os conceitos do capítulo anterior foram utilizados para estabelecer as bases da implementação do sistema.

3.1 Processo

A partir dos conceitos apresentados na seção 2.1, definiu-se o processo de desenvolvimento do sistema seguindo o esquema de iterações incrementais curtas, tendo cada uma das atividades do processo as seguintes funções:

- Comunicação: Interação com os integrantes do OBSSAN, através de reuniões presenciais ou remotas, visando levantar os requisitos do sistema utilizando *user stories*, e refinar *user stories* já criadas.
- Planejamento: Definir quais user stories seriam incluídas na iteração e os prazos de conclusão.
- Modelagem: Avaliação de quais mudanças são necessárias no sistema para incluir as novas user stories e qual a melhor forma de as acomodar na estrutura já existente.
- Construção: Efetivar as mudanças planejadas, incluindo estrutura do banco de dados, e código.
- Implantação: Disponibilizar o sistema em um servidor web para que os integrantes do OBSSAN possam avaliar o sistema.

Em cada uma dessas iterações procurou-se seguir todas as atividades do processo proposto, sendo o objetivo principal de cada iteração completar uma ou mais das user stories, e realizar ajustes referentes a avaliação da iteração anterior.

3.2 User Stories

Softwares aplicativos, por auxiliarem usuários em suas tarefas, tendem sofrer mudanças nos seus requisitos. Isso se deve ao fato de que muitas vezes o usuário não consegue

identificar de quais maneiras esse software pode ajudá-lo, passando a adquirir esse conhecimento com o uso do mesmo. Contudo, para iniciar o desenvolvimento de um novo sistema, é necessário que exista um conjunto mínimo de requisitos definidos para nortear uma versão inicial.

No sistema proposto junto ao OBSSAN, foram utilizadas as seguintes *user stories*, com o formato descrito na seção 2.1.1, para definir os requisitos iniciais:

Nº	**Como um**	**Eu quero**
1	Usuário padrão	Ver os resultados da pesquisa em um mapa.
2	Usuário padrão	Que o mapa apresente os valores do município ao passar o mouse por cima do mesmo.
3	Usuário padrão	Poder agrupar os resultados da pesquisa por mesorregiões e microrregiões.
4	Usuário padrão	Poder filtrar as buscas por região (município, mesorregião, microrregião).
5	Usuário padrão	Poder agrupar a pesquisa por tipos que possam ser somados.
6	Usuário padrão	Poder delimitar um intervalo de valores na pesquisa.
7	Usuário padrão	Poder exportar a pesquisa em uma tabela (csv).

Tabela 3.1: *User stories* do sistema

3.3 Modelo de Dados do Sistema Proposto

Os pesquisadores do OBSSAN já possuíam grande parte dos dados que compõem os indicadores da dimensão Produção de Alimentos no momento do desenvolvimento desse trabalho. Esses dados estão, em sua maioria, disponíveis na base de dados do Instituto Brasileiro de Geografia e Estatística (IBGE) e são coletados através do Sistema IBGE de Recuperação Automática (SIDRA)[1].

A obtenção dos dados foi feita através do esforço de colaboradores do OBSSAN que os organizaram por ano e município em tabelas utilizando o software OpenOffice Calc[2]. Esses dados são numéricos e representam quantidades ou área, sendo a sua granularidade de tempo anual, ou seja, a cada ano são adicionados novos dados. Apesar de já estarem a

[1] http://www.sidra.ibge.gov.br/
[2] https://www.openoffice.org/

disposição, a análise dos dados torna-se um tanto quanto difícil, pois as tabelas apresentam apenas uma visão numérica isolada de cada dado, não proporcionando uma ligação eficiente com a área geográfica a qual se referem e nem formas eficientes de agregá-los. Com isso, o objetivo do trabalho é criar um sistema web que forneça formas de se pesquisar esses dados, possibilitando agregações, quando possível, e apresentando-os em um mapa que referencie a localização geográfica a qual eles pertencem. A partir das informações obtidas com as *user stories* apresentadas na seção anterior, foi possível identificar um perfil predominante de uso do sistema, o de consultas e refinamento de consultas em dados que não sofrem alteração constante e que possuem um aspecto temporal. Logo, notou-se a semelhança desse perfil com o de um data warehouse (DW), armazém de dados em português, optando-se assim por utilizar neste trabalho a modelagem que é a padrão nesse tipo de aplicação, a modelagem dimensional. Por possuir um DW com a utilização de referência cartográfica, pode-se classificar o sistema como uma versão simplificada de uma ferramenta SOLAP.

Após a análise do documento com os dados disponibilizados pelo OBSSAN, que foi citado no início da seção, e reuniões com seus integrantes, decidiu-se utilizar um modelo derivado do esquema estrela. Isso ocorreu, pois o OBSSAN tinha interesse em ter os dados agrupados em grandes grupos de assunto em comum, para posteriormente poder fazer cruzamentos entre os mesmos e gerar relatórios mais complexos. Esse modelo é chamado de *galaxy schema*, em português esquema galáxia, no qual existe mais de um esquema estrela compartilhando dimensões sem que exista a necessidade de uma ligação direta entre os fatos (MOODY; KORTINK, 2000).

A seguir serão apresentados os fatos e dimensões que resultaram da modelagem, assim como o diagrama da mesma.

3.3.1 Fatos

Foram definidos quatro assuntos principais dentre os dados disponibilizados sobre produção alimentar. Cada um desses assuntos gerou uma tabela de fatos distinta, sendo eles:

Animais: Quantidade de animais, por exemplo, número de cabeças de bovinos.

Gêneros Alimentares: Quantidade ou área de plantação de alimentos, por exemplo, toneladas de arroz.

Estabelecimentos Agropecuários: Área ou quantidade de estabelecimentos agropecuários, por exemplo, área de pastagem.

Trabalhadores Rurais: Número de pessoas ocupadas com trabalho rural.

Em cada um desses fatos existe uma coluna "quantidade"que é onde ficam armazenados os fatos aditivos. Foi adicionada uma coluna "homologado"nas tabelas de fato,

pois era de interesse do OBSSAN que os dados inseridos pudessem ser validados antes de ficarem disponíveis ao público em geral. A tabela de fatos referente aos trabalhadores rurais, "pessoas_ocupadas", teve a sua tabela de dimensão, na qual são especificados os tipos de trabalhadores rurais, incorporada. Isso ocorreu devido a pequena quantidade de tipos de trabalhadores rurais, quatro, diminuindo assim o número de junções para retornar um resultado referente a esse fato. Além dessas, ainda foi adicionada em todas as tabelas de fatos uma coluna "ano", o que será explicado a seguir.

3.3.2 Dimensões

A partir das três tabelas de fatos que não possuíam ainda descrição, e das demais descrições necessárias para conseguir identificar cada um dos fatos corretamente, criou-se as seguintes dimensões:

Tipo de Animal: Especifica sobre qual animal o fato se refere, qual a unidade do fato, e se o fato é de agricultura familiar.

Tipo de Gênero Alimentar: Especifica sobre qual gênero alimentar o fato se refere, qual a unidade do fato, se o fato é de agricultura familiar, e se é um derivado animal.

Tipo de Estabelecimento Agropecuário: Especifica sobre qual tipo de estabelecimento agropecuário o fato se refere, qual a unidade do fato, e se o fato é de agricultura familiar.

Indicador: Especifica a qual indicador e dimensão do PLANSAN o valor se refere.

Fonte dos dados: Especifica qual o nome da fonte de dados e de que tabela dessa fonte eles foram obtidos.

Localização Geográfica: Especifica a geolocalização do fato.

Tempo: Especifica quando o fato ocorreu.

Por possuírem diversas informações transformadas em colunas, cada uma das dimensões gerou uma tabela, exceto a dimensão tempo. Uma tabela não foi necessária, pois a única informação de tempo que os dados possuem é o ano que representam. Sendo assim, uma coluna na tabela de fatos que passe a integrar a sua chave primária, é suficiente para especificar essa informação. Caso fosse necessário armazenar informações em maior quantidade e mais precisas, como dia da semana, ou dia útil, então seria essencial criar uma tabela para essa dimensão (KIMBALL, 1998). Cada uma das três primeiras dimensões dizem respeito a somente um fato, logo compõem a chave primária de cada um desses fatos. As últimas quatro são compartilhadas entre todos os fatos, visto que cada um deles necessita dessas descrições para ser corretamente identificado.

Na figura 3.1 estão representados as tabelas, juntamente com as relações entre as mesmas, criadas a partir da modelagem apresentada. As Tabelas vermelhas representam os fatos, as cinzas as dimensões específicas dos fatos, e as azuis as dimensões compartilhadas entre todos os fatos.

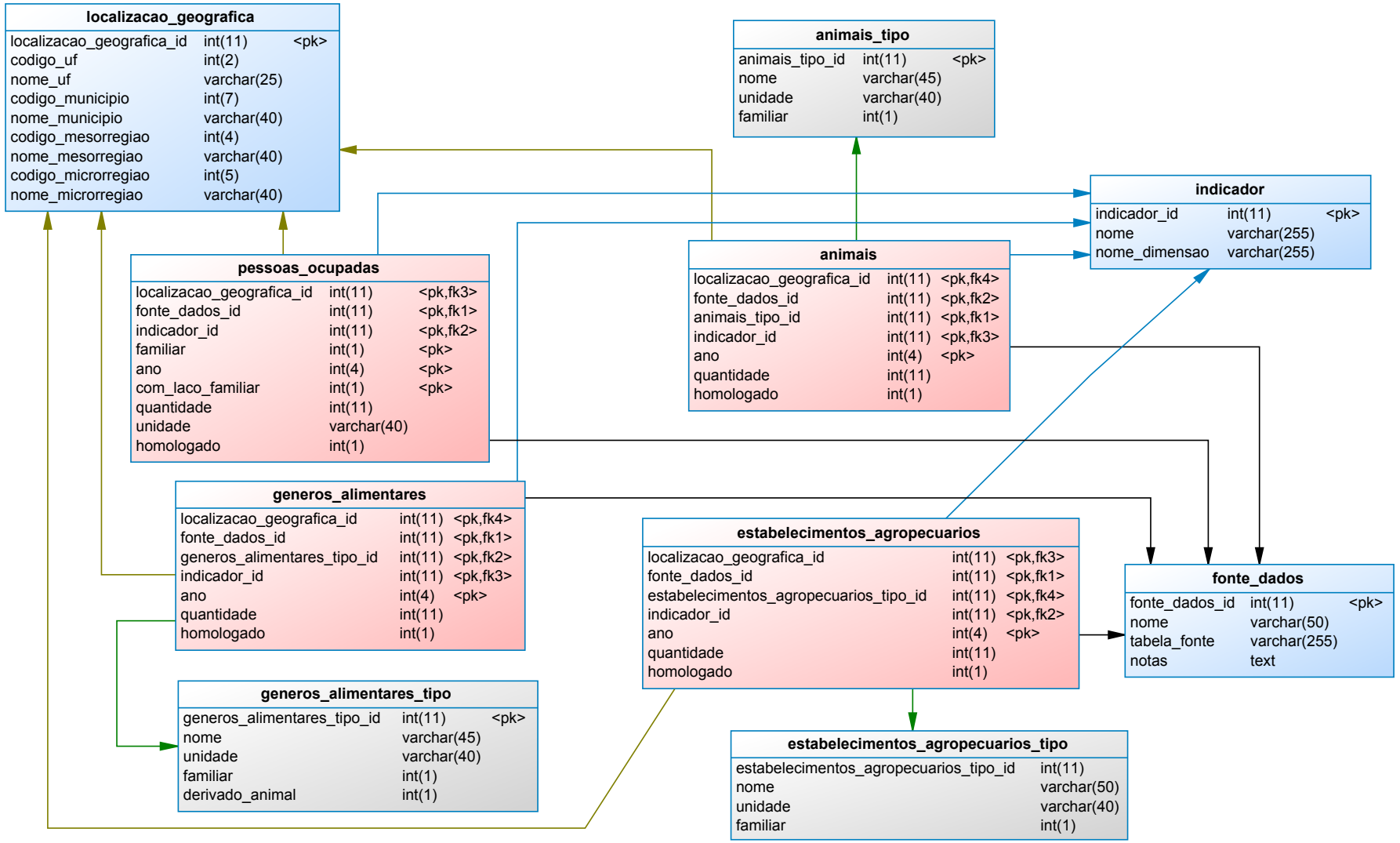


Figura 3.1: Diagrama da modelagem do banco de dados do sistema

4 IMPLEMENTAÇÃO

Nesse capítulo será apresentada a implementação do sistema, com o objetivo de demonstrar de que maneira a base de dados modelada foi unida com os requisitos definidos nas user stories para gerar o sistema final.Para isso optou-se pelo uso de ferramentas que fossem de código aberto ou gratuitas.

Como servidor web para o desenvolvimento do sistema, foi escolhido um dos softwares mais comumente utilizados nessa função, o Apache HTTP Server (2014), visto que possui suporte a utilização do PHP. O SGBD relacional escolhido para armazenar os dados da do sistema foi o MySQL (2014), pois assim como o Apache, possui uma excelente integração com o PHP. A máquina utilizada como servidor web durante o desenvolvimento do sistema foi um computador com sistema operacional Microsoft Windows 7[1] e VertrigoServ[2] instalados. O VertrigoServ é um software que instala e configura os três componentes citados acima, Apache, PHP e MySQL, acelerando assim o processo de início do desenvolvimento. Esse mesmo computador também foi utilizado como cliente para testes, fazendo uso do navegador web Google Chrome[3].

4.1 Framework Yii

Como citado anteriormente, o sistema proposto deve ser uma aplicação web. É importante frisar que se optou por utilizar a tecnologia PHP, principalmente pelo fato de que a aplicação pronta irá ficar no servidor web da UFRGS, o qual suporta essa tecnologia juntamente com poucas outras.

Como explicitado na Seção 2.2.4, ao se desenvolver uma aplicação web é importante utilizar um framework de aplicação, pois eles oferecem suporte a gerenciamento de sessão, controle de acesso por autenticação, classes que auxiliem na montagem de páginas dinamicamente através de templates, classes que forneçam interfaces abstratas para acesso a diferentes bancos de dados, entre outros recursos relevantes. Por esse motivo, e pelo fato de o CPD da UFRGS também adotar o mesmo framework em seus sistemas,

[1] http://windows.microsoft.com/en-us/windows/home
[2] http://vertrigo.sourceforge.net/
[3] https://www.google.com/intl/en-BR/chrome/browser/

escolheu-se o framework Yii, versão 1.1.14, para o desenvolvimento do sistema.

Juntamente com os motivos já apresentados, esse framework possui uma documentação muito completa, contendo diversos tutoriais, no seu próprio site, de como instalar e iniciar o desenvolvimento de um novo sistema. Além disso, ele possui um fórum online, para discussão de dúvidas referentes a sua utilização, com uma comunidade grande e ativa. Todos esses fatores não só aceleram o processo de desenvolvimento do sistema proposto, como facilitarão a adição de possíveis novos recursos, visto que haverá uma padronização do código que permitirá o autor, ou outro desenvolvedor, alterá-lo mais facilmente.

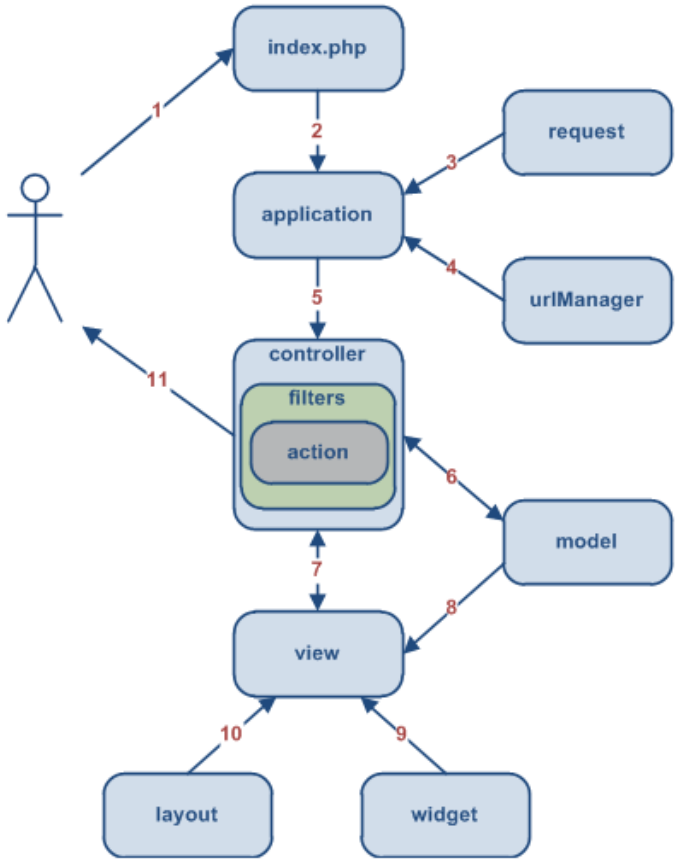


Figura 4.1: Fluxo de trabalho típico no Yii ao receber uma requisição de usuário (YII, 2014)

Um fluxo de trabalho típico dentro do framework Yii, como visto na figura 4.1, contém os seguintes passos:

1. Um usuário faz a requisição com uma URL que leve ao arquivo de inicialização do frameowrk, o index.php, localizado no servidor web.

2. O arquivo de inicialização cria uma instancia da classe Application e chama o método run.

3. A aplicação obtém detalhes da requisição do usuário através do componente chamado request.

4. Com os detalhes obtidos, e a ajuda do componente urlManager, a aplicação determina qual o arquivo *controller* e a ação que irão atender a requisição.

5. A aplicação cria uma instância do *controller* escolhido, mapeando a ação para um método na classe. São então criados e executados filtros, como controle de acesso, para verificar se a ação pode ser executada.

6. A ação solicita dados da base de dados através do *model*.

7. A ação renderiza uma *view*.

8. A *view* obtém e mostra os atributos do *model*.

9. A *view* pode executar *widgets* (componentes de apresentação)

10. A *view* renderizada é adicionada a um layout do sistema.

11. A ação é completada e o resultado é retornado para o usuário.

4.2 Google Maps

O Google Maps[4] é um serviço gratuito, desenvolvido pela empesa Google, no qual é possível fazer pesquisas e visualização de mapas e imagens de satélite, sendo disponibilizado seu acesso através da internet. Ele foi utilizado para adicionar o componente de georreferenciamento das informações do DW, sendo a principal forma de visualização das mesmas no sistema proposto. Através da API JavaScript do Google Maps v3 (2014) é possível incluir o Google Maps em uma página web, oferecendo diversos utilitários para manipulação e adição de conteúdo nos mapas através de vários serviços.

A seguir serão apresentados alguns dos utilitários utilizados na implementação do sistema.

4.2.1 Mapa Base

O primeiro passo para poder utilizar a API é carregar o arquivo JavaScript que contém todas as suas definições e símbolos. Isso é feito através da adição de uma *tag* HTML chamada script, a qual utiliza em seu atributo src a URL em que esse arquivo é disponibilizado pelo Google. Para criar um mapa, deve-se reservar um elemento HTML na página que irá receber o conteúdo gerado pela API, e então instanciar a classe google.maps.Map.

O objeto *mapOptions*, apresentado na figura 4.2, permite customizar diversos aspectos do mapa que será criado. Nessa mesma figura são definidas algumas opções como, a

[4]https://maps.google.com

```
var mapOptions = {
    center: latLng_inicial_obssan,
    zoom: zoom_inicial_obssan,
    mapTypeId: google.maps.MapTypeId.TERRAIN,
    streetViewControl: false,
    rotateControl: false,
    panControl: false
};

map = new google.maps.Map(document.getElementById("map-canvas"),mapOptions);
```

Figura 4.2: Objeto google.maps.MapOptions e instância da classe google.maps.Map

desativação de alguns dos controles que aparecem por padrão nos mapas, visto que não são necessários para a utilização que se fará do mesmo no sistema. O tipo de mapa inicial escolhido é o *TERRAIN*, por melhor apresentar as sobreposições que serão apresentadas em seguida. Esse tipo é pré-definido na API, no qual se exibe um mapa físico com base nas informações do terreno. Além disso, através da variável *zoom_inicial_obssan* é definido um valor de zoom inicial para exibição, que juntamente com a variável *latLng_inicial_obssan*, determina o posicionamento inicial do mapa. Ao instanciar a classe *google.maps.Map* são passados por parâmetro o elemento HTML, no qual será apresentado o mapa, e as opções escolhidas. Como o sistema somente utilizará dados referentes ao estado do Rio Grande do Sul, foram utilizadas coordenadas e um nível de zoom a fim de permitir um enquadramento do mesmo no elemento HTML escolhido para apresentá-los.

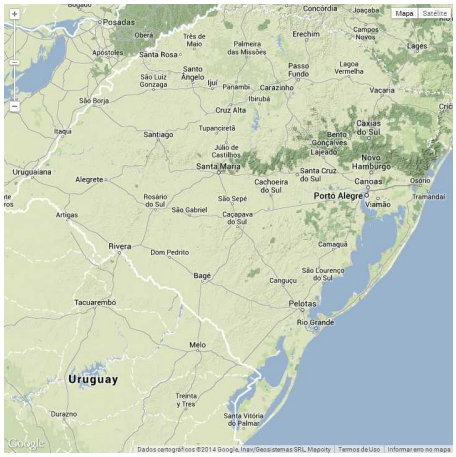


Figura 4.3: Mapa gerado a partir das opções apresentadas

4.2.2 Polígonos

A API permite a adição de sobreposições no mapa, ou seja, objetos ligados a coordenadas de latitude e longitude, que podem designar pontos, linhas, áreas ou coleções de objetos. Como as informações do DW se referem à municípios, ou às agregações deles, as mesorregiões e microrregiões, decidiu-se utilizar a sobreposição de polígonos para de-

marcar no mapa as suas localizações. Além disso, é de interesse do OBSSAN que seja demarcada a área a qual a informação se refere, algo que a os polígonos conseguem fazer.

Os polígonos consistem de uma sequencia ordenada de pontos no mapa, tendo sua posição definida por coordenadas geográficas. É feita então a união de cada ponto com seu sucessor, excetuando-se o último, o qual é unido com o primeiro, formando assim um ciclo que delimita e preenche uma região. É possível customizar a sua aparência, podendo-se alterar a cor e a opacidade de seu contorno, assim como de sua área área interna, ou de preenchimento.

```
var coordenadas = [
    new google.maps.LatLng(-30.5466388,-53.6294563),
    new google.maps.LatLng(-30.5466388,-54.6294563),
    new google.maps.LatLng(-29.5466388,-54.6294563),
    new google.maps.LatLng(-29.5466388,-53.6294563)
];

var poligonoExemplo = new google.maps.Polygon({
      paths: coordenadas,
      strokeColor: '000000',
      strokeOpacity: 0.6,
      strokeWeight: 1,
      fillColor: 'FFF945',
      fillOpacity: 0.6,
      id: 'id_ex'
    });

poligonoExemplo.setMap(map);
```

Figura 4.4: Adição de um polígono amarelo com quatro pontos no mapa

A variável *coordenadas* na figura 4.4 armazena um array com quatro instâncias da *classe google.maps.LatLng*. Esses objetos irão informar os quatro pontos que definirão o polígono. Os valores utilizados representam respectivamente a latitude e a longitude do ponto, podendo a primeira variar de -90 até 90 e a segunda de -180 até 180. Essa mesma classe foi utilizada na variável *latLng_inicial_obssan* na figura 4.2 para definir o centro do mapa.

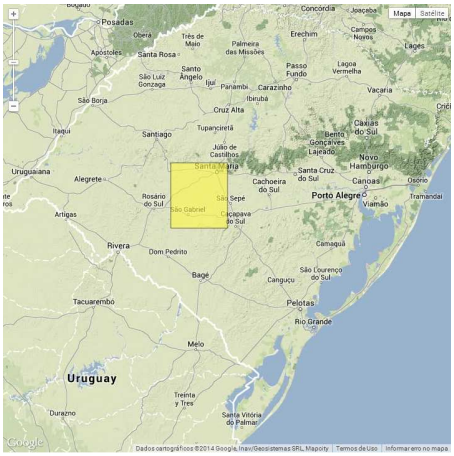


Figura 4.5: Mapa anterior com a adição do polígono

O *poligonoExemplo* é uma instância da classe *google.maps.Polygon* e armazena todas as configurações para criação do polígono. Em *paths* são definidos seus pontos, em *fillColor* sua cor de preenchimento, e em *id* um identificador único, que é utilizado para diferenciar os polígonos entre si, principalmente no momento em que são disparados eventos.

O código na linha 18 é uma chamada do método *setMap*, o qual efetua a renderização do polígono instanciado no mapa referenciado pela variável *map*.

4.2.3 Eventos

O JavaScript é uma linguagem, que quando utilizada no navegador web, geralmente só é acionada através de eventos. Esses eventos estão ligados a elementos HTML e incluem, dentre outros, a ação de apertar ou soltar o botão do mouse no elemento, a ação de colocar ou retirar o mouse de cima do elemento, a ação de apertar ou soltar uma tecla do teclado em um campo de escrita de texto. A API se aproveitou desse mesmo conceito e implementou a possibilidade de se disparar eventos em elementos do mapa, que incluem os polígonos citados acima. Entretanto, para que esses eventos possam ser utilizados, é necessário que se utilize funções chamadas de *listeners*. Essas funções aguardam que eventos sejam disparados para só então serem executadas, permitindo uma maior interação com o usuário.

```
1 google.maps.event.addListener(poligonoExemplo, 'mouseover', function(event){
2   this.setOptions({
3     fillOpacity:0.15,
4     strokeWeight: 2
5   });
6
7   abreToolTipMapsObssan('Evento Mouseover!',event.Ta.pageX,event.Ta.pageY);
8 });
```

Figura 4.6: Adição de uma função *listener* para o evento *mouseover*

Na figura 4.6 é utilizado um método da classe *google.maps.MapsEventListener* para criar uma função *listener*. Essa função está vinculada ao objeto *poligonoExemplo*, e quando o evento *mouseover* for disparado, ou seja, o cursor do mouse estiver em cima do objeto, a mesma será executada.

Nesse caso, a função realiza duas operações. Na primeira são modificados atributos de exibição do objeto do tipo polígono, afim de diferenciar o mesmo com relação a outros polígonos e chamar a atenção do usuário. Já a segunda, faz a execução de uma função criada para gerar um *tooltip* personalizado do sistema. No primeiro parâmetro da função *abreToolTipMapsObssan* é informado o texto que será exibido pela mesma, "Evento Mouseover!". Já os dois últimos parâmetros são as coordenadas X e Y na tela do navegador no momento que o cursor passou por cima do objeto. Essa função de *tooltip* cria um elemento HTML div flutuante na interface, e insere dentro dele o texto passado

por parâmetro. Após isso, os atributos de CSS *TOP* e *LEFT* do elemento, que definem o seu posicionamento na tela do navegador, são alterados de forma a que o *tooltip* fique deslocado um pouco à direita e abaixo do cursor do mouse no momento em que foi disparado o evento. Caso o elemento div já tenha sido criado, apenas os atributos CSS de posicionamento são modificados para deslocar a *tooltip* até a nova posição do cursor.

Apesar de já existir na API uma funcionalidade parecida com a da *tooltip* criada, a *InfoWindow*, optou-se por utilizar uma função personalizada. Isso se deve a um comportamento que a *InfoWindow* apresentou, o qual através de modificações na maneira de a utilizar, não se conseguiu solucionar. Quando o cursor do mouse passava por um objeto disparando o evento *mouseover*, o *tooltip* ao invés de apenas trocar de posição, produzia um efeito de piscar, como se estivesse desaparecendo e aparecendo novamente nas novas coordenadas. Por esse comportamento ser muito inconveniente para a visualização das informações do tooltip pelo usuário, não se optou por utilizar função da API.

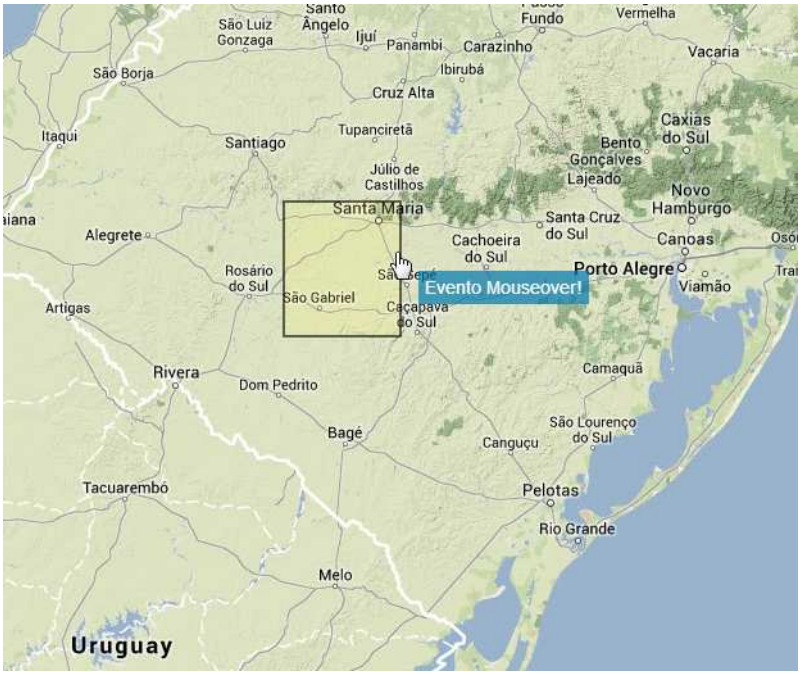


Figura 4.7: Mapa com o evento mouseover tratado pela função listener

4.3 Limites Municipais

O Google Maps permite a busca de cidades individuais por nome, mostrando seus contornos com uma linha pontilhada até um certo nível de zoom, após o qual se transforma em um marcador. Contudo, não é possível demarcar mais de um município ou alterar a cor de preenchimento ou do delimitador do mesmo. Sendo assim, a sua utilização para a visualização dos dados do sistema, que podem conter todos os municípios do estado do Rio Grande do Sul, somente é viável se for possível demarcar cada um dos municípios no mapa. Como visto na subseção 4.2.2, a API do Google Maps permite a demarcação de re-

giões com configurações de exibição, entretanto é necessário que se tenha as coordenadas geográficas de cada um dos pontos que formam seu contorno.

O IBGE, através do seu setor de geociências, disponibiliza em seu site[5] diversos mapeamentos por coordenadas geográficas do Brasil, e dentre eles, o dos municípios do estado do Rio Grande do Sul. Foi utilizado o mapeamento que consta no site como malha digital dos municípios no ano de 2013, contendo a divisão territorial do mesmo ano. Os arquivos disponibilizados pelo IBGE estão no formato shapefile. Esse formato é utilizado no armazenamento de dados geoespaciais em forma de vetor para utilização em um *Geographic Information System* (GIS), em português sistema de informação geográfica. A informação contida nesse formato, na verdade está espalhada em mais de um arquivo, contendo neste caso as seguintes extensões:

- .shp: a geometria da informação geográfica.
- .shx: índice da geometria a fim de permitir buscas mais rápidas.
- .dbf: atributos da geometria.
- .prj: sistema de coordenadas e projeção utilizada.

Como algumas dessas extensões estão em formato não textual, além da informação das coordenadas, e a quem elas pertencem, estarem em arquivos separados, optou-se por utilizar algum software que pudesse exportá-las em um formato textual de manipulação mais simples. Para isso o QGIS (2014) versão 1.8.0-Lisboa foi utilizado. O QGIS é um software gratuito de código aberto que permite a manipulação de dados georreferenciados, como mapas, utilizando camadas e sistemas de projeção diversos. Umas das suas funcionalidades é a exportação de camadas no formato *Keyhole Markup Language* (KML). Esse formato é uma extensão do *Extensible Markup Language* (XML), criado para exibir dados geográficos em um navegador da Terra como Google Earth e Google Maps, no qual as informações são armazenadas em formato textual.

Com o arquivo KML criado, foi possível desenvolver um script em PHP para extrair as coordenadas dos pontos, juntamente com o código IBGE do município e seu nome, e assim inseri-los na base de dados modelada para o sistema. Contudo, durante testes da aplicação, notou-se que a cada busca executada havia uma grande quantidade de dados sendo trazida para o navegador do usuário, graças ao grande número de coordenadas para gerar o contorno de cada município. Isso acabava causando um aumento indesejável no tempo de resposta, além de desperdício de recursos. Para solucionar essa questão, os dados das coordenadas passaram a ficar armazenados em um arquivo JavaScript da aplicação. Nesse arquivo eles estão colocados em uma variável global no formato JavaScript Object Notation (JSON), tendo sido exportados da base de dados e formatados como

[5] `http://downloads.ibge.gov.br/downloads_geociencias.htm`

```
<?xml version="1.0" encoding="utf-8" ?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <Folder>
      <name>municipios</name>
    <Schema name="municipios" id="municipios">
      <SimpleField name="Name" type="string"></SimpleField>
      <SimpleField name="Description" type="string"></SimpleField>
      <SimpleField name="ID" type="float"></SimpleField>
      <SimpleField name="CD_GEOCODM" type="string"></SimpleField>
      <SimpleField name="NM_MUNICIP" type="string"></SimpleField>
    </Schema>
    <Placemark>
      <Style>
        <LineStyle>
          <color>ff0000ff</color>
        </LineStyle>
        <PolyStyle>
          <fill>0</fill>
        </PolyStyle>
      </Style>
      <ExtendedData>
        <SchemaData schemaUrl="#municipios">
          <SimpleData name="ID">440</SimpleData>
          <SimpleData name="CD_GEOCODM">4300034</SimpleData>
          <SimpleData name="NM_MUNICIP">ACEGUÁ </SimpleData>
        </SchemaData>
      </ExtendedData>
      <Polygon>
        <outerBoundaryIs>
          <LinearRing>
            <coordinates>-54.327291703677787,-31.581645989561796
    -54.328356863880501,-31.581057942223595 ...</coordinates>
          </LinearRing>
        </outerBoundaryIs>
      </Polygon>
    </Placemark>
    </Folder>
  </Document>
</kml>
```

Figura 4.8: Exemplo de exportação dos arquivos do IBGE para KML do município de Aceguá através do QGIS

JSON através de outro script PHP desenvolvido. Por o arquivo JavaScript ser carregado pelo navegador quando o usuário acessa o sistema, no momento em que for feita a busca, as coordenadas já estarão disponíveis para que os polígonos possam ser gerados e não precisarão ser transferidas do servidor junto com as informações do resultado da busca, ocorrendo assim em uma redução do tempo de resposta inicial que se tinha.

4.4 jQuery

O JavaScript foi utilizado, não só na questão de geração dos mapas, como também para fazer requisições assíncronas ao servidor e criar, ou modificar, elementos da interface no navegador. A biblioteca JavaScript jQuery (2014) versão 1.9.1 foi utilizada para esse fim, pois além de ser gratuita e de código aberto, ela facilita a manipulação de elementos HTML, tratamento de eventos, criação de animações e requisições assíncronas, através de uma API que funciona na maior parte dos navegadores atuais, podendo ser es-

tendida. Outro fator importante para sua utilização, é que o framework Yii já possui uma integração nativa com a biblioteca, podendo utilizá-la até de maneira transparente para o desenvolvedor em alguns casos.

```
1  $.ajax({
2    type: "POST",
3    url: "http://www.sistema_exemplo.com.br/arquivo_exemplo.php",
4    data: "nome_municipio=Cotiporã",
5    success: function(resposta_servidor){
6      alert("Código IBGE do município: " + resposta_servidor);
7    },
8    error: function(jqXHR, textStatus, errorThrown){
9      alert("Ocorreu um erro!");
10   }
11 });
```

Figura 4.9: Requisição AJAX utilizando jQuery

As requisições assíncronas, comumente conhecidas como AJAX (*Asynchronous JavaScript and XML*), permitem que o navegador envie ou receba dados do servidor sem que a exibição da página seja alterada, ao contrário do comportamento padrão ao postar um formulário, por exemplo. Na figura 4.9 é apresentado um exemplo de requisição AJAX utilizando jQuery, na qual é enviado o nome de um município e devolvido o código IBGE do mesmo através de um script PHP que trata a requisição.

4.4.1 jQueryUi

A jQuery User Interface (jQuery UI) é uma coleção de componentes visuais, interações, efeitos e temas, gratuita e de código aberto, desenvolvida com a biblioteca jQuery. A versão da jQuery UI (2014) utilizada no sistema foi a 1.10.4. Uma das vantagens de utilizar a jQuery UI é que além de aproveitar todos os benefícios da jQuery, ela ainda fornece temas visuais pré-definidos e a possibilidade de modificar diversos aspectos desses temas e exportá-los para uso através do seu site[6], facilitando assim a utilização de elementos que melhoram a usabilidade do sistema. Através desses temas exportáveis, e da utilização dos componentes visuais, é facilitada a padronização da interface do sistema.

Figura 4.10: Exemplo de uso do *Accordion*

[6]http://jqueryui.com/themeroller/

O *Accordion* foi um dos componentes visuais utilizados. A partir de elementos HTML organizados em uma certa hierarquia, ele os modifica para que apresentem um comportamento de caixas de texto expansíveis com título, facilitando assim o agrupamento de informações em espaços reduzidos, como o apresentado na figura 4.10.

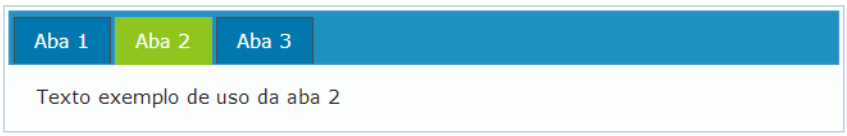


Figura 4.11: Exemplo de uso *Tabs*

O componente *Tabs* possui um funcionamento semelhante ao Accordion, modificando elementos HTML para formarem um tipo diferente de elemento na interface do sistema. Entretanto, ao invés de caixas expansíveis, ele utiliza o conceito de abas, podendo exibir um conteúdo que já está na página, ou ainda fazendo uso de uma requisição AJAX para trazê-lo, conforme a figura 4.11. Ele foi utilizado no intuito de auxiliar a organização dos formulários de busca do sistema, caso contrário, ocupariam um grande espaço na interface.

Figura 4.12: Botão normal ao lado de um botão modificado pelo *Button*

O componente *Button* atua em elementos HTML de formulários, como *inputs*, *checkboxes*, *radio buttons* e *buttons*, adicionado a eles uma aparência diferenciada. Os elementos modificados pelo componente passam a ter a aparência de botões, tendo adicionados estilos para quando estiverem ativos, com o mouse por cima, ou até ícones, vide figura 4.12. A vantagem de utilizá-lo é que como citados anteriormente, facilitam uma padronização do sistema, visto que aproveitam o mesmo tema visual já estabelecido nos demais elementos.

4.4.2 Chosen

Chosen é um *plug-in* desenvolvido utilizando jQuery, criado com o objetivo de melhorar a interface dos elementos HTML do tipo *select* e adicionar algumas funcionalidades extras. A versão do Chosen (2014) utilizado no sistema foi a 1.0.0. Com a sua utilização, os *selects* passam a ter um campo de busca embutido, permitindo digitar o que deseja-se selecionar enquanto as opções são filtradas. Outra funcionalidade interessante é a modificação da exibição de opções selecionadas, e da maneira de selecionar as mesmas em *selects* de múltipla seleção. O padrão de exibição desse tipo de campo múltiplo é uma lista com todas as opções sendo selecionadas através de clique no mouse juntamente com a tecla Shift do teclado pressionada. Já com a utilização do Chosen, o campo passa a ter o comportamento de um *select* normal, tendo suas opções selecionáveis na lista através de

um simples clique no mouse, e exibindo as opções selecionadas lado a lado. As diferenças entre os *selects* normais e os com o Chosen ativados podem ser vistos na figura 4.13. O objetivo do uso desse *plug-in* foi melhorar a usabilidade do sistema, principalmente em campos com grande número de opções e de seleção múltipla.

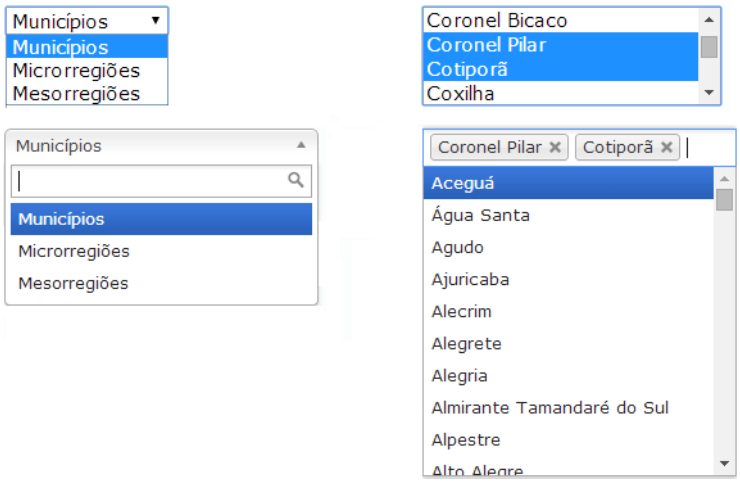


Figura 4.13: *Selects* normais acima dos selects com Chosen

4.4.3 jQuery Price Format

O jQuery Price Format (2014) é um *plug-in* desenvolvido com jQuery que permite a formatação do conteúdo de elementos HTML *input* em preços, ou seja, ao se digitar algo como '123456', ele poderá converter esse texto para 'R$ 1.234,56' no próprio campo. A versão utilizada no desenvolvimento do sistema foi a 2.0.

```
1 $('#exemplo').priceFormat({
2     prefix: '',
3     centsSeparator: '',
4     thousandsSeparator: '.',
5     centsLimit: 0,
6     limit: 10,
7     clearOnEmpty: true,
8     });
```

Figura 4.14: Aplicação do plug-in no elemento HTML com o identificador 'exemplo'

Sua utilização no sistema se deu nos campos de limitação de valor da busca graças a possibilidade de se customizar o formato no qual ele transformará o texto digitado. Como não existam campos com valores em moeda no sistema, e nem decimais, essas opções foram desativadas através das linhas 2, 3 e 5 da figura 4.14. Já as demais linhas configuram o separador de milhar como ponto, o tamanho máximo do número como 10 dígitos, e que ao ter o texto como o dígito zero, o texto será apagado.

5 GUIA DE USO

Neste capítulo é apresentada a interface da página web do sistema com a qual o usuário irá interagir para obter as informações que necessite a respeito dos dados de produção alimentar. Já existia uma página web do projeto OBSSAN [1] com seu *layout* definido, que inclui topo, rodapé e imagem de fundo. Logo, quando o sistema foi desenvolvido, somente o conteúdo da página foi criado para se adaptar ao *layout*. A interface do sistema consiste de dois elementos principais, os filtros de busca e o mapa com as informações do resultado da busca.

5.1 Busca

O processo de busca dos dados no sistema foi estabelecido de maneira semelhante ao processo de compra em um site com *E-commerce*, em português comércio eletrônico. Em geral, esses sites utilizam uma série de passos, em que o usuário vai completando informações, até que seja possível efetuar o pagamento e concluir o processo. No sistema proposto, os filtros que serão aplicados na busca foram separados em grupos, e para poder acessar o próximo grupo é necessário ter escolhido ao menos uma opção nos filtros do grupo atual. Somente ao chegar no último grupo é que pode-se efetivar a busca. Esse modelo de processo foi adotado a pedido do OBSSAN, pois essa noção de fluxo facilita o entendimento do sistema pelos usuários. As opções exibidas nos campos de seleção da busca são geradas dinamicamente, ou seja, é feita uma busca no DW para verificar quais dados estão armazenados, e a partir disso as mesmas são criadas.

Abaixo serão apresentados cada um dos três grupos que compõem os filtros de busca do sistema, Dimensões, Filtros de Dados e Filtros Adicionais.

5.1.1 Aba Dimensões

Esta aba foi definida afim de possibilitar a escolha e identificação das dimensões do PLANSAN 2012/2015 pelos usuários. O PLANSAN possui sete dimensões definidas, contudo no sistema proposto somente a primeira dimensão, Produção de Alimentos, apa-

[1] http://www.ufrgs.br/obssan/

rece como opção. Isso se deve ao fato de que somente essa dimensão foi modelada, visto que era a única que já possuía os dados que a compõem definidos e colhidos no momento da modelagem do sistema. As demais dimensões ainda precisarão ser definidas antes que se possa projetar um sistema semelhante para visualizá-las. Por ser a primeira aba, somente possui um botão para avançar à próxima.

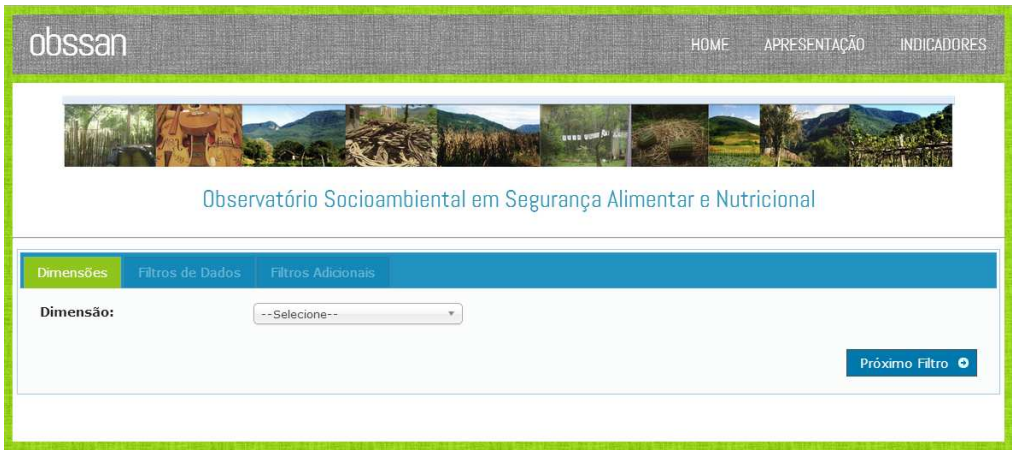


Figura 5.1: Interface do sistema no estado inicial na aba Dimensões

5.1.2 Aba Filtros de Dados

Nesta segunda aba, conforme a figura 5.2, foram colocados os filtros do assunto da busca, tendo eles organizados em quatro grupos de interesse: Produção Animal, Produção Vegetal, Estabelecimentos Agropecuários e Trabalhadores Rurais. Em cada um desses grupos existe ao menos um campo de seleção múltipla, no qual estão definidor os tipos de dados presentes no mesmo. Nos tipos do campo de Criação Animal poderiam ser encontrados, por exemplo, aves, suínos e bovinos.

Esses campos são de seleção múltipla, pois o usuário tem a possibilidade de pesquisar mais de um tipo dentro do mesmo campo, ou seja, é possível ver resultados de Aves e Bovinos na mesma pesquisa. Esse resultado é exibido no mapa como a soma dos tipos selecionados em cada município, logo é necessário que os tipos apresentem a mesma unidade de medida, caso contrário o resultado ficaria inconsistente. Para evitar que isso ocorra, toda vez que um tipo é selecionado, ocorre uma requisição AJAX para atualizar as opções disponíveis no campo, removendo assim todas as opções que tenham unidade de medida diferente da escolhida. Somente um dos campos de tipo pode ser utilizado por busca, por isso no momento em que um tipo é selecionado, os demais campos são desabilitados. Por ser uma aba que possui antecessora e predecessora, utilizou-se dois botões no fim da página, um para avançar aos próximos filtros e outro para retornar ao anterior.

Cada um dos campos de seleção irá gerar uma restrição para uma das tabelas dimensão de tipo apresentadas em 3.3.2, exceto o campo Com Laço Familiar, que pelos motivos já

citados irá restringir a própria tabela de fato. Esta restrição será aplicada na coluna nome das tabelas dimensão de tipo.

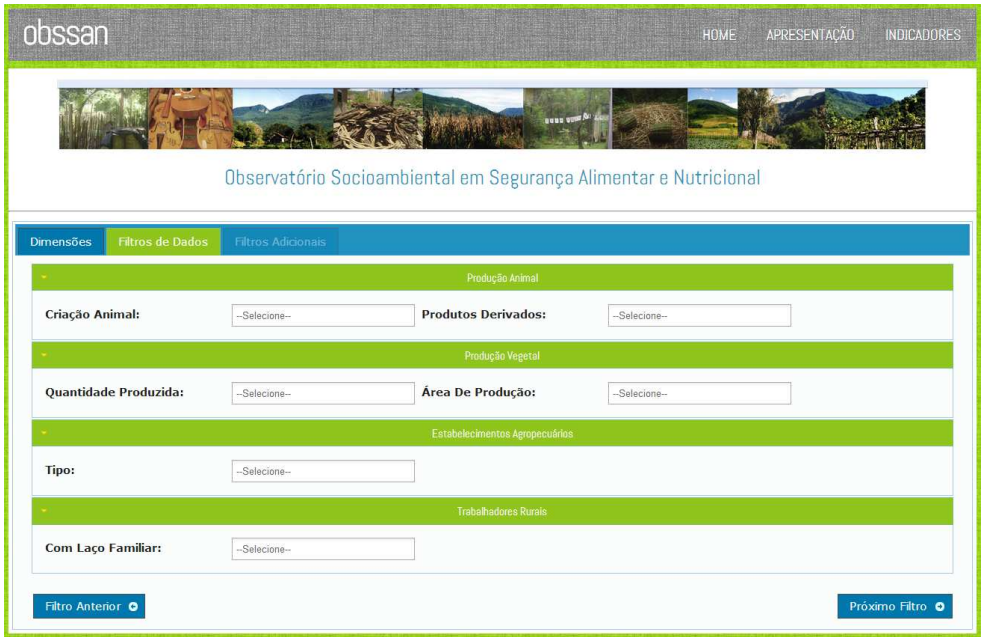


Figura 5.2: Interface da aba Filtros de Dados

5.1.3 Aba Filtros Adicionais

A terceira aba possui os filtros de restrição que serão aplicados aos tipos de dados escolhidos na aba anterior. Ao contrário das abas anteriores, todos os campos desta aba são opcionais, tendo como sua principal função auxiliar na construção de buscas mais específicas, ou seja, caso não sejam alterados, não modificarão o retorno da mesma.

Estes filtros também foram organizados em três grupos, sendo o primeiro deles o Gerais. Nele pode ser restringido se o tipo de dado escolhido na aba anterior pratica ou não agricultura familiar e sobre qual ano a busca se refere.

Já o segundo grupo, o Territoriais, apresenta todos os filtros referentes a geolocalização dos dados. Através desses filtros é possível escolher qual será a agregação utilizada para apresentar os dados, podendo-se escolher entre municípios, microrregiões e mesorregiões. Ao se modificar esse filtro serão alteradas além das informações numéricas, a exibição do mapa. Quando mesorregião ou microrregião são selecionados, todos os municípios que os compõem apresentarão a mesma cor de contorno e de preenchimento, fazendo assim com que possa identificá-los facilmente. Além disso, pode-se especificar nos três níveis territoriais, através de campos de seleção múltipla, quais dos seus integrantes serão exibidos no mapa.

O grupo Numéricos apresenta dois campos de texto para a utilização de números. Através destes campos pode-se filtrar a faixa de valores numéricos no mapa, ou seja, somente serão exibidos os valores que se encontrarem dentro da faixa definida pelo usuário.

Por ser a aba que finaliza o processo de busca, nela estão presentes os botões de voltar para a aba anterior e o botão que posta a busca para o servidor web através de uma requisição AJAX.

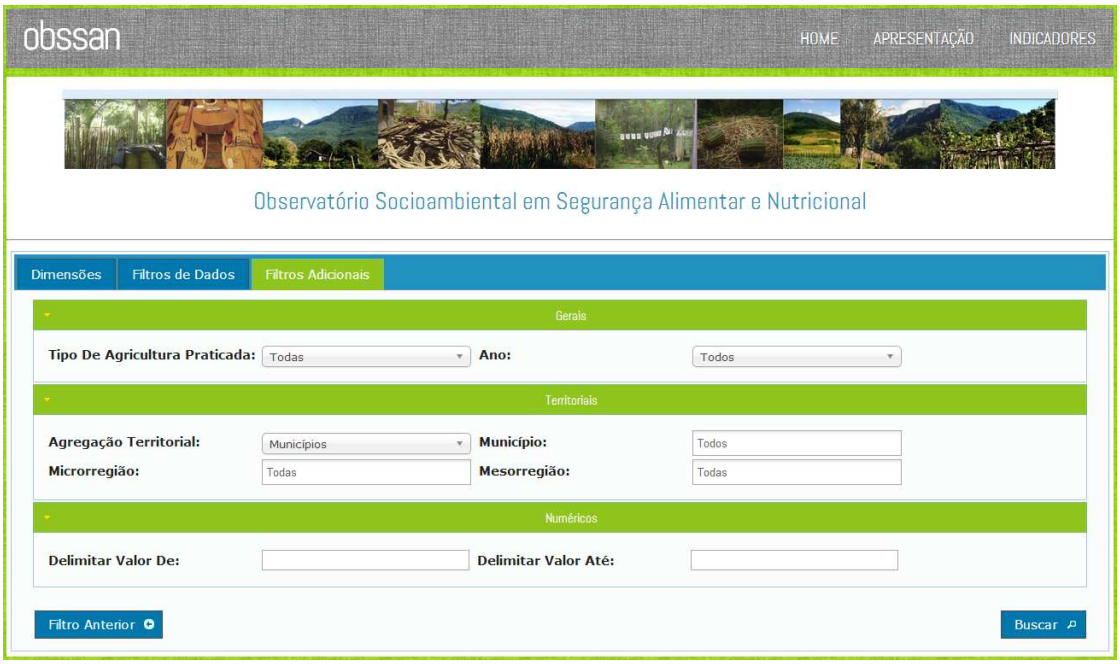


Figura 5.3: Interface da aba Filtros Adicionais

5.2 Resultado da Busca

Após o encerramento do processo de busca, é então apresentado o resultado da mesma para o usuário. O resultado consiste de dois elementos na interface, o mapa com legenda e uma coluna com funcionalidades e informações complementares as do mapa.

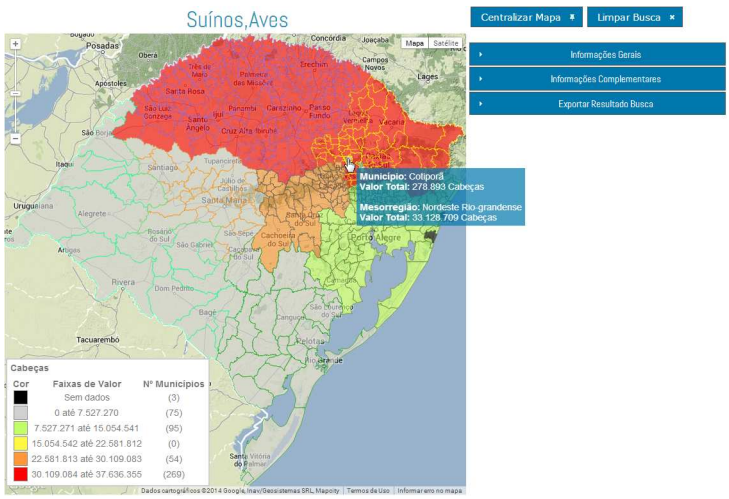


Figura 5.4: Resultado de uma busca de criação animal de aves e suínos por mesorregião

5.2.1 Mapa

O mapa é o elemento principal do resultado da busca, tendo no seu topo a listagem de todos os tipos escolhidos, como apresentado em 5.1.2. Nele é possível modificar quais informações estarão visíveis através da alteração do nível de zoom, e da possibilidade de deslocar o mapa em todas as direções. Ao passar o cursor do mouse por cima dos polígonos que representam os municípios, é apresentado um tooltip com o nome do mesmo juntamente com o valor e sua unidade retornados para os tipos pesquisados. Caso a pesquisa tenha utilizado um nível de agregação territorial diferente de município, serão apresentados também o seu nome e o valor com unidade da agregação.

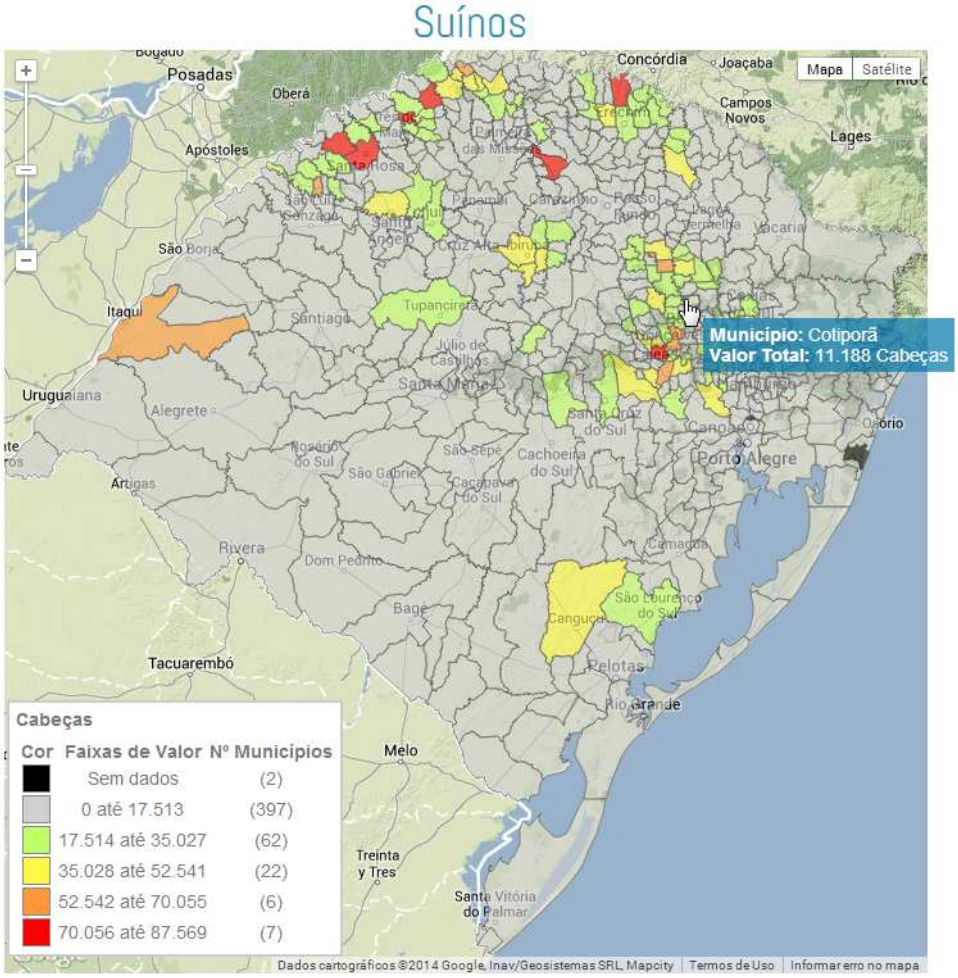


Figura 5.5: Mapa retornado para uma consulta de criação animal de suínos por município

Visando facilitar a visualização dos valores retornados nas unidades territoriais, diferentes cores de preenchimento nos polígonos foram utilizadas. A cor que será aplicada é determinada conforme uma faixa de valores, a qual é dividia em cinco intervalos. Além dessas cinco faixas, ainda existe uma sexta que representa a ausência de valor. A faixa de valores é estabelecida dinamicamente, ou seja, a cada busca é verificado o maior e menor valor retornado, os quais serão o seu inicio e fim. A partir disso a faixa é divida nos cinco intervalos de tamanho semelhante, através das quais a unidade territorial terá

seu valor classificado. Essas informações de intervalos de valores, juntamente com as cores de preenchimento, o número de municípios em cada faixa e a unidade dos valores retornados, ficam em uma legenda criada como um retângulo na parte inferior esquerda do mapa, utilizando-se elementos HTML.

5.2.2 Informações Complementares e Funcionalidades

O segundo elemento do resultado da busca é uma coluna localizada no lado direito do mapa, que possui dois botões e um *Accordion*, do tipo apresentado em 4.4.1, conforme figura 5.6. O botão Centralizar Mapa retorna o mesmo para a condição inicial de zoom e de deslocamento apresentada pelo sistema ao retornar uma busca, facilitando assim o reenquadramento do mesmo sem precisar efetuar uma nova consulta. Já o botão Limpar Busca tem como função apagar a parte da interface referente ao retorno da mesma, alterar todos os filtros para seu valor inicial e bloquear todas as abas, exceto a primeira. Assim o sistema retorna ao seu estado inicial, contribuindo para que novas pesquisas possam ser feitas de maneira mais ágil e com menos infomações na tela.

Figura 5.6: Coluna à direta do mapa

O *Accordion* possui três itens, vide figura 5.7. No primeiro item do Accordion, o Informações Gerais, são apresentadas informações relativas a busca como um todo, incluindo o número total de municípios retornados, a média de valores dos municípios, o valor do somatório dos municípios e a quais anos se referem os dados.

O item Informações Complementares apresenta algumas informações a mais do que as informadas no tooltip, que é exibido ao se passar o mouse por cima do polígono um município, visto que o mesmo se tornaria muito grande e de difícil leitura. Inicialmente não existe informações nesse item, somente sendo exibidas ao se clicar com o mouse sobre o polígono de um município. Nessas informações, sempre são apresentados o nome do município e o valor total dos dados buscado no mesmo, e caso a agregação territorial não seja a de municípios, são apresentadas também os mesmos dados a respeito da agregação escolhida. Em seguida é exibida uma lista com os tipos buscados, tendo em cada tipo informações sobre sua fonte, valor no município, porcentagem do total do tipo. Juntamente com essas informações, também pode ser exibido, quando pertinente, o nome da agregação e informação a respeito de agricultura familiar.

O terceiro e último item, Exportar Resultado Busca, gera uma tabela no formato

comma-separated values (CSV), a partir do resultado da busca. Essa tabela é disponibilizada ao usuário através de um arquivo, o qual pode ter seu *download* feito por um *link* que é gerado abaixo do ícone representando o CSV, assim que o processo de exportação se encerra. O formato CSV foi utilizado, pois pode ser lido e apresentado por uma grande quantidade de softwares existentes. Na tabela são apresentadas basicamente as unidades territoriais com os valores dos tipos buscados, juntamente com as informações de fonte dos dados, ano que representam e de agricultura familiar.

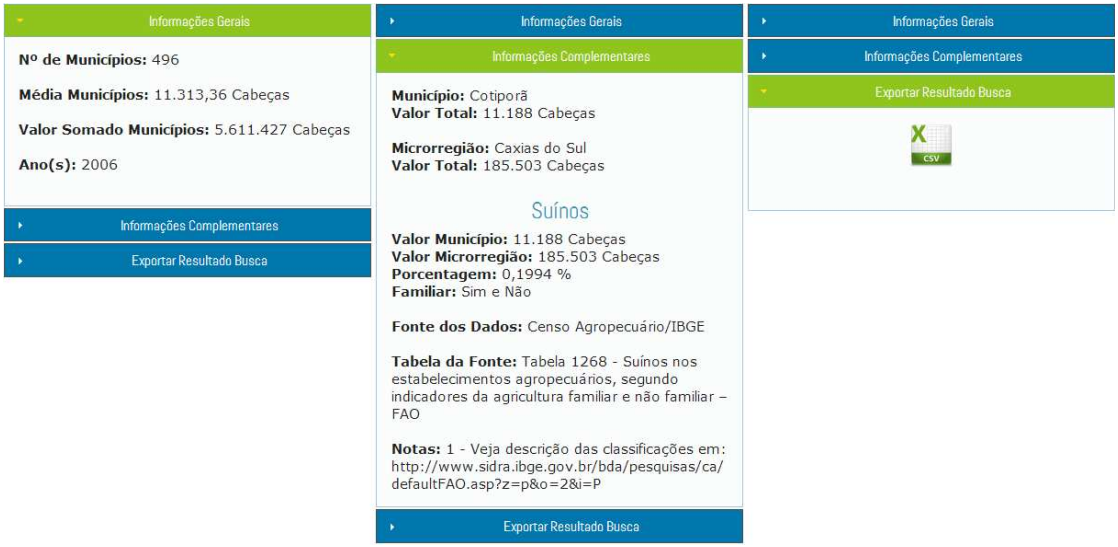


Figura 5.7: Informações complementares de uma busca de criação animal de suínos por microrregião

5.3 Questionário

Com a finalização da versão final do sistema proposto neste trabalho, pensou-se em utilizar um questionário para avaliar a sua usabilidade, e se os requisitos do sistema haviam sido alcançados. Para tanto, criou-se um formulário através do Google Drive[2], que está incluído no Apêndice A. Nele foram criadas atividades referentes a cada uma das *user stories*, as quais possuíam como resposta o nível de dificuldade para sua conclusão e se não tinha sido possível completá-las. Além disso, foi disponibilizado um campo de texto livre para que os usuário pudessem fazer comentários a respeito da sua experiência.

O questionário foi respondido pelas duas integrantes do OBSSAN que se envolveram no processo de desenvolvimento do sistema através das reuniões, sendo uma aluna de graduação em nutrição e uma doutora em desenvolvimento rural. Todas as sete atividades foram completadas ao menos uma vez, o que indica que a maior parte dos requisitos foi atendida. Somente três atividades foram completadas com facilidade por todos, o que juntamente com as duas atividades que não foram completadas por ao menos um usuário,

[2]https://drive.google.com

indica que é necessário fazer alterações na interface do sistema afim de deixá-lo mais intuitivo. No campo de texto livre também foi apontada uma deficiência do sistema com relação a combinação dos filtros de busca. Quando uma busca é feita e a combinação dos filtros não localiza uma resposta, não é indicado qual dos filtros que gerou a restrição impedindo o resultado de ser apresentado.

6 CONCLUSÃO

Neste trabalho foi apresentado o desenvolvimento de um sistema web de visualização de dados georreferenciados com o intuito de auxiliar o OBSSAN em suas análises e na disponibilização dessas informações para o público em geral. Baseado em um processo de desenvolvimento definido e nos requisitos levantados em reuniões com integrantes do OBSSAN, foi modelada a base de dados do sistema como um Data Warehouse utilizando a modelagem dimensional. Além disso, foram utilizados um framework de desenvolvimento web e diversos componentes de software para a construção da base do sistema, visando criar e utilizar padrões que facilitarão manutenções futuras. O georreferenciamento dos dados foi feito através da utilização de um mapa criado a partir do Google Maps e sua Javascript API V3. Nesse mapa foram adicionados os contornos dos municípios do Rio Grande Sul, sempre apresentando os valores respectivos aos dados buscados pelo usuário. Com isso, foi possível completar os objetivos traçados inicialmente e ter uma versão funcional do sistema para utilização. Essa versão foi apresentada para os demais integrantes do OBSSAN e teve uma boa recepção.

Apesar de se ter definido um conjunto de requisitos através de *user stories*, e até adicionado alguns novos requisitos em reuniões a cada fim de iteração, os integrantes do OBSSAN, que participaram do projeto desde o início, tinham apenas uma ideia abstrata de como poderia ser o sistema e das maneiras com as quais poderiam interagir com o mesmo. A partir do seu, uso começaram a surgir diversas novas ideias de funcionalidades e notou-se algumas limitações. Uma das limitações é a de cruzamento de dados, ou seja, só é possível analisar dados de um grupo por vez. Outra limitação é a de não ser possível efetuar outras operações aritméticas além da soma dos tipos de um mesmo grupo, por exemplo, divisão, a qual serviria para analisar proporções. De acordo com o resultado do questionário apresentado em 5.3, também foram apontadas limitações na interface, e no resultado da busca quando não se localiza nenhum dado. A segunda limitação pode ser solucionada com a verificação dos filtros possíveis de serem aplicados a cada nova seleção, desabilitando os demais.

Conforme citado acima, alguns novos requisitos foram identificados em vários aspectos do sistema. O primeiro diz respeito a manutenção da base de dados. É necessário criar

uma interface web para que os pesquisadores possam adicionar, modificar, excluir os dados da mesma, dispensando a necessidade do auxílio de uma pessoa com conhecimento técnico para isso. Além disso, viu-se a necessidade de adicionar uma nova informação ao modelo da base de dados, a de demografia dos municípios, o que provavelmente ocasionaria na criação de uma nova tabela de fatos. Notou-se a necessidade de poder customizar a quantidade e o valor das faixas e valores da legenda no mapa, visto que em diversos trabalhos de pesquisa é necessário utilizar faixas específicas. Uma outra possibilidade de customização é a do formato do arquivo de exportação do resultado da busca, ou seja, quais seriam os valores apresentados nas linhas e nas colunas. Foi da mesma maneira identificada a necessidade de se realizar um filtro pós busca no mapa, facilitando a localização de municípios e valores sem precisar realizar uma nova pesquisa. Objetivando aprofundar o sistema na relação com PLANSAN 2012/2015, também é necessário criar uma forma de exibir claramente os valores dos indicadores da primeira dimensão, definidos no plano, os quais são compostos pelos dados de produção alimentar já disponíveis dentro do sistema. Isto não foi feito, pois no momento do desenvolvimento do sistema, ainda não existia uma definição final de quais seriam os grupos de dados que comporiam cada um deles.

REFERÊNCIAS

APACHE. Disponível em: <https://httpd.apache.org/>. Acesso em: Junho 2014.

BECK, K. et al. Principles behind the agile manifesto. **Agile Alliance**, [S.l.], 2001.

CHOSEN. Disponível em: <http://harvesthq.github.io/chosen/>. Acesso em: Junho 2014.

COHN, M. **Succeeding with agile**: software development using scrum. [S.l.]: Pearson Education, 2009.

CONSEA. **Lei de Segurança Alimentar e Nutricional**: conceitos lei nº 11.346, de 15 de setembro de 2006. [S.l.]: Conselho Nacional de Segurança Alimentar e Nutricional, 2006.

DEITEL, P.; DEITEL, H. **Deitel® developer series ajax, rich internet applications, and web development for programmers**. [S.l.]: Prentice Hall Press, 2008.

GAMMA, E. et al. Design patterns: elements of reusable object-oriented software. **Reading: Addison Wesley Publishing Company**, [S.l.], 1995.

GOOGLE MAPS. **Documentação da API Javascript do Google Maps v3**. Disponível em: <https://developers.google.com/maps/documentation/javascript/>. Acesso em: Junho 2014.

INMON, W. H. **Como Construir o Data Warehouse**. Rio de Janeiro: Campus, 1997.

JQUERY. Disponível em: <http://jquery.com/>. Acesso em: Junho 2014.

JQUERY PRICE FORMAT. Disponível em: <http://jquerypriceformat.com/>. Acesso em: Junho 2014.

JQUERY UI. Disponível em: <http://jqueryui.com/>. Acesso em: Junho 2014.

KIMBALL, R. **Data warehouse toolkit**: técnicas para construção de data warehouses dimensionais. São Paulo: Makron, 1998.

MOODY, D. L.; KORTINK, M. A. From enterprise models to dimensional models: a methodology for data warehouse and data mart design. In: DMDW. **Anais...** [S.l.: s.n.], 2000. p.5.

MYSQL. Disponível em: <http://www.mysql.com/>. Acesso em: Junho 2014.

PHP. **Manual do PHP**. Disponível em: <http://www.php.net>. Acesso em: Junho 2014.

PRESSMAN, R. S.; LOWE, D. **Web engineering**: a practitioner's approach. [S.l.]: McGraw-Hill, 2009.

QGIS. Disponível em: <http://www.qgis.org/en/site/>. Acesso em: Junho 2014.

RIVEST, S.; BÉDARD, Y.; MARCHAND, P. Toward better support for spatial decision making: defining the characteristics of spatial on-line analytical processing (solap). **GEOMATICA-OTTAWA-**, [S.l.], v.55, n.4, p.539–555, 2001.

SIQUEIRA, A. C. da et al. OBSERVATÓRIO SOCIOAMBIENTAL EM SEGURANÇA ALIMENTAR E NUTRICIONAL: análise dos indicadores de produção de alimentos em nível municipal no rio grande do sul. **XV Simpósio Internacional IHU - Alimento e Nutrição no contexto dos Objetivos do Desenvolvimento do Milênio**, [S.l.], 2014.

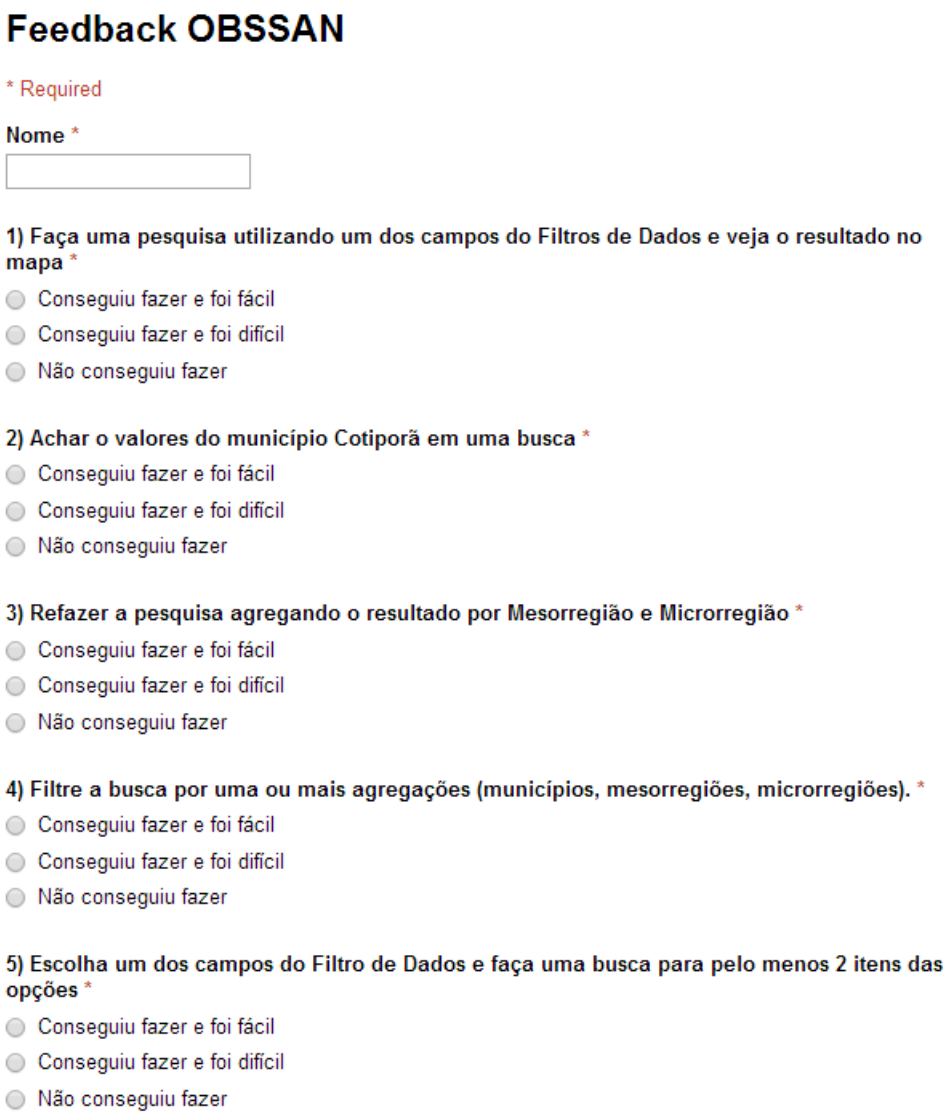
SOMMERVILLE, I. **Software Engineering.** [S.l.]: Addison Wesley, 2004.

SOMMERVILLE, I. **Software Engineering**. [S.l.]: Pearson Education, 2011.

YII. **Página do fluxo de trabalho típico no framework Yii**. Disponível em: <http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc>. Acesso em: Junho 2014.

APÊNDICE A QUESTIONÁRIO

Questionário aplicado aos integrantes do OBSSAN.



Feedback OBSSAN

* Required

Nome *

1) Faça uma pesquisa utilizando um dos campos do Filtros de Dados e veja o resultado no mapa *

- Conseguiu fazer e foi fácil
- Conseguiu fazer e foi difícil
- Não conseguiu fazer

2) Achar o valores do município Cotiporã em uma busca *

- Conseguiu fazer e foi fácil
- Conseguiu fazer e foi difícil
- Não conseguiu fazer

3) Refazer a pesquisa agregando o resultado por Mesorregião e Microrregião *

- Conseguiu fazer e foi fácil
- Conseguiu fazer e foi difícil
- Não conseguiu fazer

4) Filtre a busca por uma ou mais agregações (municípios, mesorregiões, microrregiões). *

- Conseguiu fazer e foi fácil
- Conseguiu fazer e foi difícil
- Não conseguiu fazer

5) Escolha um dos campos do Filtro de Dados e faça uma busca para pelo menos 2 itens das opções *

- Conseguiu fazer e foi fácil
- Conseguiu fazer e foi difícil
- Não conseguiu fazer

Figura A.1: Primeira parte do questionário

6) Delimite um intervalo de valores em uma busca *

- Conseguiu fazer e foi fácil
- Conseguiu fazer e foi difícil
- Não conseguiu fazer

7) Exporte o resultado de uma busca e visualize o conteúdo do arquivo *

- Conseguiu fazer e foi fácil
- Conseguiu fazer e foi difícil
- Não conseguiu fazer

Campo livre para observações

Figura A.2: Segunda parte do questionário

APÊNDICE B DICIONÁRIO DE DADOS

Tabelas de dimensão específicas de cada fato:

Tabela B.1: Tabela animais_tipo

Coluna	Tipo do Dado	Descrição	Observações
animais_tipo_id	int(11)	Identificador do animal	auto-increment
nome	varchar(45)	Nome do animal	
unidade	varchar(40)	Unidade de medida	
familiar	int(1)	Indicador de agricultura familiar	0=não, 1=sim

Tabela B.2: Tabela estabelecimentos_agropecuarios_tipo

Coluna	Tipo do Dado	Descrição	Observações
estabelecimen-tos_agropecuarios_tipo_id	int(11)	Identificador do tipo de estabelecimento agropecuário	auto-increment
nome	varchar(50)	Nome do tipo de estabelecimento agropecuário	
unidade	varchar(40)	Unidade de medida	
familiar	int(1)	Indicador de agricultura familiar	0=não, 1=sim

Tabela B.3: Tabela generos_alimentares_tipo

Coluna	Tipo do Dado	Descrição	Observações
generos_alimentares_tipo_id	int(11)	Identificador do tipo de gênero alimentar	auto-increment
nome	varchar(45)	Nome do tipo de gênero alimentar	
unidade	varchar(40)	Unidade de medida	
familiar	int(1)	Indicador de agricultura familiar	0=não, 1=sim
derivado_animal	int(1)	Indicador de derivado animal	0=não, 1=sim

Tabelas de dimensão compartilhadas entre todos os fatos:

Tabela B.4: Tabela fonte_dados

Coluna	Tipo do Dado	Descrição	Observações
fonte_dados_id	int(11)	Identificador da fonte de dados	auto-increment
nome	varchar(50)	Nome da fonte de dados	
tabela_fonte	varchar(255)	Tabela no SIDRA	
notas	text	Observações sobre a fonte de dados	

Tabela B.5: Tabela indicador

Coluna	Tipo do Dado	Descrição	Observações
indicador_id	int(11)	Identificador do indicador	auto-increment
nome	varchar(255)	Nome do indicador no PLANSAN	
nome_dimensao	varchar(255)	Nome da dimensão no PLANSAN	

Tabela B.6: Tabela localizacao_geografica

Coluna	Tipo do Dado	Descrição	Observações
localiza-cao_geografica_id	int(11)	Identificador da localização geográfica	auto-increment
codigo_uf	int(2)	Código IBGE da UF	
nome_uf	varchar(25)	Nome da UF	
codigo_municipio	int(7)	Código IBGE do município	
nome_municipio	varchar(40)	Nome do município	
codigo_mesorregiao	int(4)	Código IBGE mesorregião	
nome_mesorregiao	varchar(40)	Nome da mesorregião	
codigo_microrregiao	int(5)	Código IBGE da microrregião	
nome_microrregiao	varchar(40)	Nome da microrregião	