Rômulo de Angelis Vitoi

# SisPreço: Sistema de Preços do Incaper

Vitória, ES

2018

Rômulo de Angelis Vitoi

# SisPreço: Sistema de Preços do Incaper

Monografia apresentada ao Curso de Engenharia de Computação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Departamento de Informática

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2018

Rômulo de Angelis Vitoi
SisPreço: Sistema de Preços do Incaper/ Rômulo de Angelis Vitoi. – Vitória, ES, 2018-
53 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vítor E. Silva Souza

Monografia (PG) – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Departamento de Informática, 2018.

1. Desenvolvimento web. 2. Desenvolvimento para dispositivos móveis. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. SisPreço: Sistema de Preços do Incaper

CDU 02:141:005.7

Rômulo de Angelis Vitoi

# SisPreço: Sistema de Preços do Incaper

Monografia apresentada ao Curso de Engenharia de Computação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Trabalho aprovado. Vitória, ES, 12 de julho de 2018:

---

**Prof. Dr. Vítor E. Silva Souza**
Orientador

---

**Prof. Dr. Ricardo de Almeida Falbo**
Convidado 1

---

**César Henrique Bernabé**
Convidado 2

Vitória, ES
2018

*"I'd like to die on Mars, just not on impact."*
*(Elon Musk)*

# Resumo

Semanalmente, o Incaper (Instituto Capixaba de Pesquisa, Assistência Técnica e Extensão Rural) realiza levantamentos de preços de produtos agrícolas, de pecuária e de silvicultura em diversos municípios do estado do Espírito Santo. Além de servir como fonte de consulta para os produtores rurais e comerciantes, tais levantamentos são instrumentos balizadores de mercado, cujos preços levantados são utilizados como referência por diversos programas governamentais no estado, como o Programa de Aquisição da Agricultura Familiar (PAA) e o Programa de Garantia de Preço Mínimo (PGPM). O levantamento também é utilizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE) e Instituto Jones dos Santos Neves (IJSN) para cálculo do PIB (produto interno bruto) trimestral e anual do estado, bem como para o cálculo do PIB anual dos municípios. Atualmente estes dados de preços não estão sistematizados e não são de fácil acesso, dificultando sua manipulação. O SisPreço surge como uma solução para organizar os dados de uma maneira centralizada onde é possível manipulá-los mais facilmente.

Para a construção do sistema, foram colocadas em prática as disciplinas aprendidas ao decorrer do curso, tais como Programação, Linguagens de Programação, Banco de Dados, Engenharia de Software e Estruturas de Dados.

**Palavras-chaves**: Sistema Web, Sistema para Dispositivos Móveis, Projeto de Sistemas, Kotlin, PHP

# Lista de ilustrações

# Lista de abreviaturas e siglas

| | |
|---|---|
| AJAX | Javascript Assíncrono e XML, do inglês *Asynchronous Javascript and XML* |
| API | Interface de Programação de Aplicação, do inglês *Application Programming Interface* |
| CSS | Folhas de estilo em cascata, do inglês *Cascading Style Sheets* |
| DAO | Objeto de Acesso a Dados, do inglês *Data Access Object* |
| GPS | Sistema de Posicionamento Global, do inglês *Global Positioning System* |
| HTML | Linguagem de Marcação de Hipertexto, do inglês *HyperText Markup Language* |
| HTTP | Protocolo de Transferência de Hipertexto, do inglês *HyperText Transfer Protocol* |
| IDE | Ambiente de Desenvolvimento Integrado, do inglês *Integrated Development Environment* |
| JSON | Notação de Objetos JavaScript, do inglês *JavaScript Object Notation* |
| JVM | Máquina Virtual Java, do inglês *Java Virtual Machine* |
| MVC | Modelo-Visão-Controlador, do inglês *Model-View-Controller* |
| MVP | Modelo-Visão-Apresentação, do inglês *Model-View-Presenter* |
| NFC | Comunicação por Campo de Proximidade, do inglês *Near Field Communication* |
| ORM | Mapeamento Objeto/Relacional, do inglês *Object/Relational mapping* |
| PDF | Formato de Documento Portável, do inglês *Portable Document Format* |
| REST | Transferência de Estado Representacional, do inglês *Representational State Transfer* |
| SGBD | Sistema Gerenciador de Banco de Dados |
| UML | Linguagem de Modelagem Unificada, do inglês *Unified Modeling Language* |

| | |
|---|---|
| URI | Identificador Uniforme de Recursos, do inglês *Uniform Resource Identifier* |

# Sumário

# 1 Introdução

Semanalmente, o Incaper (Instituto Capixaba de Pesquisa, Assistência Técnica e Extensão Rural) realiza pesquisas de preços de produtos e estes dados são utilizados para a geração de relatórios. Além de servir como fonte de consulta para os produtores rurais e comerciantes, tais pesquisas são instrumentos balizadores de mercado, cujos preços levantados são utilizados como referência por diversos programas governamentais no estado, como o Programa de Aquisição da Agricultura Familiar (PAA) e o Programa de Garantia de Preço Mínimo (PGPM). O levantamento também é utilizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE) e Instituto Jones dos Santos Neves (IJSN) para cálculo do PIB (produto interno bruto) trimestral e anual do estado, bem como para o cálculo do PIB anual dos municípios.

Atualmente estes dados são inseridos em planilhas e a partir delas os relatórios são gerados. O problema desta abordagem é a manutenção e dificuldade de manipulação dessas planilhas que acabam inviabilizando a produção de relatórios mais complexos e de uma fácil visualização de dados históricos.

O SisPreço surge como uma solução para oferecer uma forma mais simples para a inserção de dados e um armazenamento unificado dos mesmos que facilita seu processamento para a geração de relatórios automatizados. Além da centralização dos dados, a criação de um sistema permite agregar várias funcionalidades, como a definição de produtos a terem seus preços levantados em um determinado município, validação das informações fornecidas pelos usuários, acompanhamento do progresso do processo de levantamentos de preços, entre outras.

O sistema, desenvolvido em conjunto com Incaper, é composto por 4 partes: a primeira uma API que acessa os dados em um banco de dados já existente no Incaper que possui dados sobre produtos, produtores e municípios; a segunda uma aplicação para dispositivos móveis Android, onde os agentes do Incaper poderão realizar a inserção de dados; a terceira uma aplicação *Web* com as mesmas funcionalidades da aplicação móvel para Android, mas contendo funcionalidades adicionais de gerenciamento; a quarta uma outra API pertencente ao SisPreço, que é a interface de comunicação das aplicações com o banco de dados do sistema.

## 1.1 Objetivos

O objetivo geral deste trabalho é desenvolver um sistema que será utilizado pelo Incaper a fim de auxiliá-los em uma tarefa rotineira de levantamento de preços de produtos.

Para isso, serão utilizados conceitos aprendidos ao longo do curso de Engenharia de Computação, em particular as disciplinas de Programação, Linguagens de Programação, Banco de Dados, Engenharia de Software e Estruturas de Dados.

São objetivos específicos deste projeto:

- Produzir o documento de especificação e análise de requisitos do software;
- Produzir o documento de projeto de software, utilizando o método FrameWeb (SOUZA, 2007), contribuindo, assim, com a avaliação deste método por meio de sua aplicação em um sistema real;
- Desenvolver o sistema de acordo com a estrutura definida no processo de Engenharia de Software, utilizando *frameworks* já existentes para auxiliar no desenvolvimento do sistema.

## 1.2 Metodologia

A metodologia utilizada no desenvolvimento deste trabalho foi composta pelas seguintes atividades:

1. *Revisão Bibliográfica:* estudo de boas práticas de Engenharia de Software e de Requisitos, Padrões de Projeto de Sistemas, Programação Orientada a Objetos, uso e projeto de Banco de Dados Relacional, Desenvolvimento *Web*, Desenvolvimento para Dispositivos Móveis, entre outros;
2. *Elaboração da Documentação do Sistema:* definição dos documentos do sistema. Em primeiro lugar, foi elaborado o Documento de Especificação de Requisitos, apresentando uma descrição geral do minimundo do sistema, definição dos requisitos funcionais e não funcionais, além das regras de negócio. Também estão neste documento os casos de uso, modelo estrutural e glossário do projeto. Por fim, foi elaborado o Documento do Projeto, contendo a arquitetura do software e projeto detalhado de cada um dos seus componentes, seguindo a abordagem FrameWeb;
3. *Estudo das Tecnologias:* estudo das tecnologias utilizadas para o desenvolvimento do sistema, tais como as linguagens de programação PHP, Javascript e Kotlin, os *frameworks* Laravel, Angular e Android, o sistema de gerenciamento de banco de dados MySQL, dentre outras;
4. *Desenvolvimento da Aplicação:* implementação das funcionalidades do sistema;

5. *Redação da Monografia:* escrita desta monografia, feita em *LaTeX*[1] e o template *abnTeX*[2] que atende os requisitos das normas da ABNT (Associação Brasileira de Normas Técnicas) para elaboração de documentos técnicos e científicos brasileiros.

## 1.3 Organização do Texto

Esta monografia é estruturada em cinco partes e contém, além da presente introdução, os seguintes capítulos:

- **Capítulo 2** – Referencial Teórico: apresenta uma revisão da literatura a respeito dos temas relevantes para o contexto deste trabalho, sendo estes: Engenharia de Software, Desenvolvimento *Web* e Desenvolvimento para Dispositivos Móveis;
- **Capítulo 3** – Especificação de Requisitos: apresenta a especificação de requisitos do sistema, descrição do minimundo, diagramas de classes e seus casos de uso;
- **Capítulo 4** – Projeto e Implementação: apresenta a arquitetura utilizada e as principais partes da implementação do sistema, ilustradas por capturas de telas;
- **Capítulo 5** – Considerações Finais: apresenta a conclusão do trabalho, dificuldades encontradas, limitações e propostas de trabalhos futuros;
- **Apêndices** – Apresenta os documentos produzidos nos processos de Engenharia de Software (Documento de Requisitos e Documento de Projeto).

[1] <http://www.latex-project.org/>.
[2] <http://www.abntex.net.br>.

# 2 Referencial Teórico

Neste capítulo são apresentados os principais conceitos teóricos que foram utilizados no desenvolvimento do sistema SisPreço. A Seção 2.1 aborda a Engenharia de *Software*, demonstrando os conceitos e processos utilizados. A Seção 2.2 apresenta os principais conceitos de desenvolvimento *Web*. A Seção 2.3 apresenta alguns conceitos de desenvolvimento para dispositivos móveis e faz um comparativo com o desenvolvimento *Web*.

## 2.1 Engenharia de *Software*

O desenvolvimento de *software* é uma atividade que vem ganhando a cada dia mais importância na sociedade. O crescimento do uso de computadores pessoais e dispositivos móveis nas diversas áreas de conhecimento humano gerou uma crescente demanda por soluções que facilitam ou automatizam processos. Visando melhorar a qualidade dos produtos de *software* e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de *Software* (FALBO, 2014).

A Egenharia de *Software* é uma área da Ciência da Computação voltada ao estudo dos processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de *software*, apoiando-se principalmente nas práticas e aplicações da área de Gerência de Projetos com o objetivo de promover melhor organização, produtividade e qualidade em todo o processo de desenvolvimento de um *software* (FALBO, 2014).

Como atividades de um processo de desenvolvimento de *software*, podemos citar a etapa de especificação e análise de requisitos e a etapa de projeto e implementação.

Nas próximas seções, serão levantados alguns pontos referentes a este processo de desenvolvimento de *software*. Na Seção 2.1.1, será abordada a etapa de especificação e análise de requisitos. Na Seção 2.1.2, a etapa retratada será a de projeto e implementação. Em ambas, serão levantados alguns conceitos importantes para a uma melhor compreensão de sua importância para o processo de desenvolvimento de *software* como um todo.

### 2.1.1 Engenharia de Requisitos

Requisitos têm um papel central no desenvolvimento de *software*, uma vez que uma das principais medidas do sucesso de um *software* é o grau no qual ele atende aos objetivos e requisitos para os quais foi construído. Requisitos são a base para estimativas, modelagem, projeto, implementação, testes e até mesmo para a manutenção. Portanto, estão presentes ao longo de todo o ciclo de vida de um *software* (FALBO, 2017).

Neste projeto foram utilizadas as técnicas de levantamento e especificação de requisitos aprendidas ao longo do curso, como descrição de minimundo, levantamento de requisitos funcionais e não funcionais, modelagem de casos de uso e modelagem conceitual estrutural. Nos parágrafos a seguir, estas técnicas são descritas brevemente.

Os requisitos de um sistema de *software* incluem descrições das funções que o sistema deve prover e das restrições que devem ser satisfeitas. Em outras palavras, os requisitos definem o que o sistema deve fazer e as circunstâncias sob as quais deve operar (SOMMERVILLE, 2010).

Requisitos são, normalmente, classificados em requisitos funcionais e não funcionais. Requisitos funcionais apontam as funções que o sistema deve prover. Já os requisitos não funcionais descrevem restrições sobre as funções oferecidas, tais como restrições de tempo, de uso de recursos, etc. De maneira geral, requisitos não funcionais de produtos de *software* referem-se a atributos de qualidade que o sistema deve apresentar, tais como confiabilidade, usabilidade, eficiência, portabilidade, manutenibilidade e segurança (FALBO, 2014).

A descrição do minimundo apresenta, em um texto corrido, uma visão geral do domínio do problema a ser resolvido e dos processos de negócio apoiados, bem como as principais ideias do cliente sobre o sistema a ser desenvolvido.

Um importante modelo neste processo é o modelo de casos de uso. O modelo de casos de uso é um modelo comportamental que demonstra as funções do sistema, mas de maneira estática. Ele é composto de dois tipos principais de artefatos: os diagramas de casos de uso e as descrições de casos de uso. Um diagrama de casos de uso é um diagrama bastante simples, que descreve o sistema, seu ambiente e como sistema e ambiente estão relacionados. As descrições dos casos de uso descrevem o passo a passo para a realização dos casos de uso e são essencialmente textuais (FALBO, 2017).

Um diagrama de classes exibe um conjunto de classes e seus relacionamentos. Diagramas de classes proveem uma visão estática da estrutura de um sistema e, portanto, são usados na modelagem conceitual estrutural. Restrições de integridade são regras de negócio e poderiam ser lançadas no Documento de Requisitos. Contudo, como elas são importantes para a compreensão e eliminação de ambiguidades do modelo conceitual, é útil descrevê-las no próprio modelo conceitual (FALBO, 2017).

### 2.1.2 Projeto Arquitetural

Com os requisitos pelo menos parcialmente capturados e especificados na forma de modelos, pode-se começar a trabalhar no domínio da solução. Muitas soluções são possíveis para o mesmo conjunto de requisitos e elas são intrinsecamente ligadas a uma dada plataforma de implementação. A fase de projeto tem por objetivo definir e especificar uma solução a ser implementada. É uma fase de tomada de decisão, tendo em vista que

muitas soluções são possíveis (FALBO, 2017). Essa fase é responsável por incorporar requisitos tecnológicos aos requisitos essenciais do sistema, tem como objetivo definir a arquitetura do *software*, tendo por base o modelo construído na fase de análise de requisitos. Essa arquitetura deve descrever a estrutura de nível mais alto da aplicação e identificar seus principais componentes. O projeto também tem como propósito detalhar o projeto do *software* para cada componente identificado, esta fase é responsável por definir como serão implementadas as funcionalidades propostas, as tecnologias a serem utilizadas para desenvolvimento e para armazenamento de dados, etc. (FALBO, 2016).

Ao final da fase de Projeto, espera-se que a arquitetura do sistema esteja definida, bem como o projeto de seus componentes, divididos geralmente em camadas a fim de assegurar a separação de responsabilidades. Durante o desenvolvimento deste projeto, o método FrameWeb, descrito na Seção 2.2.2, foi utilizado para auxiliar no projeto arquitetural com base em *frameworks*.

## 2.2 Desenvolvimento *Web*

Com o avanço e disseminação da Internet, a *Web* ganhou cada vez mais espaço e passou a abrigar aplicações cada vez mais complexas, surgindo a necessidade da chamada Engenharia *Web*, que lida com o processo de desenvolvimentos de aplicações e sistemas *Web*. Sua essência é gerenciar a diversidade e a complexidade dessas aplicações. Pode-se dizer que a Engenharia *Web* é uma abordagem proativa de desenvolvedor aplicações Web (GINIGE; MURUGESAN, 2001).

### 2.2.1 Ambiente *Web*

A arquitetura por trás da *Web* é baseada na arquitetura cliente/servidor, formada por uma rede de computadores, na qual clientes e servidores se comunicam por meio de requisições e respostas. Utilizando um navegador *Web* (um *browser*), o cliente envia uma requisição usando o protocolo de comunicação HTTP (Protocolo de Transferência de Hipertexto, do inglês *HyperText Transfer Protocol*) para uma URI (Identificador Uniforme de Recursos, do inglês *Uniform Resource Identifier*) que, geralmente, retorna uma resposta em HTML (Linguagem de Marcação de Hipertexto, do inglês *HyperText Markup Language*).

O protocolo HTTP é um dos meios de transporte de arquivos e mensagens na *Web*, é um protocolo de aplicação cliente/servidor que define um formato padrão para que recursos sejam requisitados.

A Listagem 2.1 demonstra uma requisição HTTP. A primeira linha de uma requisição HTTP é usada para informar o método HTTP, no caso *GET*, a URI requisitada e a versão do protocolo a ser utilizada. Abaixo da primeira linha são listados os cabeçalhos da requisição, onde é possível utilizar cabeçalhos pré-definidos pelo protocolo, como *Host* e

*Referer* por exemplo, ou cabeçalhos definidos pelo usuário. Também é possível enviar um corpo da requisição, usado para passar parâmetros.

Listagem 2.1 – Exemplo de requisição HTTP

```
1 GET /index.html HTTP/1.1
2 Host: www.exemplo.com.br
```

A estrutura de uma resposta HTTP é bastante parecida com a de uma requisição. A primeira linha informa a versão do protocolo e o código de status da resposta. O código de status é um número de três digitos, sendo que o primeiro dígito define o tipo da resposta, códigos de status começando com 1 são informativos, 2 são utilizados para sucesso, 3 para redirecionamento, 4 para erros do cliente e 5 para erros do servidor. Abaixo da primeira linha são listados os cabeçalhos de resposta, que contêm informações úteis de como o cliente deve exibir a informação, como por exemplo o tipo de conteúdo (*Content-Type*). Assim como na requisição, existem os cabeçalhos pré-definidos e os definidos pelo usuário. Abaixo dos cabeçalhos temos o corpo da resposta, que é o conteúdo em si da resposta. A Listagem 2.2 é o exemplo de uma resposta HTTP com um conteúdo HTML.

Listagem 2.2 – Exemplo de resposta HTTP

```
1  HTTP/1.1 200 OK
2  Date: Mon, 11 Jun 2018 22:38:34 GMT
3  Content-Type: text/html; charset=UTF-8
4  Content-Encoding: UTF-8
5  Content-Length: 102
6  Server: nginx/1.9.6
7  Connection: close
8
9  <html>
10 <head>
11   <title>Exemplo de Página</title>
12 </head>
13 <body>
14   <h1>Olá, mundo.</h1>
15 </body>
16 </html>
```

### 2.2.2 Método FrameWeb

O FrameWeb é um método de projeto para construção de sistemas de informação *Web* (*Web Information Systems* – WISs) baseado em *frameworks* proposto inicialmente por Souza (2007). O método assume que determinados tipos de *frameworks* serão utilizados durante a construção da aplicação, define uma arquitetura básica para o sistema e propõe modelos de projeto que se aproximam da sua implementação usando esses *frameworks*.

De acordo com a especificação do método, a fase de Projeto concentra as propostas principais do método: (i) definição de uma arquitetura padrão que divide o sistema em

camadas, de modo a se integrar bem com os *frameworks* utilizados; (ii) proposta de um conjunto de modelos de projeto que trazem conceitos utilizados pelos *frameworks* para esta fase do processo por meio da criação de um perfil UML que faz com que os diagramas fiquem mais próximos da implementação (SOUZA, 2007).

O FrameWeb define uma extensão ao meta-modelo UML para representar componentes típicos da plataforma *Web* e dos *frameworks* utilizados, criando um perfil UML que é utilizado para a construção de diagramas de quatro tipos, são eles:

- **Modelo de Entidades:** representa os objetos de domínio do problema e seu mapeamento para a persistência em banco de dados relacional;
- **Modelo de Persistência:** representa as classes DAO existentes, responsáveis pela persistência das instâncias das classes de domínio.
- **Modelo de Navegação:** representa os diferentes componentes que formam a camada de Lógica de Apresentação, como páginas *Web*, formulários HTML e *controladores* do modelo MVC.
- **Modelo de Aplicação:** representa as classes de serviço, que são responsáveis pela codificação dos casos de uso, e suas dependências.

Os modelos do método FrameWeb são apresentados na Seção 4.2 no contexto deste trabalho.

O método proposto por Souza (2007) evoluiu através de estudos e publicações acadêmicas, uma sintaxe da linguagem FrameWeb foi definida formalmente por meta-modelos (MARTINS; SOUZA, 2015; MARTINS, 2016), substituindo as extensões leves da UML. Essa sintaxe serviu de base para a construção de ferramentas como o *FrameWeb Editor* (CAMPOS; SOUZA, 2017), uma ferramenta CASE (*Computer-Aided Software Engineering*), com a qual pode-se produzir e validar modelos FrameWeb, e uma ferramenta de geração de código a partir de modelos FrameWeb (ALMEIDA; CAMPOS; SOUZA, 2017).

### 2.2.3 Padrão MVC de Arquitetura de Software

Um dos padrões de arquitetura de software mais populares é o MVC, abreviação de *Model-View-Controller*, em português *Modelo-Visão-Controlador*, que proporciona diversos benefícios apesar de sua simplicidade. O MVC se baseia na divisão da aplicação em três camadas, conforme mostra a Figura 1. São elas:

- **Modelo:** camada que contém a lógica da aplicação, responsável pelas regras de

negócio e pela comunicação com a fonte dos dados, sendo ele um banco de dados, uma API, entre outros;

- **Visão:** camada de interação com o usuário. Serve apenas para a exibição de dados, sendo ela por meio de HTML, JSON, XML, PDF, entre outros.
- **Controlador:** camada intermediadora entre as camadas *Modelo* e *Visão*. As requisições provenientes do *browser* são processadas pelo *controlador*, o qual acessa os *modelos* para acessar os dados ou fazer persistência deles, retornando-os para as *visões* para que os mesmos sejam exibidos.

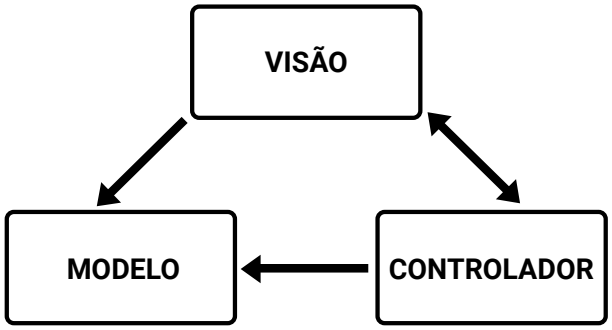



Figura 1 – Representação da arquitetura *Modelo-Visão-Controlador*

As vantagens de se usar um padrão de arquitetura de software como o MVC é a divisão de responsabilidades no código. A maneira que uma camada é implementada não deve interferir no funcionamento da outra e deve assumir apenas a sua responsabilidade. Isto permite que vários desenvolvedores trabalhem no mesmo código, em camadas diferentes, sem se preocupar como está sendo feita a implementação de uma outra camada e que uma camada possa ser testada independente da outra.

### 2.2.4 PHP e seu *framework* Laravel

O PHP[1] é uma linguagem interpretada focada principalmente em *scripts* do lado do servidor, mas que também possibilita a criação de *scripts* de linha e comando e aplicações *desktop*. Seu propósito principal é de implementar soluções *Web* velozes, simples e eficientes. O PHP é usado em mais de 80% dos sites ativos na *Web* e é a linguagem por trás de gigantes como o Facebook e Wikipedia (W3TECHS, 2017).

O Laravel[2] é um *framework* MVC escrito na linguagem PHP, de código aberto, possui uma sintaxe expressiva e atraente, sua simplicidade e variedade de recursos agiliza o processo de desenvolvimento de aplicações *Web*. Além de possuir uma ótima documentação em seu site oficial, o Laravel possui uma enorme comunidade e recursos disponíveis pela

[1] <http://php.net/>
[2] <https://laravel.com>

Internet, o que facilita muito o aprendizado.

O *framework* é composto por dezenas de componentes distribuídos em pacotes que adicionam funcionalidades a ele. Além dos componentes oficiais que compõem o *framework* Laravel, ainda é possível encontrar diversos componentes feitos pela comunidade.

Alguns dos módulos que estão incluídos no Laravel por padrão que merecem destaque por terem sido usados neste projeto são:

- **Eloquent:**[3] ORM para facilitar as interações com bancos de dados;
- **Router:**[4] provê uma maneira expressiva de roteamento URIs;
- **Cache:**[5] interface unificada para comunicação com mecanismos de *cache* como Redis[6] ou Memcached.[7]

O Laravel e outras dependências PHP utilizadas no projeto são disponibilizadas como pacotes através do repositório Packagist,[8] que é o principal repositório do gerenciador de dependências do PHP, o Composer.[9]

### 2.2.5 Docker

O Docker[10] é uma plataforma de software que permite a criação, o teste e a implantação de aplicações rapidamente. Com o Docker é possível criar pacotes de software em unidades padronizadas chamadas de contêineres. Os contêineres têm tudo o que o software precisa para ser executado, inclusive bibliotecas, código e *runtime*, com ele é possível ter certeza que código será executado em qualquer ambiente e plataforma.

## 2.3 Desenvolvimento para Dispositivos Móveis

Os dispositivos móveis crescem cada dia mais e já representam a maioria dos sistemas computadorizados atualmente. Os desenvolvedores têm que lidar com desafios como: recursos de CPU e memória limitados, bateria e diferentes tamanhos e densidades de telas. Apesar dos desafios, os desenvolvedores também contam com uma grande quantidade de dispositivos de entrada como câmeras, microfone, GPS (Sistema de Posicionamento Global, do inglês *Global Positioning System*) e sensores como acelerômetro, giroscópio,

[3] <https://laravel.com/docs/master/eloquent>
[4] <https://laravel.com/docs/master/routing>
[5] <https://laravel.com/docs/master/cache>
[6] <https://redis.io/>
[7] <https://memcached.org/>
[8] <https://packagist.org/>
[9] <https://getcomposer.org/>
[10] <https://www.docker.com/>

sensor de luminosidade, entre outros, além de dispositivos de comunicação como WiFi, telefonia, Bluetooth e NFC (Comunicação por Campo de Proximidade, do inglês *Near Field Communication*), o que proporciona a possibilidade de construção de aplicações que não seriam viáveis em outras plataformas.

Diferente do desenvolvimento para *Web*, o desenvolvimento para aplicativos móveis não necessariamente depende de um servidor para enviar os dados a serem renderizados, apesar de muitos aplicativos dependerem de um servidor para fornecerem conteúdos dinâmicos.

### 2.3.1 Padrão MVP de Arquitetura de Software

O padrão MVP, abreviação de *Model-View-Presenter*, em português *Modelo-Visão-Apresentação*, é um padrão derivado do MVC, porém mais voltado à Interface com o Usuário (IU), muito utilizado em projetos Android. A cadamada de *Apresentação* assume o papel da camada *Controlador* no MVC, a camada *Visão* assume um caráter passivo de exibição de dados e eventos com o usuário e passa a se comunicar apenas com a camada de *Apresentação*, como demonstrado na Figura 2.

Para um projeto Android, as *visões* são códigos que só rodam na plataforma Android e, portanto, necessitam de um emulador ou um dispositivo real, o que torna os testes mais lentos e custosos. O desacoplamento da *visão* com o *modelo* e a criação de uma interface para a comunicação da *visão* com a *apresentação* permite realizar testes nos módulos do código de uma maneira mais fácil, usando artifícios que simulam uma implementação e retornam dados no formato esperado da interface, este é um dos motivos do MVP ser mais usado do que o MVC no Android.

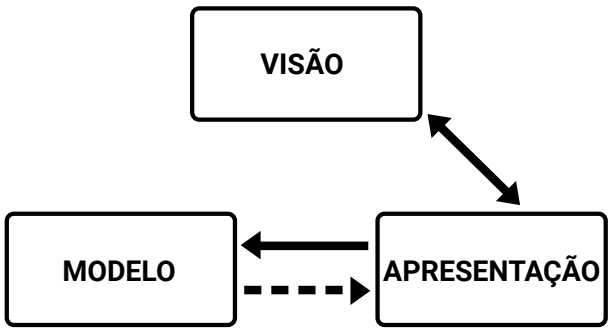



Figura 2 – Representação da arquitetura *Modelo-Visão-Apresentação*

### 2.3.2 *Framework* Android e a linguagem Kotlin

O Android[11] lidera o mercado de sistemas operacionais móveis junto com o iOS. Consiste em uma plataforma de código aberto que possui um *framework* para desen-

[11] <https://www.android.com/>

volvimento de aplicações nativas. As aplicações Android usam XML para descrever sua interface e sua principal linguagem para programação é o Java,[12] mas uma ação movida pela Oracle Corporation[13] contra o Google[14] pelo uso de APIs do Java dentro da plataforma do Android fez com que o Google buscasse por alternativas à linguagem.

Em Maio de 2017, o Google anunciou o suporte oficial à linguagem Kotlin[15] para o desenvolvimento nativo para a plataforma Android (CLERON, 2017). Desde então, a adoção da nova linguagem só vem aumentando e já estava presente em 14,7% dos aplicativos Android no fim de Setembro de 2017 (CIMPANU, 2017). O grande diferencial do Kotlin é a interoperabilidade com o Java. É possível usar classes e métodos escritos em Java dentro do Kotlin e vice-versa, o que facilita sua adoção, já que ele desfruta de todas bibliotecas já escritas em Java.

Listagem 2.3 – Definição de uma classe Cliente em Java, com *getter* e *setter*

```
1  public class Cliente {
2    private String nome;
3
4    public String getNome() {
5      return nome;
6    }
7
8    public void setNome(String nome) {
9      this.nome = nome;
10   }
11 }
```

Listagem 2.4 – Definição da classe Cliente em Kotlin

```
1 class Cliente(nome: String) {
2 }
```

A classe *Cliente* implementada em Java na Listagem 2.3 necessita de muito mais código para atingir o mesmo objetivo de sua implementação em Kotlin, demonstrada na Listagem 2.4. Outro exemplo de código mais enxuto é demonstrado nas listagens 2.5 e 2.6, que mostram a implementação de uma ação de clique em Java e Kotlin, respectivamente. Essa diminuição da verbosidade presente no Java é um dos grandes motivos para a migração de desenvolvedores Android para a linguagem Kotlin, uma vez que eles se tornam mais produtivos.

Listagem 2.5 – Definição de uma ação de clique em um botão, utilizando Java

```
1 botao.setOnClickListener(new View.OnClickListener(){
2   @Override
3   public void onClick(View v){
```

[12] <https://www.java.com/>
[13] <https://www.oracle.com>
[14] <https://www.google.com/>
[15] <https://kotlinlang.org/>

```
4       fazAlgo();
5     }
6 });
```

Listagem 2.6 – Definição de uma ação de clique em um botão, utilizando Kotlin

```
1 botao.setOnClickListener { fazAlgo() }
```

O Kotlin permite criar aplicações que rodam na JVM (Máquina Virtual Java, do inglês *Java Virtual Machine*), mas também pode ser compilado para JavaScript ou até mesmo para código nativo em diversas plataformas.

# 3 Especificação de Requisitos

Este capítulo aborda alguns resultados da Engenharia de Requisitos para a construção do SisPreço. Na Seção 3.1 é apresentado o minimundo do projeto; na Seção 3.2 é apresentado o diagrama de casos de uso; e na Seção 3.3 é apresentado o diagrama de classes. Os requisitos funcionais, requisitos não funcionais, regras de negócio e descrições detalhadas dos casos de uso podem ser encontrados no **Documento de Especificação de Requisitos** que está disponível no Apêndice ao final desta monografia.

## 3.1 Descrição do Escopo do Projeto

O Incaper (Instituto Capixaba de Pesquisa, Assistência Técnica e Extensão Rural) necessita de um sistema de informação para melhorar um processo existente de levantamento de preços de produtos e geração de relatórios. O propósito deste sistema é eliminar planilhas que são utilizadas atualmente e concentrar os dados em um banco de dados a fim de facilitar sua manipulação para a geração de relatórios mais complexos e de uma fácil visualização de dados históricos.

Ao abrir o sistema, o usuário não autenticado (visitante) será redirecionado para a página inicial, onde só possui acesso a uma funcionalidade do sistema, a consulta de preços. Existem dois tipos diferentes de relatórios de consulta de preços, a saber: Consulta de Preços Detalhada e Série Histórica de Preços.

Usuários devem poder personalizar os relatórios que desejam gerar. Para a consulta de preços detalhada deve ser possível selecionar os produtos, municípios, ano, meses e semanas. No relatório de série histórica de preço deve ser possível selecionar os produtos, municípios, ano inicial, ano final e a periodicidade dos valores: anual, mensal ou semanal. Todos os campos de personalização dos relatórios devem ser obrigatórios e em caso de campos de múltipla escolha, os mesmos devem estar pré-selecionados com todas as opções.

A princípio, os relatórios serão exibidos no formato de tabelas pela interface *Web* e poderão ser exportados no formato CSV.

Um usuário não autenticado poderá se autenticar utilizando um provedor de identidades do Incaper, provando sua identidade. O sistema deve ter acesso aos dados do usuário, incluindo o seu papel. Existem dois tipos de papéis: Administrador e Servidor.

Caso o usuário autenticado seja um Administrador, o mesmo deve ter acesso a três funcionalidades adicionais: a definição de produtos que participarão do levantamento de preços em um município, a consulta dos levantamentos de preços realizados e a definição de preços mínimos e máximos para produtos.

Anualmente, o IBGE fornece dados de produção agrícola, pecuária e de silvicultura para os municípios do estado do Espírito Santo. Um Administrador deve ser capaz de importar estes dados no sistema para que os mesmos sejam utilizados no auxílio da definição de produtos a participarem do levantamento de preços em um município. Ao tentar importar dados repetidos ou inválidos, o sistema deve alertar ao Administrador quais são as entradas problemáticas e avisá-lo que as mesmas não serão importadas.

A definição de produtos que participam do levantamento de preços pode ser realizada apenas por um Administrador e deve utilizar os dados de produção do ano anterior para sugerir os produtos participantes. O produto deve ser sugerido para participar do levantamento de preços em um município se este município está dentro do grupo dos 70% maiores produtores deste produto no estado.

Os Administradores devem poder consultar qual Servidor realizou um levantamento de preços, fornecendo o produto, ano, mês, semana e município do levantamento.

Servidores e Administradores devem ter acesso ao levantamento de preços. O levantamento de preços é realizado selecionando um dos municípios disponíveis, que serão determinados pela alocação de municípios do Servidor, um dos produtos disponíveis para o município escolhido, que serão determinados pelo Administrador, um produtor e inserindo o preço do produto. O valor inserido para o preço do produto deve ser validado de acordo com o intervalo de preços definido por um Administrador para o produto selecionado. A alocação de municípios do Servidor deve ser uma das informações fornecidas pelo provedor de identidades. Servidores precisam coletar pelo menos três amostras de preços de um mesmo produto em um município, ele deverá ser alertado caso ainda não tenha fornecido a quantidade mínima de três amostras. Caso o Servidor tente realizar um levantamento repetido, o valor do preço deve ser atualizado ao invés de ser contabilizado como um novo levantamento. Esta funcionalidade, em particular, deve também estar disponível em uma aplicação para dispositivos móveis para facilitar a inserção de dados pelos Servidores em campo.

O Incaper possui dados com as informações de produtos, beneficiários (produtores) e municípios que devem ser utilizadas no sistema. Apenas municípios do estado do Espírito Santo devem estar disponíveis no sistema. Cada beneficiário está associado a um município.

A definição de preços mínimos e máximos para produtos é utilizada parar minimizar os erros com a entrada de dados de preços por parte dos Servidores. O Administrador deve poder visualizar a faixa de valores definidas atualmente para os diversos produtos do sistema e deve poder definir uma nova faixa de valores aceitáveis.

## 3.2 Diagrama de Casos de Uso

O modelo de casos de uso visa capturar e descrever as funcionalidades que um sistema deve prover para os atores que interagem com o mesmo. No contexto deste projeto, foram detectados três tipos de atores: usuários não autenticados (Usuários), funcionários de escritórios municipais do Incaper (Servidores) e funcionários do escritório estadual do Incaper (Administradores). A Figura 3 apresenta os casos de uso do sistema com os três tipos de atores.

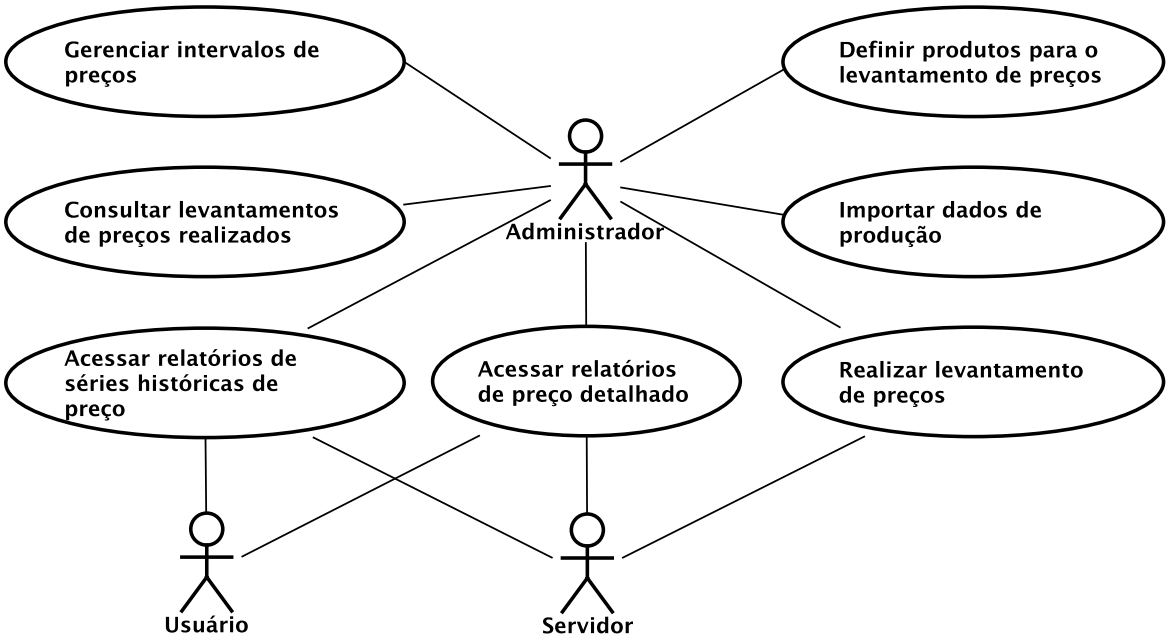



Figura 3 – Diagrama de Casos de Uso do sistema.

Administradores têm acesso a todas fucionalidades do sistema. Eles são responsáveis pela definição de quais produtos devem participar do levantamento de preços em um município, pela definição dos preços máximos e mínimos dos produtos para reduzir erros de digitação dos Servidores, por importar os dados de produção fornecidos pelo IBGE que auxiliam na definição de produtos para o levantamento e, por fim, podem consultar qual Servidor realizou um determinado levantamento de preço caso ele detecte algum valor incorreto.

Servidores são os atores responsáveis pelo levantamento de preços. Os valores informados por eles são os dados que serão utilizados posteriormente para a geração dos relatórios.

Usuários não autenticados só possuem acesso aos relatórios de preços e podem consultar os relatórios de séries históricas e de preços detalhados.

Maiores informações e detalhes sobre os casos de uso podem ser consultados no **Documento de Análise de Requisitos** que está disponível no Apêndice ao final desta

monografia.

## 3.3 Diagrama de Classes

A Figura 4 exibe o diagrama de classes do sistema. A classe *Levantamento* é o ponto principal do sistema, é ela a responsável por representar os levantamentos de preços que serão realizados pelos Servidores, e é com ela que serão calculados os preços para a geração dos relatórios. As demais classes servem de apoio tanto para o processo de levantamento de preços quanto para a geração de relatórios. As classes do *namespace Incaper* são classes que representam os dados fornecidos pelo Incaper ao SisPreço.

Um *Levantamento* é realizado para um determinado *Produto*, sendo seu preço fornecido por um *Beneficiário*, que por sua vez pertence a um *Município*. A definição de quais produtos participam do levantamento de preços em um determinado município é dado pela classe *Produto para Levantamento*. A classe *Produção* representa os dados de produção importados do IBGE, que servem de auxílio para a definição dos produtos que participam do levantamento.

A classe *Unidade Federativa* serve unicamente para identificar a qual estado um município pertence. O Incaper fornece dados de municípios de todo o Brasil, mas o SisPreço só necessita dos municípios do estado do Espírito Santo.

Um *Produto* possui uma *Unidade de Medida* que é utilizada como parâmetro para a definição de preço de um produto, por exemplo: enquanto o preço da Banana é dado por kg, o preço do Café é dado por saca de 60 kg.

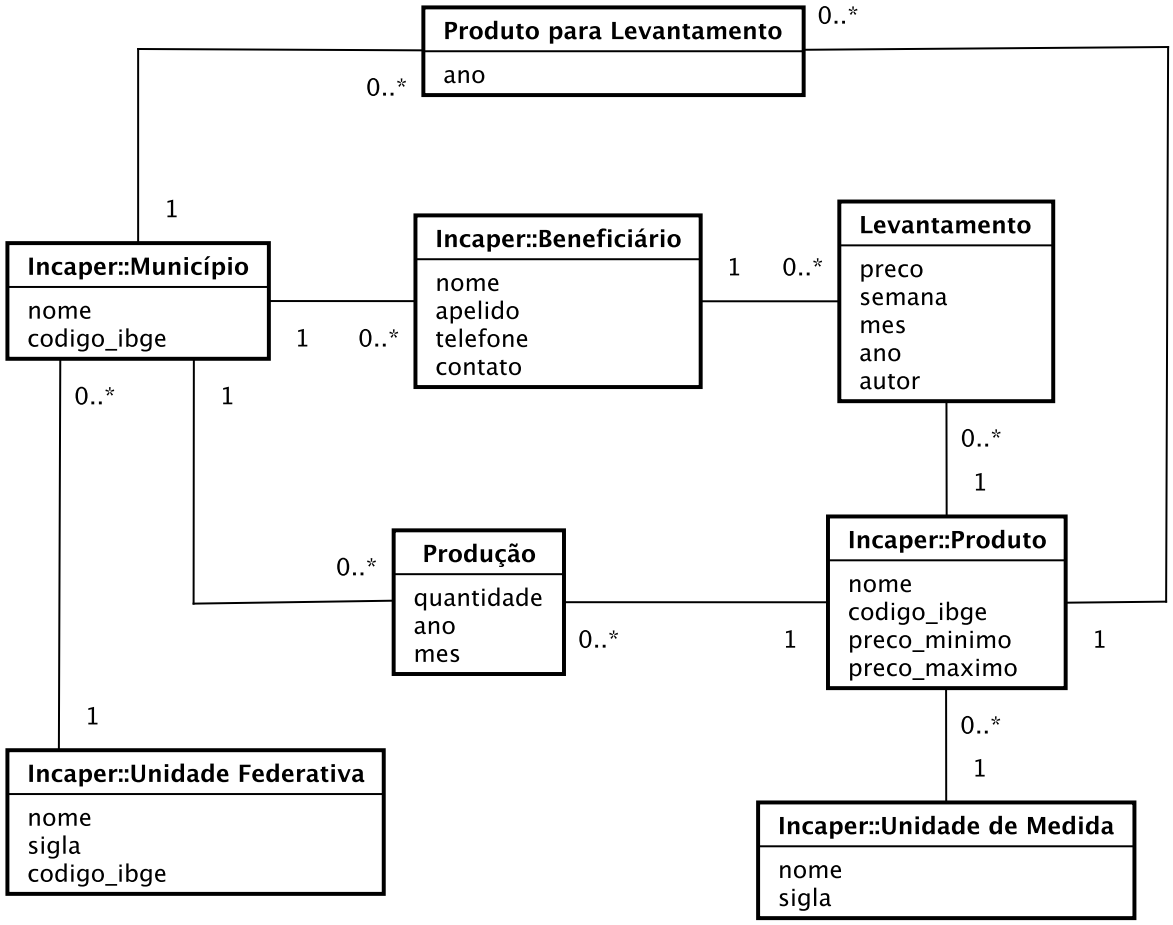



Figura 4 – Diagrama de Classes do sistema.

# 4 Projeto e Implementação

Este capítulo tem como objetivo apresentar o que foi desenvolvido, a solução proposta para o problema identificado e modelado no levantamento de requisitos. Para isso se faz necessário ter, além do conhecimento do domínio, também ter o conhecimento das tecnologias que serão utilizadas e a arquitetura a ser utilizada (FALBO, 2016).

As seções seguintes descrevem como o SisPreço foi construído. A Seção 4.1 apresenta a arquitetura do sistema, bem como todas as tecnologias envolvidas e sua implementação. A Seção 4.2 apresenta os modelos do FrameWeb criados. Por fim, a Seção 4.3 apresenta os resultados obtidos através de capturas de tela.

## 4.1 Arquitetura e Implementação

O SisPreço é composto por três subsistemas, dois subsistemas para o lado do servidor e outro para o lado do cliente. Para os dois subsistemas referentes ao lado do servidor, desenvolvidos para serem APIs, é usado o *framework* Laravel da linguagem PHP. O *framework* ORM (*Object/Relational Mapping*) usado foi o Eloquent. Já do lado do cliente é usado o Vue.js, um *framework* JavaScript para construção de interfaces com o usuário.

Para esse projeto foi utilizado o sistema de gerenciamento de banco de dados MySQL. A interação com ele é inteiramente responsabilidade dos subsistemas do lado do servidor, sendo manipulados através de modelos Eloquent.

A arquitetura do serviço é representada na Figura 5. Por trás de um *firewall*, a API do Incaper se comunica com o banco de dados para retornar dados para a API do SisPreço, através da Internet. O SisPreço, por sua vez, se comunica com seu banco de dados para a leitura e persistência de dados e utiliza um banco de dados secundário para fazer o cache de valores que ele requisita da API do Incaper.

A Listagem 4.1 mostra um exemplo simplificado de como essa arquitetura foi representada para a execução do sistema localmente utilizando o Docker.

O modelo escolhido para a construção da interface *Web* foi de uma aplicação de página única. Sua diferença para as aplicações de páginas tradicionais, ou de multi-páginas, é que toda a estrutura necessária pra a renderização da página e de seus dados é obtida de uma só vez, navegações entre seções da página se dão pela alteração do conteúdo exibido. Caso seja necessário o envio ou obtenção de dados, é feita uma requisição AJAX que retorna, normalmente, dados formatados em JSON. Em páginas tradicionais, a navegação se dá por meio de uma outra requisição ao servidor que retorna uma nova página. Os

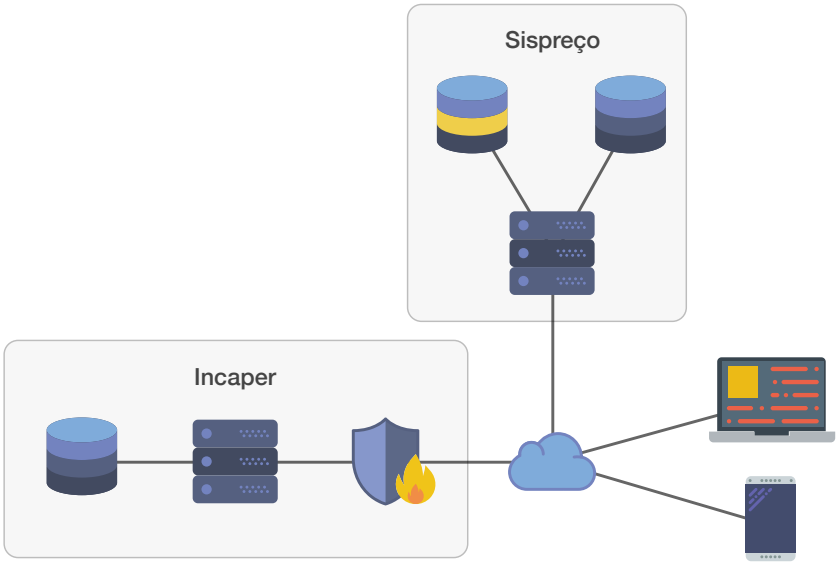



Figura 5 – Arquitetura do serviço.

Listagem 4.1 – Exemplo simplificado do arquivo docker-compose.yml

```
version: '2'

services:
  incaper-web:
    build: .
    depends_on:
      - incaper-db

  sispreco-web:
    build: .
    depends_on:
      - incaper-web
      - sispreco-db
      - sispreco-redis

  incaper-db:
    image: mysql:5.7

  sispreco-db:
    image: mysql:5.7

  sispreco-redis:
    image: redis:latest
```

ciclos de vida desses dois tipos de aplicações *Web* são mostrados na Figura 6.

A utilização do *framework* Vue.js facilitou muito a construção de uma aplicação de página única, utilizando suas funcionalidades de renderização condicional e de templates, demonstradas nas Listagens 4.2 e 4.3.

Apesar de serem subsistemas diferentes, as APIs do SisPreço e Incaper compartilham de alguns modelos, de códigos e configurações em comum para a inicialização do *framework*, e por este motivo foram implementados utilizando a mesma base de código. Em situações onde se viu necessária uma diferente implementação para os subsistemas, foram utilizadas

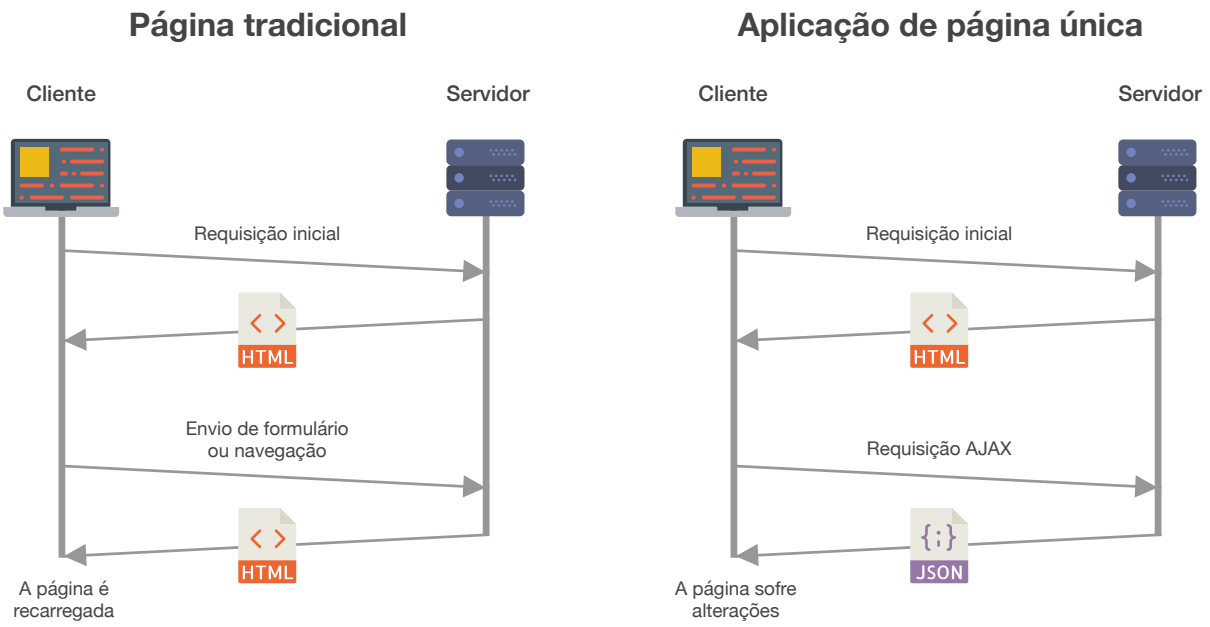



Figura 6 – Ciclo de vida de páginas tradicionais e de aplicações de página única.

Listagem 4.2 – Renderização condicional no Vue.js

```
<a v-if="user" v-on:click="$emit('logout')">Sair</a>
<a v-else v-on:click="$emit('login')">Entrar no sistema</a>
```

Listagem 4.3 – Renderização de templates no Vue.js

```
<table>
  <thead>
  <tr>
    <th>Ano</th>
    <th>Produto</th>
    <th>Preço</th>
  </tr>
  </thead>
  <tbody>
  <tr v-for="resultado in resultados">
    <td>{{ resultado.ano }}</td>
    <td>{{ resultado.produto }}</td>
    <td>{{ resultado.preco }}</td>
  </tr>
  </tbody>
</table>
```

estruturas condicionais baseadas no valor de uma variável de ambiente. A Listagem 4.4 mostra o caso da inclusão das rotas da API do SisPreço apenas se o modo de execução for *sispreco*.

A Listagem 4.5 mostra exemplos de como são definidas algumas rotas da API do Incaper, o mesmo é feito para as rotas do SisPreço. Através do módulo Route do Laravarel, é possível definir as rotas de uma maneira simples e expressiva. Por exemplo, a linha 1 da Listagem 4.5 define uma rota do método *GET* para o caminho */municipios* e indica que

Listagem 4.4 – Inclusão das rotas da API do SisPreço

```
if (config('app.mode') === 'sispreco') {
    Route::prefix('api')
        ->middleware('api')
        ->namespace($this->namespace)
        ->group(base_path('routes/sispreco_api.php'));
}
```

Listagem 4.5 – Exemplos de definições das rotas da API do Incaper

```
Route::get('/municipios', 'MunicipioController@all');

Route::get('/beneficiarios', 'BeneficiarioController@all');

Route::get('/produtos', 'ProdutoController@all');
```

Listagem 4.6 – Controlador resposável pelos Beneficiários da API do Incaper

```
class BeneficiarioController extends Controller
{
    public function all()
    {
        return BeneficiarioResource::collection(Beneficiario::all());
    }
}
```

ela deve ser tratada pelo método *all* da classe *MunicipioController*.

Os controladores criados durante a implementação do sistemas são mostrados na Figura 7, eles são responsáveis pelo tratamento das requisições recebidas. A Listagem 4.6 apresenta a implementação de uma ação que apresenta todos os beneficiários cadastrados no banco de dados.

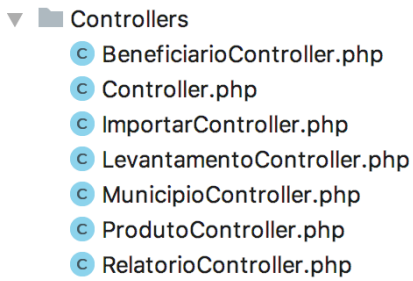



Figura 7 – Estrutura da pasta *app/Http/Controllers*.

A Figura 8 apresenta os modelos implementados no sistema. A implementação do modelo *Produto* é mostrada na Listagem 4.7 para demonstrar a implementação dos modelos. A Listagem 4.8 mostra como é simples realizar uma operação a partir de um modelo Eloquent, neste exemplo é obtida a soma das produções de um determinado produto e um ano.

A Figura 9 mostra a estrutura da pasta *app/database* que contém apenas a pasta

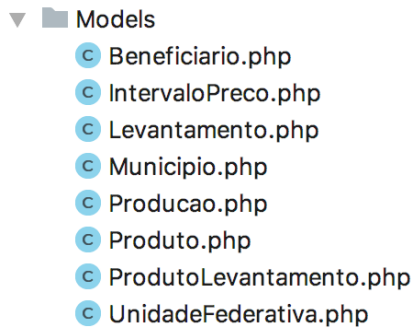



Figura 8 – Estrutura da pasta *app/Models*.

Listagem 4.7 – Modelo *Producao*

```
class Producao extends Eloquent\Model
{
    protected $table = 'producao';
    protected $primaryKey = 'id';
    public $timestamps = false;

    protected $fillable = [
        'produto',
        'municipio',
        'quantidade',
        'ano',
        'mes',
        'tipo',
    ];
}
```

Listagem 4.8 – Utilização dos métodos do Eloquent

```
$quantidadeTotal = Producao::where([
    'ano' => $ano,
    'produto' => $produto,
])->sum('quantidade');
```

de migrações do banco de dados. As migrações são mudanças que devem ser feitas na estrutura do banco de dados para acomodar as mudanças que ocorreram em uma nova versão do código. O Laravel mantém uma tabela no banco de dados para fazer o controle do versionamento, a versão do banco é dada pela última migração que ele executou e é definida pela data de criação da migração no início do nome do arquivo.

Apesar de ser possível definir migrações usando a linguagem SQL, a classe *Migration* do pacote *Database* do Laravel provê uma API para a manipulação das migrações. A Listagem 4.9 contém um exemplo utilizado no sistema para a criação de uma nova tabela. O método *up* é executado ao realizar a migração, criando a tabela *produtos_levantamento*, já o método *down* é executado caso haja a necessidade de reverter a migração, removendo a tabela criada.

Apesar de não ser uma parte muito relevante do sistema, o cliente para Android serve como uma prova de conceito de que sistemas contruídos totalmente em cima de APIs

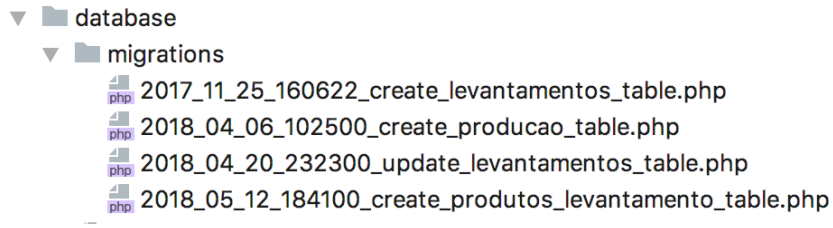



Figura 9 – Estrutura da pasta *app/database*.

Listagem 4.9 – Exemplo de uma migração do banco do dedados

```
class CreateProdutosLevantamentoTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('produtos_levantamento', function (Blueprint $table) {
            $table->increments('id');
            $table->string('produto');
            $table->integer('municipio');
            $table->integer('ano');

            $table->unique(['produto', 'municipio', 'ano']);
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('produtos_levantamento');
    }
}
```

podem ser fácilmente portados para outras plataformas, já que a implementação do cliente é totalmente independente da implementação do sistema e pode se comunicar com ele através de protocolos já consolidados.

Os códigos-fonte dos sistemas estão disponíveis no repositório do Incaper: <https://gitlab.com/incaper/>.

## 4.2 Modelos FrameWeb

Nesta seção, serão apresentados diagramas no padrão proposto pelo *FrameWeb* (SOUZA, 2007) de forma adaptada ao *framework* Laravel sendo utilizado em uma aplicação de página única.

O Modelo de Domínio é um diagrama de classes da UML que representa os objetos

de domínio do problema e seu mapeamento para a persistência em um banco de dados relacional. A partir dele são implementadas as classes da camada de Domínio.

Diferentemente da abordagem original proposta em 2007, todos os atributos que não podem ser nulos tiveram a tag `not null` omitida e aqueles que podem tiveram a tag `null` adicionada de forma a reduzir o impacto visual nos diversos diagramas. Pelo mesmo motivo, foi considerada que a estratégia padrão de recuperação de uma associação é do tipo *lazy*, e não *eager* como proposto pelo FrameWeb.

Todas as classes de domínio estendem uma classe do pacote Eloquent do Laravel, responsável pelo mapeamento entre classes e tabelas, por associações, dentre outras funcionalidades. Essa herança não é mostrada nos diagramas com o intuito de não poluí-los com várias associações, mas é possível saber mais sobre o Eloquent em <https://laravel.com/docs/5.6/eloquent>.

O modelo de domínio do sistema foi dividido em dois subsistemas pois os dados que são providos pelo Incaper são armazenados em um banco de dados diferente do SisPreço. A Figura 10 representa o modelo de domínio do subsistema Incaper.

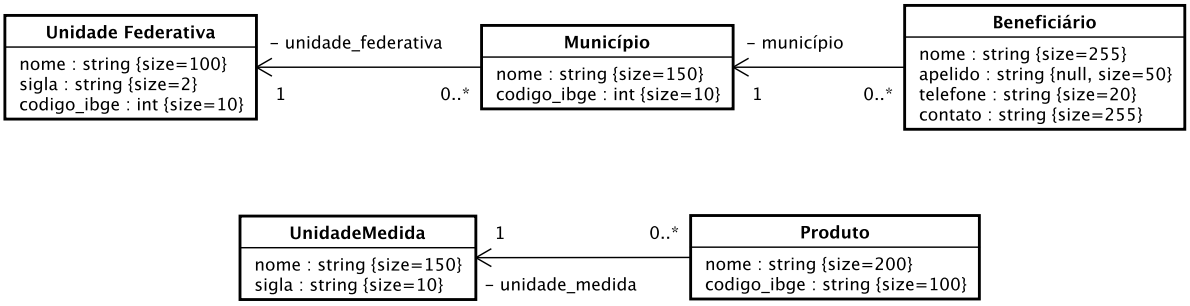



Figura 10 – Modelo de Domínio do susbsistema Incaper.

A seguir, a Figura 11 representa o modelo de domínio do subsistema SisPreço, onde as classes *Beneficiário*, *Produto* e *Município* são apenas referências para classes que existem no subsistema Incaper.

O Modelo de Persistência é um diagrama de classes da UML que representa as classes DAO existentes, responsáveis pela persistência das instâncias das classes de domínio. Diferente do que indica o método FrameWeb, não foram utilizadas classes DAO para a camada de acesso a dados, pois os modelos, como mencionado anteriormente, estendem um modelo Eloquent que já provê todas as funcionalidades necessárias para persistência dos dados, e portanto, não foram modelados os modelos de persistência. A Figura 12 apresenta a classe abstrata Eloquent, com alguns de seus métodos relevantes para a obtenção e persistência dos dados.

O Modelo de Navegação é um diagrama de classe da UML que representa os diferentes componentes que formam a camada de Apresentação, como páginas *Web*,

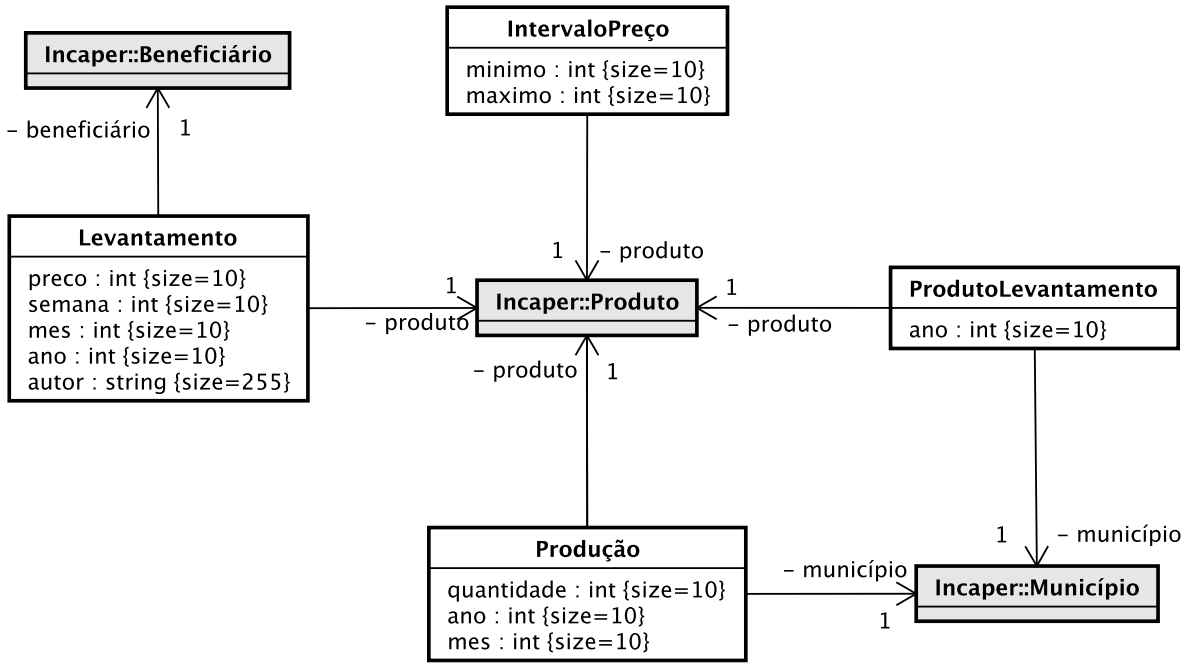



Figura 11 – Modelo de Domínio do subsistema SisPreço.

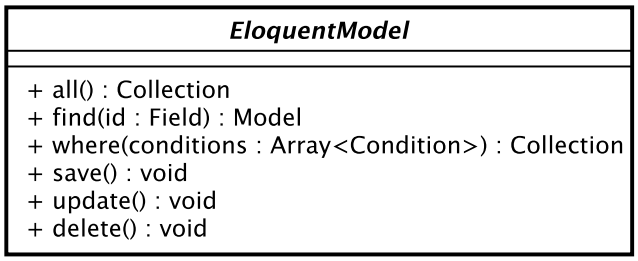



Figura 12 – Classe abstrata Eloquent.

formulários HTML e classes de ação. Esse modelo é utilizado pelos desenvolvedores para guiar a codificação das classes e componentes das camadas de View e Controller. A classe de ação é o principal componente deste modelo: suas associações de dependência ditam o controle de fluxo quando uma ação é executada.

Por estarmos lidando com uma aplicação de página única, foi adotada a nomenclatura `/#fragmento` para descrever qual o fragmento da página está sendo exibido. Todos os métodos representados nos modelos de navegação a seguir são chamados por requisições AJAX e o retorno de dados é sempre por objetos JSON.

A Figura 13 apresenta o modelo de navegação para o fluxo *Acessar relatórios de preço detalhado* do caso de uso *Acessar relatórios de preço detalhado*. O sistema, ao exibir o fragmento de consulta de preços, executa os métodos *getProdutos* e *getMunicipios*, que são necessários para popular as opções do formulário. Na submissão do formulário, o método *gerar* é executado, retornando os dados do relatório para a exibição.

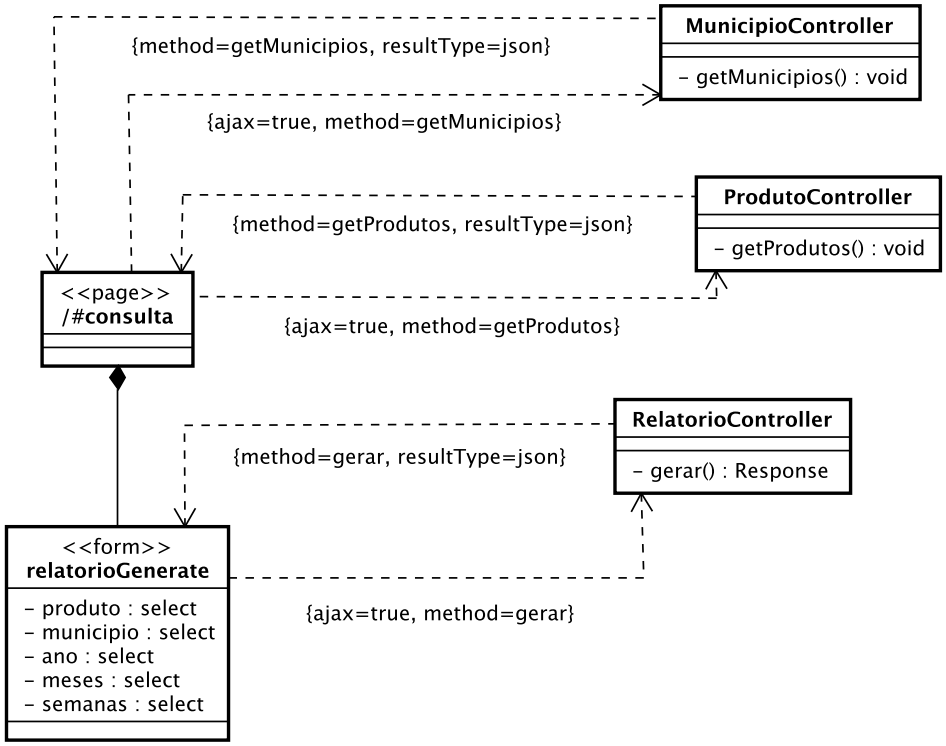



Figura 13 – Modelo de Navegação - Visualização de relatórios.

A Figura 14 exibe os modelos de navegação para os fluxos *Visualizar intervalos de preços* e *Definir intervalos de preços* do caso de uso *Gerenciar intervalos de preços* e o fluxo *Realizar levantamento de preços* do caso de uso *Realizar levantamento de preços*. Os fragmentos *levantamento* e *intervalo*, assim como o exemplo anterior, necessitam de dados adicionais para popular formulários ou listas. No fragmento *levantamento*, os métodos *getProdutos* e *getFromMunicipio* são executados para a obtenção dos produtos participantes no levantamento de um município e os beneficiários de um município, respectivamente. Ainda no fragmento *levantamento*, a submissão do formulário de levantamento de preços executa o método *create* para enviar os dados para o sistema. O fragmento *intervalo* executa o método *getIntervaloPreco* para obter os intervalos de preços e *postIntervaloPreco* ao submeter o formulário para atualizar um intervalo.

Por fim, a Figura 15 apresenta o modelo de navegação para o fluxo *Importar dados de produção* do caso de uso *Importar dados de produção*. Ele se inicia ao acessar a página de importação de dados de produção, representada pelo fragmento *producao*, neste fragmento tem-se a opção de fazer a importação dos dados enviando um arquivo com os dados de produção e selecionando o tipo do relatório. A importação é dada pelos métodos *pevs*, *ppm* e *lspa*, nomeados de acordo com o tipo do relatório selecionado. O resultado contém a informação se a importação foi bem sucedida e a lista de erros, caso contrário.

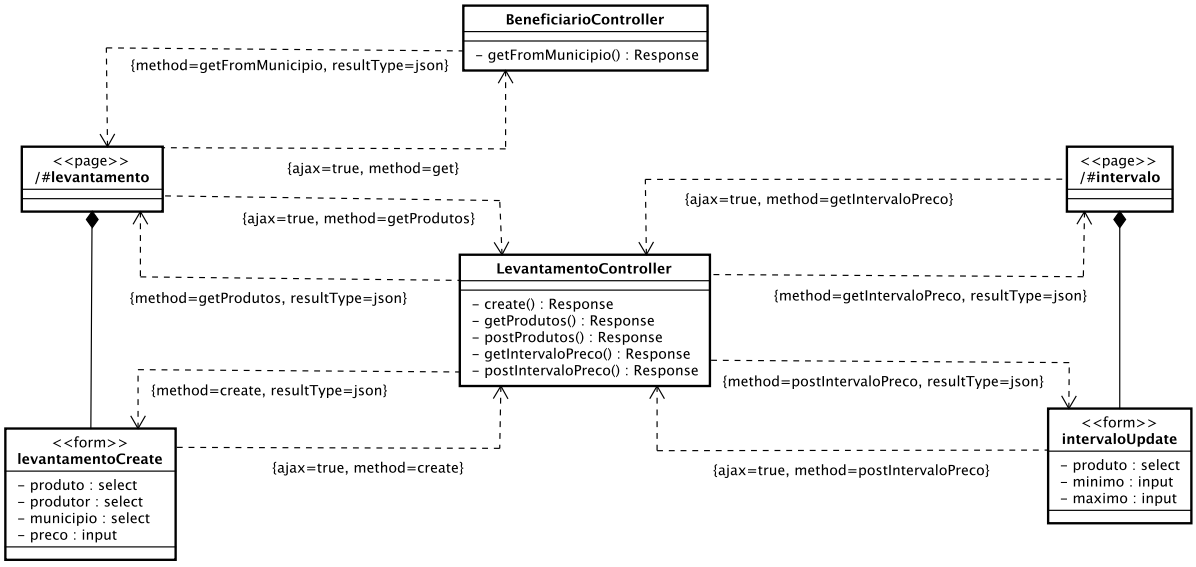



Figura 14 – Modelo de Navegação - Levantamento e definição de intervalo de preços.

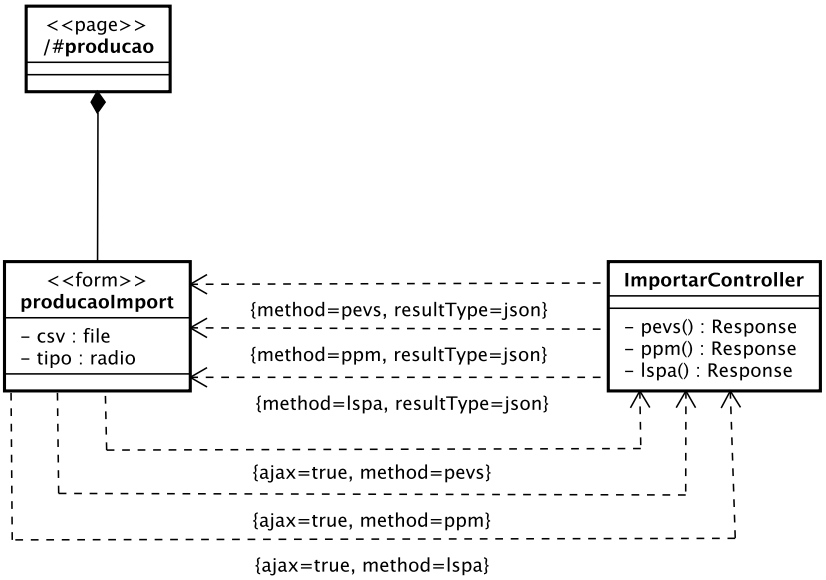



Figura 15 – Modelo de Navegação - Importação de dados de produção.

## 4.3 Apresentação

Nesta seção o sistema é apresentado por meio de capturas de tela, a fim de demonstrar o resultado obtido.

### 4.3.1 Sistema Web

Ao clicar em *Entrar no sistema* o usuário é redirecionado para o provedor de identidades do Incaper. A Figura 16 exibe a tela de login do provedor de identidades, nela o usuário deve inserir suas credenciais e, em caso de sucesso, é redirecionado de volta para o SisPreço. Caso o usuário não lembre sua senha, é possível recuperá-la através do

formulário de recuperação de senha, mostrado na Figura 17. Após preencher o formulário o usuário receberá um e-mail com as instruções para recuperar sua senha. Ao acessar o *link* de recuperação por *e-mail*, o usuário terá acesso à tela demonstrada na Figura 18 onde será possível alterar sua senha.


Figura 16 – SisPreço - *Login*.

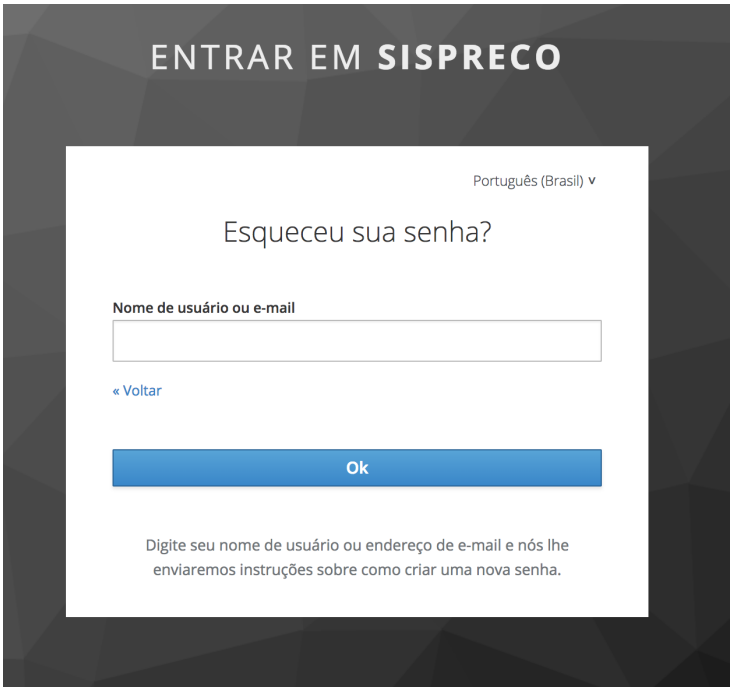



Figura 17 – SisPreço - Recuperar senha.

O sistema conta com um menu de navegação horizontal que, ao passar o mouse


Figura 18 – SisPreço - Alterar senha.

sobre seus itens, exibe as demais opções relacionadas, este comportamento é mostrado na Figura 19.


Figura 19 – SisPreço - Menu.

A Figura 20 apresenta a tela dos parâmetros de preços mínimos e máximos dos produtos do sistema. Esta tela exibe a lista de todos os produtos presentes no sistema, juntamente com sua unidade de medida e os valores definidos para o preço mínimo e máximo. Ao clicar no botão *Editar* de um produto, seus campos de preço mínimo e máximo

se tornam editáveis e é possível alterar os valores, o botão *Editar* é transformado em um botão *Salvar* que ao ser clicado salva os valores definidos.


Figura 20 – SisPreço - Intervalo de Preços.

A Figura 21 apresenta a tela de importação de dados de produção. O usuário deve selecionar um arquivo CSV de seu computador que contenha os dados de produção a serem importados e deve também selecionar o tipo de relatório que está sendo importado. Caso ocorra algum erro durante a importação, o usuário é apresentado à lista de erros que ocorreram e tem a opção de prosseguir com a importação ignorando as entradas que apresentaram erros, como mostra a Figura 22.


Figura 21 – SisPreço - Importar dados de produção.

A Figura 23 exibe a tela de definição de produtos a participarem do levantamento de preços em um município. Ao selecionar um município a partir das opções, é carregada


Figura 22 – SisPreço - Erro ao importar dados.

a lista de produtos, cada um acompanhando de uma caixa de seleção que determina se o mesmo deve participar do levantamento. Ao clicar em *Salvar*, os dados são salvos.


Figura 23 – SisPreço - Definição de produtos.

A Figura 24 apresenta a tela de levantamento de preços. O usuário deve selecionar o município, produto e produtor a partir da lista de opções e inserir o preço do produto. Ao clicar em *Enviar* os dados são enviados ao servidor e os campos são limpos para que possa ser realizado um novo levantamento. Caso o preço do produto fornecido pelo usuário esteja fora do intervalo de preços aceitáveis, ele será alertado, como demonstrado.


Figura 24 – SisPreço - Levantamento de preços.

A Figura 25 exibe a tela de consulta aos levantamentos de preços realizados. O usuário, ao preencher os campos, pode consultar qual Servidor realizou um levantamento de preços específico.

As Figuras 26 e 27 apresentam as tela de geração de relatórios de séries históricas e de preços detalhados, respectivamente. O usuário pode personalizar os parâmetros e escolher se deseja exibir o resultado na interface do sistema ou exportá-lo para um arquivo CSV.

### 4.3.2 Aplicativo Android

Esta seção apresenta as capturas de tela da aplicação móvel para Android.

A Figura 28 apresenta a tela inicial do sistema quando o usuário ainda não está autenticado. Ao clicar no botão *ENTRAR*, o usuário é encaminhado para uma página Web externa ao aplicativo onde deve se autenticar no provedor de identidades do Incpaper, em caso de sucesso na autenticação o usuário é redirecionado de volta ao aplicativo e é apresentado à tela de menu inicial.

Na Figura 29 é apresentada a tela de menu inicial, esta tela é acessada após a

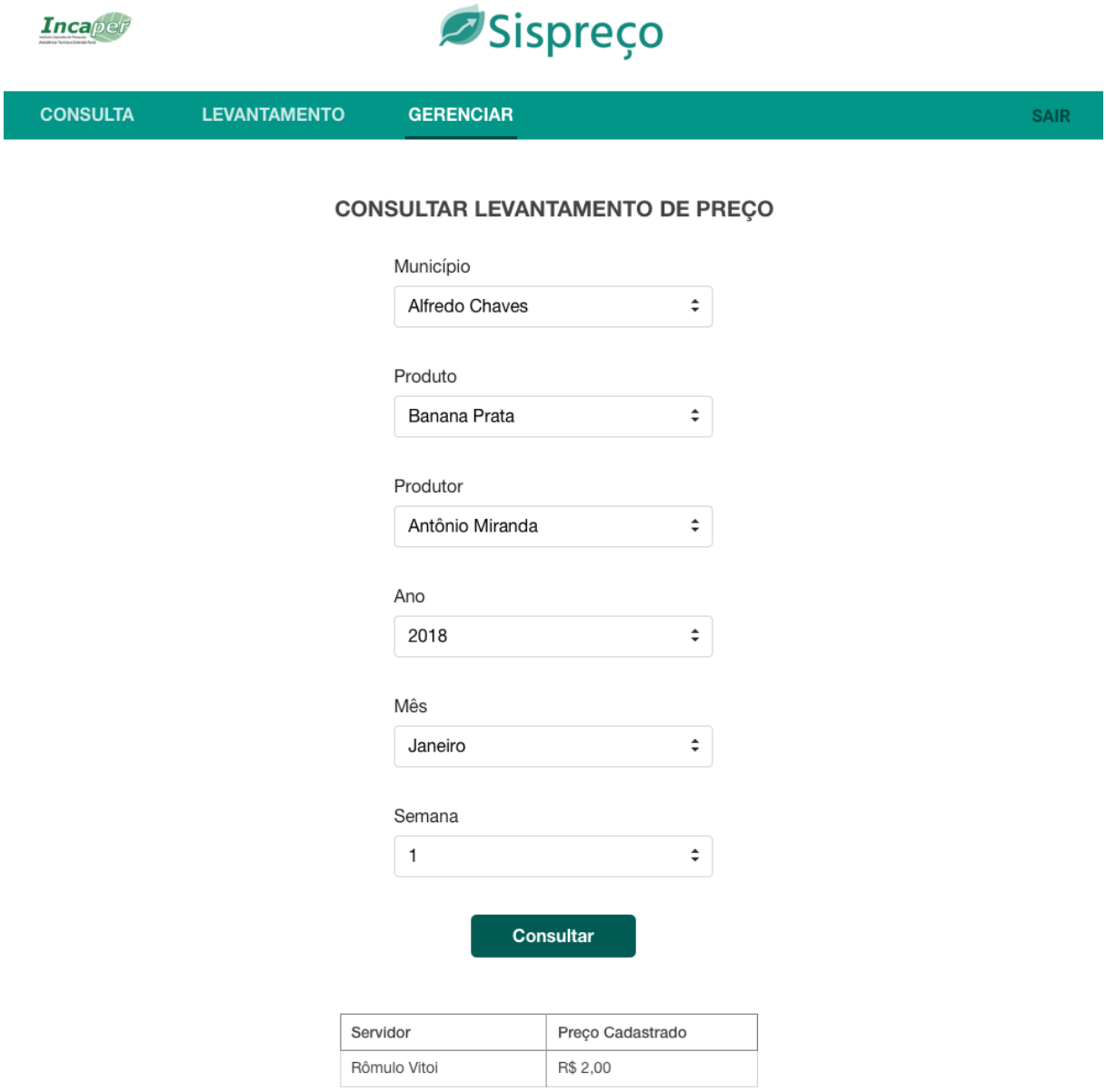



Figura 25 – SisPreço - Consulta de levantamento de preço.

realização do *login*, no caso do usuário já ter realizado o *login* no aplicativo anteriormente, esta é a tela inicial. Nesta tela é exibida a única funcionalidade atualmente presente no aplicativo, o levantamento de preços. Ao clicar em uma opção o usuário é redirecionado para a respectiva tela.

A Figura 30 apresenta a tela de lavantamento de preços, onde o usuário deve preencher quatro campos para realizar o levantamento, os campos *Município*, *Produto* e *Produtor* são campos auto-completáveis, que exibem as possibilidades de valores a serem selecionados de acordo com o que o usuário digita. Ao selecionar o botão *INCLUIR* uma requisição é enviada ao servidor com as informações e os campos são limpos para que o usuário possa realizar um novo levantamento. A seta no canto superior esquerdo retorna o

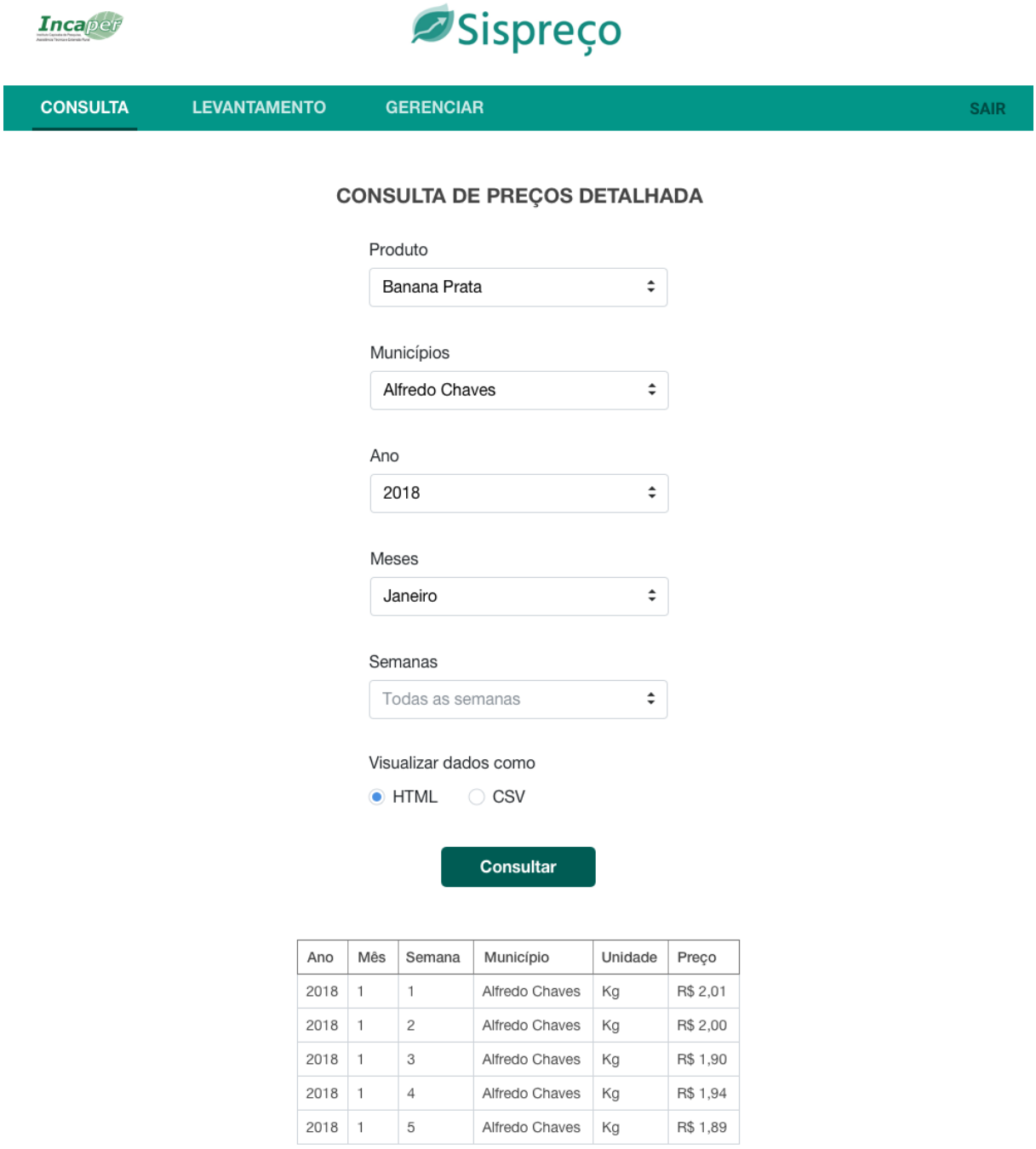
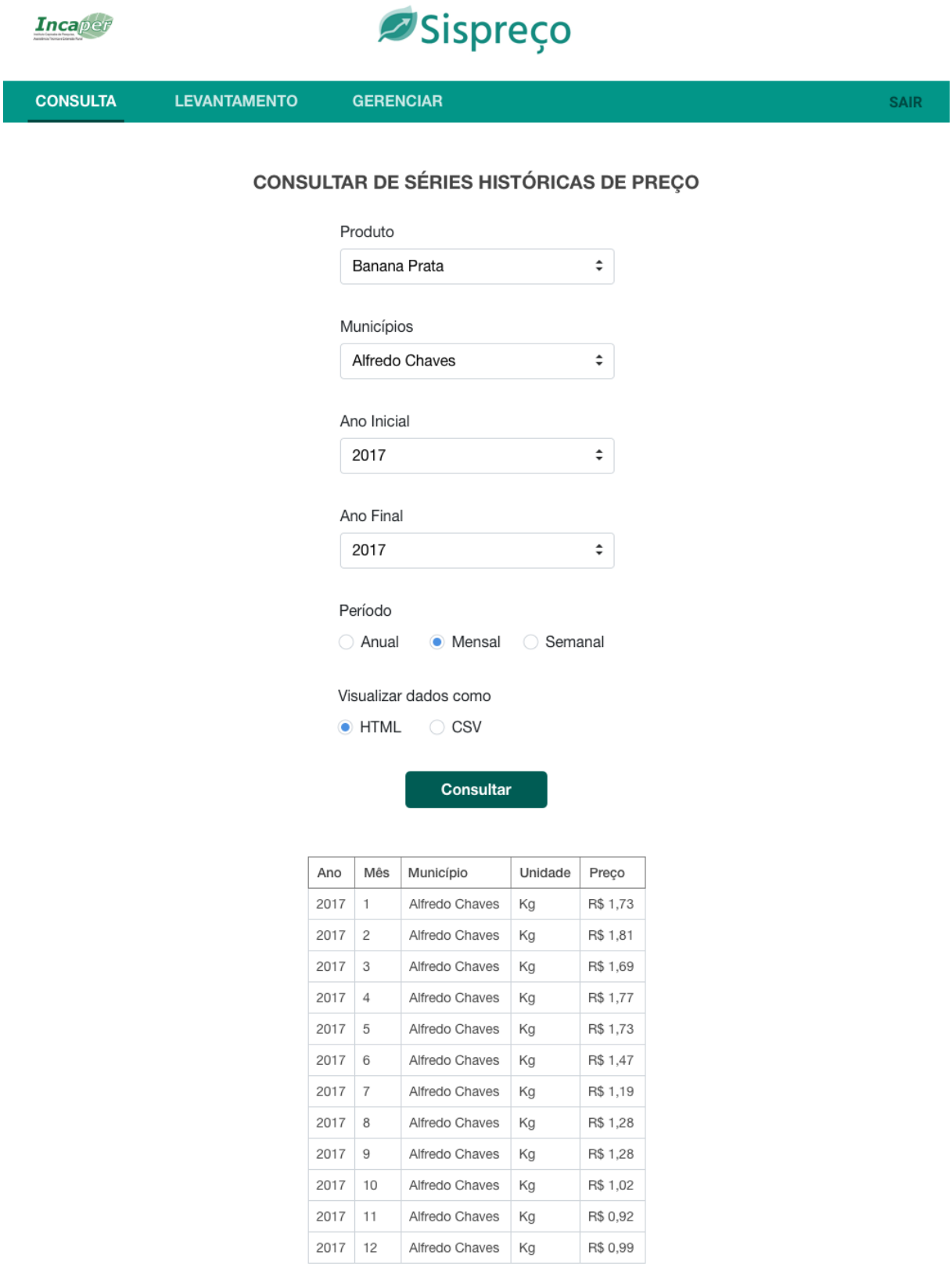



Figura 26 – SisPreço - Consulta de preços detalhados.

usuário para a tela anterior.




| Ano | Mês | Município | Unidade | Preço |
|---|---|---|---|---|
| 2017 | 1 | Alfredo Chaves | Kg | R$ 1,73 |
| 2017 | 2 | Alfredo Chaves | Kg | R$ 1,81 |
| 2017 | 3 | Alfredo Chaves | Kg | R$ 1,69 |
| 2017 | 4 | Alfredo Chaves | Kg | R$ 1,77 |
| 2017 | 5 | Alfredo Chaves | Kg | R$ 1,73 |
| 2017 | 6 | Alfredo Chaves | Kg | R$ 1,47 |
| 2017 | 7 | Alfredo Chaves | Kg | R$ 1,19 |
| 2017 | 8 | Alfredo Chaves | Kg | R$ 1,28 |
| 2017 | 9 | Alfredo Chaves | Kg | R$ 1,28 |
| 2017 | 10 | Alfredo Chaves | Kg | R$ 1,02 |
| 2017 | 11 | Alfredo Chaves | Kg | R$ 0,92 |
| 2017 | 12 | Alfredo Chaves | Kg | R$ 0,99 |

Figura 27 – SisPreço - Consulta de série histórica de preço.

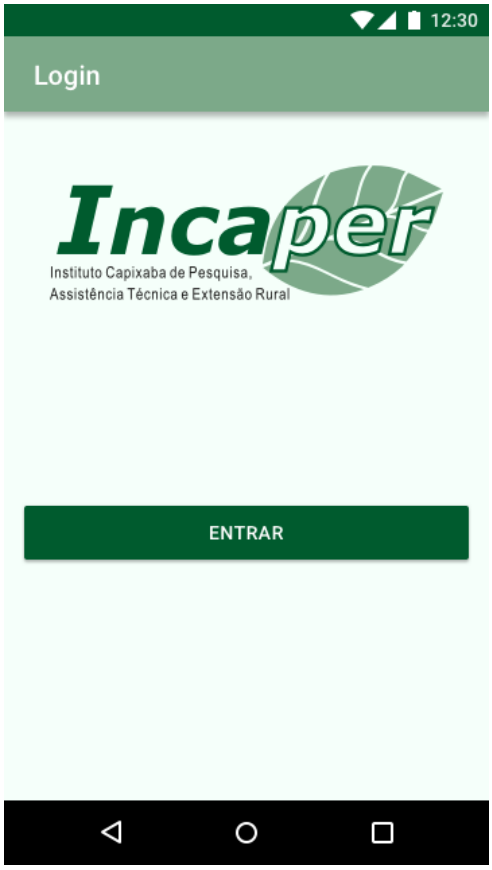



Figura 28 – Aplicativo Android - *Login*.


Figura 29 – Aplicativo Android - Menu inicial.

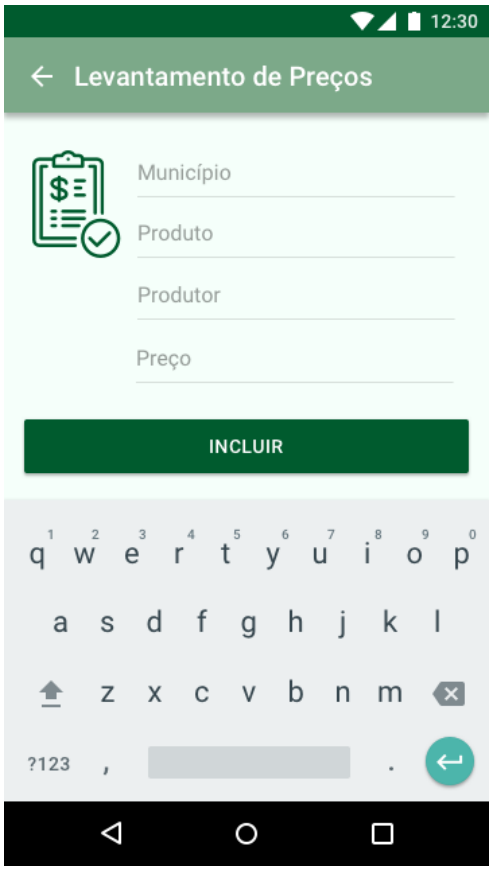



Figura 30 – Aplicativo Android - Levantamento de Preços.

# 5 Considerações Finais

Neste capítulo são apresentadas as considerações finais relacionadas ao trabalho feito. Por fim, são apresentadas suas limitações e perspectivas de possíveis trabalhos futuros.

## 5.1 Conclusões

Dada a necessidade de melhor organizar os dados de levantamentos realizados pelo Incaper, o SisPreço provê uma funcionalidade básica para a manipulação desses dados e cria uma base para uma plataforma que pode evoluir para melhor atender as necessidades do Incaper.

Todos os objetivos levantados no Capítulo 1 foram atingidos, com exceção da implementação da funcionalidade de alertar o usuário sobre o progresso da realização dos levantamentos. Durante o desenvolvimento, também foram surgindo idéias de novas funcionalidades que seriam úteis para os usuários do sistema, algumas delas acabaram sendo incluídas dentro do escopo do projeto e outras são descritas na Seção 5.2 como perspectivas de trabalhos futuros. Toda a documentação proposta foi produzida de acordo com os padrões de Engenharia de Software adotados. Foram levantados os requisitos, em seguida foi feita a análise de tais requisitos, produzindo, então, o Documento de Especificação de Requisitos. Este documento contém informações sobre requisitos funcionais, não funcionais, regras de negócio, descrição do sistema, definição de atores, casos de uso e diagramas de classe. Com o documento de requisitos finalizado, foi criado o Documento de Projeto, contendo todas as informações relacionadas à arquitetura do sistema e foram criados os modelos propostos na Seção 2.2.2, seguindo a abordagem FrameWeb.

Ao longo de todo o desenvolvimento, diversas dificuldades foram encontradas: o aprendizado de novos *frameworks* como o Laravel e Vue.js, desconhecidos até então, foram as maiores barreiras encontradas; a falta de prática e o longo período sem ter contato com os modelos de Engenharia de Requisitos também foram fatores que dificultaram o projeto; além disso, o FrameWeb se mostrou um método diferente dos quais são utilizados normalmente no desenvolvimento de software, ainda mais por tentá-lo aplicar em um *framework* e modelo de aplicação *Web* diferentes da proposta inicial.

Apesar da dificuldade, o método FrameWeb é bastante útil para o entendimento de como o sistema irá se comportar na *Web*, seu uso teria sido mais bem aproveitado se a modelagem tivesse ocorrido antes da implementação do sistema para auxiliar nas tomadas de decisão.

A execução do projeto poderia ter sido mais simples se todos os elementos estivessem presentes em um mesmo sistema, mas esta abordagem serve como exemplo de sistemas modernos que estão cada vez mais distribuídos, sendo necessária a comunicação entre sistemas.

Por fim, é importante citar como as disciplinas cursadas durante o curso ajudaram no desenvolvimento deste trabalho. Uma das coisas mais difíceis no final de um curso é conseguir utilizar todos os conhecimentos adquiridos em 5 anos e colocá-los em prática ao mesmo tempo, visto que muitas vezes os conceitos são vistos de maneiras isoladas. Este trabalho proporcionou isso, sendo possível colocar em prática boa parte das disciplinas estudadas durante o curso.

## 5.2 Limitações e Trabalhos Futuros

O desenvolvimento de software é um processo iterativo que passa por várias etapas como a refatoração para melhorias de performance ou até mesmo legibilidade do código, correções de falhas que não são detectadas pelo desenvolvedor durante o período de testes, adição de novas funcionalidades e melhoria de funcionalidades e da usabilidade do sistema a partir do *feedback* dos usuários.

A partir dos resultados obtidos e das percepções durante o desenvolvimento pôde-se verificar alguns pontos que possibilitam que novos projetos e trabalhos possam ser realizados. Esses pontos são listados abaixo.

- Implementar a funcionalidade de importar dados de levantamentos que já existem no Incaper;
- Implementar a funcionalidade de alertar ao Servidor a quantidade de levantamentos necessários restantes;
- Realizar testes unitários e testes fim-a-fim;
- Implementar as demais funcionalidades do sistema para o aplicativo móvel na plataforma Android;
- Implementar um aplicativo móvel para a plataforma iOS;
- Possibilitar que os relatórios sejam exportados em outros formatos como PDF e XLS (planilhas do Excel);
- Implementar novos tipos de relatórios.

# Referências

ALMEIDA, N. V. de; CAMPOS, S. L.; SOUZA, V. E. S. A Model-Driven Approach for Code Generation forWeb-based Information Systems Built with Frameworks. In: *Proc. of the 23rd Brazilian Symposium on Multimedia and the Web*. Gramado, RS, Brazil: ACM, 2017. p. 245–252. Citado na página 18.

CAMPOS, S. L.; SOUZA, V. E. S. FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb. In: *Anais do 16º Workshop de Ferramentas e Aplicações, 23º Simpósio Brasileiro de Sistemas Multimedia e Web*. Gramado, RS, Brazil: SBC, 2017. p. 199–203. Citado na página 18.

CIMPANU, C. *Kotlin Expected to Surpass Java as Android Default Programming Language for Apps*. 2017. Disponível em: <https://www.bleepingcomputer.com/news/mobile/kotlin-expected-to-surpass-java-as-android-default-programming-language-for-apps/>. Citado na página 22.

CLERON, M. *Android Announces Support for Kotlin*. 2017. Disponível em: <https://android-developers.googleblog.com/2017/05/android-announces-support-for-kotlin.html>. Citado na página 22.

FALBO, R. d. A. *Engenharia de Software*. [s.n.], 2014. 144 p. Disponível em: <https://inf.ufes.br/~falbo/files/ES/Notas_Aula_Engenharia_Software.pdf>. Citado 2 vezes nas páginas 14 e 15.

FALBO, R. d. A. *Projeto de Sistemas*. [s.n.], 2016. 138 p. Disponível em: <https://inf.ufes.br/~falbo/files/PSS/Notas_Aula_Projeto_Sistemas_2016.pdf>. Citado 2 vezes nas páginas 16 e 29.

FALBO, R. d. A. *Engenharia de Requisitos*. [s.n.], 2017. 178 p. Disponível em: <https://inf.ufes.br/~falbo/files/ER/Notas_Aula_Engenharia_Requisitos.pdf>. Citado 3 vezes nas páginas 14, 15 e 16.

GINIGE, A.; MURUGESAN, S. Web engineering: An introduction. *IEEE multimedia*, IEEE, v. 8, n. 1, p. 14–18, 2001. Citado na página 16.

MARTINS, B. F. *Evolução do Método FrameWeb para o Projeto de Sistemas de Informação Web Utilizando uma Abordagem Dirigida a Modelos*. Vitória, ES, Brazil, 2016. Citado na página 18.

MARTINS, B. F.; SOUZA, V. E. S. A Model-Driven Approach for the Design of Web Information Systems based on Frameworks. In: *Proc. of the 21st Brazilian Symposium on Multimedia and the Web*. Manaus, AM, Brazil: [s.n.], 2015. p. 41–48. Disponível em: <http://dl.acm.org/citation.cfm?id=2820439>. Citado na página 18.

SOMMERVILLE, I. *Software Engineering*. 9th. ed. USA: Addison-Wesley Publishing Company, 2010. ISBN 0137035152, 9780137035151. Citado na página 15.

SOUZA, V. E. S. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Dissertação (Mestrado) — Programa de Pós-Graduação em

Informática — Universidade Federal do Espírito Santo, 2007. Citado 4 vezes nas páginas 12, 17, 18 e 34.

W3TECHS. *Usage Statistics and market share of PHP for websites*. 2017. Disponível em: <https://w3techs.com/technologies/details/pl-php/all/all>. Citado na página 19.

# Apêndices


Documento de Especificação de Requisitos

# SisPreço

Registro de Alterações:

| Versão | Responsável | Data | Alterações |
|---|---|---|---|
| 1.0 | Rômulo de Angelis Vitoi | 28/06/2017 | Versão inicial da documentação |
| 1.1 | Vítor E. Silva Souza | 29/06/2017 | Primeira revisão |
| 1.2 | Rômulo de Angelis Vitoi | 30/06/2017 | Versão corrigida |
| 1.3 | Rômulo de Angelis Vitoi | 30/06/2017 | Adicionadas as Descrições de Caso de Uso. |
| 1.4 | Vítor E. Silva Souza | 03/07/2017 | Segunda revisão |
| 1.5 | Rômulo de Angelis Vitoi | 03/07/2017 | Versão corrigida |
| 1.6 | Vítor E. Silva Souza | 03/07/2017 | Revisão final |

Vitória, ES

2018

# 1 Introdução

Este documento apresenta os requisitos de usuário e a análise dos requisitos do sistema SisPreço. A atividade de análise de requisitos foi conduzida aplicando-se técnicas de modelagem de casos de uso, modelagem de classes e levando em consideração as boas práticas de programação relacionadas ao desenvolvimento de aplicações *Web*. Os modelos apresentados foram elaborados usando a UML.

Na Seção 2 deste documento descreve-se de forma geral o sistema, apresentando superficialmente suas principais características. Já na Seção 3 é apresentada a descrição do minimundo. A Seção 4 lista os requisitos de usuário do sistema (funcionais, não funcionais e suas regras de negócio). A Seção 5 apresenta o modelo de casos de uso, incluindo descrições de atores, os diagramas de casos de uso e suas respectivas descrições. Na Seção 6 estão apresentados os modelos conceituais estruturais do sistema na forma de diagramas de classes. Por fim, a Seção 7 apresenta o dicionário do projeto, contendo as definições das classes identificadas.

# 2 Descrição do Propósito do Sistema

Semanalmente, o Incaper (Instituto Capixaba de Pesquisa, Assistência Técnica e Extensão Rural) realiza pesquisas de preços de produtos e estes dados são utilizados para a geração de relatórios. Atualmente estes dados são inseridos em planilhas e a partir delas os relatórios são gerados, o problema desta abordagem é a manutenção e dificuldade de manipulação dessas planilhas que acabam inviabilizando a produção de relatórios mais complexos e de uma fácil visualização de dados históricos.

O SisPreço surge como uma solução para oferecer uma forma mais simples para a inserção de dados e um armazenamento unificado dos mesmos que facilita seu processamento para a geração de relatórios automatizados. Além da centralização dos dados, a criação de um sistema permite agregar várias funcionalidades, como a definição de produtos a terem seus preços levantados em um determinado município, validação das informações fornecidas pelos usuários, acompanhamento do progresso do processo de levantamentos de preços, entre outras.

# 3 Descrição do Minimundo

O Incaper (Instituto Capixaba de Pesquisa, Assistência Técnica e Extensão Rural) necessita de um sistema de informação para melhorar um processo existente de levantamento de preços de produtos e geração de relatórios. O propósito deste sistema é eliminar planilhas que são utilizadas atualmente e concentrar os dados em um banco de dados a fim de facilitar sua manipulação para a geração de relatórios mais complexos e de uma fácil visualização de dados históricos.

Ao abrir o sistema, o usuário não autenticado (visitante) será redirecionado para a página inicial, onde só possui acesso a uma funcionalidade do sistema, a consulta de preços. Existem dois tipos diferentes de relatórios de consulta de preços, a saber: Consulta de Preços Detalhada e Série Histórica de Preços.

Usuários devem poder personalizar os relatórios que desejam gerar. Para a consulta de preços detalhada deve ser possível selecionar os produtos, municípios, ano, meses e semanas. No relatório de série histórica de preço deve ser possível selecionar os produtos, municípios, ano inicial, ano final e a periodicidade dos valores: anual, mensal ou semanal. Todos os campos de personalização dos relatórios devem ser obrigatórios e em caso de campos de múltipla escolha, os mesmos devem estar pré-selecionados com todas as opções.

A princípio, os relatórios serão exibidos no formato de tabelas pela interface *Web* e poderão ser exportados no formato CSV.

Um usuário não autenticado poderá se autenticar utilizando um provedor de identidades do Incaper, provando sua identidade, o sistema deve ter acesso aos dados do usuário, incluindo o seu papel. Existem dois tipos de papéis: Administrador e Servidor.

Caso o usuário autenticado seja um Administrador, o mesmo deve ter acesso a três funcionalidades adicionais: a definição de produtos que participarão do levantamento de preços em um município, a consulta dos levantamentos de preços realizados e a definição de preços mínimos e máximos para produtos.

Anualmente, o IBGE fornece fornece dados de produção agrícola, pecuária e de silvicultura para os municípios do estado do Espírito Santo. Um Administrador deve ser capaz de importar estes dados no sistema para que os mesmos sejam utilizados no auxílio da definição de produtos a participarem do levantamento de preços em um município. Ao tentar importar dados repetidos ou inválidos, o sistema deve alertar ao Administrador quais são as entradas problemáticas e avisá-lo que as mesmas não serão importadas.

A definição de produtos que participam do levantamento de preços pode ser realizada apenas por um Administrador e deve utilizar os dados de produção do ano

anterior para sugerir os produtos participantes. O produto deve ser sugerido para participar do levantamento de preços em um município se este município está dentro do grupo dos 70% maiores produtores deste produto no estado.

Os Administradores devem poder consultar qual Servidor realizou um levantamento de preços, fornecendo o produto, ano, mês, semana e município do levantamento.

Servidores e Administradores devem ter acesso ao levantamento de preços. O levantamento de preços é realizado selecionando um dos municípios disponíveis, que serão determinados pela alocação de municípios do Servidor, um dos produtos disponíveis para o município escolhido, que serão determinados pelo Administrador, um produtor e inserindo o preço do produto. O valor inserido para o preço do produto deve ser validado de acordo com o intervalo de preços definido por um Administrador para o produto selecionado. A alocação de municípios do Servidor deve ser uma das informações fornecidas pelo provedor de identidades. Servidores precisam coletar pelo menos três amostras de preços de um mesmo produto em um município, ele deverá ser alertado caso ainda não tenha fornecido a quantidade mínima de três amostras. Caso o Servidor tente realizar um levantamento repetido, o valor do preço deve ser atualizado ao invés de ser contabilizado como um novo levantamento. Esta funcionalidade, em particular, deve também estar disponível em uma aplicação para dispositivos móveis para facilitar a inserção de dados pelos Servidores em campo.

O Incaper possui dados com as informações de produtos, beneficiários (produtores) e municípios que devem ser utilizadas no sistema. Apenas municípios do estado do Espírito Santo devem estar disponíveis no sistema. Cada beneficiário está associado a um município.

A definição de preços mínimos e máximos para produtos é utilizada parar minimizar os erros com a entrada de dados de preços por parte dos Servidores, o Administrador deve poder visualizar a faixa de valores definidas atualmente para os diversos produtos do sistema e deve poder definir uma nova faixa de valores aceitáveis.

# 4 Requisitos de Usuário

Tomando por base o contexto do sistema acima e considerando como principais *stakeholders* os funcionários do Incaper, foram identificados os seguintes requisitos de usuário e regras de negócio:

Tabela 1 – Requisitos Funcionais

| ID | Descrição | Prioridade | Depende |
|---|---|---|---|
| RF-1 | Servidores e Administradores devem ser capazes de se autenticar no sistema. | Alta | |
| RF-2 | Administradores devem ser capazes de importar dados de produção de produtos através de arquivos CSV. | Alta | RF-1 |
| RF-3 | O sistema deve alertar o Administrador caso ocorra algum problema durante a importação dos dados de produção, informando quais entradas não serão importadas. | Média | RF-2 |
| RF-4 | Administradores devem ser capazes de definir quais produtos participarão do levantamento de preços no ano corrente. | Alta | RF-1 |
| RF-5 | O sistema deve sugerir que um produto participe do levantamento de preços em um determinado município se este município estiver dentro do grupo dos 70% maiores produtores deste produto no estado. | Média | RF-4 |
| RF-6 | Administradores devem ser capazes de definir intervalos de preços aceitáveis para produtos, a fim de reduzir erros de digitação ao realizar um levantamento. | Alta | RF-1 |
| RF-7 | Administradores devem ser capazes de visualizar os intervalos de preços que já foram definidos. | Alta | RF-6 |
| RF-8 | Servidores e Administradores devem ser capazes de realizar o levantamento de preços, contendo: produto, produtor e preço. | Alta | RF-4, RF-6 |
| RF-9 | O valor inserido em um levantamento deve estar dentro do intervalo de preço definido. | Média | RF-8, RF-6 |
| RF-10 | O sistema deve alertar caso ainda não tenha sido realizado pelo menos três levantamentos de preços para um produto em um município. | Baixa | RF-8 |

| | | | |
|---|---|---|---|
| RF-11 | Ao realizar um levantamento com um conjunto de parâmetros de município, produto e produtor já existente na semana, o sistema deve atualizar o preço fornecido ao invés de adicionar uma nova entrada. | Alta | RF-8 |
| RF-12 | Administrador devem poder consultar o autor de um levantamento fornecendo o produto, ano, mês, semana e município do levantamento. | Baixa | RF-8 |
| RF-13 | Visitantes devem poder gerar relatórios de preço detalhado. | Alta | RF-8 |
| RF-14 | O visitante pode, ao solicitar o relatório de preços detalhado, especificar os seguintes parâmetros: produtos, municípios, ano, meses e semanas. | Alta | RF-13 |
| RF-15 | Visitantes devem poder gerar relatórios de série histórica de preços | Alta | RF-8 |
| RF-16 | O visitante pode, ao solicitar o relatório de série histórica de preços, especificar os seguintes parâmetros: produtos, municípios, ano inicial, ano final e periodicidade. | Alta | RF-15 |
| RF-17 | O visitante pode exportar os relatórios gerados no formato CSV. | Baixa | RF-13, RF-15 |

Tabela 2 – Requisitos Não Funcionais

| ID | Descrição | Categoria | Escopo | Prioridade |
|---|---|---|---|---|
| RNF-1 | O sistema deve estar disponível como uma aplicação Web, acessível a partir dos principais navegadores disponíveis no mercado. | Portabilidade | Sistema | Alta |
| RNF-2 | O levantamento de preços deve estar disponível como uma aplicação móvel. | Usabilidade | Sistema | Baixa |
| RNF-3 | A ferramenta dever ser de aprendizado fácil, não sendo necessário nenhum treinamento especial para seu uso. | Facilidade de Aprendizado | Sistema | Alta |

| | | | | |
|---|---|---|---|---|
| RNF-4 | O sistema deve controlar o acesso às suas funcionalidades por meio de autenticação e autorização, utilizando o papel de cada usuário para definir a quais funcionalidades ele terá acesso. | Segurança | Sistema | Alta |
| RNF-5 | O sistema deve garantir a consistência de dados por meio da obrigatoriedade no momento da inserção de certos dados. | Confiabilidade | Sistema | Alta |
| RNF-6 | A plataforma deve ser de fácil operação, não sendo necessário uso contínuo para uma boa operação do sistema. | Usabilidade | Sistema | Média |
| RNF-7 | O desenvolvimento do sistema deve facilitar manutenções futuras, utilizando padrões de design de software. | Manutenibilidade | Sistema | Média |

Tabela 3 – Regras de Negócio

| ID | Descrição | Prioridade | Depende |
|---|---|---|---|
| RN-1 | O sistema deve utilizar os dados de produtos, beneficiários e municípios já existentes no Incaper. | Alta | |
| RN-2 | O sistema deve exibir apenas municípios do estado do Espírito Santo em suas operações. | Alta | |
| RN-3 | A autenticação deve ser feita através de um provedor de identidade do Incaper. | Alta | RF-1 |
| RN-4 | Servidores só podem realizar levantamentos para os municípios em que estão alocados. | Alta | RF-8 |
| RN-5 | Não pode existir mais de um levantamento para um produto e produtor em um mesmo período. | Alta | RF-8 |
| RN-6 | O preço de um levantamento não pode estar fora do intervalo definido para o produto. | Alta | RF-6 |
| RN-7 | O limite inferior de preço deve ser maior do que o limite superior e ambos devem ser maiores do que 0. | Alta | RF-6 |
| RN-8 | Não pode existir mais de um dado de produção para um produto e município em um mesmo período. | Alta | RF-2 |

| | | | |
|---|---|---|---|
| RN-9 | Só deverão ser importados dados de produção para produtos e municípios que existam na base de dados. | Alta | RF-2 |
| RN-10 | Dados de produção de produtos que não existam na base de dados mas que tenham uma grande semalhança de nome com algum produto cadastrado deverão ser assimilados. | Baixa | RF-2 |

# 5 Modelo de Casos de Uso

O modelo de casos de uso corresponde a uma tentativa de descrever a relação das funcionalidades do sistema com cada um de seus atores. Os atores identificados no contexto deste projeto estão descritos na Tabela 4.

Tabela 4 – Atores

| Ator | Descrição |
|---|---|
| Usuário | Qualquer usuário que acesse o sistema e não tenha se autenticado. |
| Servidor | Funcionários de escritórios municipais do Incaper. |
| Administrador | Funcionários do escritório estadual do Incaper. |

A Figura 1 apresenta o diagrama de casos de uso.

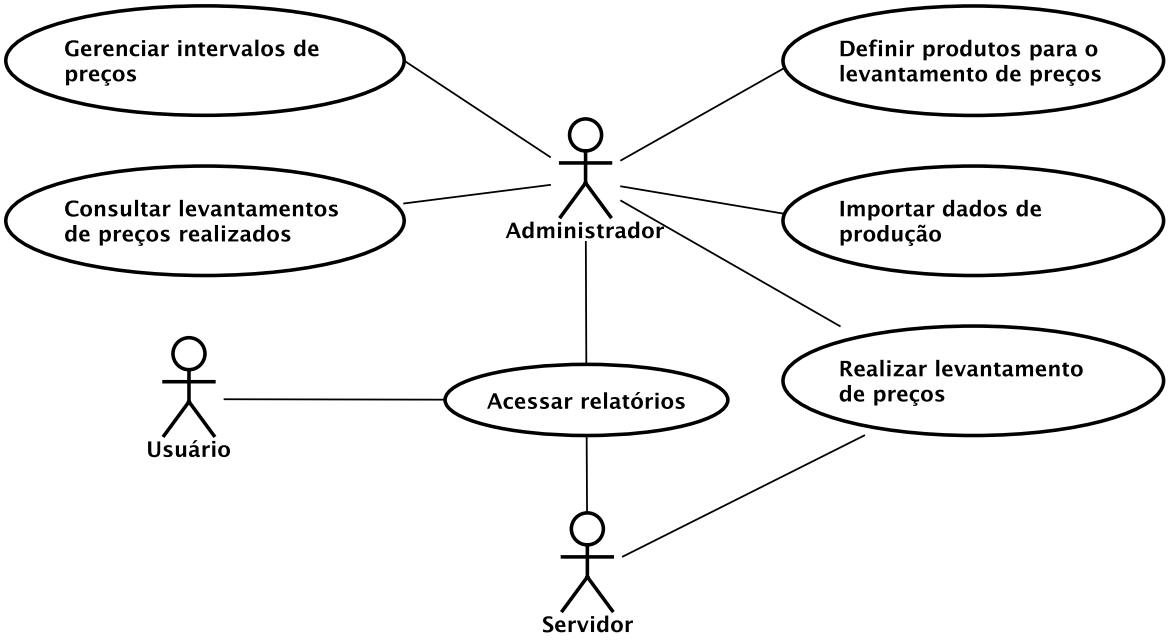



Figura 1 – Diagrama de Casos de Uso do sistema.

**Descrição de Caso de Uso**

**Projeto:** SisPreço
**Identificador do Caso de Uso:** UC-1
**Caso de Uso:** Gerenciar intevalos de preços
**Descrição Sucinta:** Este caso de uso permite que um administrador visualize e defina os intervalos de preços para produtos.

Tabela 5 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Visualizar intervalos de preços | O usuário deve estar logado no sistema e ser um administrador. | 1. O sistema deve exibir uma lista com os produtos e seus intervalos de preços. |
| Definir intervalos de preços | O usuário deve estar logado no sistema e ser um administrador. | 1. O administrador deve selecionar o produto que deseja definir o intervalo de preços.<br>2. O administrador deve inserir o preço máximo e o mínimo.<br>3. O sistema irá salvar o intervalo de preços definido no banco de dados. |

Tabela 6 – Fluxos de Eventos Variantes

| Nome do Fluxo | Variante | Descrição |
|---|---|---|
| Definir intervalos de preços | 2. O administrador não fornece algum dos valores. | 1. O sistema irá exibir uma mensagem informando que todos os campos são obrigatórios. |
| Definir intervalos de preços | 2. O administrador insere um valor máximo menor que o valor mínimo. | 1. O sistema irá exibir uma mensagem informando que o valor máximo deve ser maior que o valor mínimo. |

**Requisitos Relacionados:** RF-6, RF-7
**Classes Relacionadas:** Produto.

**Descrição de Caso de Uso**

**Projeto:** SisPreço
**Identificador do Caso de Uso:** UC-2
**Caso de Uso:** Definir produtos para o levantamento de preços
**Descrição Sucinta:** Este caso de uso permite que um administrador defina quais produtos irão participar do levantamento de preços no ano corrente.

Tabela 7 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Definir produtos para levantamento de preços em um município | O usuário deve estar logado no sistema e ser um administrador. | 1. O administrador deve selecionar o município para o qual deseja definir os produtos do levantamento.<br>2. O sistema pré-seleciona os produtos em que o muncípio selecionado está entre os 70% maiores produtores.<br>3. O administrador modifica a lista de produtos selecionados.<br>4. O sistema salva os municípios selecionados para o ano corrente no banco de dados. |

**Requisitos Relacionados:** RF-4, RF-5
**Classes Relacionadas:** Produção, Produto, Produto para Levantamento, Município.

**Descrição de Caso de Uso**

**Projeto:** SisPreço
**Identificador do Caso de Uso:** UC-3
**Caso de Uso:** Importar dados de produção
**Descrição Sucinta:** Este caso de uso permite que um administrador importe dados de produção para o sistema.

Tabela 8 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Importar dados de produção | O usuário deve estar logado no sistema e ser um administrador. | 1. O administrador enviará um arquivo CSV contendo os dados de produção.<br>2. O sistema salva os dados no banco de dados. |

Tabela 9 – Fluxos de Eventos Variantes

| Nome do Fluxo | Variante | Descrição |
|---|---|---|
| Importar dados de produção | 1. Os dados de produção contêm municípios não cadastrados no sistema. | 1. O sistema irá exibir uma mensagem informando que os dados não serão importados. |
| Importar dados de produção | 1. Os dados de produção contêm produtos não cadastrados no sistema. | 1. O sistema irá exibir uma mensagem informando que os dados não serão importados. |
| Importar dados de produção | 1. Os dados de produção já existem no sistema. | 1. O sistema irá exibir uma mensagem informando que os dados não serão importados. |

**Requisitos Relacionados:** RF-2, RF-3
**Classes Relacionadas:** Município, Produção, Produto.

**Descrição de Caso de Uso**

**Projeto:** SisPreço
**Identificador do Caso de Uso:** UC-4
**Caso de Uso:** Realizar levantamento de preços
**Descrição Sucinta:** Este caso de uso permite que um administrador ou servidor realize o levantamento de preços.

Tabela 10 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Realizar levantamento de preços | O usuário deve estar logado no sistema e ser um administrador ou servidor. | 1. O usuário preenche os dados do levantamento: município, produto, produtor (beneficiário) e preço.<br>2. O sistema salva os dados no banco de dados, registrando a semana, mês e ano atuais e o usuário autor do levantamento. |

Tabela 11 – Fluxos de Eventos Variantes

| Nome do Fluxo | Variante | Descrição |
|---|---|---|
| Realizar levantamento de preços | 1. O preço informado está fora do intervalo definido para o produto. | 1. O sistema irá exibir uma mensagem informando que o valor é inválido. |
| Realizar levantamento de preços | 1. O usuário ainda não realizou três levantamentos para um produto em um município. | 1. O sistema irá exibir uma mensagem informando que mais levantamentos são necessários. |
| Realizar levantamento de preços | 2. Já existe um levantamento para um mesmo conjunto de parâmetros de município, produto e produtor na mesma semana. | 1. O sistema atualizará o valor do levantamento. |

**Requisitos Relacionados:** RF-8, RF-9, RF-10, RF-11,
**Classes Relacionadas:** Beneficiário, Levantamento, Município, Produto, Produto para Levantamento.

**Descrição de Caso de Uso**

**Projeto:** SisPreço
**Identificador do Caso de Uso:** UC-5
**Caso de Uso:** Consultar levantamentos de preços realizados
**Descrição Sucinta:** Este caso de uso permite que um administrador consulte qual usuário realizou um determinado levantamento de preços.

Tabela 12 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Consultar levantamentos de preços realizados | O usuário deve estar logado no sistema e ser um administrador. | 1. O administrador preencherá os dados: produto, ano, mês, semana e município.<br>2. O sistema exibe o usuário que realizou o levantamento. |

Tabela 13 – Fluxos de Eventos Variantes

| Nome do Fluxo | Variante | Descrição |
|---|---|---|
| Consultar levantamentos de preços realizados | 1. Algum dado não foi fornecido. | 1. O sistema irá exibir uma mensagem alertando que todos os campos são obrigatórios. |
| Consultar levantamentos de preços realizados | 2. Não existe um levantamento com os dados fornecidos. | 1. O sistema irá exibir uma mensagem informando que não existe um levantamento com os dados fornecidos. |

**Requisitos Relacionados:** RF-12
**Classes Relacionadas:** Beneficiário, Levantamento, Município, Produto.

**Descrição de Caso de Uso**

**Projeto:** SisPreço
**Identificador do Caso de Uso:** UC-6
**Caso de Uso:** Acessar relatórios de preço detalhado
**Descrição Sucinta:** Este caso de uso permite que um usuário acesse os relatórios de consulta de preço detalhado.

Tabela 14 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Acessar relatórios de preço detalhado | | 1. O usuário seleciona os valores desejados para os filtros: produto, municípios, ano, meses e semanas.<br>2. O sistema exibe o relatório de preços a partir dos filtros que foram selecionados, contendo as seguintes colunas: ano, mês, semana, município, unidade de medida e preço. |
| *Download* de relatórios de preço detalhado | | 1. O usuário seleciona os filtros desejados como descrito no fluxo *Acessar relatórios de preço detalhado* e seleciona a opção de realizar o *download* do relatório em formato CSV.<br>2. O sistema gera um arquivo no formato CSV com os dados do relatório, como descrito no fluxo *Acessar relatórios de preço detalhado*. |

**Requisitos Relacionados:** RF-13, RF-14, RF-17
**Classes Relacionadas:** Levantamento, Município, Produto, Unidade de Medida.

**Descrição de Caso de Uso**

**Projeto:** SisPreço
**Identificador do Caso de Uso:** UC-7
**Caso de Uso:** Acessar relatórios de séries históricas de preços
**Descrição Sucinta:** Este caso de uso permite que um usuário acesse os relatórios de séries históricas de preços.

Tabela 15 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Acessar relatórios de séries históricas de preços | | 1. O usuário seleciona os valores desejados para os filtros: produto, municípios, ano inicial, ano final e periodicidade dos valores. As periodicidades disponíveis são: anual, mensal e semanal.<br>2. O sistema exibe o relatório de preços a partir dos filtros que foram selecionados, contendo as seguintes colunas: período, município, unidade de medida e preço. |
| *Download* de relatório de séries históricas | | 1. O usuário seleciona os filtros desejados como descrito no fluxo *Acessar relatórios de séries históricas de preços* e seleciona a opção de realizar o *download* do relatório em formato CSV.<br>2. O sistema gera um arquivo no formato CSV com os dados do relatório, como descrito no fluxo *Acessar relatórios de séries históricas de preços*. |

**Requisitos Relacionados:** RF-15, RF-16, RF-17
**Classes Relacionadas:** Levantamento, Município, Produto, Unidade de Medida.

# 6 Modelo Estrutural

O modelo conceitual estrutural visa capturar e descrever as informações (classes, associações e atributos) que o sistema deve representar para prover as funcionalidades descritas na seção anterior. A seguir, é apresentado o diagrama de classes identificado no contexto deste projeto. Na Seção 7 – Dicionário de Projeto – são apresentadas as descrições das classes, atributos e operações presentes nos diagramas apresentados nesta seção.

## 6.1 Diagrama de Classes

A Figura 2 apresenta o diagrama de classes do sistema. As classes do *namespace* *Incaper* são classes que representam os dados fornecidos pelo Incaper ao Sispreço.

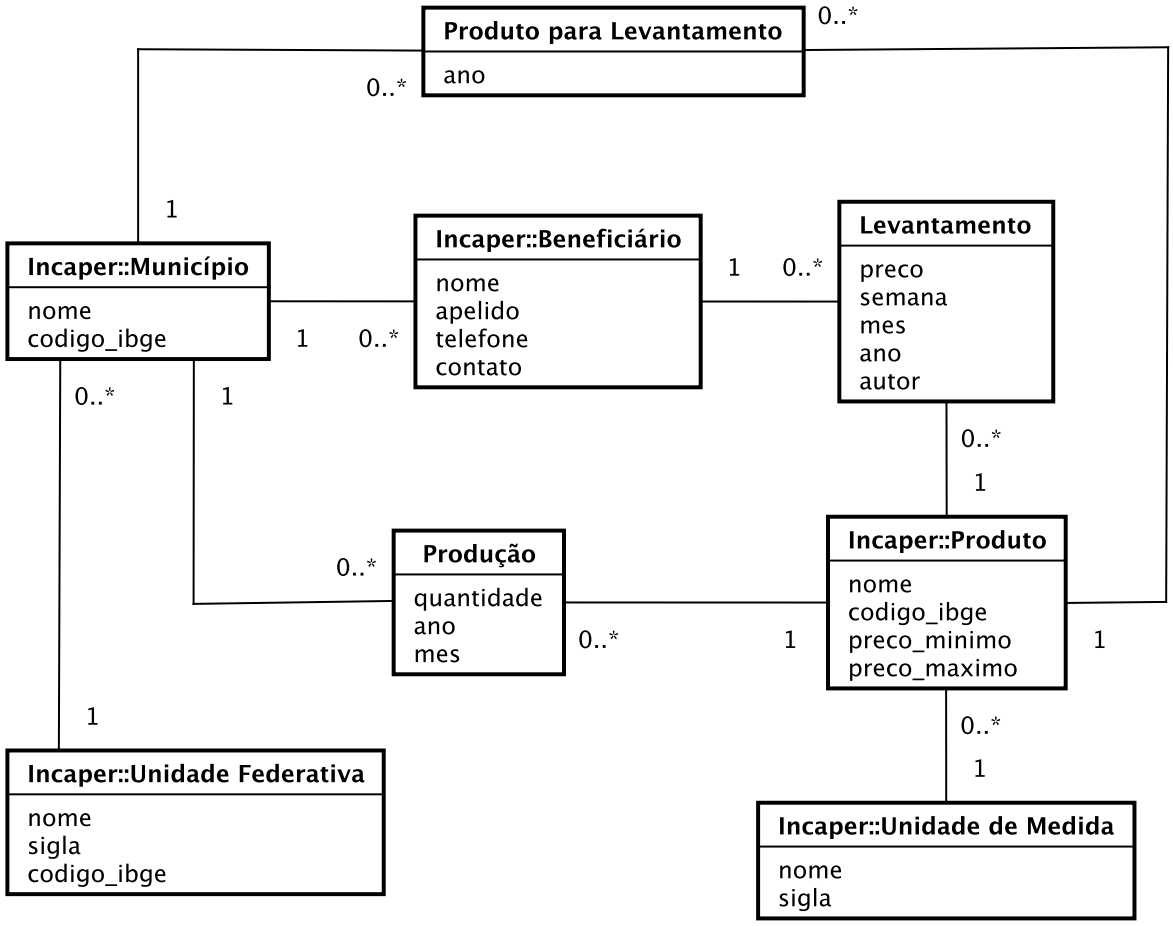



Figura 2 – Diagrama de Classes do sistema.

# 7 Dicionário de Projeto

Esta seção apresenta as definições das classes (e seus atributos), servindo como um glossário do projeto. Vale destacar que eventuais operações que estas classes vierem a ter não são listadas e descritas nesta fase do projeto.

## 7.1 Beneficiário

Um Beneficiário representa um produtor que fornece informações para um usuário realizar o levantamento de preços.

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| nome | Texto | x | Nome completo do beneficiário. |
| apelido | Texto | | Nome alternativo. |
| telefone | Texto | x | Telefone de contato do beneficiário. |
| contato | Texto | | Pessoa a ser contatada pelo telefone. |
| municipio | Município | x | Município que o beneficiário pertence. |
| levantamentos | Levantamento | | Levantamentos de preço feitos com o beneficiário. |

## 7.2 Levantamento

Um Levantamento representa uma pesquisa de preço realizada por um Usuário do sistema.

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| produto | Produto | x | Produto o qual o levantamento se refere. |
| beneficiario | Beneficiário | x | Beneficiário que forneceu o preço do produto. |
| preco | Número | x | Preço fornecido pelo beneficiário para o produto. |
| semana | Número | x | Semana do mês em que o levantamento foi realizado. |
| mes | Número | x | Mês do ano em que o levantamento foi realizado. |
| ano | Número | x | Ano em que o levantamento foi realizado. |
| autor | Texto | x | Nome do autor do levantamneto. |

## 7.3 Município

Um Município é uma subdivisão administrativa de uma Unidade Federativa.

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| nome | Texto | x | Nome do município. |
| codigo_ibge | Texto | x | Código do IBGE do município. |
| unidade_federativa | Unidade Federativa | x | Unidade Federativa a qual o município pertence. |
| beneficiarios | Beneficiário | | Produtores do município. |
| producoes | Produção | | Dados de produção referentes ao município. |
| produtos_levantamento | Produto para Levantamento | | Associação a produtos que devem participar do levantamento de preços no município. |

## 7.4 Produção

A classe Produção representa um dado de produção de um determinado produto em um município em um período.

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| quantidade | Número | x | Quantidade produzida. |
| produto | Produto | x | Produto produzido. |
| municipio | Muncípio | x | Município de produção. |
| ano | Número | x | Ano de produção. |
| mes | Número | | Mês de produção. |

## 7.5 Produto

Produtos representam items produzidos na agricultura, pecuária ou silvicultura que são de interesse para o levantamento de preços do Incaper.

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| nome | Texto | x | Nome do produto. |
| codigo_ibge | Texto | x | Código do IBGE do produto. |
| unidade_medida | Unidade de Medida | x | Unidade de medida do produto. |
| preco_minimo | Número | | Preço mínimo que o produto pode ter. |
| preco_maximo | Número | | Preço máximo que o produto pode ter. |
| produtos_levantamento | Produto para Levantamento | | Associação à municípios que devem levantar o preço do produto. |

## 7.6 Produto para Levantamento

A classe Produto para Levantamento representa a definição de um produto que deve participar do levantamento de preços em um município em um determinado ano.

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| produto | Produto | x | Produto a ser levantado o preço. |
| municipio | Município | x | Município referente ao levantamento. |
| ano | Número | x | Ano da definição. |

## 7.7 Unidade de Medida

Unidades de Medida são utilizadas como referencial para a definição e exibição de preços. São exemplos de unidades de medidas: kilograma, litro, etc.

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| nome | Texto | x | Nome da unidade de medida. |
| sigla | Texto | x | Sigla da unidade de medida. |
| produtos | Produto | | Produtos que possuem a unidade de medida |

## 7.8 Unidade Federativa

Unidades Federativas são representações dos estados do Brasil, é utilizada no sistema para identificar os municípios do estado do Espírito Santo.

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| nome | Texto | x | Nome da unidade federativa. |
| sigla | Texto | x | Sigla de 2 dígitos. |
| codigo_ibge | Texto | x | Código do IBGE da unidade federativa. |
| municipios | Município | | Municípios pertencentes à unidade federativa. |


Documento de Projeto de Sistema

# SisPreço

Registro de Alterações:

| Versão | Responsável | Data | Alterações |
|---|---|---|---|
| 1.0 | Rômulo de Angelis Vitoi | 28/06/2018 | Versão Inicial |
| 1.1 | Vítor E. Silva Souza | 29/06/2018 | Primeira revisão |
| 1.2 | Rômulo de Angelis Vitoi | 01/07/2018 | Seção Arquitetura |
| 1.3 | Vítor E. Silva Souza | 02/07/2018 | Revisão da seção Arquitetura |
| 1.4 | Rômulo de Angelis Vitoi | 03/07/2018 | Correções da seção Arquitetura |
| 1.5 | Vítor E. Silva Souza | 03/07/2018 | Revisão final |

Vitória, ES

2018

# 1 Introdução

Este documento apresenta o documento de projeto (*design*) do sistema SisPreço. Este documento está organizado da seguinte forma: a Seção 2 apresenta a plataforma de software utilizada na implementação da ferramenta; a Seção 3 trata de táticas utilizadas para tratar requisitos não funcionais (atributos de qualidade); por fim, a Seção 4 apresenta o projeto da arquitetura de software e suas subseções explicam cada uma de suas camadas.

# 2 Plataforma de Desenvolvimento

Na Tabela 1 são listadas as tecnologias utilizadas no desenvolvimento da ferramenta, bem como o propósito de sua utilização.

| Tecnologia | Versão | Descrição | Propósito |
|---|---|---|---|
| PHP | 7.1.18 | Linguagem de programação interpretada, multi-paradigma, de tipagem dinâmica. Projetada para o desenvolvimento de aplicações Web. | Desenvolvimento de aplicação em linguagem de programação de codificação rápida. |
| Composer | 1.5.5 | Gerenciador de dependências PHP. | Instalar pacotes PHP. |
| Yarn | 1.5.1 | Gerenciador de dependências Node.js. | Instalar pacotes Node.js. |
| Apache | 2.4.33 | Servidor web gratuito, de código aberto e multi-plataforma. | Servir como ponte de ligação entre requisições e respostas HTTP com os scripts PHP. |
| Laravel | 5.5.1 | Framework PHP utilizado para o desenvolvimento Web. Utiliza a arquitetura Modelo-Visão-Controlador (MVC), fornecendo estruturas padrão para banco de dados, serviços e páginas Web. | Reduzir a complexidade do desenvolvimento, implantação e gerenciamento da aplicação. |
| Bootstrap | 4.1.1 | Framework HTML/CSS/JavaScript para desenvolvimento de sites responsivos e *mobile-first*. | Melhorar a produtividade, permitindo a construção de interfaces para Web usando um conjunto de componentes pré-construídos e de caráter responsivo. |
| Vue | 2.1.10 | Framework JavaScript para desenvolvimento de interfaces Web, adapta e estende o HTML tradicional para uma melhor experiência com conteúdo dinâmico, com ligação direta e bi-direcional de dados. | Melhorar a produtividade, evitando a manipulação do DOM (*Document Object Model*, modelo de objetos que compõem os documentos HTML). |
| SASS | 4.5.3 | Extensão CSS que ajuda a reduzir problemas de repetição e manutenção de CSS tradicionais. | Melhorar a produtividade por meio da redução de repetição de código. |
| MySQL Server | 5.7.22 | Sistema Gerenciador de Banco de Dados Relacional. | Persistência dos dados manipulados pela ferramenta. |
| Redis Server | 4.0.9 | Sistema Gerenciador de Banco de Dados Chave-Valor. | Cache da aplicação. |
| Git | 2.15.1 | Sistema de controle de versão distribuído projetado para lidar com velocidade e eficiência. | Assegurar o controle de versão da aplicação. |
| Docker | 18.05.0 | Plataforma de software que permite a criação de contêineres. | Facilitar a configuração do ambiente de desenvolvimento. |
| Docker Compose | 1.21.2 | Ferramenta para criação e execução de múltiplos contêineres Docker. | Facilitar a configuração do ambiente de desenvolvimento. |

Tabela 1 – Plataforma de Desenvolvimento e Tecnologias Utilizadas

Na Tabela 2 vemos os *softwares* que apoiaram o desenvolvimento de documentos e também do código fonte.

| Software | Versão | Descrição | Propósito |
|---|---|---|---|
| JetBrains PhpStorm | 2018.1.2 | Ambiente de desenvolvimento (IDE) para a linguagem PHP. | Facilitar a atividade de implementação de software. |
| Postman | 5.5.2 | Ferramenta para a realização de requisições HTTP. | Facilitar o teste de APIs. |
| GitKraken | 3.6.4 | Interface gráfica para git. | Facilitar a manipulação e visualização do versionamento. |
| Astah Professional | 7.2.0 | Ferramenta para modelagem em UML | Modelar os diagramas de classes, casos de uso, etc. |
| Sketch | 50 | Ferramenta de desenho vetorial | Desenho de diagramas e prototipação de interfaces. |
| TeXstudio | 2.12.8 | Editor de LaTeX. | Escrever a documentação do sistema. |

Tabela 2 – Softwares de Apoio ao Desenvolvimento do Projeto

# 3 Atributos de Qualidade e Táticas

Na Tabela 3 são listados os atributos de qualidade considerados neste projeto, com uma indicação se os mesmos são condutores da arquitetura ou não e as táticas a serem utilizadas para tratá-los.

| Categoria | Requisitos Não Funcionais | Condutor da Arquitetura | Tática |
|---|---|---|---|
| Portabilidade | RNF-1, RNF-2 | Sim | Utilizar uma linguagem que seja compatível com os principais navegadores do mercado. |
| Facilidade de Aprendizado, Usabilidade | RNF-3, RNF-6 | Sim | 1 - Prover ao usuário a capacidade de entrar com comandos que permitam operar o sistema de modo mais amigáveis. Para tal, as interfaces do sistema devem permitir, sempre que possível, a entrada por meio de seleção ao invés da digitação de campos.<br>2 - Desenvolver um aplicativo para a plataforma Android. |
| Segurança | RNF-4 | Não | O sistema deverá alertar ou redirecionar o usuário quando este tentar realizar uma operação que esteja fora de alcance para o seu papel. |
| Confiabilidade | RNF-5 | Sim | Sempre que o usuário inserir informações em um formulário, o sistema irá garantir que todos os campos obrigatórios tenham sido devidamente preenchidos e que os mesmos sejam válidos, envitando qualquer tipo de incosistência. |
| Manutenibilidade | RNF-7 | Sim | Organizar a arquitetura da ferramenta segundo uma combinação de camadas e partições. |

Tabela 3 – Atributos de Qualidade e Táticas Utilizadas

# 4 Arquitetura de Software

A arquitetura de software do Sispreço baseia-se no padrão MVC (*Modelo, Visão, Controlador*). Nas próximas seções, serão apresentados diagramas no padrão proposto pelo *FrameWeb* (SOUZA, 2007) de forma adaptada ao *framework* Laravel sendo utilizado em uma aplicação de página única.

## 4.1 Camada *Model*

A seguir, são apresentados os modelos de domínio seguindo o método FrameWeb. Diferentemente da abordagem original proposta em 2007, todos os atributos que não podem ser nulos tiveram a tag `not null` omitida e aqueles que podem tiveram a tag `null` adicionada de forma a reduzir o impacto visual nos diversos diagramas. Pelo mesmo motivo, foi considerada que a estratégia padrão de recuperação de uma associação é do tipo *lazy*, e não *eager* como proposto pelo FrameWeb.

Todas as classes de domínio estendem uma classe do pacote Eloquent do Laravel, responsável pelo mapeamento entre classes e tabelas, por associações, dentre outras funcionalidades. Essa herança não é mostrada nos diagramas com o intuito de não poluí-los com várias associações, mas é possível saber mais sobre o Eloquent em <https://laravel.com/docs/5.6/eloquent>.

O modelo de domínio do sistema foi dividido em dois subsistemas pois os dados que são providos pelo Incaper são armazenados em um banco de dados diferente do Sispreço. A Figura 1 representa o modelo de domínio do subsistema Incaper.

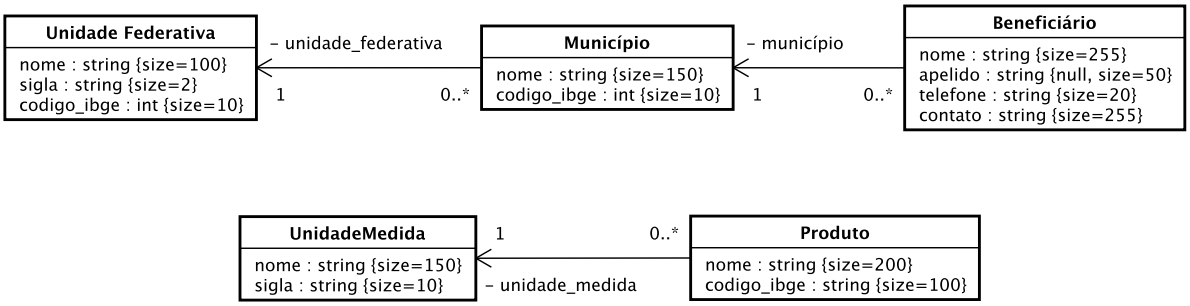



Figura 1 – Modelo de Domínio do susbsistema Incaper.

A seguir, a Figura 2 representa o modelo de domínio do subsistema Sispreço, onde as classes *Beneficiário*, *Produto* e *Município* são apenas referências para classes que existem no subsistema Incaper.

Diferente do que indica o método FrameWeb, não foram utilizadas classes DAO

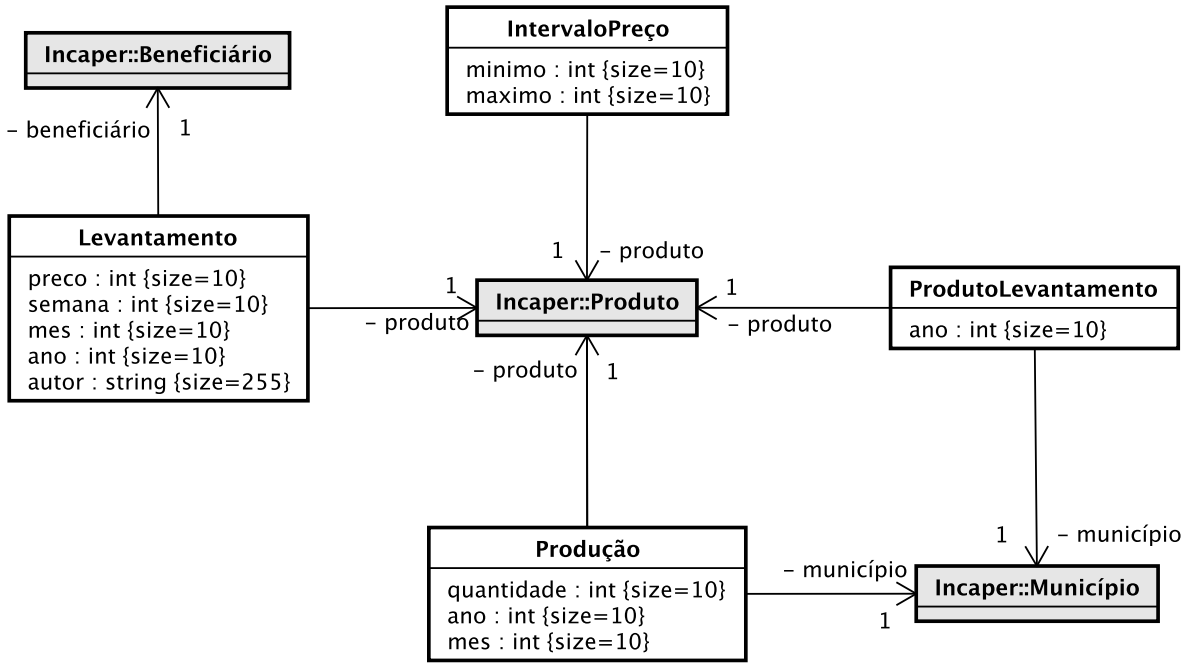



Figura 2 – Modelo de Domínio do subsistema Sispreço.

para a camada de acesso a dados, pois os modelos, como mencionado anteriormente, estendem um modelo Eloquent que já provê todas as funcionalidades necessárias para persistência dos dados, e portanto, não foram modelados os modelos de persistência. A Figura 3 apresenta a classe abstrata Eloquent, com alguns de seus métodos relevantes para a obtenção e persistência dos dados.

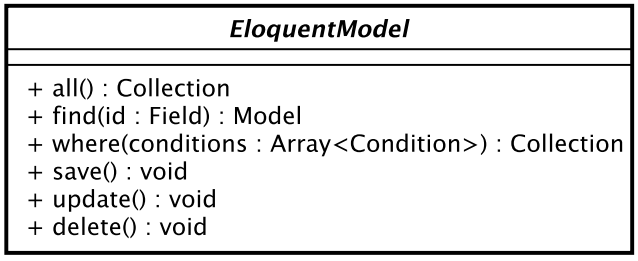



Figura 3 – Classe abstrata Eloquent.

## 4.2 Camadas de *Visão* e *Controle*

Esta seção apresenta alguns dos modelos de navegação que fazem parte da comunicação entre as camadas de *Visão* e *Controle*. Por estarmos lidando com uma aplicação de página única, foi adotada a nomenclatura `/#fragmento` para descrever qual o fragmento da página está sendo exibido.

A Figura 4 apresenta o modelo de navegação para os fluxos *Acessar relatórios* e *Download de relatórios* do caso de uso *Acessar relatórios*.

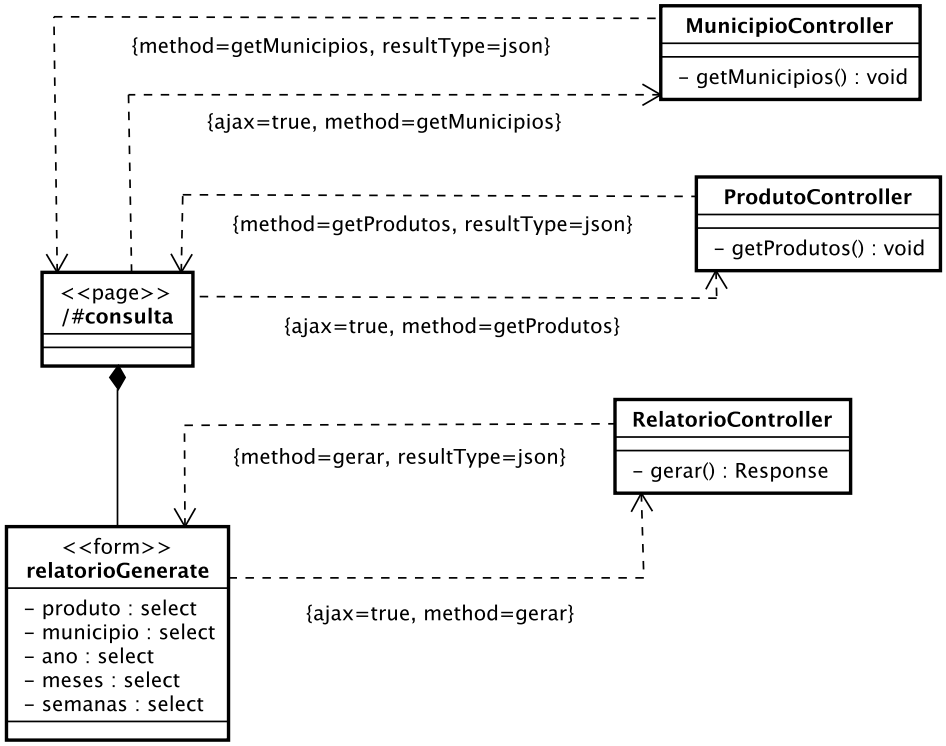



Figura 4 – Modelo de Navegação - Visualização de relatórios.

A Figura 5 exibe os modelos de navegação para os fluxos *Visualizar intervalos de preços* e *Definir intervalos de preços* do caso de uso *Gerenciar intevalos de preços* e o fluxo *Realizar levantamento de preços* do caso de uso *Realizar levantamento de preços*.

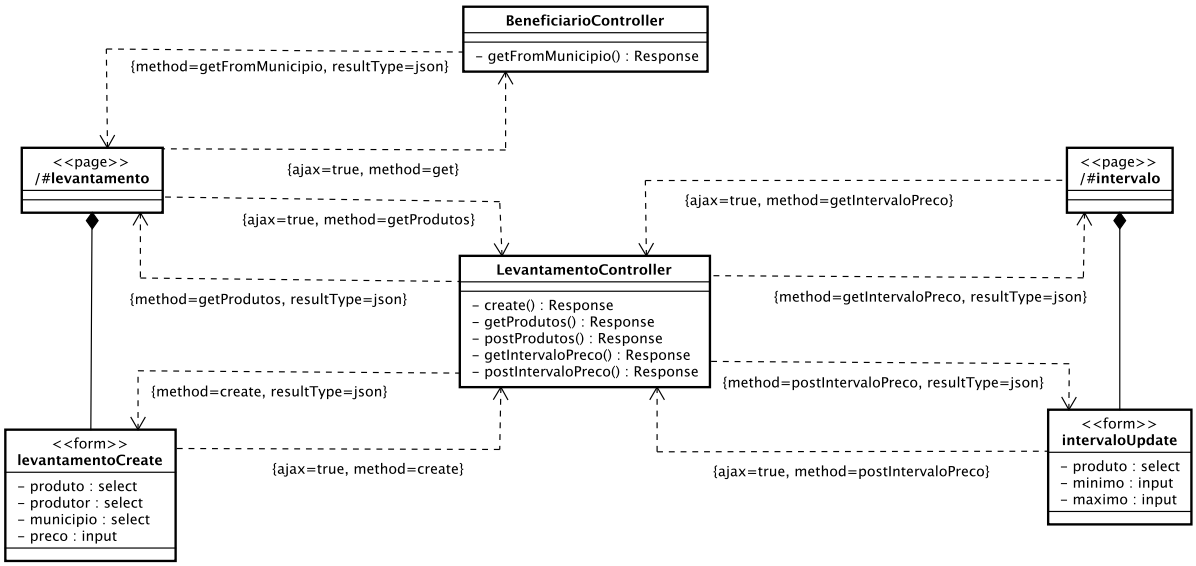



Figura 5 – Modelo de Navegação - Levantamento e definição de intervalo de preços.

Por fim, a Figura 6 apresenta o modelo de navegação para o fluxo *Importar dados de produção* do caso de uso *Importar dados de produção*.

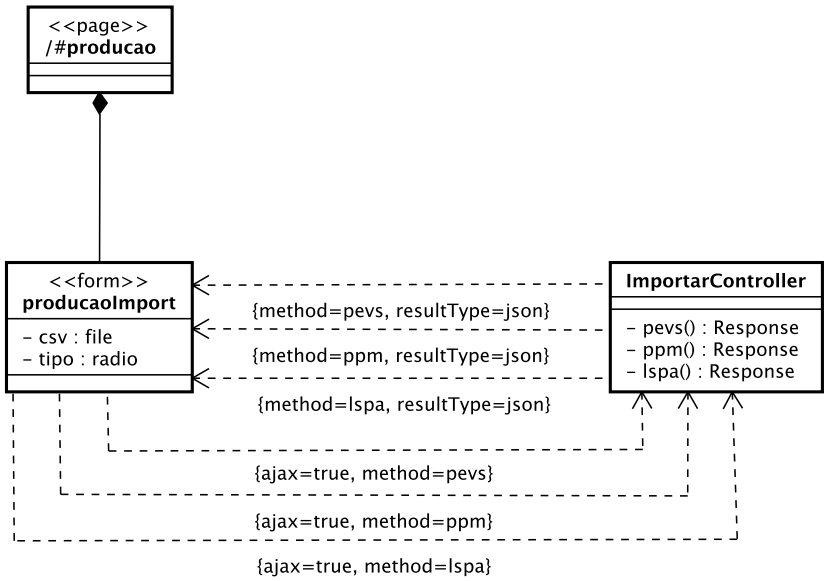



Figura 6 – Modelo de Navegação - Importação de dados de produção.

# Referências

SOUZA, V. E. S. *FrameWeb – A Framework-based Design Method for Web Engineering*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2007. Citado na página 5.