**DataViva**

Visão e Escopo

V0.1

Núcleo de Sistemas e Gestão

Fevereiro 2013

Histórico de Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| DATA | VERSÃO | DESCRIÇÃO | AUTOR |
| 29/07/2013 | 1.0 | Criação do documento | Mario H. Mol |
| 19/06/2013 | 1.1 | Alteração da Arquitetura | Mario H. Mol |
| 31/10/2013 | 1.0 | Versão base | Mario H. Mol |

ÍNDICE

[1. INTRODUÇÃO 4](#_Toc371440614)

[2. ARQUITETURA 5](#_Toc371440615)

[2.1. Camada de Visão 7](#_Toc371440616)

[2.2. Banco de Dados 9](#_Toc371440617)

[3. DESENVOLVIMENTO 11](#_Toc371440618)

[3.1. Equipe 11](#_Toc371440619)

[4. PRODUÇÃO 13](#_Toc371440620)

[4.1. Servidor de Aplicação 13](#_Toc371440621)

[4.2. Equipe 14](#_Toc371440622)

[5. API 15](#_Toc371440623)

INTRODUÇÃO

Está sendo implantado em Minas Gerais o *Project Space* (PS), em parceria com a *Growth Ventures* (GV) e o Governo do Estado de Minas Gerais. Criado pelos professores Ricardo Hausmann (Universidade de Harvard) e Cesar Hidalgo (Instituto de Tecnologia de Massachusetts), no ano de 2007, o Product Space é um conjunto de ferramentas, conceitos e metodologia que tenta explicar o desenvolvimento desigual de estados ou regiões e criar instrumentos para que esse processo seja feito de maneira mais uniforme.

Como insumo desta análise temos informações do Estado de Minas Gerais e do Governo Federal Brasileiro, como dados de Exportação do *Ministério do Desenvolvimento, Indústria e Comércio* (MDIC), de estabelecimentos e empregados disponibilizado pelo *Ministério do Trabalho e Emprego* (MTE), e informações de municipalidade fornecido pelo *Instituto Brasileiro de Deografia e Estatística* (IBGE).

A estrutura produtiva é definida pelo conjunto de habilidades específicas (capital, trabalho, tecnologia, instituições, infraestrutura, existência de relações sociais) que possuem. O conjunto de habilidades necessárias para a produção de bens e serviços é que gera o nível de sofisticação dos mesmos. Assim é possível que os formuladores de políticas públicas, industriais e tecnológicas possam ter uma ferramenta de análise crítica para trazer prosperidade a essas localidades.

A equipe parceira, a *Growth Ventures*, tem todo o conhecimento teórico para realizar cálculos e obter estes resultados. Eles estão construindo também uma aplicação Web para compartilhar informações de forma rápida, além de gráficos para consulta aberta ao público. Por fim será construída também uma *API* para integração de outras plataformas com esta, ou seja, a possibilidade de outros sites apresentarem também informações do DataViva.

Todos estes programas, para calcular, Web e *API*, serão disponibilizados para o Escritório de Prioridades Estratégicas, que é o órgão responsável pela implantação deste projeto no Governo do Estado. Para que este escritório possa receber esta tecnologia, para então ter como passar para outra empresa ou área dentro do Estado, é necessário um estudo e análise destas entregas.

Neste documento teremos então uma análise técnica de toda a solução tecnológica proposta para o projeto, com informações para que uma equipe de desenvolvimento possa realizar manutenção e evolução. Será contemplado também informações técnicas para a equipe de Infraestrutura, ou seja, a ser utilizado como um guia na instalação de servidores ou até mesmo para o Gestor de TI ter uma orientação na contratação de serviços para dar continuidade a este projeto.

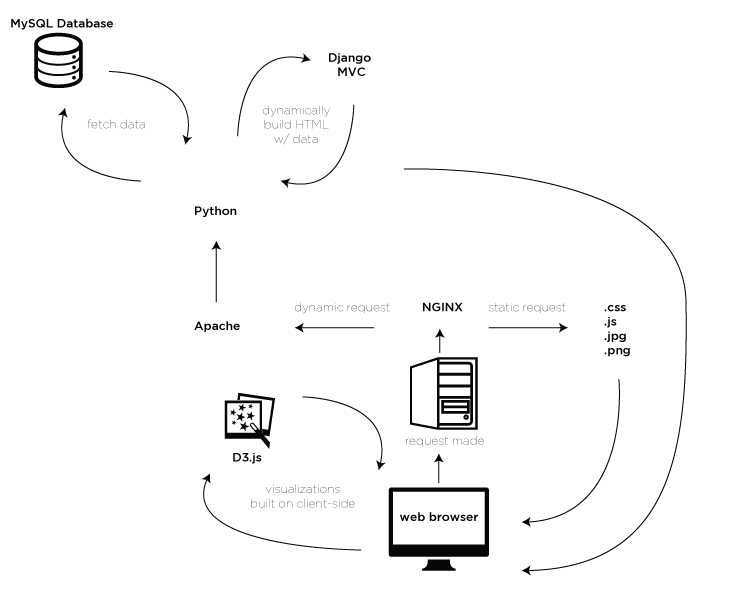
O objetivo deste documento não é detalhar a fundo todos os temas abordados aqui, pois o este é um documento voltado principalmente para gestores, que não tem necessariamente um conhecimento técnico de TI. Existem documentos em anexo que, são apontados neste documento, com uma profundidade maior nos temas técnicos.

ARQUITETURA

A arquitetura de um sistema é a definição da sua estrutura base, ou seja, responsável por viabilizar o objetivo proposto com uma boa produtividade de desenvolvimento. O objetivo desta arquitetura é possibilitar a construção de um sistema Web para exibir de forma rápida e interativa informações que a metodologia do *Product Space* pode produzir.

Este sistema deve ser capaz também de prover uma interface de comunicação com outros sistemas, seja para atualização dos dados, para exportação de gráficos para outros sites ou até mesmo a exportação das informações em seu formato original, com intuito de servir como insumos para processos de outros sistemas.

Vamos começar exibindo uma visão macro de como a arquitetura foi montada e durante este capítulo teremos informações mais detalhadas:



**Figura 1** – Desenho da arquitetura do sistema.

Para este projeto foi utilizado a arquitetura *Flask* Versão 0.10, que proporciona uma boa produtividade no desenvolvimento e performance de acesso para uma grande número de visitantes. Quando dizemos que um sistema utiliza uma arquitetura quer dizer que ele foi construído como uma aplicação compatível com esta arquitetura, no caso deste projeto foi utilizado o *Flask*. Isto quer dizer que o sistema construído desta forma não funciona por sí só, ou seja, para que ele funcione é necessário que seja instalado também a arquitetura utilizada. Existem diversos benefícios que uma arquitetura proporciona no desenvolvimento de um software, mas pensando em um objetivo macro a arquitetura dá reaproveitamento de código, isto é, existem diversas funcionalidades no *Flask* que é comum a muitos sistemas e assim não é necessário desenvolver novamente, basta utilizar algo já pronto. Para ajudar a entender vamos exemplificar pensando que seu sistema precisa de um mecanismo para os usuários informem seu nome e senha. Ao invés de desenvolver esta funcionalidade, basta utilizar o mecanismo que já existe na plataforma *Flask*.



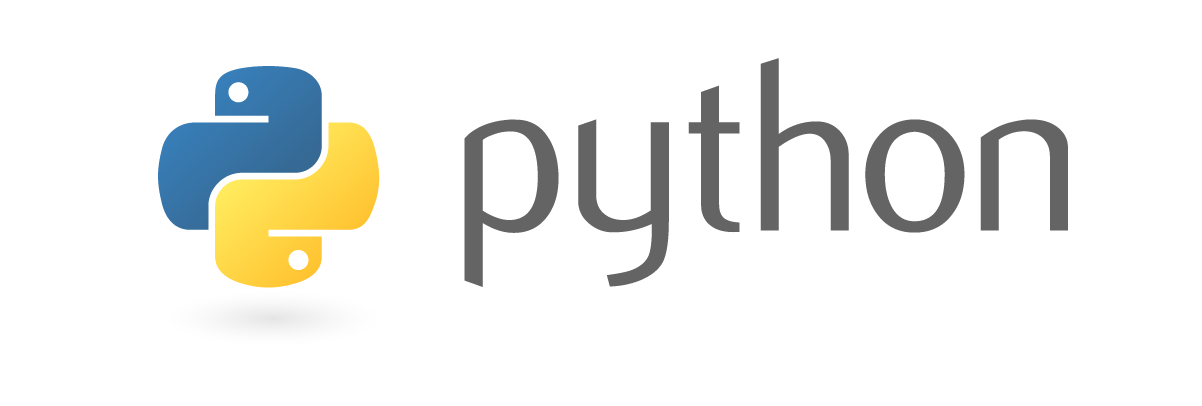
**Figura 2** – Logomarca do principal framework utilizado nesta solução.

Para mais informações em acesse o site do fornecedor:

* http://flask.pocoo.org/

A linguagem em que o Flask foi construído é o *Python* na versão 2.7.3. Sendo assim podemos considerar que a principal linguagem de programação utilizada no PS é o Python, para mais informações acesse o site do fornecedor:

* https://www.python.org /



**Figura 3** – Linguagem utilizada na construção do sistema.

Sistema operacional pode ser tanto *Windows* ou baseados em *Linux*, como por exemplo o *Debian* e o *Ubuntu*. Abaixo temos uma análise de qual sistema operacional utilizar para determinados casos.

* 1. Camada de Visão

A camada de visão de uma arquitetura é a parte responsável por criar a visualização do sistema, ou seja, gráficos, formulários e listagem de dados na tela. No desenho deste projeto a arquitetura é construída em duas partes:

* Primeiro é gerado no *Server-side*, por meio do *Flask*, uma interface nos padrões Web.
* Relatórios e infográfos são construídos no client-side, através do *D3*.

A criação de formulários para preenchimento, listagem de dados e a estrutura de layout do sistema são realizados por código que está junto com a arquitetura *Flask*, mas utilizando padrões Web convencionais, como as tecnologias abaixo:

* *HTML*
* *JavaScript*
* *CSS*
* *SVG*

Neste projeto temos gráficos muito bem elaborados e diferentes do convencional, construídos com uma nova e moderna plataforma chamada *Data Driven Document* (*D3).* Esta plataforma constrói os gráficos no Client-Side*,* diferente de boa parte dos componentes de mercado que realizam a construção do gráfico no Server-Side.

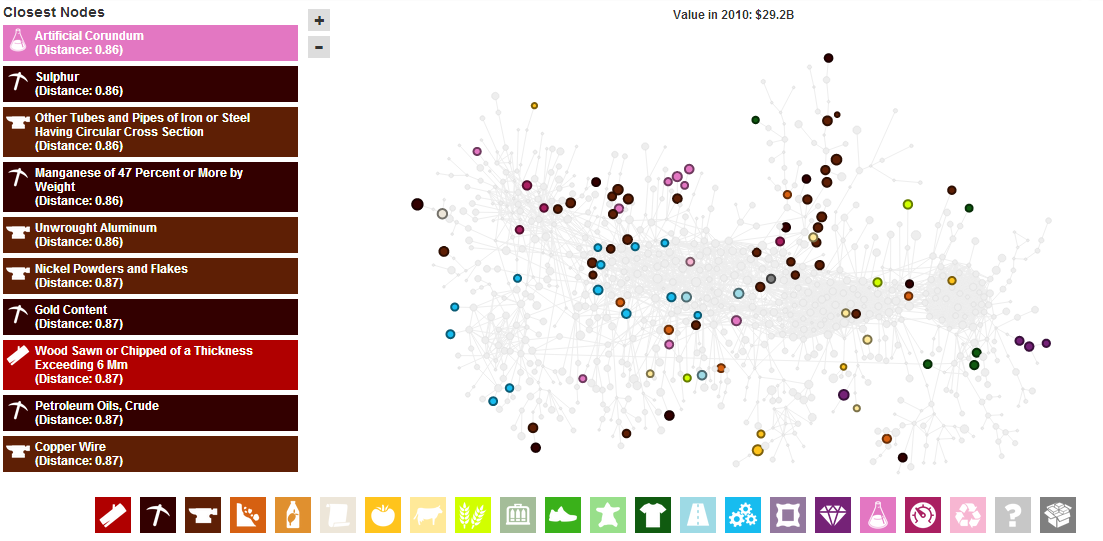
* <http://d3js.org/>

Este componente utiliza as seguintes tecnologias:

* *Javascript* 
  + Componentes de Terceiros
* *SVG*
  + Tecnologia para exibição e animação de formas geométricas
* *HTML*
  + Os gráficos são realizados no próprio HTML no navegador, sem utilização de componentes de terceiros, como por exemplo o *Adobe Flash*.

Os gráficos utilizados nesta solução estão listados abaixo, com exemplos e também com o link no site oficial do componente com mais informações de como funciona e de como desenvolver.

* Gráfico Network
  + Componente FishEye
  + <http://bost.ocks.org/mike/fisheye/>



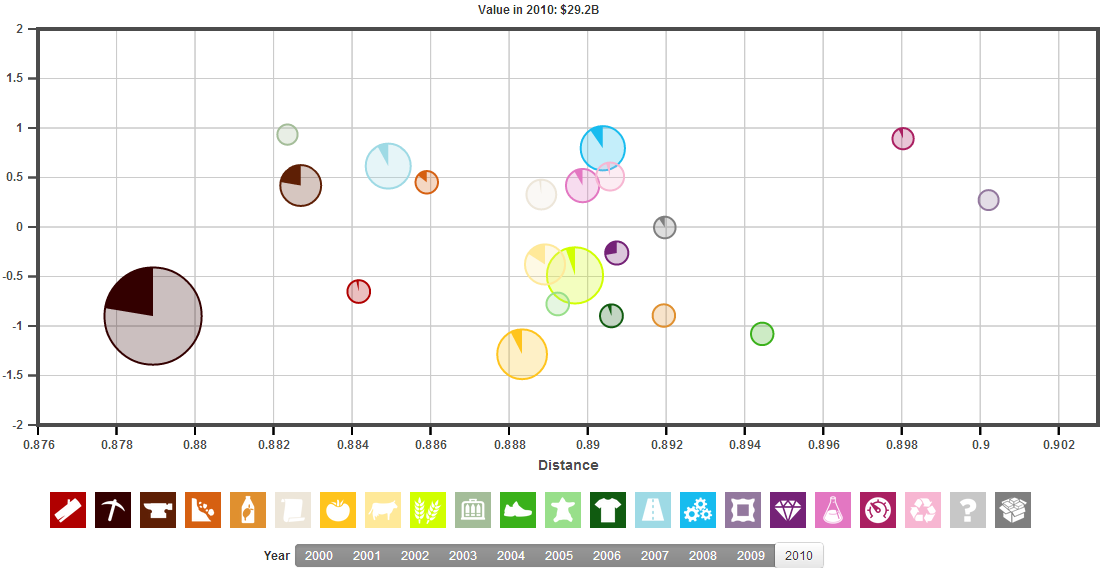
**Figura 4** – Exemplo de gráfico do tipo *Network*.

* Gráfico Treemap:
  + Componente Zoomable Treemap
  + http://mbostock.github.com/d3/talk/20111018/treemap.html



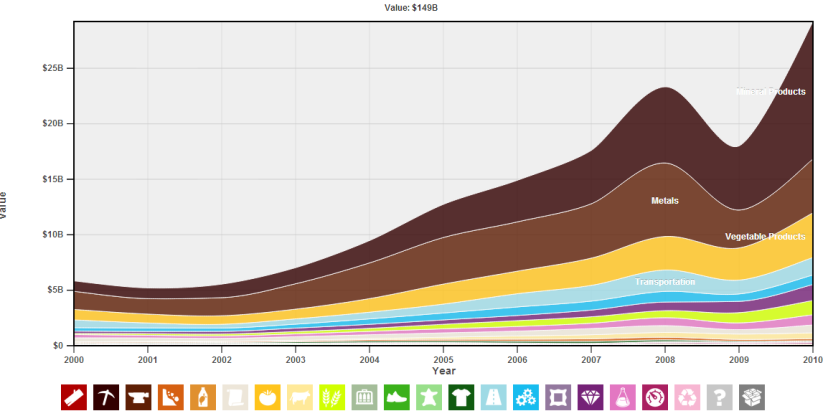
**Figura 5** – Exemplo de gráfico do tipo TreeMap.

* Gráfico ScatterPlot:
  + Componente Scatter Plot :
  + <http://bl.ocks.org/mbostock/3887118>



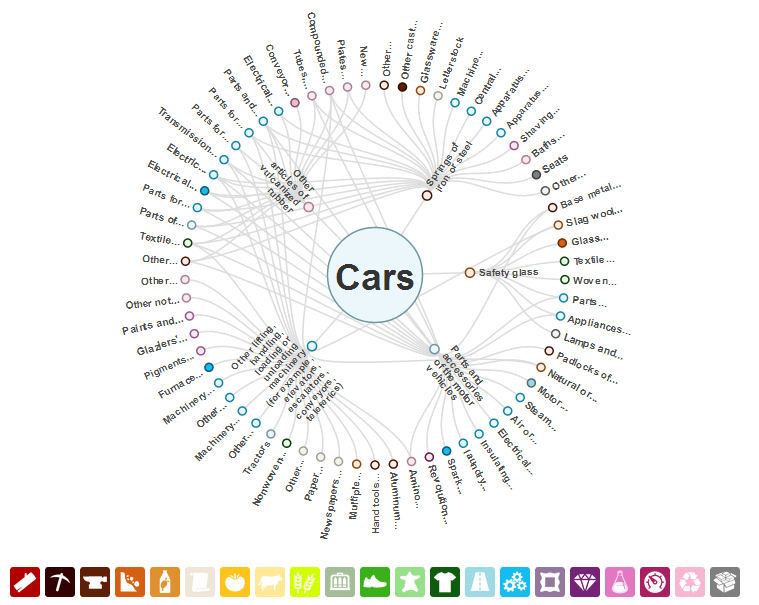
**Figura 6** – Exemplo de gráfico do tipo ScatterPlot.

* Gráfico Stacked
  + Componente Stacked Area Chart
  + http://bl.ocks.org/mbostock/3885211



**Figura 7** – Exemplo de gráfico do tipo Stacked.

* Gráfico Cluster
  + Componente Cluster
  + http://mbostock.github.io/d3/talk/20111018/cluster.html



**Figura 8** – Exemplo de gráfico do tipo Cluster.

* 1. Banco de Dados

O banco de dados utilizado neste projeto é o *MySQL Community Server 5.6* e nele tem a estrutura e os dados utilizados pelo PS. Este sistema de banco de dados é largamente utilizado no mercado, inclusive com exemplos de aplicações de grande porte. Portanto é uma ferramenta com muita documentação e alta confiabilidade. Para mais informações acessar a página do fabricante:

* http://www.mysql.com



**Figura 8** – Sistema de banco de Dados utilizado nesta solução.

Existem também algumas tabelas de sistema no banco de dados, como por exemplo para autenticação de usuário. O tamanho esperado desta base de dados, com todas as informações atualizadas, é de 100 Gb. A base é relativamente grande, mas não a ponto de ser considerada um “*Big Data*”. O importante é que, como teremos apenas visualizações de informações, o banco de dados será otimizado para atender a esta demanda. A atualização das informações no banco de dados deve ocorrer em uma periodicidade semestral, portanto não tem problema que seja demorado para atualizar os dados, desde que a consulta às informações seja rápida.

DESENVOLVIMENTO

Consideramos como ambiente todos os preparativos de máquina, processos, ferramentas e documentação para que o produto possa ser instalado para consulta de usuários ou para que o desenvolvedor possa trabalhar em alterações.

Neste caso o desenvolvedor utilizará uma máquina Windows, já que é o padrão da estação de trabalho para o Governo de Minas. Existe um documento de Ambiente de Desenvolvimento para este projeto que fala em ferramentas utilizadas para suportar este produto e também descreve os procedimentos de instalação na máquina de um desenvolvedor.

Para viabilizar um ambiente de desenvolvimento mais produtivo é necessário ter um servidor onde seja instalado ferramentas auxiliares, como:

* Repositório de arquivos
  + Neste projeto é utilizado a tecnologia GIT (<http://git-scm.com/>), inclusive com uma interface Web para gerenciamento;
* Gestão de desenvolvimento
  + Ferramenta que auxilia na gestão, onde é possível criar atividade, gerir seu ciclo de vida e associar para determinadas pessoas da equipe. Utilizamos hoje o Mantis Bug Tracker (http://[www.mantisbt.org](http://www.mantisbt.org));
* Geração de versão para instalação
  + Com o código fonte do sistema é necessário realizar alguns procedimentos técnicos para então gerar a versão de instalação do sistema, chamamos este processo de *Build*. No caso a ferramenta que estamos utilizando é o GITLAB (<http://gitlabhq.com/>).

O acesso a estas ferramentas pelo Escritório de Prioridades Estratégicas pode ser realizado através de um servidor próprio e existem mais informações no documento de ambiente de desenvolvimento.

* 1. Equipe

A equipe de desenvolvimento necessária para dar continuidade neste projeto deve ter algumas competências em específico e com diferentes níveis de conhecimento em cada área. Abaixo listo quais são os conhecimentos necessários e o nível de proficiência recomendado:

* *HTML*
  + Nível Intermediário: Esta tecnologia é exigida em todo sistema Web, portanto não tem como trabalhar sem um pré-requisito de conhecimento intermediário de *HTML*.
* *CSS*
  + Nível Básico: Sistema foi construído de forma muito simplista e não tem grande complexidade no *CSS*.
* *Javascript*
  + Nível Avançado: Existe uma integração entre os dados montados pela plataforma *Flask* com a criação de gráficos com o componente D3. Por isso é necessário um conhecimento acima do básico neste projeto.
* *Python*
  + Nível Intermediário: Como a grande parte do código fica na plataforma *Flask*, inclusive as interfaces de integração com outros sistemas, é necessário que tenha conhecimento nesta linguagem. Mas os códigos que estão escrito no *Flask* são até fáceis de entender, por isso o nível requisitado aqui é intermediário.
* *MySQL* 
  + Nível Intermediário: Necessário saber dar manutenção em consultas um pouco mais complexas, por isso a necessidade acima do básico.
* *Flask*
  + Nível Intermediário: Como a arquitetura já está montada, com exemplos de telas funcionando e um guia de desenvolvimento facilita muito o trabalho, por isso a necessidade de conhecimento Básico
  + Se for para ter grandes evoluções neste sistema, recomenda-se uma pessoa na equipe com maiores conhecimento nesta arquitetura, portanto neste caso um nível intermediário passa a ser necessário.

PRODUÇÃO

No ambiente de produção temos o desenho da infraestrutura necessária para disponibilizar ao público o acesso ao sistema. Diferentemente do ambiente de desenvolvimento, onde não há uma preocupação com performance por ser utilizado apenas pelo desenvolvedor, no ambiente de produção é necessário pensar no volume de visitantes. O volume de visitantes tem influência direta nas ferramentas e máquinas utilizadas para montar este ambiente.

Como servidor de aplicação Web deve ser utilizado o *NginX*, que parece muito com o mais popular *Apache*, mas com grande ganho de performance e facilidade de configuração.

A equipe de desenvolvimento da GV utiliza o Ubuntu Server, mas utilizamos o Windows como ambiente de desenvolvimento por ser o padrão na Cidade Administrativa. Diferente do ambiente de produção, que será descrito para utilizar em Linux, termos aqui um cenário totalmente diferente para configuração do ambiente de desenvolvimento.

* 1. Servidor de Aplicação

Será utilizado o Ngnx para realiza a requisição com o navegador, fornecendo apenas conteúdo estático e realizando um Proxy para arquivos dinâmicos para o Apache resolver.



**Figura 9** – E Servidor de Aplicação Web utilizado em desenvolvimento.

Para servidor de aplicação será utilizado o NginX:

* http://nginx.org/en/download.html



**Figura 10** – Servidor de Aplicação Web utilizado em produção.

Existe uma forma de conectar o Flask diretamente com o Nginx, mas foi optado pela equipe da GV o uso do Apache com o Flask.

* https://code.djangoproject.com/wiki/DjangoAndNginx
  1. Equipe

A equipe de infraestrutura necessária para dar manutenção no ambiente de produção deste projeto deve ter algumas competências em específico e com diferentes níveis de conhecimento em cada área. Abaixo temos quais são os conhecimentos necessários e o nível de proficiência recomendado:

* LINUX – Red Hat Server
  + Nível Intermediário: Conhecimento necessário para dar manutenção normal em um servidor Linux, considerando acompanhamento de métricas de uso de memória RAM, espaço físico e tamanho de log’s. Para criar novos servidores existe a necessidade de conhecimento intermediário para instalação e configuração de alguns módulos.
* Apache
  + Nível Básico: Configuração de um Server simples e também com documentação para a montagem.
* MySQL
  + Nível Intermediário: Banco de dados com um grande volume de informações pode necessitar de intervenção de uma pessoa com perfil mais de DBA, para ajudar a otimizar as consultas e as informações.

API

Chamamos de *Application Programming Interface* (*API*) uma interface de comunicação que um sistema abre para que outros, sistema ou pessoa, possam interagir. Por exemplo, os *Correios* têm um sistema de consulta de CEP que pode ser feito através de um formulário no site. Mas se você tem um sistema com um cadastro de clientes e deseja validar o *CEP* do cliente, seu sistema pode utilizar a API do Correios para que faça isso automaticamente, sem a necessidade de uma intervenção humana.

Neste caso será utilizado este conceito de API para que:

* Obter dados de relatórios formatados para exibição ao usuário final com informações do DataViva. Esta funcionalidade pode ser utilizada por:
  + Outros sites do Governo do Estado
  + Sistemas parceiros
* Exportar os dados da tabela da maneira como eles estão, sem um tratamento de visualização como no caso acima.
* Importar novas informações atualizando a base de dados, utilizado provavelmente duas vezes ao ano.
  + Este caso é indicado para quando tivermos novos dados coletados pelos órgãos governamentais

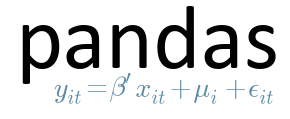
A tecnologia adotada para realizar a construção desta API é por meio de WebService REST, que trabalha com o padrão de comunicação de internet e pode ser utilizada por qualquer outra tecnologia. Ou seja, independente da tecnologia utilizada no sistema que está consumindo esta API, deverá funcionar sem mais problemas.

PROCESSO DE CÁLCULO

O processo de cálculo é realizado inteiramente por meio da linguagem Python. O framework Flask não é utilizado neste momento, até porque esta arquitetura é utilizada para criação de aplicações Web, e não para realizar cálculo.

Existem bibliotecas específicas em Python para viabilizar a criação de processos de cálculos complexos, os principais são:

Pandas é um framework que facilita o tratamento, manipulação e visualização de dados estruturados. Seu maior diferencial é ter uma alta performance para um grande volume de dados.



**Figura 11** – Pandas – Python Data Analysis Library.

NumPy é um framework para realizar cálculos em dados que contêm várias dimensões.



**Figura 12** – NumPy – Base N\_dimensional package.

Outra biblioteca muito importante para o processo de cálculo é SciPy, que contém métodos de cálculo científicos.



**Figura 13** – SciPy – Fundamental library for scientific computing.