### Arquitetura do Sistema

A arquitetura do REV3RSO está contida em três modelos básicos que se segue:

**Cliente-Servidor:**

Modelo computacional onde utiliza-se uma rede de computadores para se ter o melhor desempenho e separar camadas de negócio da apresentação, servidor e cliente. Cada instância de um cliente pode enviar requisições de dado para algum dos servidores conectados e esperar pela resposta. Por sua vez, algum dos servidores disponíveis pode aceitar tais requisições, processá-las e retornar o resultado para o cliente.

O servidor executa um ou mais programas que partilham os seus recursos com os clientes.

O cliente não compartilha de seus recursos, mas solicita o conteúdo de um servidor ou um serviço.

**Modelo em três camadas:**

Recebe esta denominação quando um sistema cliente-servidor é desenvolvido retirando-se a camada de negócio do lado do cliente. Isso faz com que as respostas sejam mais rápidas nas requisições, excelente desempenho tanto em sistemas que rodam na Internet ou em intranet e mais controle no crescimento do sistema.

A arquitetura do sistema REV3RSO é uma arquitetura cliente-servidor de três camadas (cliente, servidor aplicação e servidor de banco de dados).

Os requisitos para funcionamento do sistema devem respeitar:

**Cliente**

Web browser Firefox, Google Chrome ou Internet Explorer.

**Servidor de aplicação**

Apache 2 ou superior.

PHP 5 ou superior.

**Servidor de Banco Dados**

Mysql 5 ou Maria DB 5

Todos acima devem estar conectados em rede**.**

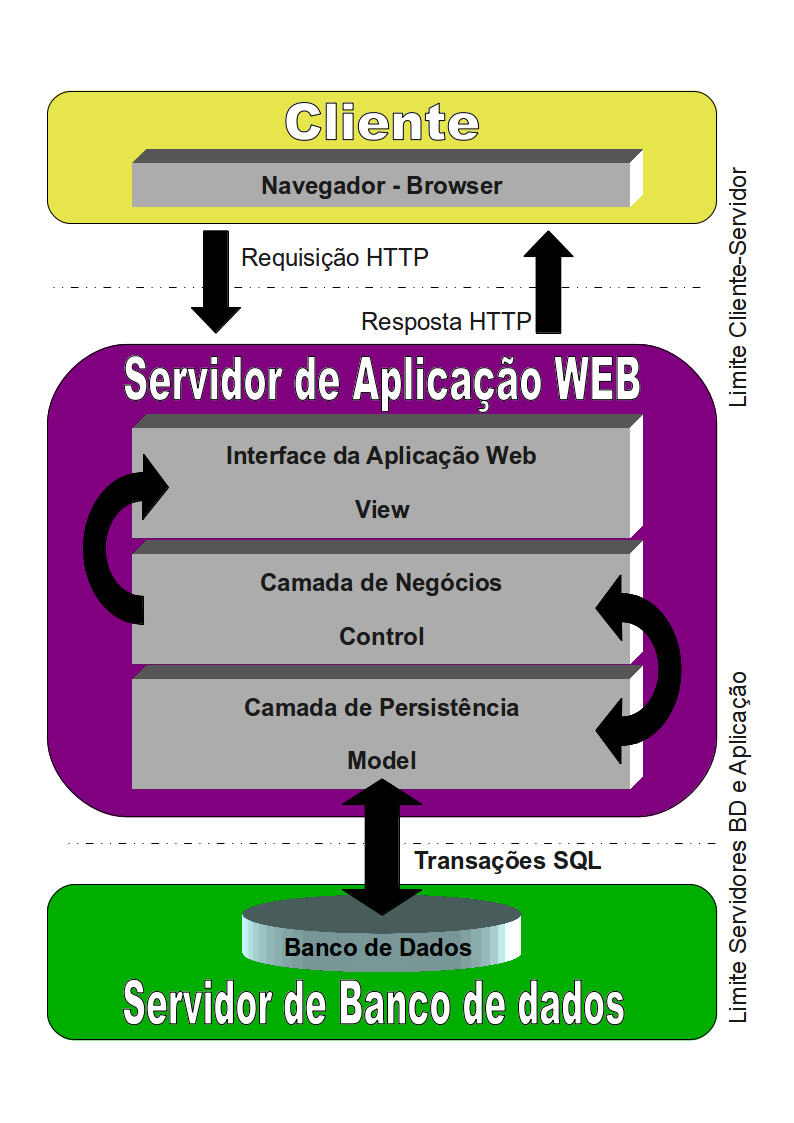
****

Figura 1 - Arquitetura do Sistema

### Tecnologias utilizadas do Sistema

A aplicação fora desenvolvida em PHP num modelo MVC (model, view e controller) que é um padrão de desenvolvimento onde a lógica é separada dos dados e da interface do software.

Além de dividir a aplicação em três tops de componentes, o desenho MVC define as interações entre eles.

* Um controlador (controller) pode enviar comandos para sua visão associada para alterar a apresentação da visão do modelo (por exemplo, percorrendo um documento). Ele também pode enviar comandos para o modelo para atualizar o estado do modelo (por exemplo, editando um documento).
* Um modelo (model) notifica suas visões e controladores associados quando há uma mudança em seu estado. Esta notificação permite que as visões produzam saídas atualizadas e que os controladores alterem o conjunto de comandos disponível. Uma implementação passiva do MVC monta estas notificações, devido a aplicação não necessitar delas ou a plataforma de software não suportá-las.
* A visão (view) solicita do modelo a informação que ela necessita para gerar uma representação de saída. foi utilizado com o objetivo de dividir as responsabilidades de cada parte do código e garantir organização ao projeto, aumentando sua escalabilidade e garantindo uma manutenção mais rápida e eficiente.