**SEI**

**Documento de Arquitetura**

**Para WebServices**

**Versão do Documento 1.0**

Data de Publicação: 21/03/2016

Histórico da Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versão | Descrição | Autor |
| 21/03/2016 | 1.0 | Criação do documento | MBAMobi – Paulo Borges |

# Introdução

Esse documento tem como objetivo fornecer uma visão geral de arquitetura de sistema apresentando elementos técnicos utilizados na construção de Webservices que utilizam a arquitetura REST Level 1 para o projeto SEI.

# Metas e Restrições de Arquitetura

## Metas

A meta principal da Arquitetura deste software é proporcionar entendimento comum entre os desenvolvedores e desse modo designar a estrutura de projetos para inclusão de WebServices.

Para a proposta da arquitetura, foram considerados fatores como a finalidade do sistema, tipo de acessos e ambiente de execução. Sendo assim, a arquitetura a ser adotada precisa atender às seguintes características:

* Disponibilidade: o sistema deve ser desenvolvido através de uma API (Interface de Programação de aplicações), com disponibilidade dos principais recursos da aplicação para acessos externos e internos, de acordo com a política de segurança do software.
* Reusabilidade: a arquitetura do sistema deve permitir a utilização de classes e componentes em outros projetos, favorecendo o tempo de produção e a qualidade do produto gerado.
* Integração: o sistema deve ser facilmente integrado a outros sistemas ou dispositivos através da API.

## Restrições

As restrições aqui reportadas são impostas pelo ambiente de operação do software e também pela característica e criticidade da aplicação.

* Utilização da linguagem de programação PHP a partir da versão 5.4.0, o recomendado é a versão 5.6.0 ou superior;
* Utilização do framework Zend Framework na versão 2.3.x;
* Utilização da arquitetura REST (Representational State Transfer) level 1.

# Requisições, Respostas e Especificações

## API (Application Programming Interface)

A API é uma forma de comunicação via protocolo HTTP. A API especifica uma forma de expor serviços via URIs, através de métodos HTTP.

Através das APIs os recursos do software são acessados via requests e enviam responses para cada solicitação.

Entre diversas formas de se acessar as APIs iremos utilizar o padrão REST(Representational State Transfer) Level 1.

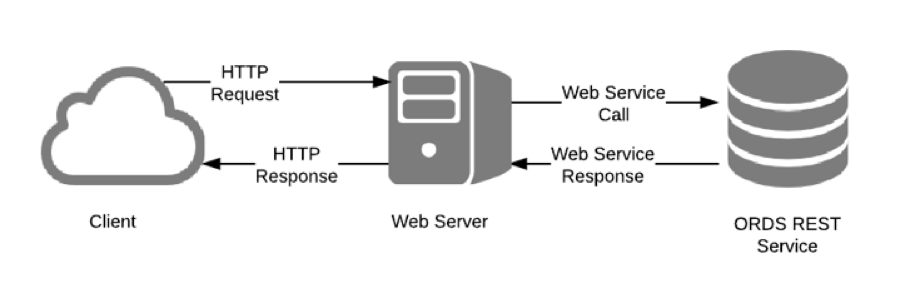
## REST (Representational State Transfer)

É uma abstração da representação www. Trata os itens disponíveis da aplicação como recursos que podem ser acessados via API, mais conhecido como RESTFull.

O software é baseado nessa representação de dados, para facilitar o acesso aos usuários.

## Requisições (Requests)

Segue uma represenção resumida do ciclo de vida da requisição na arquitetura REST:



Todas os cabeçalhos das requisições da API possuem características relevantes para o envio da resposta.

No padrão REST, os métodos HTTP deixam implicitas as ações a serem realizadasna API.

Existem 9 (nove) tipos de métodos para requisições HTTP, são eles: *GET*, *HEAD*, *POST*, *PUT*, *DELETE*, *TRACE*, *OPTIONS*, *CONNECT*, *PATCH*.

Para a API nesta arquitetura serão considerados somente 3 tipos de método: *GET*, POST e *PUT*.

O método OPTION não deve ser permitido devido a privacidade de informações desta API.

Todas as requisições são enviadas com base em um recurso disponibilizado pela API. Basicamente os recursos simbolizam uma entidade de valor para o sistema:

- Entidade Anotação => Identificador (http://ws-sei.mec.gov.br/anotacao);

- Entidade Usuário => Identificador (http://ws-sei.mec.gov.br/usuario);

- Entidade Processo => Identificador (http://ws-sei.mec.gov.br/processo);

As ações realizadas nas requisições seguem o seguinte padrão com base nos métodos HTTP:

*Método GET*

* O método GET será utilizado para listagem ou consulta de recursos.

*Método POST*

* O método POST será utilizado para submissão de dados e autenticação.
* No processamento da requisição para inclusão de dados devem ser realizadas a filtragem e validação de dados.

*Método PUT*

* O método PUT será utilizado para edição de dados.
* No processamento da requisição para edição de dados devem ser realizadas a filtragem e validação de dados.

## Respostas (Response)

As respostas devem seguir a especificação Level 1 do Modelo de Maturidade Richardson para o padrão REST ( <http://martinfowler.com/articles/richardsonMaturityModel.html> ).

Na API iremos utilizar como padrão para comunicação e retornos o mediatype JSON.

O Mediatype para JSON será application/json

## Tratamento de erros (Response)

Como padrão para respostas será utilizado o formato JSON com o mediatype application/json.

Para toda exceção ou erro devem ser disponibilizados os parâmetros:

* status: Código HTTP para status [(lista de status HTTP);](https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)
* message: Detalhes da exceção ou do erro.

## Versionamento de API

Não haverá versionamento para esta API por se tratar de uma versão estática que não será modificada. Posteriormente a API será retirada de dentro do projeto para se tornar um projeto a parte.

# Visão de Integração

## Autenticação

A autenticação será realizada através de uma chamada SOAP ao Sistema Eletrônico de Informações (SEI) que está incluída dentro da plataforma de desenvolvimento.

# Visão Lógica

## Visão de Componentes

|  |  |  |
| --- | --- | --- |
| **Componente** | **Versão** | **Referência** |
| Zend Framework 2 | 2.3.x | <http://framework.zend.com> |
| SEI | v1.0.0 | https://softwarepublico.gov.br/social/sei |

**Tabela 1 - Lista de componentes**

## Camadas

O modelo a ser utilizado na aplicação é o MVC (Model, View e Controllers).

A divisão de camadas devem ser realizadas para organizar as responsabilidades de acordo com os seus contextos.

# Visão de Processos de Sistema

O ponto focal na construção e uso da visão de processos são as questões de simultaneidade, o tempo de resposta, o *deadlock*, a taxa de transferência de dados, a tolerância a falhas e a escalabilidade.

# Visão de implantação

Não se aplica

# Visão de Solução de Requisitos

Não se aplica.

## Auditoria

Não se aplica.

# Deployment

Os ambientes para deployment devem atender aos mínimos requisitos descritos nas restrições de software.

O deploy será realizado pela ferramenta Jenkins.

São 4 ambientes de publicação:

* Desenvolvimento
* Teste
* Treino
* Producão (sendo que a publicação em produção é realizada somente pela equipe de infraestrutura do cliente).

Toda a implantação de ambientes de deploy é de responsabilidade do cliente.

# Segurança

## HTTPS

Não será incluída nesta versão o HTTPS.

## Criptografia de Requests e Responses

Devido a solicitações da área gestora, não haverá codificação e criptografia das requisições.

## Rate Limit (limitação de requisições)

Não haverá limitação na utilização desta API.

# Riscos

## Testes de carga

Os testes de carga da aplicação, para serem validados devem ser realizados em ambiente de produção, e a responsabilidade de realização dos testes de carga são do cliente.

## Deployment

O ambiente de deploy deve ser configurado corretamente com as configurações mínimas descritas na seção **restrição**, o risco para o deploy depende da configuração que será realizada pelo próprio cliente ou por terceiros**.**

## Auditoria

Não se aplica.

## Sobrecarga de servidores

Um fator a ser levado em consideração é a quantidade de acessos ao WebService. Com a inclusão de acessos via webservice, principalmente para dispositivos mobile, devem ser avaliados a quantidade de acessos para adequar a infraestrutura a nova realidade do projeto.

# Não escopo

Os seguintes itens estão fora do escopo deste software e dependem de fatores externos alheios ao desenvolvimento:

* Implementação do HTTPS (SSL no domínio do Serviço);
* Balanceamento de carga de servidores;
* Restrições de infraestrutura, como bloqueio de IP`s, firewalls e demais configurações relacionadas a infraestrutura;
* Log e Auditoria de acessos a API;
* Implantação em ambiente de produção;
* Configurações de banco de dados;

# Referências

* REST (<https://pt.wikipedia.org/wiki/REST> );
* Zend Framework 2 (<http://framework.zend.com/manual/1.12/en/manual.html> );
* HTTP Status Code (<https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html> );

# Aprovações

|  |  |
| --- | --- |
| **APROVAÇÃO** | |
| **Área Requisitante** | **Área de TI** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***<Nome>***  ***<Cargo>***  **Matrícula.:** ***<Matr.>*** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***<Cargo>*** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_ de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ de 20\_\_\_\_\_ | |

|  |
| --- |
| **CIÊNCIA** |
| **Contratada (quando aplicável)** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***<Nome>***  ***<Cargo>*** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_ de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ de 20\_\_\_\_\_ |