**Reino do Código**

**Documento de Arquitetura do Sistema**

**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 21/06/2015 | 1.0 | Primeira versão do documento de arquitetura do sistema. | Julian Sansat, Jefferson Augusto, Allyson José, Débora Arruda, Wendell Holden, Mateus Oliveira. |
| 05/06/2015 | 2.0 | Segunda versão do documento de arquitetura do sistema. | Jullian Sansat, Jefferson Augusto, Mateus Oliveira, Renato Soares, Wendell Holden |

**1 Arquitetura**

* 1. **1.1 Visão do ambiente computacional e físico**

O sistema será hospedado em um servidor Heroku, e será utilizado como desenvolvimento os computadores pessoais de cada integrante da equipe.

* 1. **1.2 Visão macro dos pacotes ou subsistemas**

Será utilizado o framework Ruby on Rails nesse projeto, esse framework é considerado modular**,** pois internamente ele divide as responsabilidades em estruturas distintas. Essas estruturas são nomeadas de módulose entre elas estão: **ActiveRecord**, **ActionPack**, **ActiveModel**, **ActiveSupport** e o **ActionMailer**.

**2 Estilo Arquitetural**

* 1. **2.1 Descrição do Estilo Arquitetural**

Foi escolhido o M.V.C.(Model-View-Controller), pois esse tipo de arquitetura proporciona uma divisão que é interessante na hora de partilhar as tarefas entre os desenvolvedores e na manutenção do código.

**3 Pacotes ou Sub-Sistemas**

**3.1 Pacote de interface com o usuário**

### O módulo principal para a interface com o usuário é o ActionPack, onde ele é responsável pelo tratamento de rotas, requisições, geração de repostas a essas requisições e o renderização das views.

### **3.1 Pacote de negócio (serviços do sistema)**

### O módulo ActiveModel é responsável pela camada de negócio, além de servir como meio intermediário entre o ActiveRecorder(Persistencia), e o ActionPack(Interface).

### **3.1 Pacote de persistência**

### O módulo ActiveRecoder fica responsável pela persistencia, e ele é responsável pelo ORM (object-relational mapping), ou seja, um mapeador entre objetos e registros de uma tabela, onde cada classe de modelo possui uma tabela correspondente a ela no banco de dados.

### **4 API**

### 4.1 Enviando Badge (Reino do Codigo)

### A API utiliza o modulo "net/http", envia um JSON utilizando o metodo post para o controller do Badge Wall resposavel por criar o objeto com as informações recebidas. A configuração da API como a url é definida no método badge wall share

### code.png

### 

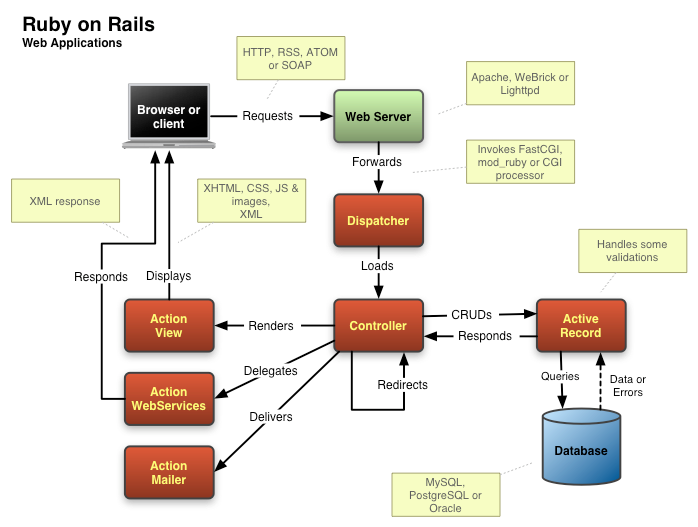
### 

### 

### 4.2 Recebendo Badge (Badge Wall)

O Badge Wall recebe o JSON enviado pelo reino do codigo e cria um objeto com estes dados.

**5 Anexos**



**Casos de uso**

