**Documento de Arquitetura**

Acompanhamento Projetos Parlamentares

|  |
| --- |
| **APP** |

**Versão: 1.0** | 13/09/2013

**Histórico de Alterações**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Horário início** | **Horário fim** | **Autor** |
| 13/09/2013 | 0.1 | Criação do Documento | 18:00 | 18:05 | Cleiton Gomes |
| 24/09/2013 | 0.2 | Tópicos 1, 1.1, | 19:31 | 19:57 | Cleiton Gomes |
| 24/09/2013 | 0.3 | Tópicos 1.2, 1.5 | 19:58 | 20:13 | Vanessa Barbosa |
| 24/09/2013 | 0.4 | Tópicos 2, 2.1 | 20:21 | 20:56 | Simião Carvalho |
| 24/09/2013 | 0.5 | Tópicos 7, 7.1, 7.2, 8 | 21:04 | 21:16 | Marco William |
| 28/09/2013 | 0.6 | Tópico 3 | 21:03 | 21:23 | João Paulo |
| 28/09/2013 | 0.7 | Tópico 4 | 21:24 | 21:42 | Vanessa Barbosa |
| 29/09/2013 | 0.8 | Tópico 5, 5.1 | 13:40 | 15:23 | Matheus Godinho |
| 29/09/2013 | 0.9 | Tópico 5.2 | 13:50 | 15:07 | Caio Nardelli |
| 30/09/2013 | 1.0 | Tópico 6 | 21:10 | 22:40 | Matheus Godinho |
| 30/09/2013 | 1.1 | Revisão final | 20:40 | 22:53 | Hebert Douglas |

**Conteúdo**

1. Introdução

1.1 Escopo

1.2 Referências

1.3 Visão Geral

2. Representação Arquitetural

2.1 Visões Arquiteturais

3. Metas e Restrições da Arquitetura

4. Visão de Casos de Uso

5. Visão Lógica

5.1 Visão Geral

5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura

6. Visão de Processos

7. Visão da Implementação

7.1 Visão Geral

7.2 Camadas

7.3 Camada de Visão

7.4 Camada de Controle

7.5 Camada de Modelo

**1. Introdução**

Este documento descreve as diretrizes adotadas para se montar uma arquitetura eficiente para o projeto APP. Tais diretrizes dão ênfase para que o sistema seja iterativo e incremental. Sendo assim, é de se esperar que a arquitetura seja simples e bem estrutura para fornecer uma base sólida e consistente para o APP.

**1.1 Finalidade**

O papel da arquitetura adotada pelo APP é fazer com que o sistema seja desenvolvido de forma sistemática e coesa. A equipe APP poderá facilmente identificar as ferramentas e outros atributos descritos na arquitetura e realmente entender a mesma e realizar o projeto com sucesso.

**1.2 Escopo**

O escopo do APP concentra-se na emissão de informações relativas a projetos parlamentares. A emissão dessas informações devem ocorrer de forma clara e objetiva. Para que a execução do projeto seja realizada com sucesso, deve-se levar em consideração não somente o escopo, mas também as variáveis de tempo, custo e qualidade. Essas variáveis modem modificar o escopo ao longo do tempo. Sendo assim, a arquitetura irá basear-se nos casos de uso mais importantes para que os riscos arquiteturais do projeto sejam minimizados ao máximo possível.

Como poderá ser observado no decorrer do documento, as ferramentas adotadas pelo grupo APP foram selecionadas de forma que agregam valor e maximiza a chance de cada iteração ser bem sucedida.

**1.3 Referências**

*1. Internet <*<http://pt.scribd.com/doc/56665358/Documento-de-Arquitetura-de-Software-Projudi>*>, acesso em 24/09/2013. Autor desconhecido*

*2. BOOCH, G; RUMBAUGH, J e JACOBSON, I: UML, Guia do Usuário: tradução; Fábio Freitas da Silva, Rio de Janeiro, Campus ,2000.*

*3. Luiz, R. (2005) “Obtendo Qualidade de Software com o RUP”, TCC, Universidade de Uberaba.*

*4. Runeson, P. and Greberg, P. (2004) "Extreme Programming and Rational Unified Process – Contrasts or Synonyms?", Lund University, Sweden.*

**1.4 Visão Geral**

A arquitetura será composta em 3 camadas: visão, controle e modelo. A organização das camadas será feita de tal forma que durante o processo de integração das mesmas seja realizado facilmente com sucesso.

Logo em seguida, será mostrada as visões do projeto APP: Caso de Uso, Lógica, Processos, Implantação, Implementação.

**2. Representação Arquitetural**

A arquitetura em camadas deve permitir que as funcionalidades do APP sejam desenvolvidas de maneira eficiente e com facilidade de manutenção, além de estabelecer limites lógicos entre as camadas, facilitando futuros incrementos no software ou modificações em camadas superiores. Dessa maneira, é maximizada as vantagens que a arquitetura pode oferecer.

A modularidade, manutenibilidade, extensibilidade e reusabilidade são características da arquitetura em camadas que permitem sucessivamente dividir a aplicação em módulos, reduzir o custo de manutenção da aplicação, permitir que novas funcionalidades sejam adicionadas sem impactar significativamente as existentes e por fim, permitir que classes e componentes sejam reusados em outros módulos de uma aplicação ou em parte de alguma outra aplicação.

**2.1. Visões Arquiteturais**

Para representar as estruturas que compõem a arquitetura do APP, as unidades lógicas de código, suas composições, relacionamentos e dependências, além de processos e iterações entre componentes e de que forma o software é mapeado no hardware, serão apresentados neste documento cinco diferentes visões do Sistema. As visões necessárias para a arquitetura são: Visão de Casos de Uso, Visão Lógica, Visão de Processos, Visão de Implantação e Visão de Implementação. As principais características de cada visão são respectivamente, fornecer uma base para o planejamento do conteúdo técnico de iterações, compreender a estrutura e a organização do design do sistema, fornecer uma base que permite compreender a organização do processo do sistema, ilustrar a distribuição do processamento em um conjunto de nós do sistema e por fim, captar as decisões de arquitetura tomadas para a implementação. A Visão de Casos de Uso foca na disciplina de Requisitos e as demais na disciplina de Análise e Design.

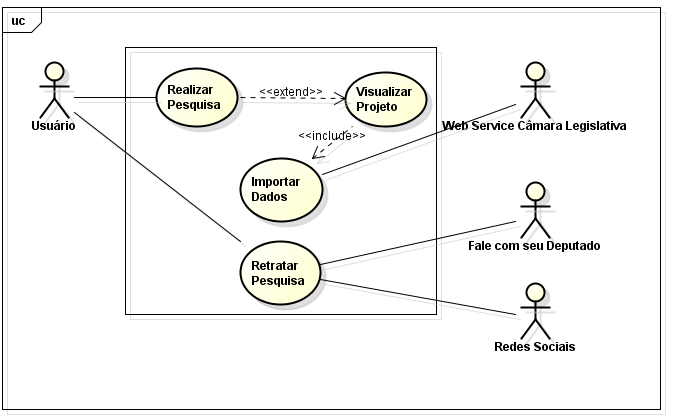
**3. Metas e Restrições da Arquitetura**

Dentre as restrições arquiteturais, tem-se:

* O sistema será desenvolvido em JAVA para atender principalmente os requisitos de portabilidade e devido desenvolver para o aplicativo android.
* O SGBD MySqL LITE será utilizado para manter a base de dados do APP por ser grátis e oferecer um conjunto adequado de características de desempenho e segurança necessário ao APP.
* Haverá restrições de usuários com objetivo de não permitir alterações indesejadas na base de dados.
* Os requisitos de usabilidade, descritos no documento de Visão Geral, devem ser considerados no desenvolvimento de qualquer componente da camada visão.
* A camada visão não deve acessar diretamente os elementos da camada Model ou do pacote DAO
* As camadas inferiores não devem conhecer as camadas superiores.

**4. Visão de Casos de Uso**

A visão de Caso de Uso aqui apresentada visa expor a visão funcional do software. A seguir é apresentado o diagrama de Casos de Uso do APP e uma breve descrição de cada Caso de Uso.



O caso de uso Realizar Pesquisa é responsável pela realização da pesquisa de um projeto através da passagem de alguns parâmetros como nome, ano, data de início e etc. O caso de uso Visualizar Projeto irá mostrar o projeto que foi pesquisado com sucesso para o usuário de forma interativa, como através de gráficos, por exemplo, ou até mesmo através de textos, desde os mesmos estão claros e objetivos. O caso de uso Importar Dados possui a função de guardar os dados de maneira normalizada para que o mesmos possam ser usados pelo caso de uso visualizar projeto. Por fim, o caso de uso Retratar Pesquisa é responsável por divulgar a pesquisa em redes sociais ou permitir que o usuário fale com um deputado relacionado aquela pesquisa.

De acordo com o diagrama de casos de uso apresentado, a direita está o ator principal e a esquerda está os atores secundários.

**5. Visão Lógica**

A Visão Lógica que será descrita aqui, busca exibir as partes significativas da arquitetura do APP, detalhando sua divisão em subsistemas e pacotes, sendo assim representando os componentes lógicos que formam a arquitetura do APP.

Os pacotes do projeto APP podem ser visualizados na figura 1 e os mesmos estão organizados de tal forma que as classes fiquem ordenadas e utilizadas de forma conveniente. dentro de seus objetivos na arquitetura.

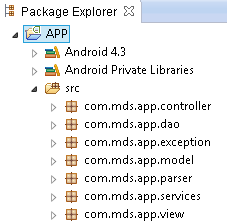


Figura 1: Pacotes projeto App

A Tabela abaixo ilustra os pacotes e suas respectivas classes mais significativas.

|  |  |
| --- | --- |
| Pacote | Classes |
| com.mds.app.controller | ParlamentarController, PartidoController, Projeto Controller. |
| com.mds.app.model | ParlamentarModel, ProcuraParlamentarModel, Partido Model, Projeto Model, ProcuraPartidoModel, ProcuraProjetoModel. |
| com.mds.app.view | MainActivity |
| com.mds.app.services | Endereco, ProcuraProjeto, RecebeHTTP, XMLParser |
| com.mds.app.dao | ParlamentarDao, ProjetoDao, PartidoDao |
| com.mds.app.parse | ParlamentarParse, ProjetoParse, PartidoParse |
| com.mds.app.exceptions | ProjetoExceptions, ParlementarExceptions |

A tabela abaixo mostra algumas classes mais significativas com suas responsabilidades e respectivos atributos.

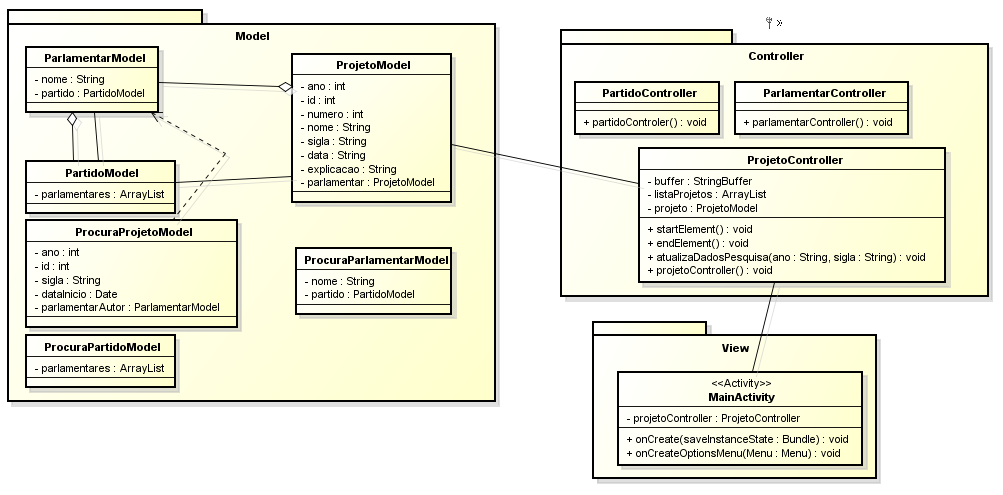
|  |  |  |
| --- | --- | --- |
| Classe | Atributos | Responsabilidade |
| ProjetoModel | ano : int; id : int; numero : int; ; nome: String; sigla : String; explicacao : String; parlamentar : ParlamentarModel | Responsavel por conter as caracteristicas necessarias de um projeto. |
| PartidoModel |  | responsavel por detalhar um partido. |
| ParlamentarModel | nome : String; partido : PartidoModel | responsável por conter as características de um parlamentar |
| ProcuraProjetoModel | ano : int; id : int; sigla : String; dataInicio : Date; parlamentar : ParlamentarModel | realiza a pesquisa de um projeto de acordo com suas caractetisticas. |
| MainActivity |  | Testa a execução do projeto. |

**5.1 Visão Geral**

O design do software APP é dividido em basicamente três camadas compostas pelos pacotes Model, View, e Controller. São dentro desses três pacotes que ocorre todo o funcionamento do sistema ordenado de acordo com as suas funções, o pacote Controller é o responsável pela interação de todos os componentes do APP, fazendo a ligação entre Model e View, além de coordenar o que e como é feito dentro de sua utilização. Enquanto a Model e responsável por caracterizar todos os componentes com o que eles são feitos, como são feitos e com o que pode interagir, ou seja, é a responsável por caracterizar quaisquer mudanças de estados. E então por ultimo o pacote View é responsável por gerar uma saída, é a que mostra o resultado das mudanças de estados feitas e que a controller pediu que fosse atualizado este estado.

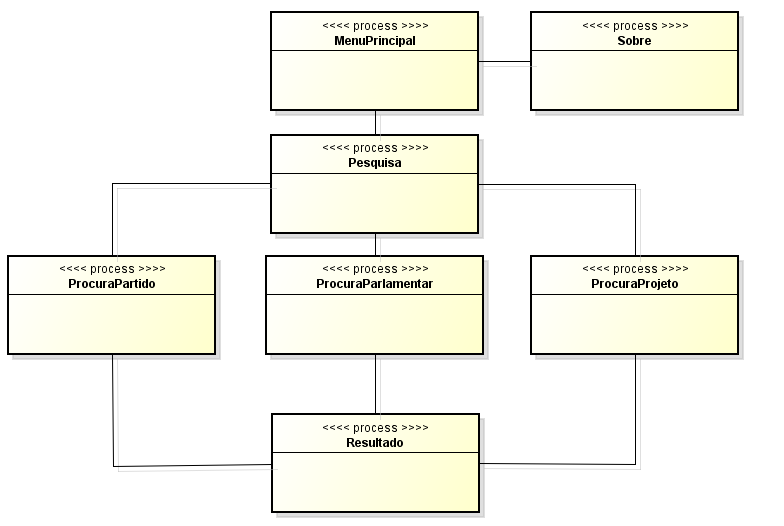
**5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura**

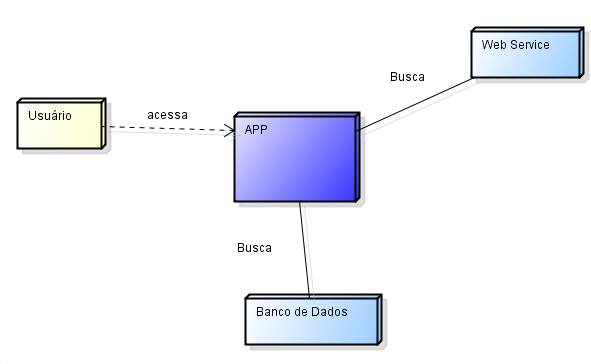
Na figura abaixo é mostrado os pacotes com grau de importância mais elevados e suas respectivas classes mais significativas.



**6. Visão de Processos**

A Visão de Processos descreve variáveis ligadas a decomposição do sistema e essas variáveis estão ligadas aos fluxos de execução. Por fim, pode-se obter a representação dos relacionamentos, comportamentos, além dos componentes ligados ao APP.





**7.** **Visão de Implementação**

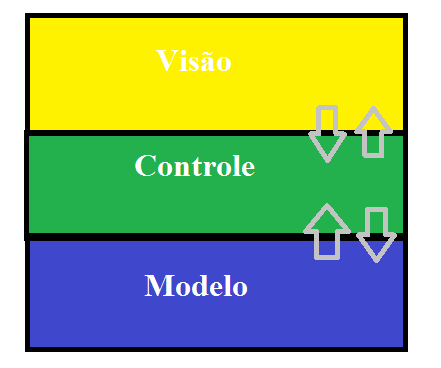
Na Visão de Implementação é descrito a estrutura geral do modelo de implementação, a divisão do software em camadas e subsistemas no modelo de implementação e todos os componentes significativos do ponto de vista da arquitetura.

**7.1 Visão Geral**

O modelo de arquitetura MVC é largamente utilizado em projetos. Esse padrão isola os componentes de apresentação da lógica de negócio facilitando a manutenção do sistema.

**7.2 Camadas**

As camadas em que se baseiam a arquitetura do projeto (visão, controle e modelo) são esquematizadas na figura abaixo.



**7.3 Camada de Visão**

Camada de Visão é a chamada GUI (Graphical User Interface), ou simplesmente interface. Esta camada é responsável pela interação com o usuário. No APP, a camada de visão irá implementar uma interface de entrada e saída para a interação da aplicação com usuário. As informações fornecidas pelo usuário serão validadas e consequentemente o usuário terá acesso aos serviços disponíveis pela camada de negócio.

**7.4 Camada de Controle**

A camada de controle deve implementar a lógica da aplicação, expondo esta lógica para a camada de visão ou para outras aplicações clientes remotas.

**7.5 Camada de Modelo**

Camada responsável pela persistência e acesso aos dados da aplicação. Ela isola o resto da aplicação do meio de armazenamento usado (memória, arquivos, SGBD, aplicações legadas, etc.) de maneira que, se o meio de armazenamento for trocado, apenas as classes desta camada precisarão ser modificadas ou substituídas.

**8.** **Exceptions**

A utilização do recurso exception no contexto do APP provê mecanismos para tratamento de erros e falhas para que não haja interrompimento nas ações do usuário no sistema ou até mesmo inconsistência de estado após alguma ação incorreta. Serão usados os proprios recursos de tratamento de exceções pelo pacote exceptions fornecido pelo JAVA, além de algumas alteras para a criação de nossas próprias exceções utilizando destes recursos disponíveis.