## Histórico de Versões

| Data | Versão | Descrição | Autor | Revisor | Aprovado por |
|---|---|---|---|---|---|
| 23/03/2018 | 1.0 | Preenchimento do documento | Anderson | Equipe | Equipe |
| 07/04/2018 | 1.1 | Continuação do preenchimento | Anderson | Equipe | Equipe |
| 11/04/2018 | 1.2 | Inserir diagramas | Anderson | Equipe | Equipe |
| 18/04/2018 | 1.3 | Inserindo plano de teste e diagrama de classes | Anderson, Joziel, Everton, Fernando e Allison | Equipe | Equipe |
| 29/04/2018 | 1.4 | Inclusão de Diagramas (item 2.1) e texto (item 1,2, e 3) | Everton | Equipe | Equipe |
| 01/05/2018 | 1.5 | Criação de Caso de Teste (item 4.1) | Everton | Equipe | Equipe |
| | | | | | |
| | | | | | |

Índice

## 1. Objetivo do Documento

Prover uma visão geral do projeto Recruta-if com os elementos de engenharia do RUP. Durante este documento iremos detalhar as partes mais significativas do sistema.

## 2. Objetivos e Restrições da Arquitetura

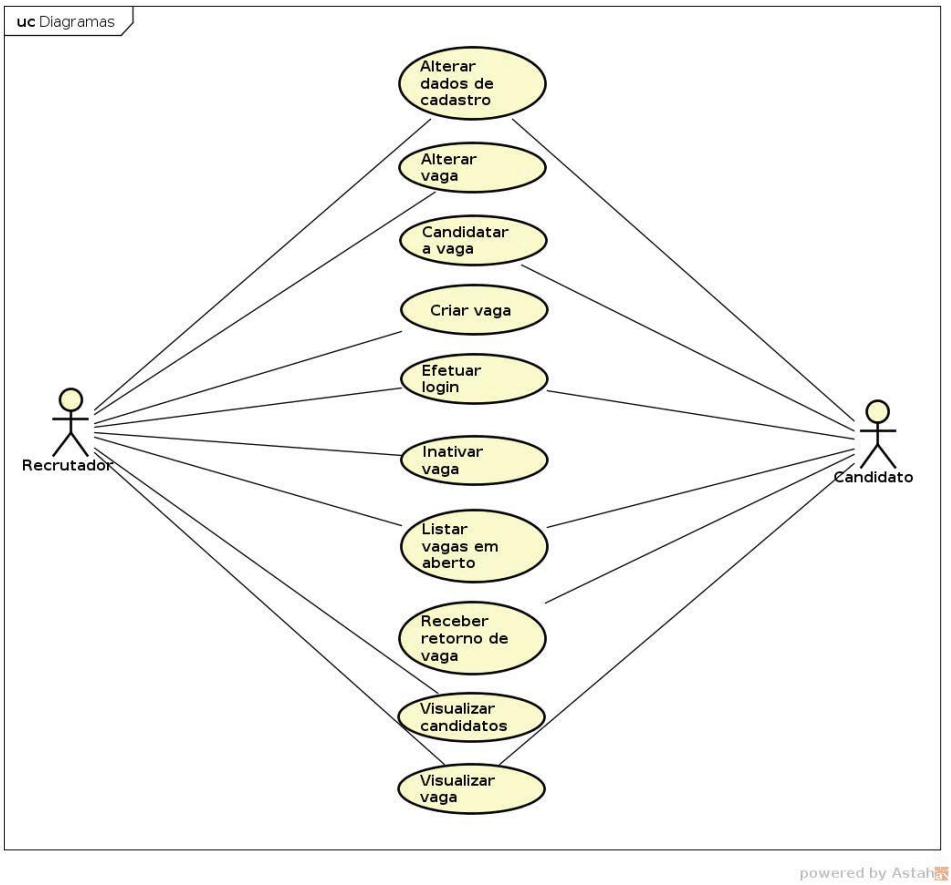
O objetivo é desenvolver o software utilizando as tecnologias abaixo:

- Java EE - Linguagem de programação principal;
- Wildfly - Servidor de aplicação;
- Hibernate - Entidade de persistência de dados no Banco;
- Rest - Padrão de comunicação entre o frontend e o backend;
- HTML5 – Linguagem de marcação, para gerar as telas da aplicação;
- CSS3 – Linguagem de estilização das páginas em HTML.

As principais restrições são:

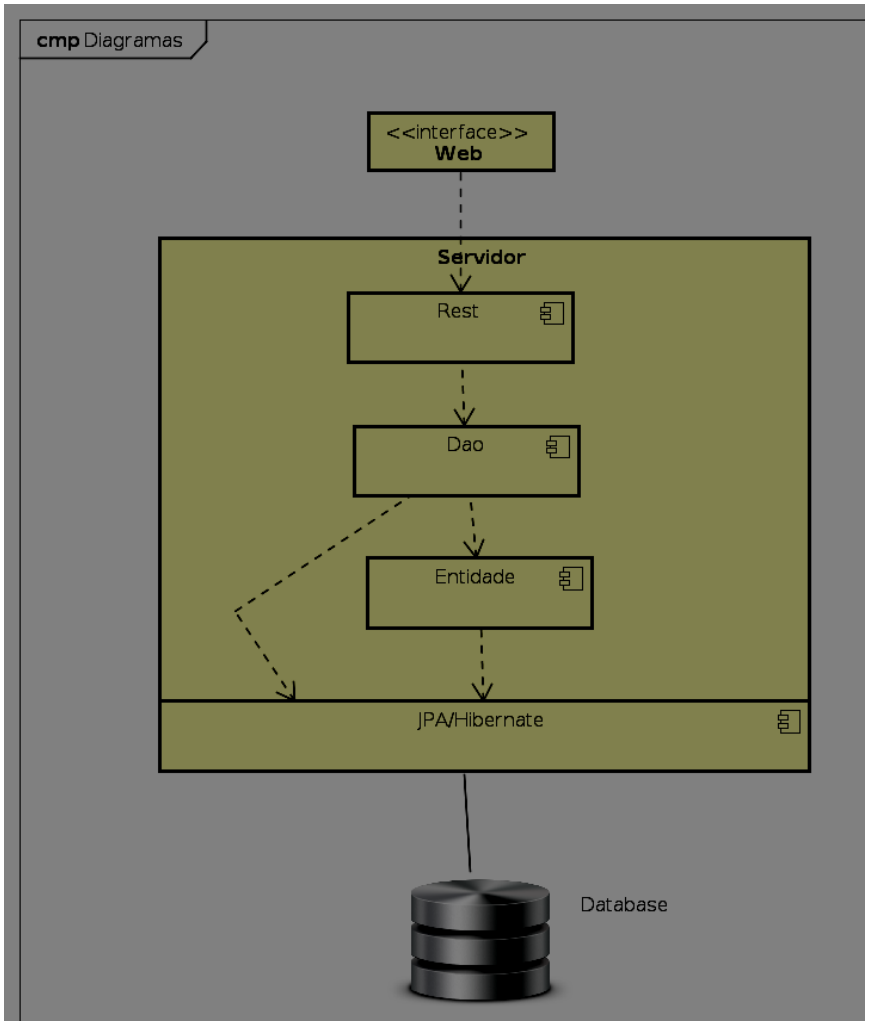
- Promover um ambiente de desenvolvimento que trabalhe nos sistemas operacionais Debian 9 e Windows 10;
- Desenvolver métodos que alimentem o nosso banco de dados com dados de terceiros;
- Utilizar o Maven como gerador de dependências e build de nosso projeto;
- Integrar a equipe para desenvolver em conjunto.

## 2.1. Diagrama Casos de Uso



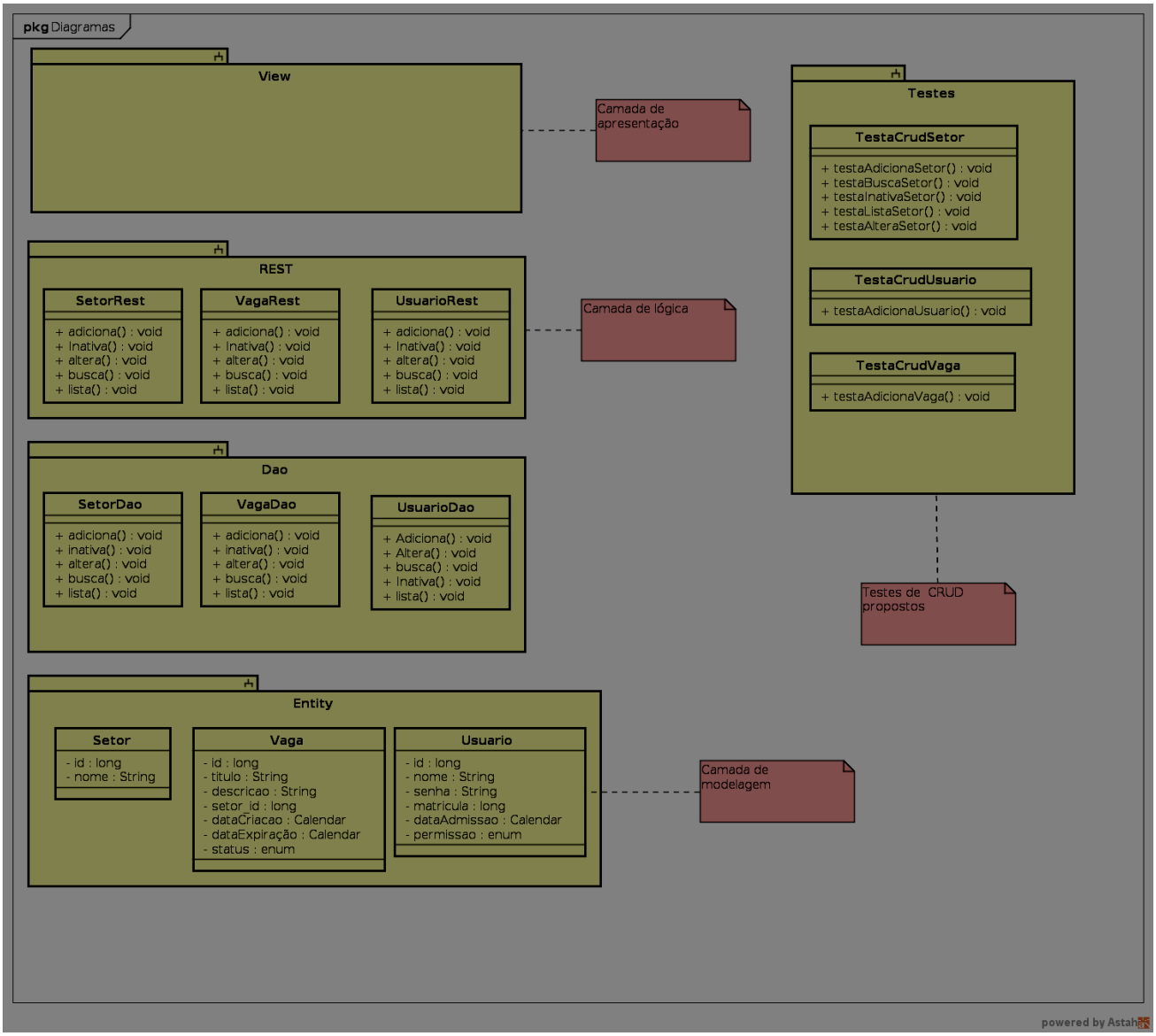
## 3. Elementos Arquiteturalmente Significativos

O sistema será desenvolvido em linguagem de programação JAVA Web para acessar pelos navegadores Chrome (a partir da versão: 65.0.3325.162) e Firefox (a partir da versão: 52.6.0). Plataforma Windows 10 64 bits e Linux Debian 9. Banco de dados Postgree SQL.



# 4. Descrição da Arquitetura

## Camadas, Subsistemas e testes:



### 4.1. Plano de testes seguindo modelo JUnit:

Utilizaremos o JUnit como suíte de testes. Para comparar resultados obtidos e esperados usaremos os métodos assertEquals, assertTrue, assertNul.

Códigos comuns de referência:

| | |
|---|---|
| C01 | Campo não pode estar em branco |
| C02 | Campo não deve conter números |
| C03 | Deve possuir no máximo 500 caracteres. |
| C04 | Não deve ultrapassar 50 caracteres. |
| C05 | Deve retornar uma exceção com uma mensagem de erro |
| C06 | A data informada deve estar no formato dd/MM/yyyy, sem minutos e segundos. |
| C07 | A data informada não deve ser maior que 6 meses. |
| C08 | A data informada não deve ser menor que o dia atual. |

| | |
|---|---|
| Caso de Teste | 001 – Criar Vaga |
| Descrição | Testar se a vaga é criada dentro dos parâmetros sem exceções dentre os cenários citados |
| Pacote de teste | br.org.iel.recrutaif.dao.teste |
| Classe de teste | TesteVagaDao |
| Cenário | Cenário 01 - Título da Vaga: C02<br>Cenário 02 - Título da Vaga: C04<br>Cenário 03 - Título da Vaga: C01<br>Cenário 04 - Descrição da vaga: C01<br>Cenário 05 - Descrição da vaga: C03<br>Cenário 06 - Setor da vaga – C01<br>Cenário 07 - Data expiração da vaga: C01<br>Cenário 08 - Data expiração da vaga: C06<br>Cenário 09 - Data expiração da vaga: C07<br>Cenário 10 - Data expiração da vaga: C08 |
| Resultado esperado | Todos os cenários devem retornar: C05 |
| Resultado obtido | NA |

| | |
|---|---|
| Caso de Teste | 002 – Alterar Vaga |
| Descrição | Testar se a vaga é alterada dentro dos parâmetros sem exceções dentre os cenários citados |
| Pacote de teste | br.org.iel.recrutaif.dao.teste |
| Classe de teste | TesteAlterarVagaDao |
| Cenário | Cenário 01 - Título da Vaga: C01<br>Cenário 02 - Descrição da vaga: C01<br>Cenário 03 - Descrição da vaga: C03<br>Cenário 04 - Setor da vaga – C01<br>Cenário 05 - Data expiração da vaga: C01<br>Cenário 06 - Data expiração da vaga: C06<br>Cenário 07 - Data expiração da vaga: C07<br>Cenário 08 - Data expiração da vaga: C08 |
| Resultado esperado | Todos os cenários devem retornar: C05 |
| Resultado obtido | NA |

| | |
|---|---|
| Caso de Teste | 003 – Alterar dados de cadastro |
| Descrição | Testar se os dados do cadastro são alterados dentro dos parâmetros sem exceções dentre os cenários citados |
| Pacote de teste | br.org.iel.recrutaif.dao.teste |
| Classe de teste | TesteAlterarDadosDao |
| Cenário | Cenário 01 - Senha: Não pode possuir menos de 8 caracteres<br>Cenário 02 - Senha: C01<br>Cenário 03 – Senha: não pode ser preenchida com espaços em branco.<br>Cenário 04 - Senha: não pode ser igual a anterior |
| Resultado esperado | Todos os cenários devem retornar: C05 |
| Resultado obtido | NA |

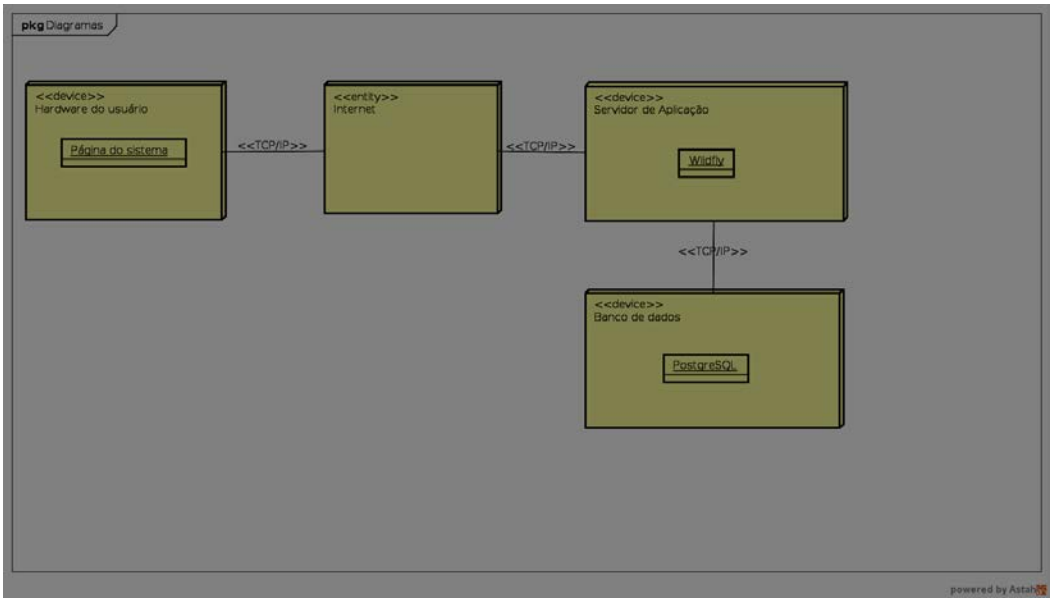
| | |
|---|---|
| Caso de Teste | 004 – Permitir que o usuário altere sua senha |
| Descrição | Testar se os dados do cadastro são alterados dentro dos parâmetros sem exceções dentre os cenários citados |
| Pacote de teste | br.org.iel.recrutaif.dao.teste |
| Classe de teste | TesteAlterarSenhaDao |
| Cenário | Cenário 01 - Senha: Não pode possuir menos de 8 caracteres<br>Cenário 02 - Senha: C01<br>Cenário 03 – Senha: não pode ser preenchida com espaços em branco.<br>Cenário 04 - Senha: não pode ser igual a anterior |
| Resultado esperado | Todos os cenários devem retornar: C05 |
| Resultado obtido | NA |

Classes de equivalência:

```
// Cenário 001-02: Campo contendo números.
@Test
public void addCreateTest() {
   assertFalse(addCreateTest (1, 1));
}


// Cenário 002-04: Verifica se o setor é listado

Vector<ActivityVagaBuscaAttibute> activityBuscaVagaAttibuteVector;

      activityAssociationAttibuteVector = br.org.iel.recrutaif.dao
                  .getAttributesByFilter(null, false);

      assertFalse(activityVagaBuscaAttibuteVector.elementAt(0)
                  .getActivity().getCode() == 1);
}
```

### 4.2. Padrões e Mecanismos Arquiteturais

### 4.3. Topologia




## 5. Decisões e Justificativas

### 5.1. Backend

Acordamos pelo uso do JAVA EE para desenvolvimento, banco de dados PostgreSQL, framework Hibernate para persistir os dados e Wildfly como servidor de aplicação.

Escolhemos todos os citados acima por serem aplicações muito utilizadas em projetos para web usando JAVA.

### 5.2. Frontend

Entramos em consenso na criação com HTML5 e CSS com Bootstrap. Porém houve uma alteração e estamos usando framework Angular, além do JavaScript.