EEL873 - Engenharia de Software - 2016.2

ConsertaAê

Equipe Chile

Antonio Pereira de Brito Galvão

Eduardo Fernando dos Santos Araujo

Eric Reis Figueiredo

João Henrique Franco

Sumário

Documentação Preparada pela Equipe

- ➔ Especificação de Requisitos e Casos de Uso
- ➔ Modelos de Projeto (Alto Nível)
- ➔ Plano de Testes
- ➔ Código Gerado e Evidência dos Testes
- ➔ Sistema Entregue

Sumário

Cenário Geral de Desenvolvimento

- ➔ Papéis dos Membros da Equipe e Esforço Individual
- ➔ Utilização dos Modelos de Projeto
- ➔ Dificuldades
- ➔ Facilidades
- ➔ Comentários Adicionais

Especificação de Requisitos e Casos de Uso

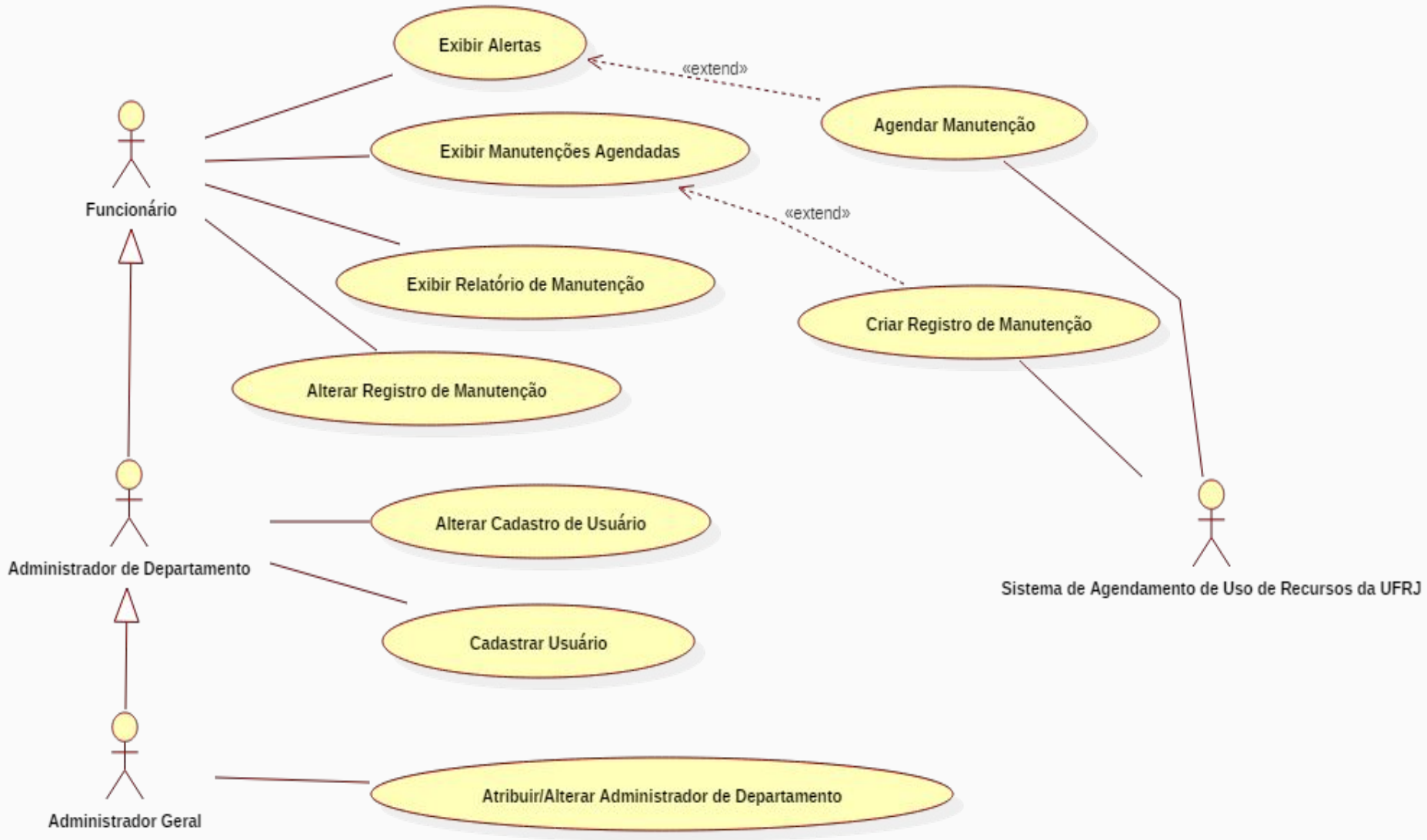


Diagrama de Casos de Uso

Especificação de Requisitos e Casos de Uso

- Ao longo das fases iniciais do projeto esses documentos foram discutidos e corrigidos constantemente, entretanto após esse período não foram mais alterados.

Modelos de Projeto (Alto Nível)

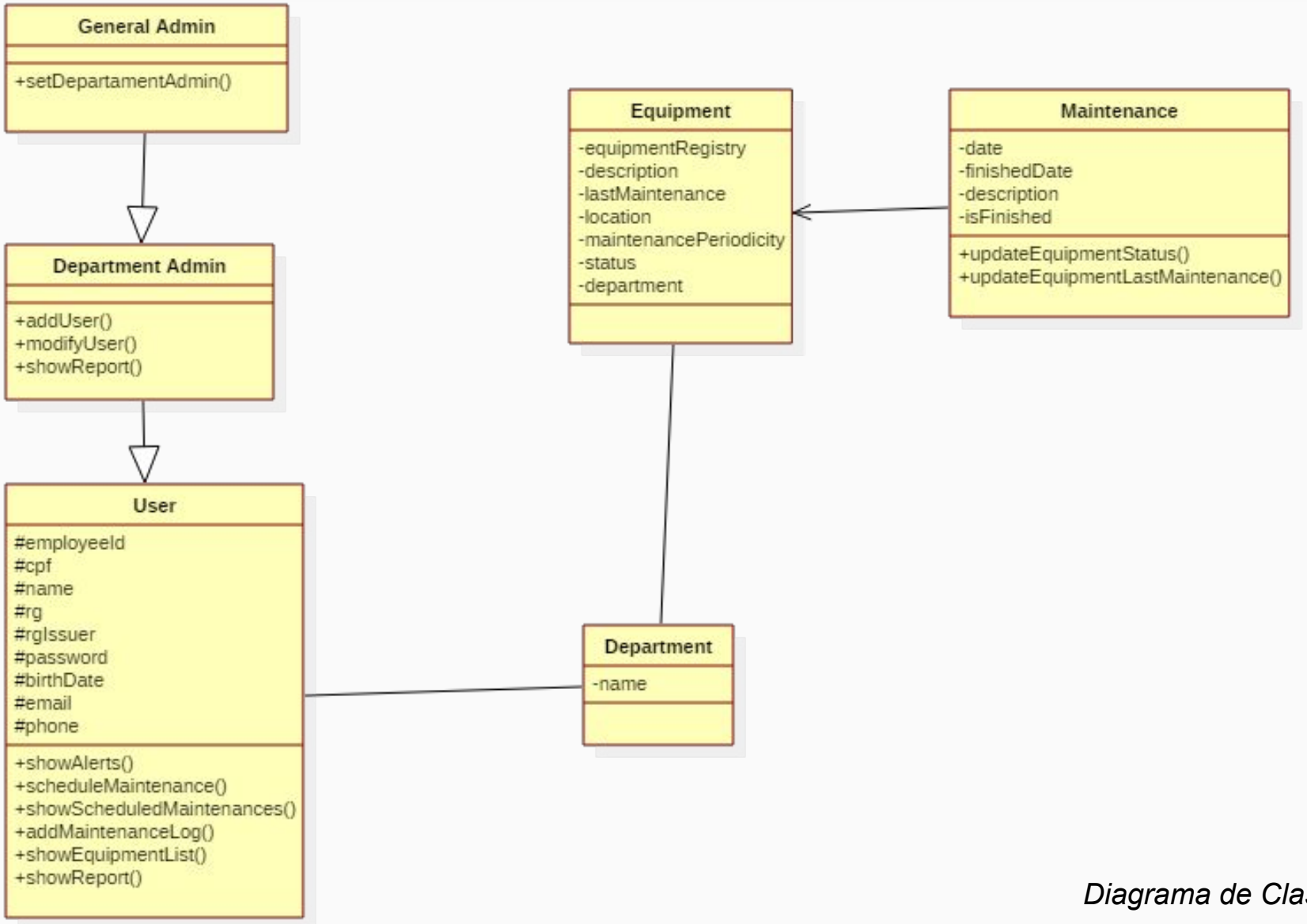


Diagrama de Classes

Modelos de Projeto

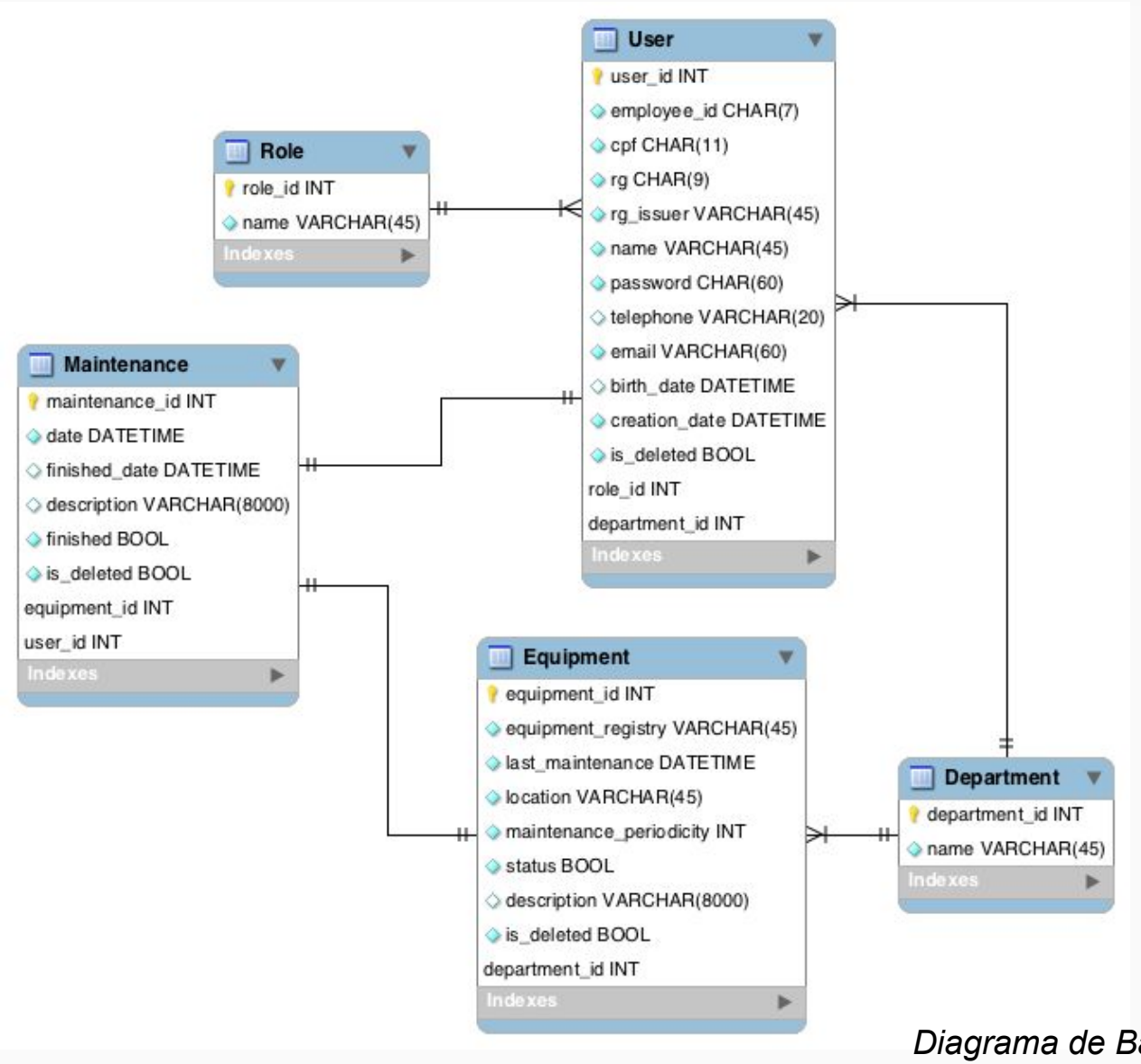


Diagrama de Banco de Dados

Modelos de Projeto (Alto Nível)

- ➔ Os modelos são importantes pois asseguram que toda a equipe compreendeu além do problema, a solução de software proposta e os detalhes de sua implementação.
- ➔ Durante o projeto nas fases de codificação e testagem por vezes percebemos que alguma especificação previamente acordada não era a melhor solução possível para um problema ou não atendia satisfatoriamente aos requisitos.

Código Gerado e Evidências de Testes

- Um exemplo de teste de unidade que planejamos e executamos é o teste de login, implementado utilizando a framework *JUnit*.

Código Gerado e Evidência de Testes

Entradas		Resultados	
Usuário	Senha	Esperado	Obtido
admin	admin	Sucesso	Sucesso
asdjhfklajs	admin	Falha	Falha
admin	“ ”	Falha	Falha
12345689049876512 36	admin	Falha	Falha
123456	admin	Falha	Falha
12345678	admin	Falha	Falha
“ “	admin	Falha	Falha
1234567	1234567	Sucesso	Sucesso

Código Gerado e Evidência de Testes

```
@Test
    public void loginTest() throws SQLException
    {
        User expected = null;

        LoginDTO loginDTO = new LoginDTO();

        loginDTO.setEmployeeId(admin");
        loginDTO.setPassword(admin");
        Assert.assertNotEquals(expected,
this.loginService.login(loginDTO));

        loginDTO.setEmployeeId(asdjhfklajs");
        loginDTO.setPassword(admin");
        Assert.assertEquals(expected,
this.loginService.login(loginDTO));

        loginDTO.setEmployeeId(admin");
        loginDTO.setPassword(         ");
        Assert.assertEquals(expected,
this.loginService.login(loginDTO));
```

```
loginDTO.setEmployeeId('1234568904987651236');
        loginDTO.setPassword(admin");
        Assert.assertEquals(expected,
this.loginService.login(loginDTO));

loginDTO.setEmployeeId('123456");
        loginDTO.setPassword(admin");
        Assert.assertEquals(expected,
this.loginService.login(loginDTO));

        loginDTO.setEmployeeId(12345678");
        loginDTO.setPassword(admin");
        Assert.assertEquals(expected,
this.loginService.login(loginDTO));

        loginDTO.setEmployeeId(        ");
        loginDTO.setPassword(admin");
        Assert.assertEquals(expected,
this.loginService.login(loginDTO));

        loginDTO.setEmployeeId(1234567");
        loginDTO.setPassword(1234567");
        Assert.assertNotEquals(expected,
this.loginService.login(loginDTO));
    }
```

Código Gerado e Evidência de Testes

Nosso projeto foi desenvolvido/testado utilizando...

- Java
- MySQL
- AngularJS
- TomCat 8
- Jenkins
- AWS
- JUnit
- Selenium IDE

Código Gerado e Evidência de Testes

- Essas decisões foram tomadas baseando-se não só no que era mais adequado ao projeto, mas também no prévio conhecimento e experiência da equipe.

Código Gerado e Evidência de Testes

- ➔ Os testes foram relevantes pois asseguraram que erros fossem descobertos e corrigidos.
- ➔ Além disso testes unitários nos permitiram isolar as causas desses erros.

Código Gerado e Evidência de Testes

- ➔ Adotamos uma estratégia focando em testes unitários e em posteriores testes de integração.

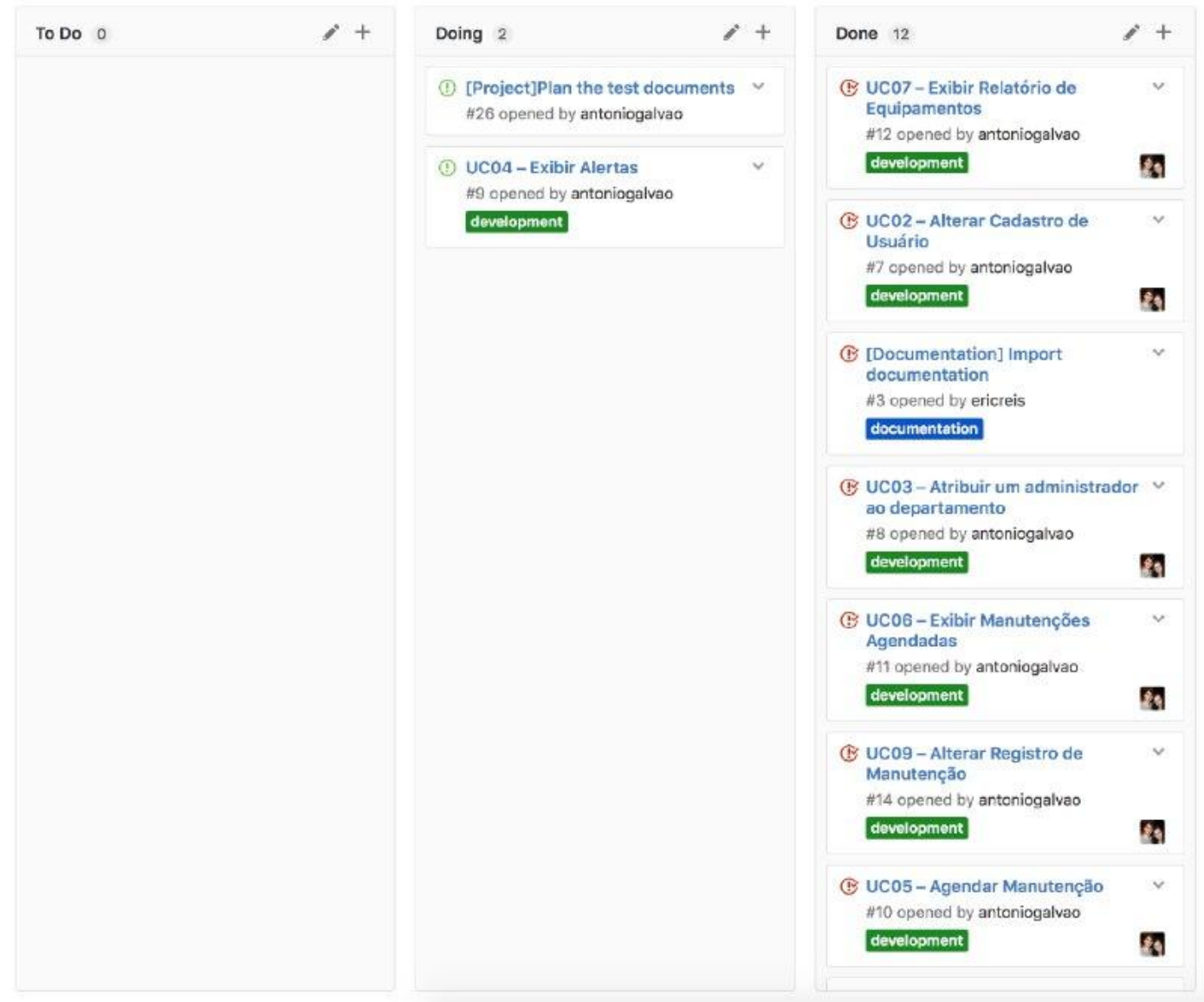
Demonstração do Sistema

http://ec2-34-192-86-33.compute-1.amazonaws.com:8081/consertae/

Papéis dos Membros da Equipe e Esforço Individual

- ➔ Um de nossos integrantes já possuía certa experiência com projetos do tipo portanto desde o início do desenvolvimento tínhamos uma estimativa realista quanto ao esforço e tempo necessários.
- ➔ Esperávamos um cronograma dificultado e tal expectativa se concretizou.

Papéis dos Membros da Equipe e Esforço Individual



Print do Kambam no estágio final do projeto, disponível no GitHub

Papéis dos Membros da Equipe e Esforço Individual

- A equipe dividiu as tarefas de forma orgânica, onde cada integrante realizava a parte do projeto que tivesse mais aptidão no momento e estivesse mais confortável desenvolvendo.
- Isso garantiu que cada membro da equipe trabalhasse em diversos tipos de atividades assim compreendesse melhor o funcionamento do projeto como um todo, além de não sobrecarregar nenhum integrante.

Papéis dos Membros da Equipe e Esforço Individual

- A maior parte do desenvolvimento foi realizada de forma assíncrona, devido a dificuldade de se encontrar horários em que todos pudessem trabalhar juntos.
- Entretanto para algumas decisões e fases críticas do projeto foram realizadas reuniões tanto presencialmente quanto via *hangouts*.

Papéis dos Membros da Equipe e Esforço Individual

- ➔ Tal estratégia facilitou o andamento do projeto de forma mais dinâmica e eficiente.
- ➔ Teve como ponto negativo a maior dificuldade de comunicação entre os membros dado que cada um estava disponível em um momento diferente.

Papéis dos Membros da Equipe e Esforço Individual

- Considerarmos que a equipe trabalhando em um mesmo local tem uma maior produtividade e seria a melhor solução hipotética.
- Entretanto dadas as circunstâncias do projeto como prazos muito curtos e a disponibilidade não coincidente dos membros, julgamos a estratégia seguida como a melhor opção.

Construção dos Modelos de Projeto

- ➔ Sentimos dificuldades desde a captura dos requisitos, devido a complexidade do sistema e a divergência de opiniões entre os *stakeholders*. Essas dificuldades acabaram se estendendo para os outros modelos.
- ➔ Adotamos uma estratégia similar ao desenvolvimento ágil, na qual escrevíamos os documentos na medida que percebíamos a necessidade de alguma nova funcionalidade ou alteração de alguma já existente.

Construção dos Modelos de Projeto

- ➔ Preferimos adotar este procedimento, ao invés de primeiro escrevermos os documentos para depois implementarmos, para não ficarmos presos a apenas um modo de implementação.

Construção dos Modelos de Projeto

- Acreditamos que para o desenvolvimento de um projeto como este, com uma equipe pequena e com tempo muito limitado para desenvolvimento, é mais interessante seguir um modelo de desenvolvimento ágil, como o *Scrum*. Isso permitiria que a evolução do sistema ficasse mais visível e a alocação do esforço fosse mais eficiente.

Construção dos Modelos de Projeto

- Classificação dos modelos:
  - Diagrama de Casos de Uso: Essencial para compreendermos o que o sistema deve oferecer para os usuários.
  - Diagrama de Classes: Essencial para compreendermos como são definidos os agentes do sistema e como eles se relacionam
  - Diagrama de Sequência: Desejável caso precisássemos seguir a risca um fluxo de chamadas (pré-definido) de uma funcionalidade na sua implementação.

Construção dos Modelos de Projeto

- Diagrama de Atividades: Desejável caso precisássemos seguir a risca um fluxo de ações e reações (pré-definido) de uma funcionalidade na sua implementação.
- Diagrama de Componentes: Desejável para documentar as relações de integração entre diferentes sistemas e, até mesmo, entre as diferentes camadas de um único sistema.

Construção dos Modelos de Projeto

- ➔ De uma forma geral, os modelos apresentados no curso supriram bem as necessidades do projeto.
- ➔ Entretanto, sentimos falta de um maior foco em modelagem de banco de dados, que utilizamos bastante ao longo do desenvolvimento.