**Documento de Requisitos**

**Bolão Web**

**Versão2.00**

|  |  |
| --- | --- |
| Integrantes | R.A. |
| ALISON ORITA | 69657 |
| DANILO PASSAFARO LARROSA | 61347 |
| JUAN CARLOS BIAZOTO | 67595 |
| LUCIO MASSAO GOMES ITIKAWA | 65135 |

**Sumário**

1 Introdução

1.1 Objetivo do documento

1.2 Escopo do Produto

2 Descrição Geral do Produto

2.1 Interfaces com o Produto

2.2 Requisitos Funcionais

[RF001] Gerenciar cadastro de administradores

[RF002] Gerenciar cadastro de operadores

[RF003] Gerenciar cadastro de campeonatos

[RF004] Gerenciar cadastro de times

[RF005] Gerenciar cadastro de partidas

[RF006] Cadastrar novo usuário apostador

[RF007] Gerenciar cadastro de apostadores

[RF008] Gerenciar cadastro de palpite

[RF009] Cadastrar placar e distribuir pontos

[RF010] Ranking de apostadores

[RF011] Consulta fácil de palpites

[RF012] Tela de acesso (login)

2.3 Requisitos Não Funcionais

[NF001] Plataforma web

[NF002] Integração com redes sociais

3 Estudo de Viabilidade

3.1 Quanto as contribuições à empresa

3.2 Quanto a viabilidade econômica

3.3 Quanto a viabilidade técnica

3.4 Quanto a legalidade

3.5 Quanto as alternativas

4 Avaliação dos Riscos

5 Diagrama de pacotes

6 Arquitetura de software

7 Elaboração

7.1 Modelo de casos de uso

Identificação e priorização dos casos de uso

Diagrama de casos de uso

Detalhamento dos casos de uso

USE CASE<gerenciar cadastro de administradores> - Danilo

USE CASE<gerenciar cadastro de operadores> - Juan

USE CASE<gerenciar cadastro de campeonatos> - Alison

USE CASE<gerenciar cadastro de times> - Lucio

USE CASE<gerenciar cadastro de partidas> - Danilo

USE CASE<cadastrar novo usuário apostador> - Juan

USE CASE<gerenciar cadastro de apostador> - Alison

USE CASE<gerenciar cadastro de palpite> - Lucio

USE CASE<processar resultados e distribuir pontos> - Danilo

USE CASE<consultar ranking de apostadores> - Alison

USE CASE<consulta fácil de palpites> - Lucio

7.2 Modelo de análise

USE CASE<gerenciar cadastro de administradores> - Danilo

USE CASE<gerenciar cadastro de operadores> - Juan

USE CASE<gerenciar cadastro de campeonatos> - Alison

USE CASE<gerenciar cadastro de times> - Lucio

USE CASE<gerenciar cadastro de partidas> - Danilo

USE CASE<cadastrar novo usuário apostador> - Juan

USE CASE<gerenciar cadastro de apostador> - Alison

USE CASE<gerenciar cadastro de palpite> - Lucio

USE CASE<processar resultados e distribuir pontos> - Danilo

USE CASE<consultar ranking de apostadores> - Alison

USE CASE<consulta fácil de palpites> - Lucio

7.3 Planejamento das iterações

7.4 Diagrama de classes – Modelo de análise

8 Modelo de projeto

8.1 Padrões de projeto

8.2 Frameworks

JSF

Spring

Hibernate

8.3 Diagrama de componentes

8.4 Diagrama de classes de projeto

8.5 Diagrama de sequência - USE CASE<processar resultados e distribuir pontos>

8.6 Diagrama de implantação

9 Planejamento das Sprints

9.1 Detalhamento das estórias e tarefas

RF001 - Gerenciar cadastro de administradores (Danilo)

RF002 –Gerenciar cadastro de operadores(Juan)

RF003 – Gerenciar cadastro de campeonato (Alison)

RF004 - Gerenciar cadastro de times (Lucio)

RF005 - Gerenciar cadastro de partidas (Danilo)

RF006 - Cadastrar novo usuário apostador (Juan)

RF007 – Gerenciar cadastro de apostador (Alison)

RF008 - Gerenciar cadastro de palpite (Lucio)

RF009 - Cadastrar placar e distribuir pontos (Danilo)

RF010 – Ranking de apostadores (Alison)

RF011 - Consulta fácil de palpites (Lucio)

RF012 - Tela de acesso (login)

9.2 Product backlog e Planejamento das sprints

10 Planejamento dos testes

10.1 Detalhamento dos testes

TESTE 1 – Validação do LOGIN (Alison)

TESTE 2 – Validação XXX

TESTE 3 – Validação XXX

TESTE 4 – Validação XXX

1. **Introdução**

Este documento especifica os requisitos do **Bolão Web**, fornecendo aos desenvolvedores as informações necessárias para o projeto e implementação, assim como para a realização dos testes e homologação da aplicação.

O Bolão Web será uma aplicação disponível na rede mundial de computadores (*internet*) através do endereço [***http://bolao.anguloweb.com.br***](http://bolao.anguloweb.com.br), publicamente, através de um registro de usuário. O seu principal objetivo é angariar visibilidade e acessos ao domínio da empresa **Anguloweb Soluções Para Internet Ltda.** na internet, promovendo engajamento e diversão com o bom e velho bolão, porém, de uma maneira inovadora através de um sistema web.

* 1. **Objetivo do documento**

Este documento tem como público alvo todos os envolvidos no projeto, desde sua análise até a concepção, com o intuito de guiar a todos em direção a seu objetivo e, consequentemente, sua conclusão satisfatória.

* 1. **Escopo do Produto**

A Anguloweb é uma empresa de natureza *startup*, com foco no desenvolvimento de soluções web personalizadas e sob encomenda.

O objetivo do projeto é elevar a audiência de seu domínio na internet, gerando visibilidade e credibilidade à empresa, para alavancar o crescimento da mesma através da obtenção de novos clientes e projetos.

O projeto tem como foco a diversão, e com integração as redes sociais, pretendemos obter engajamento dos usuários, levando a um aumento exponencial dos novos registros e, consequentemente, dos acessos.

1. **Descrição Geral do Produto**
   1. **Interfaces com o Produto**

|  |  |
| --- | --- |
| **Ator** | **Descrição** |
| *Administrador* | *Usuário do sistema que deverá moderar e gerenciar os usuários e definir políticas de uso e pontuação das apostas.* |
| *Operador* | *Usuário do sistema que deverá manter os cadastros da aplicação atualizados e atender as solicitações dos usuários apostadores.* |
| *Apostador* | *Usuário final do sistema que fará as apostas do bolão e receberá pontos de acordo com seus resultados.* |
| *App do Facebook* | *Rede social integrada através de um aplicativo para possibilitar o cadastro de novos apostadores de forma automatizada, divulgação de resultados e divulgação do sistema.* |
| *Twitter* | *Rede social integrada para possibilitar que o apostador divulgue seus resultados.* |
| *Contas do Google* | *Integração para possibilitar o cadastro de novos apostadores de forma automatizada.* |

* 1. **Requisitos Funcionais**

**[RF001] Gerenciar cadastro de administradores**

O sistema deverá permitir incluir, alterar, excluir, e consultar os usuários administradores. O cadastro deve conter um identificador único gerado pelo sistema, nome, e-mail e senha.

Obs.: Por padrão, já existe um usuário administrador cadastrado com e-mail *admin.bolao@anguloweb.com.br* e senha padrão *123mudar*.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Administrador.* | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF002] Gerenciar cadastro de operadores**

O sistema deverá permitir incluir, alterar, excluir, e consultar os usuários operadores. O cadastro deve conter um identificador único gerado pelo sistema, nome, e-mail e senha.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Administrador.* | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF003] Gerenciar cadastro de campeonatos**

O sistema deverá permitir incluir, alterar, excluir, e consultar os campeonatos. O cadastro deve conter um identificador único gerado pelo sistema, nome, escopo (regional, estadual, nacional, continental ou mundial), data de início, data de fim, tipo (pontos ou mata-mata) e observações.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Administrador, Operador.* | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF004] Gerenciar cadastro de times**

O sistema deverá permitir incluir, alterar, excluir, e consultar os times. O cadastro deve conter um identificador único gerado pelo sistema, tipo (time ou seleção), nome e local de origem. O local de origem depende do campo *tipo:* Se for time, considera-se um campo composto por cidade, estado e país. Se for uma seleção, considera-se apenas o país.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Administrador, Operador.* | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF005] Gerenciar cadastro de partidas**

O sistema deverá permitir incluir, alterar, excluir, e consultar as partidas. O cadastro deve conter um identificador único gerado pelo sistema, o identificador de ambos os times, a data e hora da partida e a quantidade de gols marcados por cada time.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Administrador, Operador.* | | | | | |
| **Requisitos associados:** | [RF004] | | | | | |

**[RF006] Cadastrar novo usuário apostador**

O sistema deverá permitir o cadastro de novos usuários apostadores em sua página inicial através de preenchimento manual de formulário web ou integração com conta do Facebook ou Google. O cadastro deve conter um identificador único gerado pelo sistema, nome, sobrenome, apelido, e-mail, data de nascimento, senha e foto.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Apostador, App do Facebook, Contas do Google.* | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF007] Gerenciar cadastro de apostadores**

O sistema deverá permitir que o apostador, desde que esteja logado, exclua ou altere os dados de sua conta.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Apostador.* | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF008] Gerenciar cadastro de palpite**

O sistema deverá permitir o apostador incluir, alterar, excluir, e consultar seus palpites de resultado para cada jogo. A alteração e exclusão podem ser feitas até o momento de início do jogo.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Apostador.* | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF009] Cadastrar placar e distribuir pontos**

Ao fim de cada partida, os usuários operadores serão responsáveis em registrar o resultado do jogo, e ao salvar o sistema deverá processar esse resultado e distribuir os pontos aos apostadores que deram um palpite em determinada partida. O apostador recebe dois pontos caso acerte o placar exato e um ponto caso acerte o time vencedor.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Operador* | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF010] Ranking de apostadores**

O sistema deverá disponibilizar de forma pública uma página com o ranking dos apostadores classificado em ordem da quantidade de pontos de forma decrescente. A página deve conter a classificação, nome e quantidade de pontos de cada apostador. Caso o apostador esteja logado, a posição dele deve estar destacada.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** |  | Essencial | X | Importante |  | Desejável |
| **Ator (es):** | *Apostador.* | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF011] Consulta fácil de palpites**

O sistema deverá possuir uma tela inicial para o apostador constando seus palpites, os resultados dos jogos já realizados, dados de registro, posição no ranking e quantidade de pontos.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Apostador.* | | | | | |
| **Requisitos associados:** |  | | | | | |

**[RF012] Tela de acesso (login)**

O sistema deverá possuir uma tela inicial para o apostador/operador logar no sistema e então realizar suas ações conforme a permissão atribuída.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Ator (es):** | *Operador, Apostador.* | | | | | |
| **Requisitos associados:** |  | | | | | |

* 1. **Requisitos Não Funcionais**

**[NF001] Plataforma web**

O sistema deverá ser implantado em plataforma web, para que seja possível acesso de qualquer lugar, sem a necessidade de instalação do software e também através de dispositivos móveis.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Requisitos associados:** |  | | | | | |

**[NF002] Integração com redes sociais**

O sistema deverá ser integrado a redes sociais para facilitar a divulgação e engajamento dos usuários.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Requisitos associados:** |  | | | | | |

1. **Estudo de Viabilidade**
   1. **Quanto as contribuições à empresa**

O projeto visa, para o cliente, gerar visibilidade ao seu domínio, uma crescente em termos de usuários e credibilidade perante aos mesmos. Visando também garantir futuros novos clientes e projetos. Pois a partir do objetivo traçado, o usuário final poderá interagir e, muitas vezes, se informar sobre o esporte predileto da maioria dos brasileiros, o futebol.

* 1. **Quanto a viabilidade econômica**

O projeto acaba por ter um custo relativamente baixo, pois é feito através de um sistema web, deixando de lado problemas mais custosos e físicos. Ressaltando que mesmo por tal motivo ainda há custos tal qual: recursos humanos, manutenção.

* 1. **Quanto a viabilidade técnica**

A tecnologia atual oferece uma gama de recursos que pode contribuir com a construção do projeto. Estamos em um momento que está sendo muito discutido desenvolvimento web, armazenamento em nuvem e tantos outros assuntos relativos ao tema.

Em relação a alguma restrição de custo de processamento, cremos que não haverá maiores problemas, no nosso entendimento os requisitos para o uso do sistema é básico, praticamente nulo – necessitando apenas acesso à internet e um navegador web.

A questão do prazo é um ponto importante, pois como se trata de um evento com data estabelecida, existe um prazo válido para a entrega do projeto. Mas considerando as necessidades e o tempo hábil, cremos que o tempo é suficiente.

* 1. **Quanto a legalidade**

O projeto é baseado em um sistema de palpites, que não visa lucro direto ao cliente ou, de forma alguma, prejuízo ao usuário. Tratando-se apenas de uma opção de descontração e informação. Em nenhum momento foi constatada alguma infração ou ilegalidade em tempo de desenvolvimento do projeto.

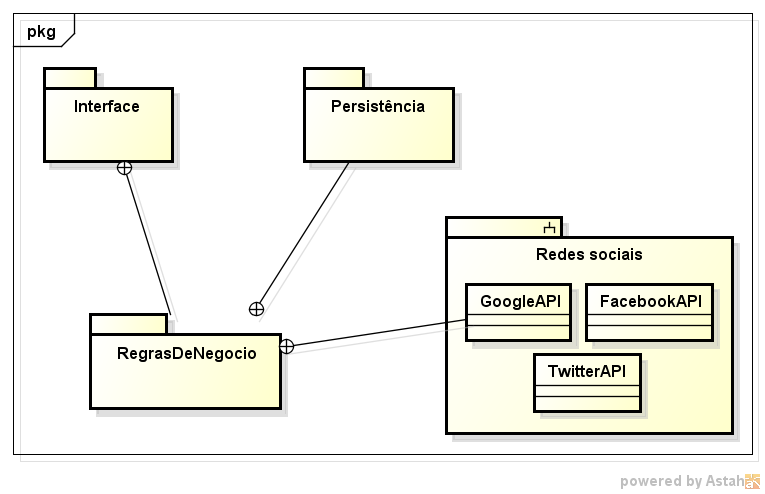
* 1. **Quanto as alternativas**

O desenvolvimento web nos oferece muitas possibilidades, dentre as que mais se sobressaíram durante o estudo foram as linguagens PHP e JAVA. As duas nos oferecem codificação no servidor, um ponto importante para o desenvolvimento que visamos. Revisados todos os nossos fatores que colocamos em consideração, chegamos a decisão em usar a tecnologia de php. Por sua familiaridade entre os desenvolvedores do projeto, por não ser um projeto de grande porte, oferecer mais rapidez em seu desenvolvimento, entre outros fatores menos específicos.

1. **Avaliação dos Riscos**

* **Risco de projeto**
  + Falta de tempo, uma vez que somos todos universitários e trabalharemos no contra turno.
  + Como os nossos usuários são pessoas aleatórias, não temos como fazer um estudo muito aproximado do que irá agrada-lo e a falta desse estudo poderá gerar uma constante mudança no escopo do projeto.
* **Risco técnico**
  + Falta de informação, caso aconteça de não conseguirmos informações com tempo suficiente para cadastrar os jogos no Bolão WEB e os apostadores fazerem suas apostas.

1. **Diagrama de pacotes**



1. **Arquitetura de software**

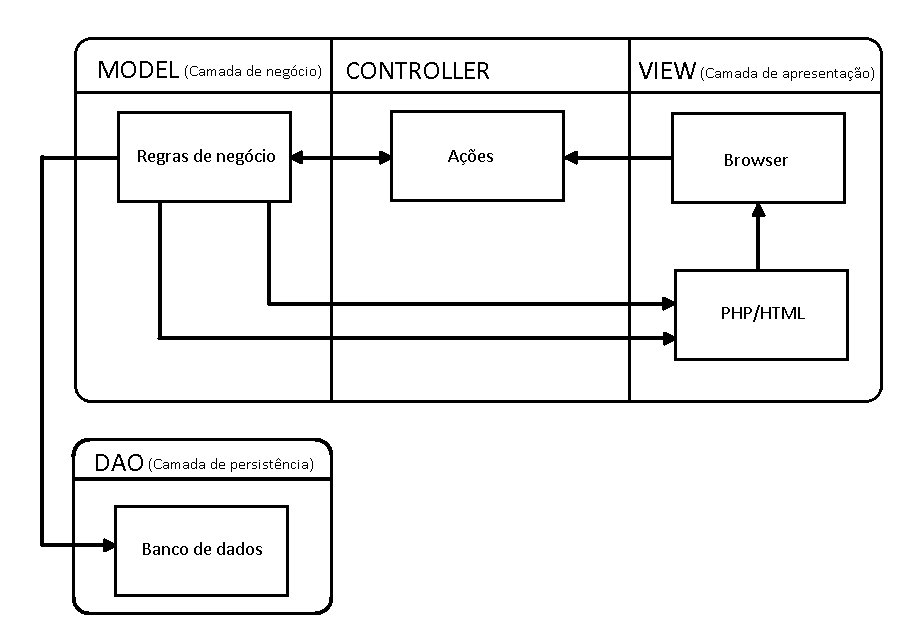
Definimos que o tipo de arquitetura de sistema que utilizaremos para o desenvolvimento do sistema web é o *Cliente x Servidor*. Onde especificamos que:

* **Cliente**: através de um *browser* (navegador web), as estações clientes (usuários), poderão ter acesso ao pacote de servidor;
* **Servidor**: Servidor web que será o responsável pelo tratamento dos dados enviados pelas estações clientes através do *browser;*
* **Sistema Gerenciador de Banco de dados**: Será o responsável em armazenar os dados passados pelo servidor.

Como padrão arquitetônico, definimos que será utilizado o *Thin Client* (Cliente Magro). Basicamente o servidor que fará toda a lógica de negócio, restando a estação cliente apresentar os dados através de uma página dinâmica.

Utilizaremos a arquitetura em camadas, onde podemos dividi-las em três camadas, conforme apresentado a seguir:

* **Camada de Apresentação:** Será a camada responsável com a interação com o usuário, a partir de um navegador. Responsável pela entrada de dados, requisitar os serviços ao servidor e apresentar os resultados;
* **Camada de Aplicação**: Será a responsável em definir e realizar as lógicas de negócio, no lado do servidor. Ressaltando que utilizaremos o padrão de projeto *Model View Controller* (Modelo, Visão e Controle);
* **Camada de Persistência**: Será a responsável em armazenar os dados que serão tratados pela camada de aplicação, os quais recebidos pela camada de aplicação.

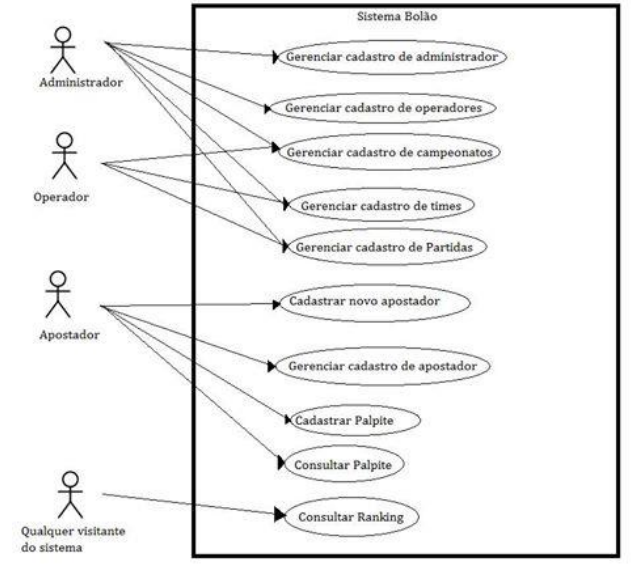
A seguir ilustraremos, com um diagrama, como será a arquitetura que visamos para desenvolver o projeto web:

1. **Elaboração**
   1. **Modelo de casos de uso**

**Identificação e priorização dos casos de uso**

1. USE CASE<gerenciar cadastro de administradores> - Danilo
2. USE CASE<gerenciar cadastro de operadores> - Juan
3. USE CASE<gerenciar cadastro de campeonatos> - Alison
4. USE CASE<gerenciar cadastro de times> - Lucio
5. USE CASE<gerenciar cadastro de partidas> - Danilo
6. USE CASE<cadastrar novo usuário apostador> - Juan
7. USE CASE<gerenciar cadastro de apostador> - Alison
8. USE CASE<gerenciar cadastro de palpite> - Lucio
9. USE CASE<processar resultados e distribuir pontos> - Danilo
10. USE CASE<consultar ranking de apostadores> - Alison
11. USE CASE<consulta fácil de palpites> - Lucio

**Diagrama de casos de uso**



**Detalhamento dos casos de uso**

**USE CASE<gerenciar cadastro de administradores> - Danilo**

**Objetivo:**

Permitir ao usuário administrador incluir, alterar, excluir, e consultar outros usuários administradores.

**Pré-condições:**

O usuário deve estar logado no sistema e possuir permissão de administrador.

**Iniciado por:**

Usuário administrador devidamente logado no sistema, com a intenção de incluir, alterar, excluir ou consultar outros usuários administradores.

**Fluxo principal:**

1. O usuário acessa a opção de gestão de administradores.
2. O sistema abre a tela com as opções incluir, alterar, excluir e consultar administradores.
3. O usuário seleciona a opção para incluir novo usuário.
4. O sistema exibe o formulário de cadastro.
5. O usuário informa o nome, e-mail e senha do novo usuário e seleciona a opção para gravação do novo usuário.
6. O sistema gera o identificador do usuário
7. O sistema salva as informações no banco de dados.
8. O sistema exibe uma mensagem informando o sucesso da operação e que vai enviar uma mensagem de confirmação de registro para o e-mail informado.
9. O sistema envia uma mensagem de confirmação de registro para o e-mail informado.
10. O usuário recebe o e-mail e acessa o link de confirmação.
11. O sistema atualiza o usuário como ativo e mostra uma mensagem informando o sucesso da operação

**Fluxo alternativo: Alterar usuário administrador**

No item 3, caso o usuário selecione a opção de alterar um usuário, é executado o fluxo alternativo de consulta, e ao fim, a execução volta para o item 4, porém, com todos os dados já previamente preenchidos para edição. Não é executado o item 6.

**Fluxo alternativo: Excluir usuário administrador**

No item 3, caso o usuário selecione a opção de excluir um usuário, é executado o fluxo alternativo de consulta, e ao fim, a execução volta para o item 4, porém com todos os dados já previamente preenchidos e com uma mensagem de confirmação exclusão. Após isso, é executado o passo 8.

**Fluxo alternativo: Consultar usuário administrador**

No item 3, caso o usuário selecione a opção de consultar um usuário, é exibido um diálogo solicitando o campo para ser pesquisado (id, nome ou e-mail) e o valor a ser pesquisado. Ao fim, a execução volta para o item 4, porém com todos os dados já previamente preenchidos e opções para o usuário alterar ou excluir o usuário.

**Fluxo alternativo: E-mail duplicado ou inválido**

Ao fim do item 5, caso o sistema identifique que o e-mail informado já existe, está em branco ou é inválido, deve ser exibida uma mensagem de erro e abortado o processo. A execução retorna para o item 5.

**Fluxo alternativo: Excluir o ultimo, ou o próprio usuário administrador**

No fluxo alternativo de exclusão, caso o sistema identifique que o usuário que será excluído é o ultimo do sistema, ou seja, o próprio executor do processo. Deve ser exibida uma mensagem de erro e abortado o processo. A execução retorna para o item 5.

**Pós-condições:**

Um usuário administrador foi criado, excluído, alterado ou consultado caso o usuário não tenha abortado o processo.

**USE CASE<gerenciar cadastro de operadores> - Juan**

**Objetivo:**

Permite o gerenciamento e manipulação (incluir, alterar e consultar) de usuários responsáveis pela alimentação das informações operacionais do sistema. Estas informações operacionais são pertinentes ao lançamento de resultados, gerenciamento de times, gerenciamento de partidas, gerenciamento de campeonatos e processamento de apostas.

**Pré-condições:**

O usuário deve estar logado no sistema e possuir permissão de administrador.

**Iniciado por:**

Qualquer usuário administrador no intuito de efetuar uma inclusão, alteração ou consulta por usuários operacionais.

**Fluxo principal:**

1. O usuário administrador acessa a opção de controle de usuários operacionais;
2. O sistema mostra a tela para escolha da rotina que será manipulada (inclusão, alteração ou consulta);
3. O usuário escolhe a opção para inclusão de novo usuário operacional;
4. O sistema mostra o formulário de inclusão de novo usuário operacional;
5. O usuário informa os dados de inclusão de novo usuário, com atenção especial aos atributos obrigatórios;
6. O sistema valida a corretude dos atributos, e verifica o preenchimento dos atributos obrigatórios;
7. O sistema grava as informações na devida tabela do BD;
8. O sistema retorna uma mensagem para a tela do usuário administrador confirmando a operação.
9. O sistema envia uma mensagem para o e-mail cadastrado informando as credenciais (login e senha) de usuário operador.
10. O sistema solicitará alteração de senha no primeiro acesso do usuário operador;
11. O usuário operador insere uma nova senha e confirma a operação;
12. O sistema envia uma mensagem para a tela do usuário operador informando o sucesso da operação e atualiza o usuário como *ativo*.

**Fluxo alternativo: Alteração de usuário operacional**

No item 2, caso o usuário escolha a opção *Alteração*, o usuário administrador deverá poder buscar o usuário operador já cadastrado para fazer a alteração. O sistema deverá carregar no formulário de inscrição os dados do usuário operador desejado e o fluxo é continuado a partir do item 4.

**Fluxo alternativo: Consulta de usuário operacional**

No item 2, caso o usuário escolha a opção *Consulta*, o usuário administrador deverá poder buscar os dados do usuário operador somente para visualização. É mostrada numa tela todos os dados do usuário operador somente para consulta e o fluxo é interrompido.

**Fluxo alternativo: Validação de atributos**

É comum ocorrer problemas de validação de formulários e falta de preenchimento de atributos obrigatórios. No processamento do item 6, caso o sistema encontre qualquer empecilho, o sistema exibe uma mensagem de erro e informa quais atributos devem ser revistos. O fluxo só é retomado após a aprovação do item 6.

**Pós-condições:**

Um usuário administrador deve ser criado e estar devidamente ativo.

**USE CASE<gerenciar cadastro de campeonatos> - Alison**

**Objetivo:**

O sistema deverá permitir incluir, alterar, excluir, e consultar os campeonatos através da página do Bolão. O cadastro deve conter um identificador único gerado pelo sistema, nome, escopo (regional, estadual, nacional, continental ou mundial), data de início, data de fim, tipo (pontos ou mata mata) e observações.

**Pré-Condições:**

O cadastramento de Campeonato é uma atividade exclusiva para usuários Administradores ou Operadores.

**Iniciado Por:**

O Administrador ou o Operador, que estiver designado a cadastrar os campeonatos, fará o cadastro manualmente através do site.

**Fluxo principal:**

1. O usuário acessa a página iniciado do sistema, o site do Bolão.
2. O usuário seleciona a opção de cadastrar campeonato
3. O sistema apresenta um formulário de cadastramento
4. O usuário preenche o formulário de campeonato e submete o formulário
5. O sistema valida os dados cadastrais
6. O sistema reapresenta os dados ao usuário e pede confirmação
7. O usuário confirma os dados
8. O sistema efetua a inclusão dos dados informados no cadastro de campeonato
9. O sistema agradece o cadastramento e convida o usuário a cadastrar um novo campeonato.

**Fluxo alternativo:** Corrigir dados cadastrais

No item 5, o sistema ao validar os dados cadastrais, detecta que alguns dados considerados como obrigatórios não foram preenchidos ou que dados foram preenchidos incorretamente.

5a. O sistema reapresenta os dados cadastrais indicando o(s) erro(s)

5b O usuário corrige o(s) erro(s) cadastrais

Retorna ao fluxo principal no item 6

**Fluxo alternativo:** Cancelar operação

Nos itens 2 ou 4 ou 7, o usuário cancela a operação. O sistema retorna à página inicial do sistema.

**Fluxo alternativo:** Exceder tempo

No item 7, caso o usuário não confirme os dados e fique mais de 10 minutos inativo, a operação é cancelada pelo sistema. Nesse caso o sistema retorna à sua página inicial.

**Pós-condições:**

O campeonato foi cadastrado no sistema ou o sistema exibiu a sua página inicial, por cancelamento da operação.

**USE CASE<gerenciar cadastro de times> - Lucio**

**Objetivo:**

Permitir ao usuário realizar a inscrição, alteração, excluir e consultar as esquipes no sistema.

**Pré-Condições:**

Usuário logado no sistema e com permissões ao nível operacional ou de administrador.

**Iniciado Por:**

Usuário com nível de permissão devidamente logado no sistema, com a intenção de incluir, alterar, excluir ou consultar as equipes no sistema.

**Fluxo principal:**

1. O usuário seleciona a opção de gerenciamento de times;
2. O sistema abre a tela com as opções incluir, alterar, excluir e consultar as equipes;
3. O usuário seleciona a opção para inserir, alterar ou excluir um time no sistema;
4. O sistema retorna ao usuário uma tela de formulário para inserção, alteração ou exclusão;
5. O usuário informa que tipo de equipe será inclusa, alterada ou excluída, se clube ou federação (país);
6. O usuário informa o nome, dados de origem da equipe e seleciona a opção para gravação no sistema;
7. O sistema gera o identificador do time inscrito, caso inclusão, ou gera “decremento” do identificador, caso exclusão;
8. O sistema salva as informações no banco de dados;
9. O sistema exibe uma mensagem de confirmação da operação.

**Fluxo alternativo**: Cancelar cadastro.

Em qualquer momento, o usuário pode cancelar o cadastro da equipe, com o sistema retornando ao início do formulário com os campos vazios.

**Fluxo alternativo**: Dados da equipe

No item seis, o sistema ao validar encontra uma inconsistência, sendo um dado inválido, time já inserido ou a falte de um campo requerido.

**Pós-condições**:

Um time foi criado, excluído, alterado ou consultado caso o usuário não tenha abortado o processo.

**USE CASE<gerenciar cadastro de partidas> - Danilo**

**Objetivo:**

Permitir ao usuário administrador ou operador incluir, alterar, excluir, e consultar as partidas.

**Pré-condições:**

O usuário deve estar logado no sistema, deve existir ao menos um campeonato cadastrado e ao menos dois times cadastrados.

**Iniciado por:**

Usuário administrador ou operador com a intenção de incluir, alterar, excluir ou consultar as partidas.

**Fluxo principal:**

1. O usuário acessa a opção de gestão de partidas.
2. O sistema abre a tela com as opções incluir, alterar, excluir e consultar partidas.
3. O usuário seleciona a opção para incluir nova partida.
4. O sistema exibe o formulário de cadastro.
5. O usuário seleciona o campeonato na qual a partida faz parte, o time da casa e o time visitante, a data e hora da partida.
6. O sistema gera o identificador da partida.
7. O sistema salva as informações no banco de dados.
8. O sistema exibe uma mensagem informando o sucesso da operação.

**Fluxo alternativo: Alterar partida**

No item 3, caso o usuário selecione a opção de alterar uma, é executado o fluxo alternativo de consulta, e ao fim, a execução volta para o item 4, porém, com todos os dados já previamente preenchidos para edição. Não é executado o item 6.

**Fluxo alternativo: Excluir partida**

No item 3, caso o usuário selecione a opção de excluir uma partida, é executado o fluxo alternativo de consulta, e ao fim, a execução volta para o item 4, porém com todos os dados já previamente preenchidos e com uma mensagem de confirmação exclusão. Após isso, caso a partida não possua nenhum palpite vinculado, é executado o passo 8.

**Fluxo alternativo: Consultar partida**

No item 3, caso o usuário selecione a opção de consultar uma partida, é exibido um diálogo solicitando o campo para ser pesquisado (id, nome do time visitante, nome do time da casa, nome de ambos os times, data e hora da partida) e o valor a ser pesquisado. Ao fim, a execução volta para o item 4, porém com todos os dados já previamente preenchidos e opções para o usuário alterar ou excluir o registro.

**Fluxo alternativo: Data e hora da partida são inferiores a atual**

Antes do item 7, caso o sistema identifique que a data e hora da partida são menor que a atual (partida já ocorreu), o sistema deve obrigar o usuário a informar o placar do hoje antes de salvar o registro.

**Pós-condições:**

Uma partida foi criada, excluída, alterada ou consultada caso o usuário não tenha abortado o processo.

**USE CASE<cadastrar novo usuário apostador> - Juan**

**Objetivo:**

Permite que um devido apostador se cadastre junto ao sistema para efetuar suas apostas. As apostas só podem ser realizadas após o devido cadastro e identificação do apostador em questão.

**Pré-condições:**

Deverá haver alguma competição ou partida disponível para aposta.

**Iniciado por:**

Pelo apostador que esteja interessado em realizar uma aposta nas partidas disponíveis no sistema.

**Fluxo principal:**

1. O apostador acessa a url do sistema (http://bolao.anguloweb.com.br)
2. O sistema mostra a tela inicial o sistema (index.php) onde o apostador pode escolher entre “*Fazer login*” caso já tenha sido cadastrado, e “*Efetuar cadastro*” para realizar seu cadastro.
3. O apostador escolhe a opção “*Efetuar cadastro*”.
4. O sistema mostra o formulário de cadastro de novo apostador;
5. O apostador informa seus dados pessoais necessários para inclusão, com atenção especial aos atributos obrigatórios;
6. O sistema valida a corretude dos atributos, e verifica o preenchimento dos atributos obrigatórios;
7. O sistema grava as informações na devida tabela do BD;
8. O sistema retorna uma mensagem para a tela do usuário administrador confirmando a operação.
9. O sistema envia uma mensagem para o e-mail cadastrado informando as credenciais (login e senha) para o usuário normal e um link para ativação de cadastro.
10. O usuário abre o e-mail e clica no link de ativação.
11. O sistema envia uma mensagem para a tela do apostador informando o sucesso da operação e atualiza o status como *ativo*.

**Fluxo alternativo: Validação de atributos**

É comum ocorrer problemas de validação de formulários e falta de preenchimento de atributos obrigatórios. No processamento do item 6, caso o sistema encontre qualquer empecilho, o sistema exibe uma mensagem de erro e informa quais atributos devem ser revistos. O fluxo só é retomado após a aprovação do item 6.

**Pós-condições:**

O sistema deverá estar permitindo o cadastro de apostadores.

**USE CASE<gerenciar cadastro de apostador> - Alison**

**Objetivo:**

O sistema deverá permitir que um apostador altere ou exclua os dados de sua conta sempre que estiver logado.

**Pré-Condições:**

O usuário deve estar logado.

**Iniciado Por:**

O Apostador, quando necessário, através do site do Bolão.

**Fluxo principal:**

1. O usuário acessa a página iniciado do sistema, o site do Bolão.
2. O usuário seleciona a opção de gerenciar cadastro de apostador
3. O sistema apresenta um formulário com as informações do usuário
4. O usuário altera ou exclui os dados
5. O sistema valida a alteração/exclusão dos dados
6. O sistema reapresenta os dados ao usuário e pede confirmação
7. O usuário confirma a alteração/exclusão
8. O sistema efetua a alteração/exclusão dos dados do usuário
9. O sistema agradece a atualização dos dados e volta à página inicial do sistema

**Fluxo alternativo:** Corrigir dados

No item 5, o sistema ao validar os dados, detecta que alguns dados considerados como obrigatórios não foram preenchidos ou que dados foram preenchidos incorretamente.

5a. O sistema reapresenta os dados indicando o(s) erro(s)

5b O usuário corrige o(s) erro(s)

Retorna ao fluxo principal no item 6

**Fluxo alternativo:** Cancelar operação

Nos itens 2 ou 4 ou 7, o usuário cancela a operação. O sistema retorna à página inicial do sistema.

**Fluxo alternativo:** Exceder tempo

No item 7, caso o usuário não confirme os dados e fique mais de 10 minutos inativo, a operação é cancelada pelo sistema. Nesse caso o sistema retorna à sua página inicial.

**Pós-condições:**

Os dados do usuário foram alterados no sistema ou o sistema exibiu a sua página inicial, por cancelamento da operação.

**USE CASE<gerenciar cadastro de palpite> - Lucio**

**Objetivo:**

Permitir que o usuário tente adivinhar o placar dos jogos que estiverem disponíveis no site do Bolão.

**Pré-Condições:**

O usuário deve estar logado no sistema.

**Iniciado Por:**

De acordo com cada partida cadastrada pelo Administrador no sistema, o usuário poderá definir seu palpite até um dia antes da rodada corrente.

**Fluxo principal:**

1. O usuário acessa a página inicial do sistema, leia-se o site do Bolão;
2. <<include>>Autenticar usuário;
3. O usuário seleciona a opção no sistema de fazer o palpite;
4. O usuário seleciona a partida na qual deseja cadastrar um palpite;
5. O usuário informa o palpite da partida selecionada;
6. O sistema mostra as informações dos dados inseridos e das partida selecionadas;
7. O usuário confirma os dados passados ao sistema;
8. O sistema realiza a inclusão dos dados informados no cadastro de palpite na partida desejada;
9. O sistema confirma o palpite e convida o usuário a fazer um novo ou retornar a página inicial do site.

**Fluxo alternativo:** Corrigir dados do palpite

No item 7, o sistema ao validar os dados do palpite, detecta que alguns dados considerados como obrigatórios não foram preenchidos ou que dados foram preenchidos incorretamente.

5a. O sistema reapresenta os dados do palpite indicando o(s) erro(s);

5b. O usuário corrige o(s) erro(s) do palpite;

Retorna ao fluxo principal no item 6.

**Fluxo alternativo:** Cancelar operação

Nos itens 3 ou 4 ou 5 ou 7, o usuário cancela a operação. O sistema retorna à página inicial do sistema.

**Fluxo alternativo:** Exceder tempo

No item 7, caso o usuário não confirme os dados e fique mais de 10 minutos inativo, a operação é cancelada pelo sistema. Nesse caso o sistema retorna à sua página inicial.

**Pós-condições:**

O palpite do usuário foi cadastrado no sistema ou o sistema exibiu a sua página inicial, por cancelamento da operação.

**USE CASE<processar resultados e distribuir pontos> - Danilo**

**Objetivo:**

Ao fim de uma partida, os operadores são alertados e devem cadastrar o placar e o sistema deve percorrer todos os palpites da respectiva partida distribuindo aos apostadores dois pontos caso acerte o placar exato e um ponto caso acerte o time vencedor.

**Pré-condições:**

Existir uma partida com pelo menos um palpite e com data e hora inferior a atual.

**Iniciado por:**

Processo do sistema que verifica constantemente se existe alguma partida já ocorrida.

**Fluxo principal:**

1. O sistema identifica uma partida com data e hora inferior a atual.
2. O sistema notifica a todos os operadores por e-mail com um link direto para a partida.
3. O usuário recebe o e-mail e acessa o link.
4. O usuário informa o placar do jogo e seleciona a opção para salvar os dados.
5. O sistema atualiza a partida no banco de dados.
6. O sistema exibe uma mensagem informando o sucesso da operação.
7. O sistema carrega todos os palpites relacionados a essa partida.
8. O sistema, para cada palpite, soma aos pontos do apostador dois pontos caso acerte o placar exato e um ponto caso acerte o time vencedor e envia um e-mail ao apostador informando de seus resultados.

**Fluxo alternativo: Cadastro do placar já efetuado**

Após o item 3, caso a partida já possua cadastro de placar, é exibida uma mensagem informando o usuário e solicitando que ele valide e confirme o placar. Após isso, o processo é finalizado.

**Pós-condições:**

A partida já realizada fica com o placar cadastrado e os apostadores que deram palpites nela, recebem os seus pontos.

**USE CASE<consultar ranking de apostadores> - Alison**

**Objetivo:**

O sistema deverá permitir consultar o ranking dos Apostadores através da página do Bolão. A consulta é permitida a qualquer visitante do site. No ranking deve constar classificação, nome e pontos de cada apostador em ordem decrescente da classificação.

**Pré-Condições:**

Nenhuma pré-condição, o usuário simplesmente deve ir ao sistema e realizar a consulta.

**Iniciado Por:**

O qualquer visitante do site.

**Fluxo principal:**

1. O usuário acessa a página iniciado do sistema, o site do Bolão.
2. O usuário seleciona a opção de consultar ranking de apostadores
3. O sistema apresenta o ranking

**Fluxo alternativo:** Mostrar posição

No item 3, caso o usuário logado seja um apostador, a posição dele deverá estar em destaque.

**Pós-condições:**

O ranking foi exibido na tela.

**USE CASE<consulta fácil de palpites> - Lucio**

**Objetivo:**

Permitir ao usuário realizar consultados palpites já inseridos no sistema, assim como as partidas já realizadas da rodada, posição no ranking e a quantidade de pontos.

**Pré-Condições:**

Usuário logado no sistema e com permissões ao nível de apostador.

**Iniciado Por:**

Usuário com nível de permissão devidamente logado no sistema.

**Fluxo principal:**

1. O sistema exibe a tela inicial do site;
2. O usuário seleciona a consulta fácil de palpites;
3. O sistema retorna uma página onde constam as informações de palpites já realizados, pontuação e a posição no ranking geral;
4. O usuário confere os dados e seleciona voltar, retornando a página inicial do sistema.

**Fluxo alternativo:** Exceder tempo

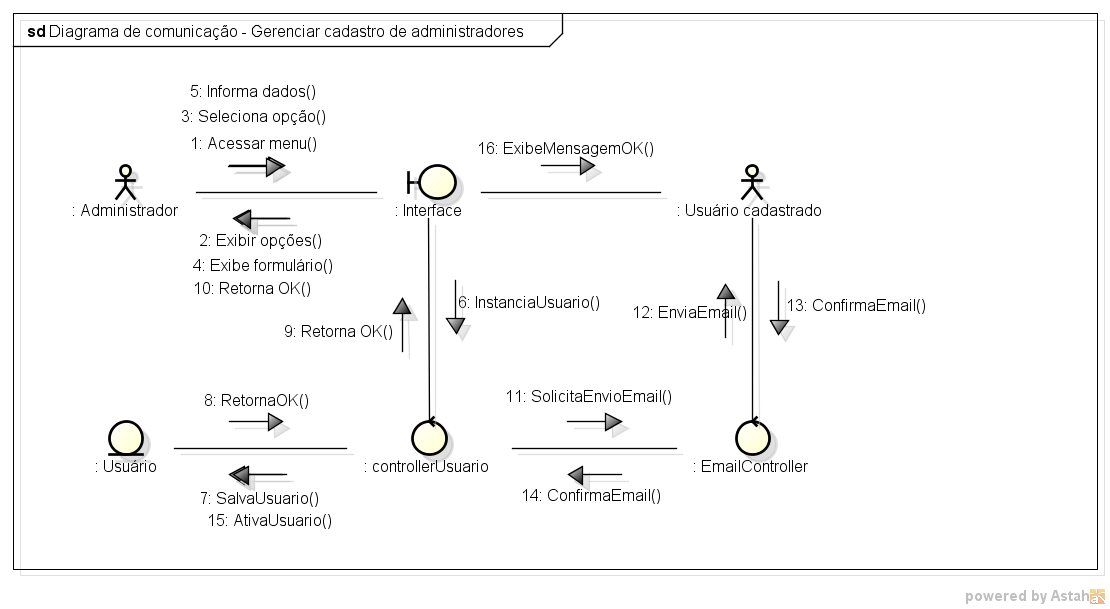
No item 7, caso o usuário não retorne nenhuma ação dentro de 10 minutos, a operação é cancelada pelo sistema. Nesse caso o sistema retorna à sua página inicial.

**Pós-condições:**

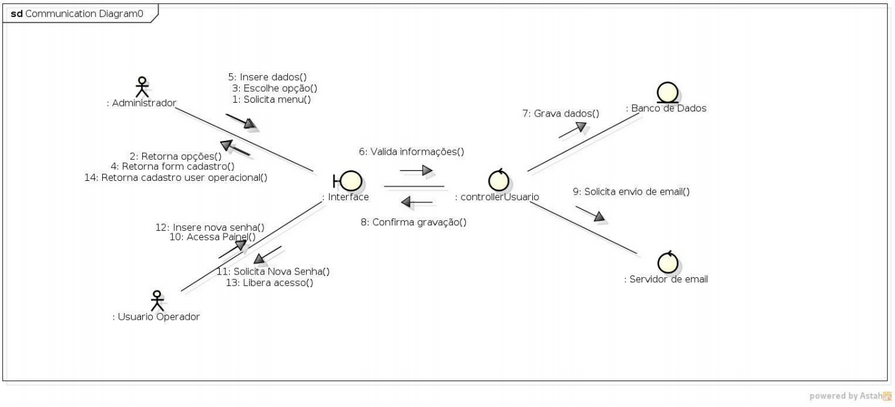
O sistema retorna a página inicial do sistema após o usuário conferir os dados ou exceder o tempo limite.

* 1. **Modelo de análise**

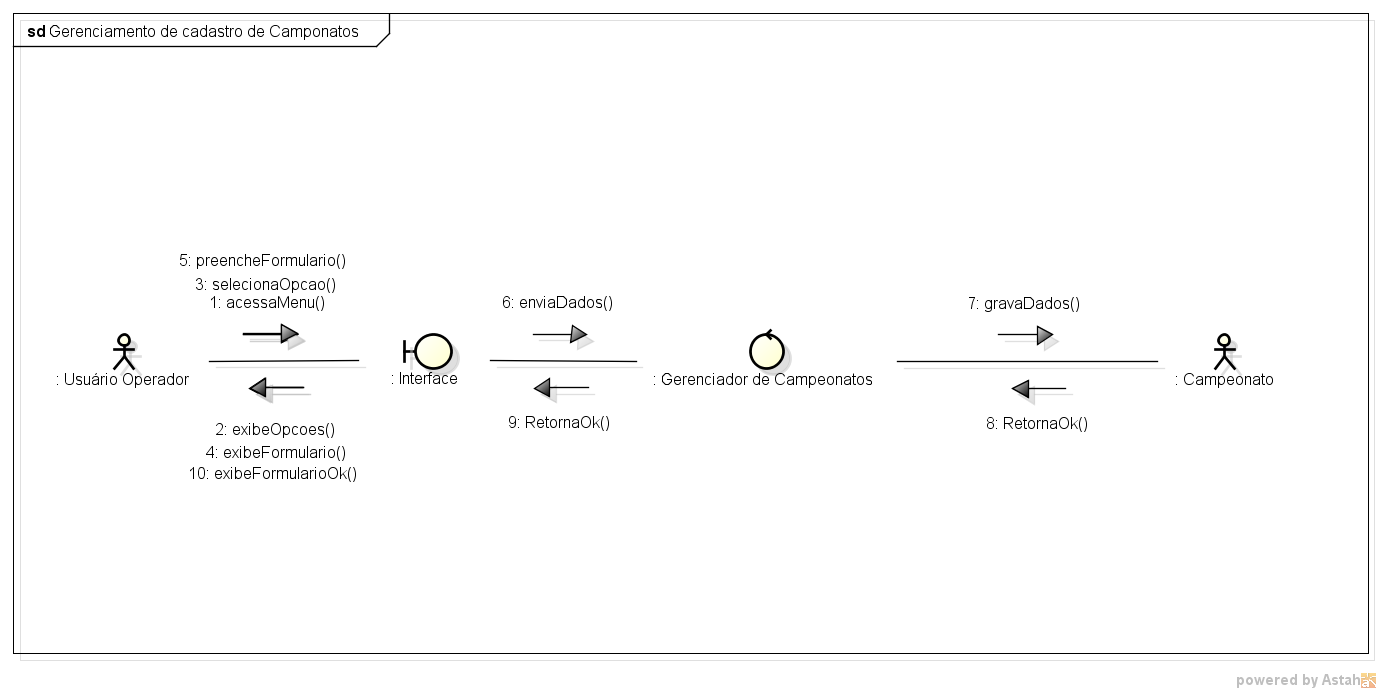
**USE CASE<gerenciar cadastro de administradores> - Danilo**



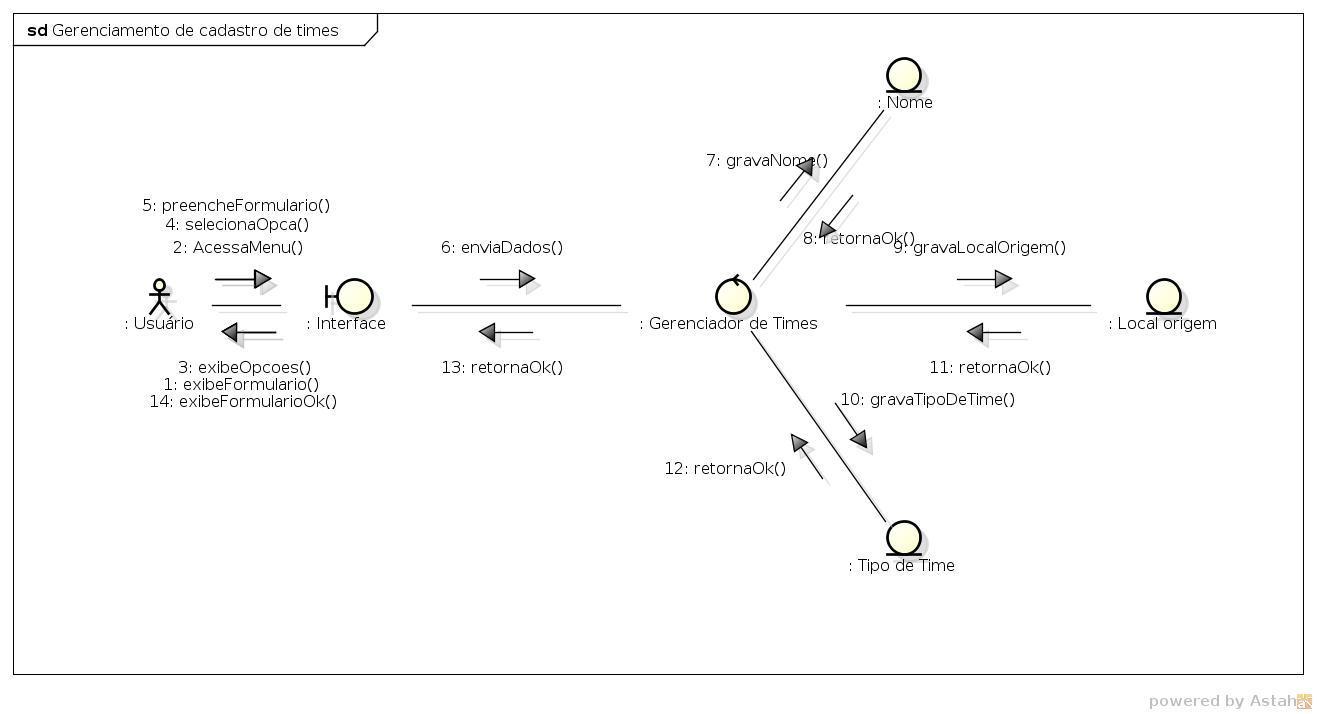
**USE CASE<gerenciar cadastro de operadores> - Juan**



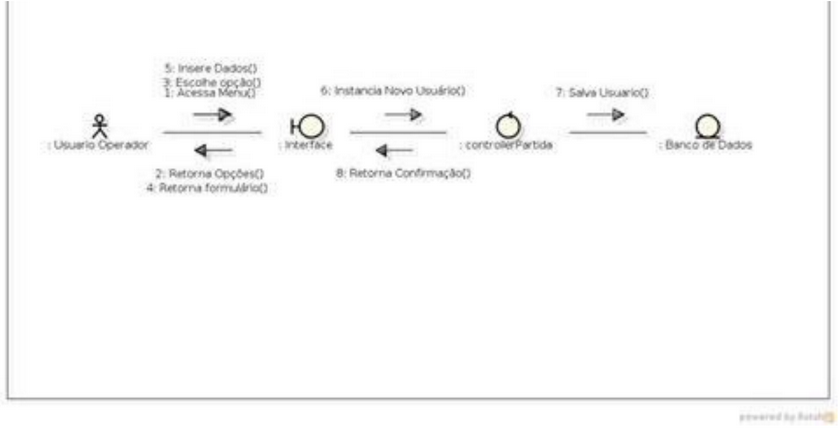
**USE CASE<gerenciar cadastro de campeonatos> - Alison**



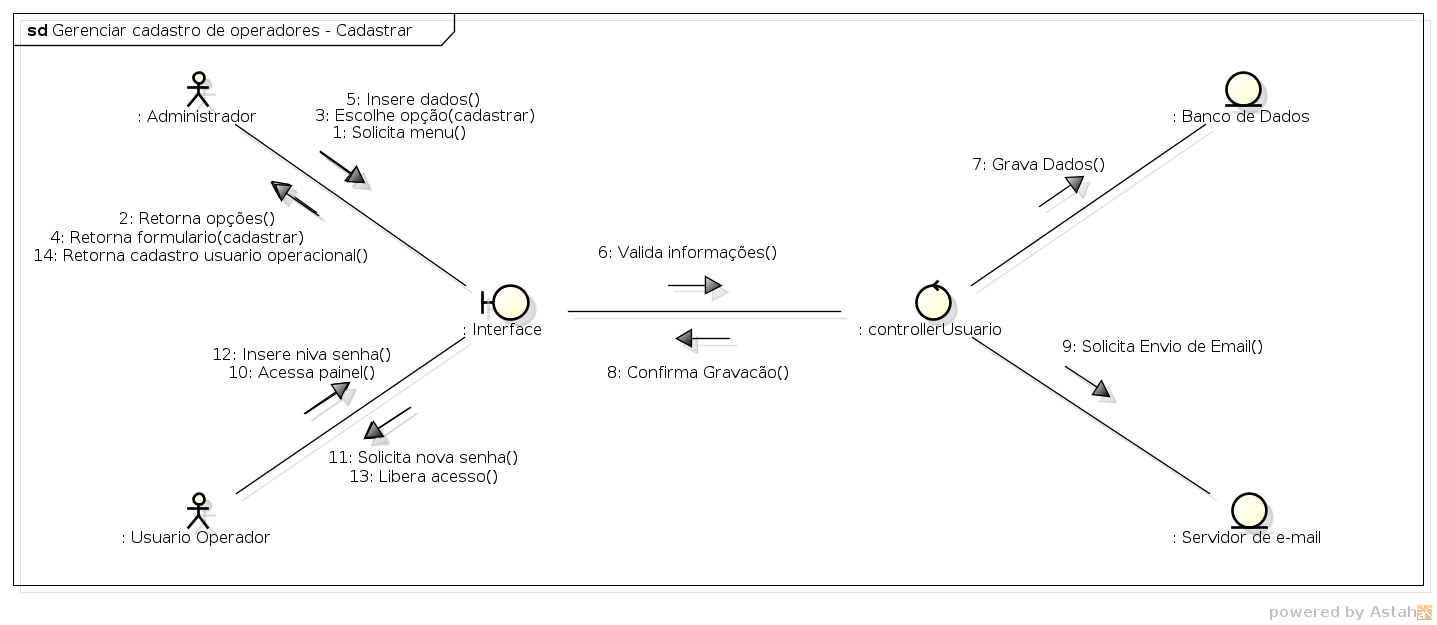
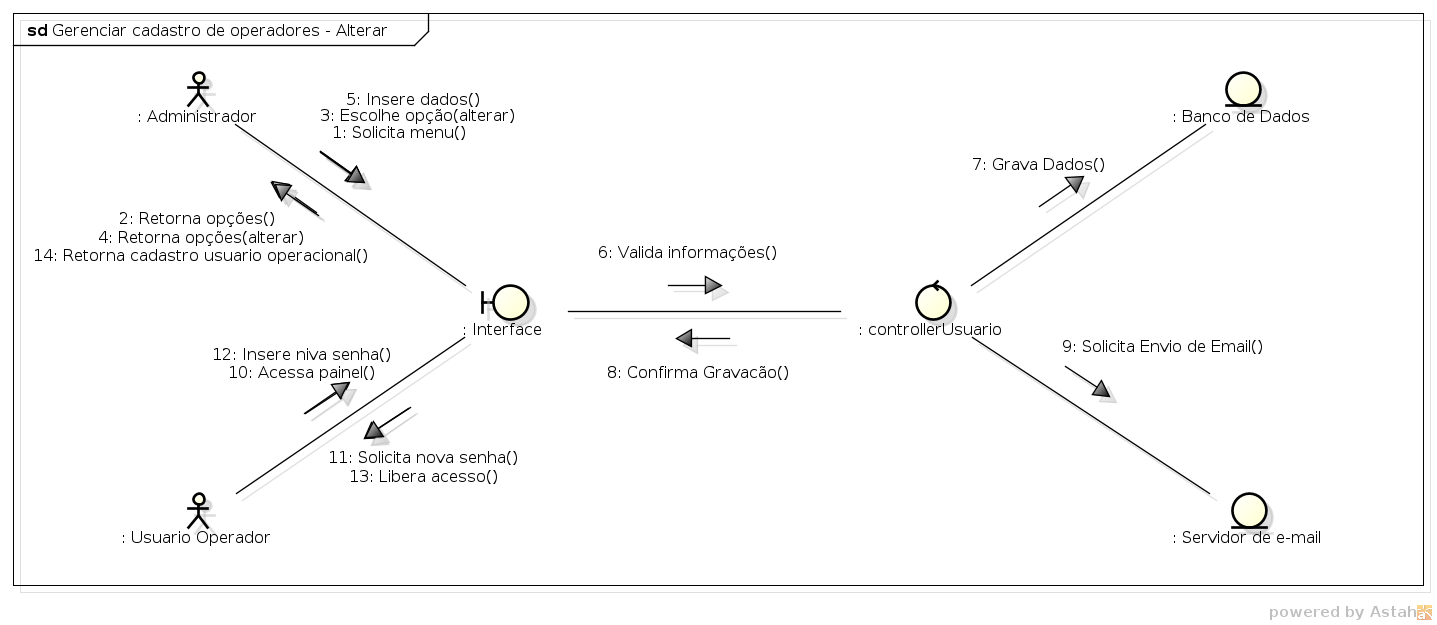
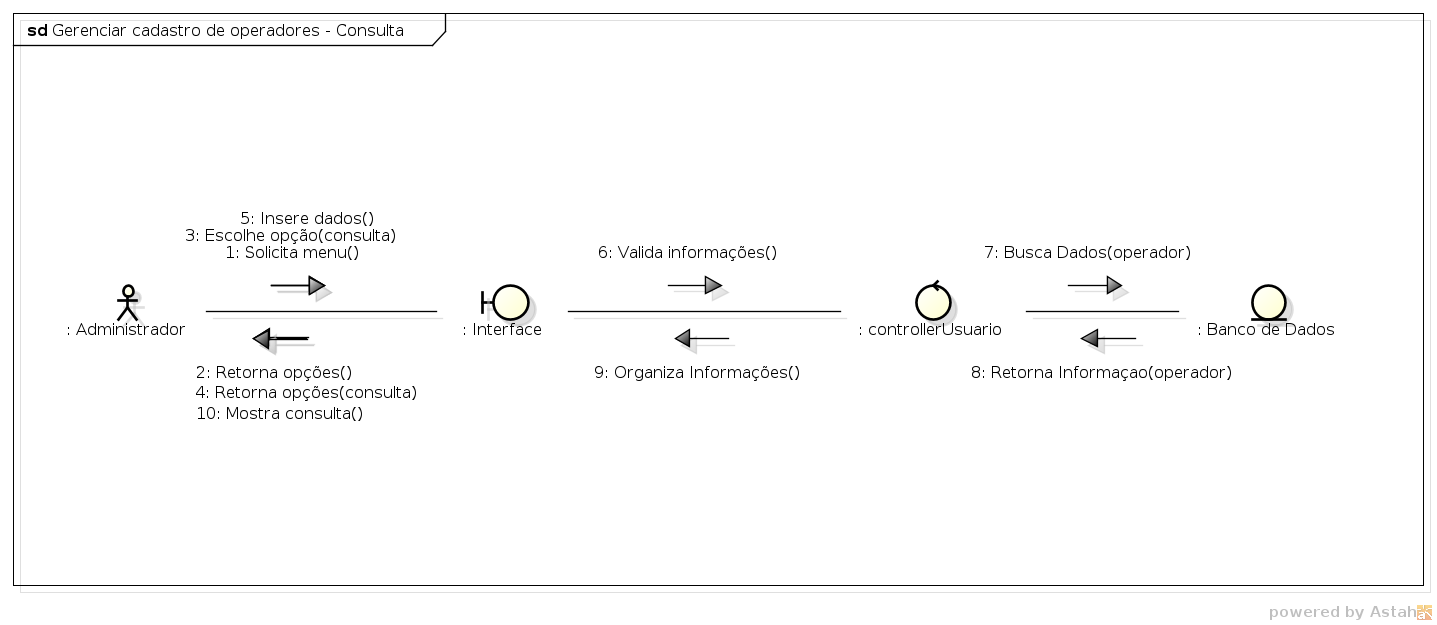
**USE CASE<gerenciar cadastro de times> - Lucio**



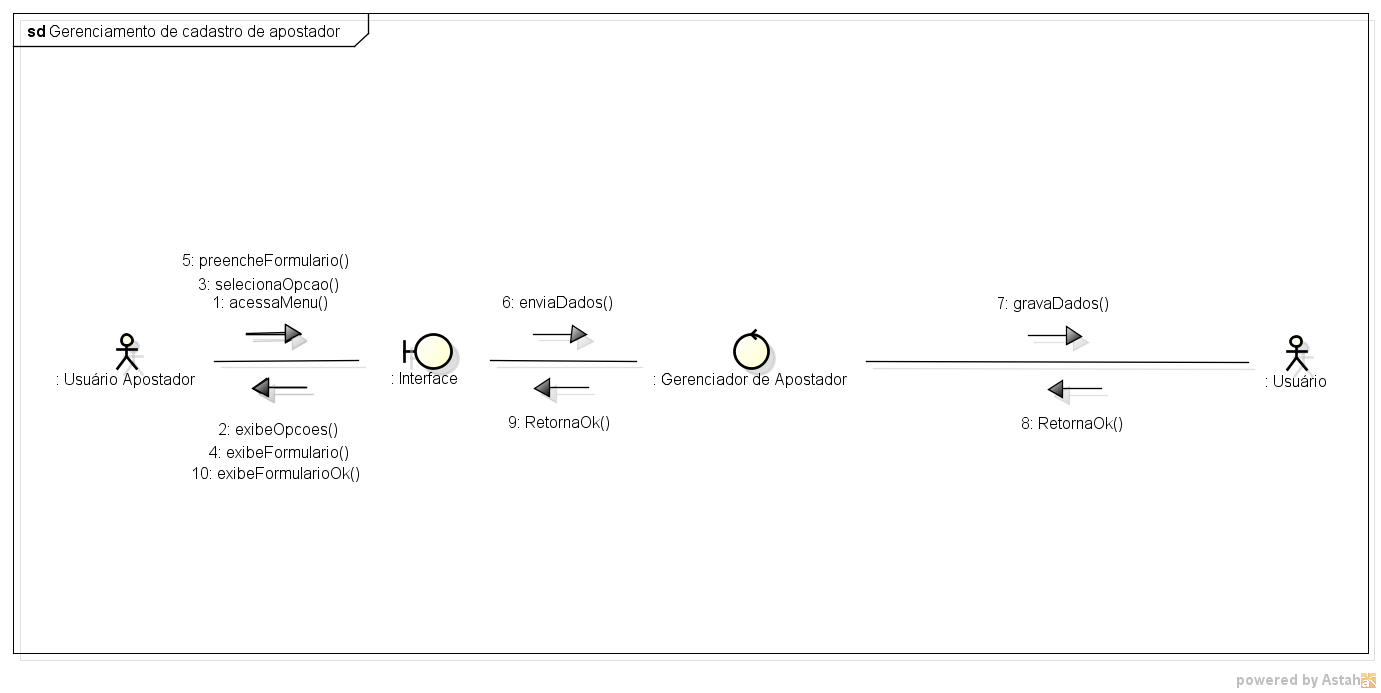
**USE CASE<gerenciar cadastro de partidas> - Danilo**



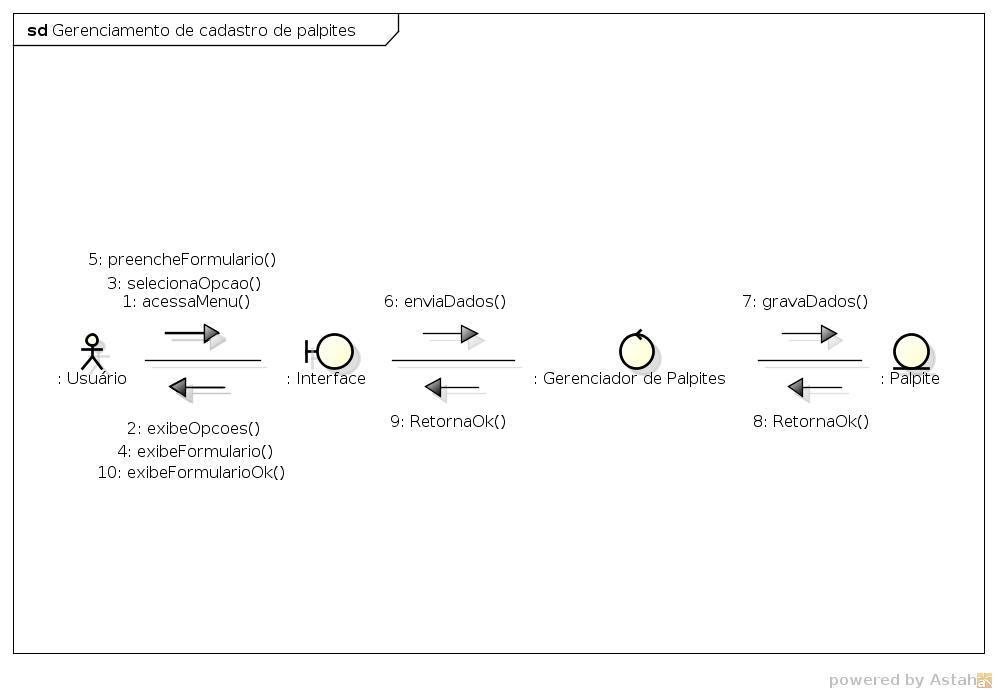
**USE CASE<cadastrar novo usuário apostador> - Juan**



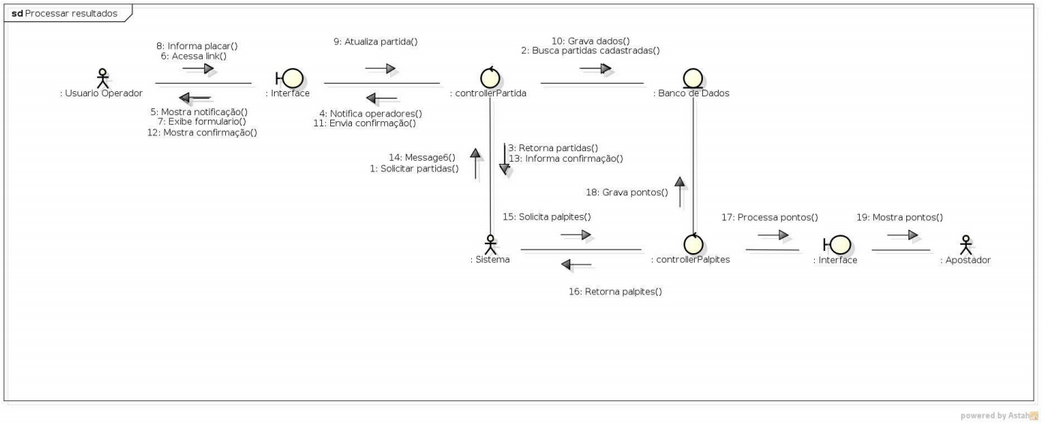
**USE CASE<gerenciar cadastro de apostador> - Alison**



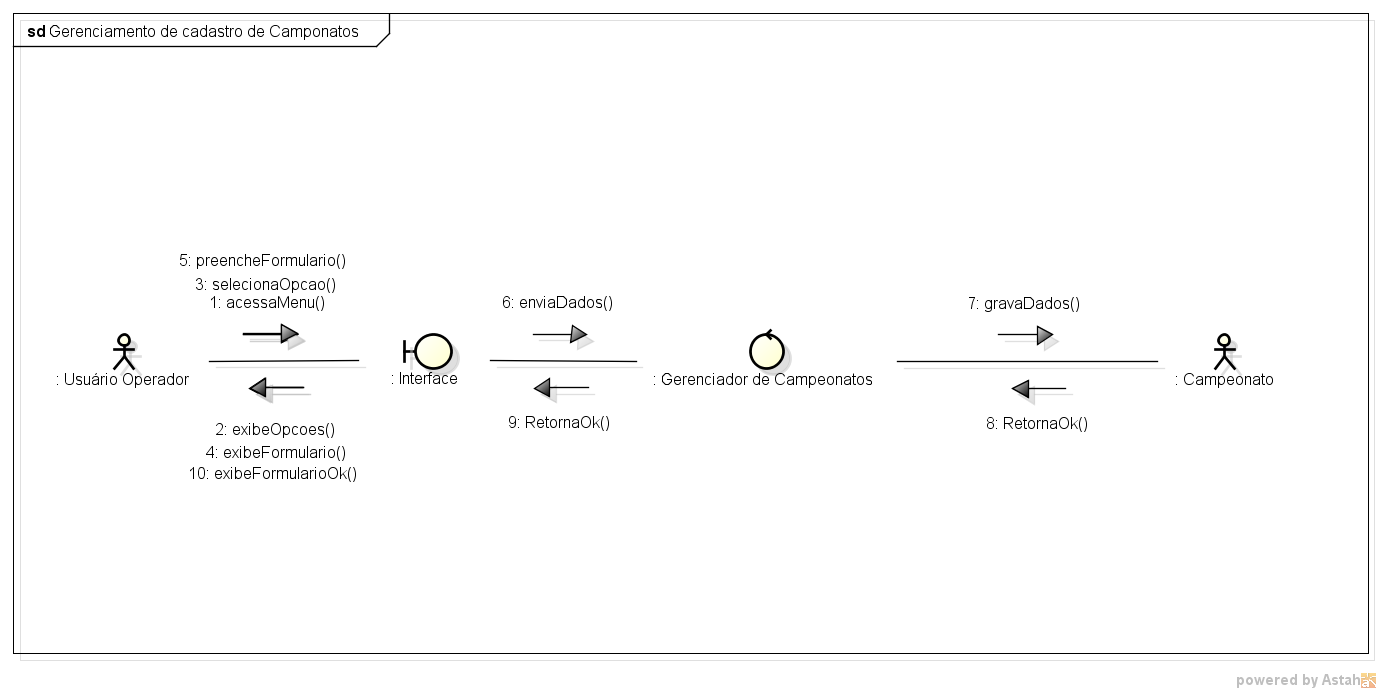
**USE CASE<gerenciar cadastro de palpite> - Lucio**



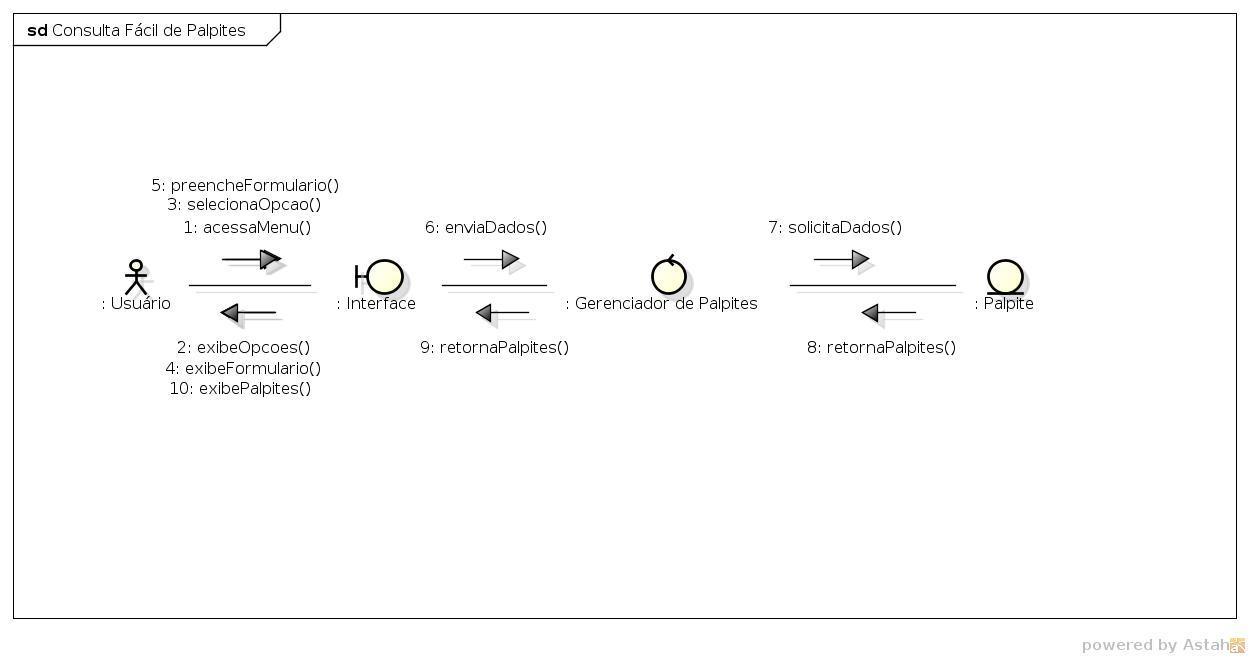
**USE CASE<processar resultados e distribuir pontos> - Danilo**



**USE CASE<consultar ranking de apostadores> - Alison**



**USE CASE<consulta fácil de palpites> - Lucio**



* 1. **Planejamento das iterações**

**ITERAÇÃO 1:**

USE CASE<gerenciar cadastro de administradores> - Danilo

USE CASE<gerenciar cadastro de operadores> - Juan

USE CASE<gerenciar cadastro de campeonatos> - Alison

USE CASE<gerenciar cadastro de times> - Lucio

USE CASE<gerenciar cadastro de partidas> - Danilo

**ITERAÇÃO 2:**

USE CASE<cadastrar novo usuário apostador> - Juan

USE CASE<gerenciar cadastro de apostador> - Alison

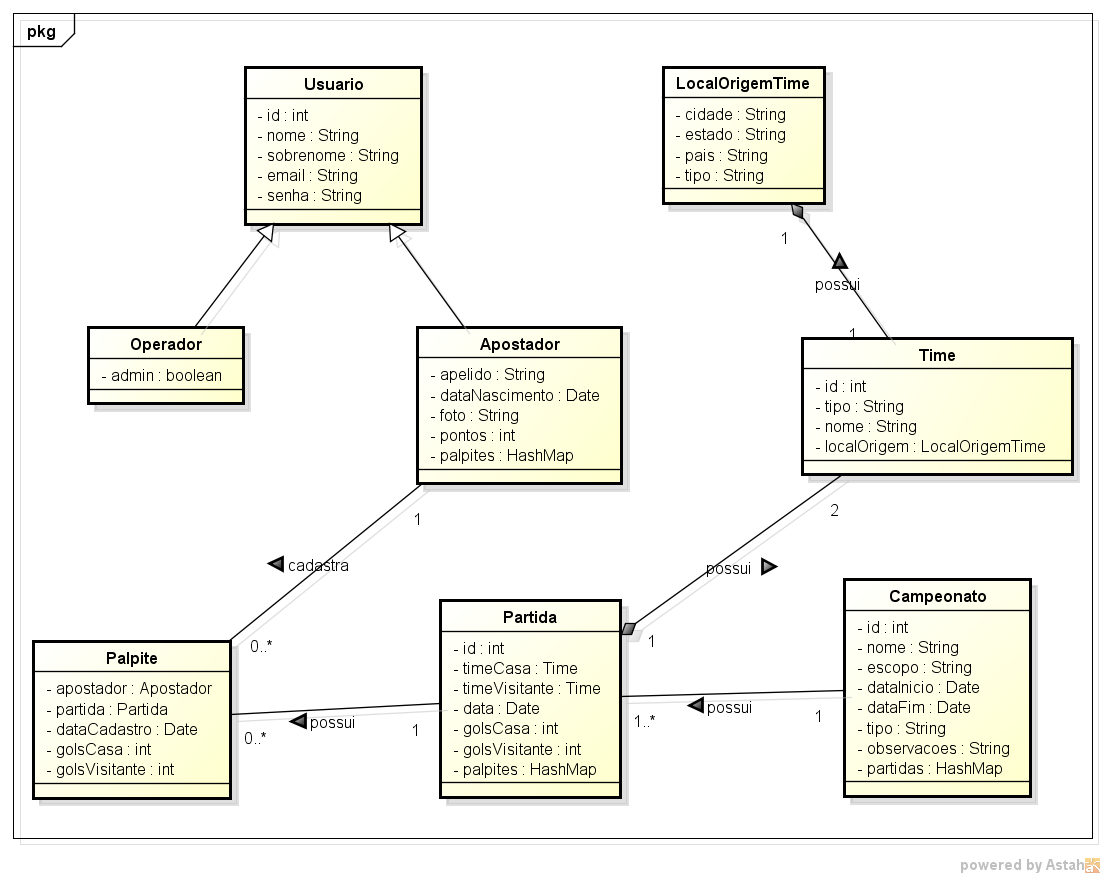
USE CASE<gerenciar cadastro de palpite> - Lucio

USE CASE<processar resultados e distribuir pontos> - Danilo

USE CASE<consultar ranking de apostadores> - Alison

USE CASE<consulta fácil de palpites> - Lucio

* 1. **Diagrama de classes – Modelo de análise**



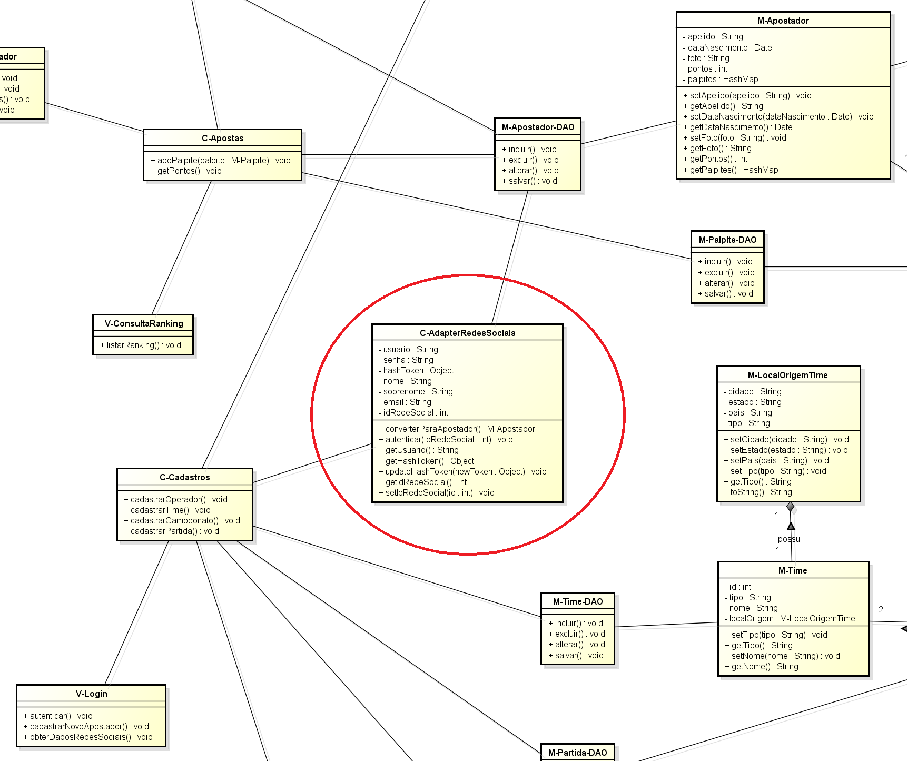
1. **Modelo de projeto**
   1. **Padrões de projeto**

Vamos desenvolver a criação de usuários em nosso projeto utilizando a integração com redes sociais. Para que o apostador não precise ficar preenchendo formulários, vamos utilizar dos dados já existentes em seus perfis.

Para consumir esses serviços, vamos utilizar o padrão de projeto **Adapter**. Mais especificamente, será desenvolvida uma classe responsável por comunicar-se com o webservice da rede social desejada, autenticar o usuário e traduzir o resultado em um objeto do tipo *Apostador*.

Utilizaremos desse padrão para que não seja necessário o desenvolvimento de uma classe especifica para os apostadores criados com base nos dados de redes sociais, assim aumentando a coesão do projeto e impedindo a redundância de classes.

Abaixo está um fragmento do *Diagrama de Classes de Projeto* destacando como esse padrão de projeto será implementado.



A classe *C-AdapterRedesSociais* irá prover uma interface compatível o modelo do apostador.

* 1. **Frameworks**

**JSF**

O JavaServer Faces (JSF) é uma tecnologia que nos permite criar aplicações Java para Web utilizando componentes visuais pré-prontos, de forma que o desenvolvedor não se preocupe com o Javascript e HTML, o que define o JSF como um framework. Basta adicionar os componentes que forem necessários ao projeto e eles serão renderizados e exibidos em formato HTML.

Com a característica stateful podemos criar formulários de várias páginas e navegar nos vários passos dos componentes com o estado das telas sendo mantidos automaticamente.

A separação de camadas de apresentação e de aplicação também é uma característica muito marcante da arquitetura do JSF. Pensando no modelo MVC, o JSF possui uma camada de visualização bem separada do conjunto de classes de modelo.

O JSF ainda tem a vantagem de ser uma especificação do Java EE, isto é, todo servidor de aplicações Java tem que vir com uma implementação dela e há diversas outras disponíveis. A implementação mais famosa do JSF e também a implementação de referência, é a Oracle Mojarra. Outra implementação famosa é a MyFaces da Apache Software Foundation.

**Hibernate**

O Hibernate é um framework para o mapeamento objeto-relacional. Este framework facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, mediante o uso de arquivos (XML) ou anotações.

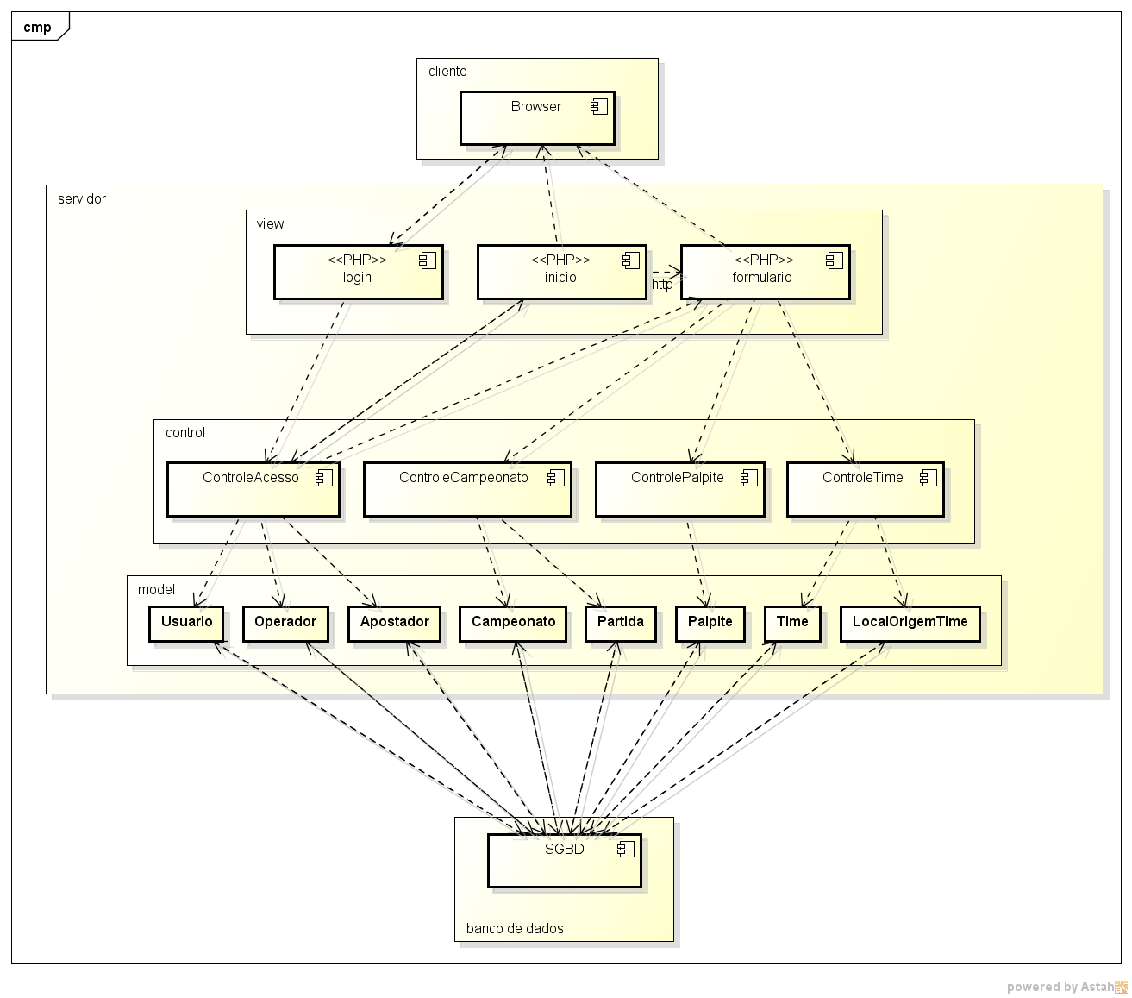
O objetivo do Hibernate é diminuir a complexidade entre os programas Java, baseado no modelo orientado a objeto, que precisam trabalhar com um banco de dados do modelo relacional. Em especial, no desenvolvimento de consultas e atualizações dos dados.

Sua principal característica é a transformação das classes em Java para tabelas de dados (e dos tipos de dados Java para os da SQL). O Hibernate gera as chamadas SQL e libera o desenvolvedor do trabalho manual da conversão dos dados resultante, mantendo o programa portável para quaisquer bancos de dados SQL, porém causando um pequeno aumento no tempo de execução.

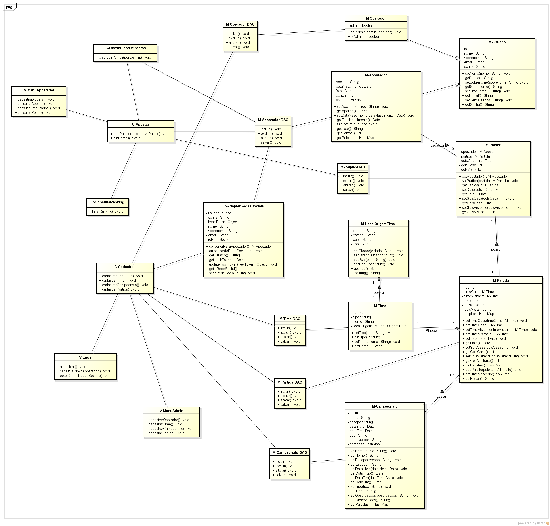
**BootsFaces**

O BootsFaces é um poderoso framework JSF que carrega o melhor do Bootstrap 3 e jQuery UI para desenvolver os Front-End das aplicações mais rápido e fácil.

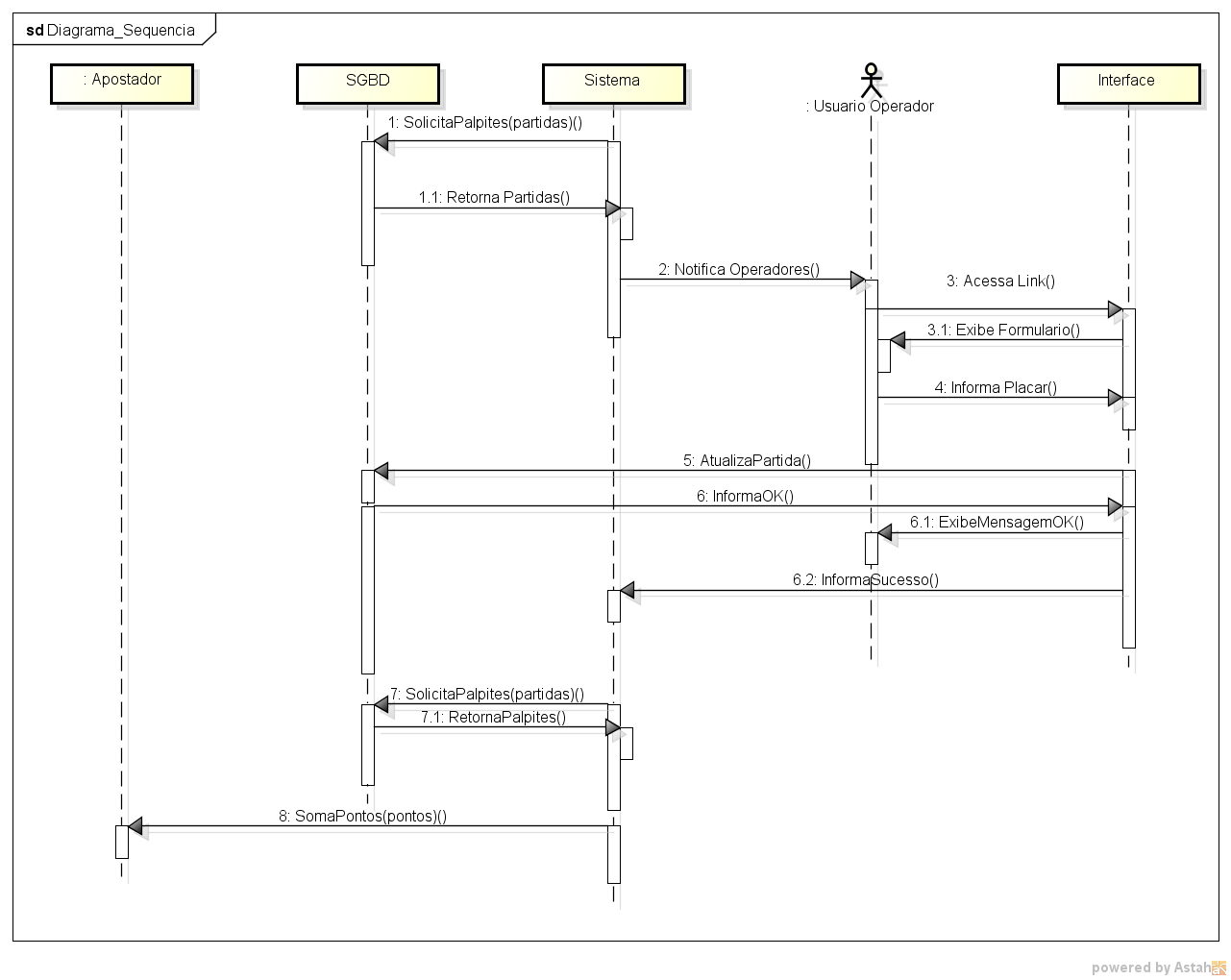
* 1. **Diagrama de componentes**



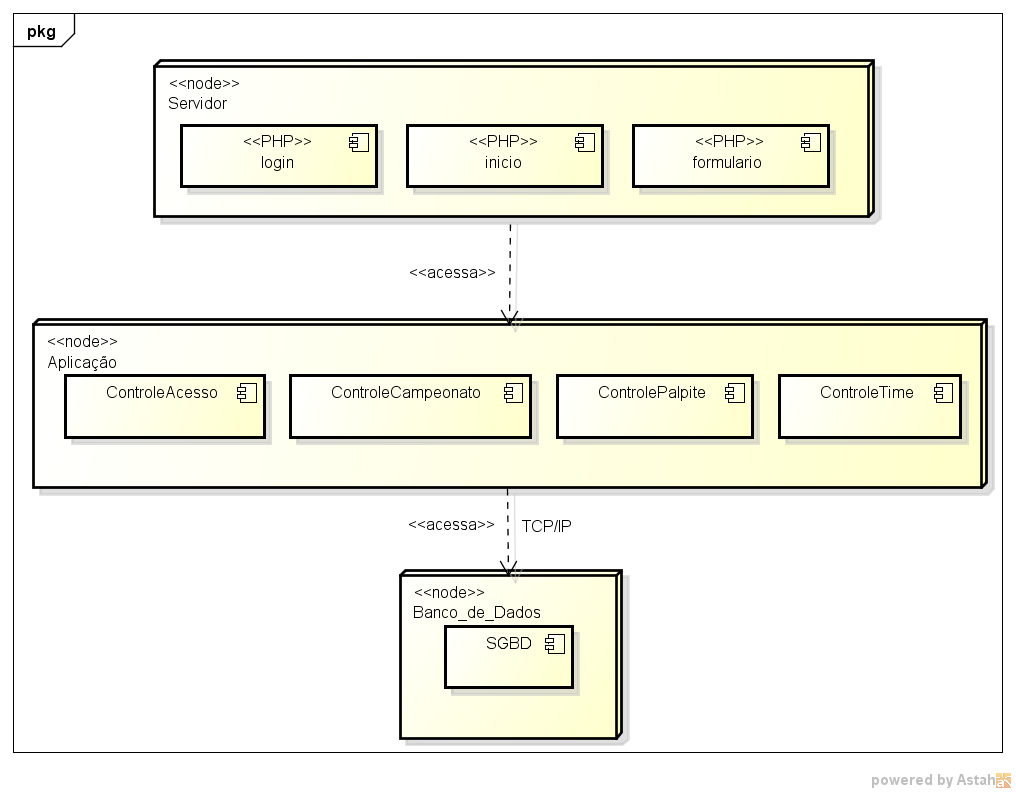
* 1. **Diagrama de classes de projeto**



* 1. **Diagrama de sequência - USE CASE<processar resultados e distribuir pontos>**



* 1. **Diagrama de implantação**



1. **Planejamento das Sprints**
   1. **Detalhamento das estórias e tarefas**

**RF001 - Gerenciar cadastro de administradores (Danilo)**

SPRINT 1:

* Criar CRUD de administradores (1/3)
  + Criar tabela USUARIOS
    - ID: INT (PK)
    - NOME: STRING
    - SOBRENOME: STRING
    - EMAIL: STRING
    - SENHA: STRING
    - ADMIN: BOOLEAN
  + Criar classes MVC e DAO
    - M-USUARIO
      * GETS E SETS
    - M-OPERADOR
      * GETS E SETS
      * ISOPERADOR(): BOOLEAN
    - M-OPERADOR-DAO
      * INCLUIR
      * EXCLUIR
      * ALTERAR
      * SALVAR

Duração: 4 horas

SPRINT 2:

* Alterar Interface de administradores (2/3)
  + Criar interface para consulta dos administradores
    - Campos:
      * Consulta (por ID, nome, sobrenome ou email)
      * Grid para exibição dos registros (uma coluna por campo do registro)
    - Botões
      * Buscar (executa o filtro)
      * Novo (direciona para a tela de cadastro de novo administrador)
  + Criar interface para cadastro de administradores
    - Campos:
      * ID (gerado automaticamente e não editável)
      * NOME (obrigatório)
      * SOBRENOME (obrigatório)
      * EMAIL (obrigatório)
      * SENHA (Campo criptografado e obrigatório)
    - Botões
      * Confirmar (Grava os dados e retorna para a interface de consulta)
      * Voltar (Cancela a operação e retorna para a interface de consulta)

Duração: 10 horas

SPRINT 3:

* Criar Dinâmica de administradores (3/3)
  + Implementar dinâmica do usuário e validações
    - Interface de consulta
      * Limitar acesso apenas a usuários administradores
      * Ao clicar no botão NOVO, direcionar para a interface de cadastro com todos os campos vazios.
      * Ao clicar em BUSCAR, aplicar o filtro definido na grid.
      * Ao dar dois cliques em algum registro do grid, direcionar para a tela de cadastro com os dados do registro selecionado.
    - Interface de cadastro
      * Ao clicar em “Confirmar” validar os dados preenchidos pelo Usuário, caso contenha erro de preenchimento indicar, senão exibir mensagem de confirmação. Após confirmação voltar para a interface de Consulta.
      * Ao clicar em “Voltar” cancelar a operação e voltar para a interface de Consulta.

Duração: 3 horas

RF002 – GERENCIAR CADASTRO DE OPERADORES

SPRINT 1:

**+ JUAN – CRIAR CRUD DE OPERADORES– 1/3**

OBJETIVO:

CRIAR TABELA OPERADORES:

- ID: INT(PK)

- NOME: STRING

- EMAIL:STRING

- SENHA: STRING

CRIAR CLASSES MVC E DAO:

M-OPERADORES

- GET()

- SET()

M-CAMPEONATODAO

- INCLUIR()

- CONSULTAR()

- ALTERAR()

DURAÇÃO: 4 horas

SPRINT 2:

**+ JUAN – ALTERAR INTERFACE DE OPERADORES – 2/3**

OBJETIVO:

CRIAR INTERFACE DE CADASTRO DE OPERADORES

CAMPOS:

- ID (ID, NÃO EDITAVEL)

- Nome (NOME, OBRIGATORIO)

- EMAIL(EMAIL, OBRIGATORIO)

-SENHA(SENHA,OBRIGATORIO)

BOTÕES:

- Confirmar (Grava os dados e retorna para a interface de consulta de operadores)

- Voltar (Cancela a operação e retorna para a interface de consulta de operadores)

CRIAR INTERFACE DE CONSULTA DE OPERADORES

CAMPOS:

- Buscar Operador (Filtro por ID/NOME)

- Grid (Colunas com ID/Nome/Email)

BOTÕES

- Buscar (Executa o filtro)

- Novo Operador(Direciona para a interface de cadastro de operadores)

DURAÇÃO: 10horas

SPRINT 3:

**+ JUAN – CRIAR DINÂMICA DE OPERADORES – 3/3**

OBJETIVO: IMPLEMENTAR DINÂMICA DO USUÁRIO E VALIDAÇÕES:

INTERFACE CONSULTA:

- Interface disponível para Administradores

- Ao clicar no botão “Novo Operador” direcionar para tela de Cadastro com todos os campos vazios

- Ao clicar no botão “Buscar” filtrar o grid por ID, Nome ou Email do Operador

- Ao dar dois cliques em algum registro do Grid, direcionar para a tela de Cadastro com os respectivos dados do operador

INTERFACE CADASTRO:

- Interface disponível apenas para Administradores

- Ao clicar em “Confirmar” validar os dados preenchidos pelo Usuário, caso contenha erro de preenchimento indicar, senão exibir mensagem de confirmação. Após confirmação voltar para a interface de Consulta de operadores

- Ao clicar em “Voltar” cancelar a operação e voltar para a interface de Consulta

- Caso o Usuário não responda por 10 minutos retornar para a interface de Consulta de operadores

ENTRADA: Dados do operador

SAÍDA: Operador registrado no sistema

DURAÇÃO: 4horas

**RF003 – Gerenciar cadastro de campeonato (Alison)**

SPRINT 1:

* CRIAR CRUD DE CAMPEONATO – 1/3
  + CRIAR TABELA CAMPEONATO:
    - - ID: INT(PK)
    - - NOME: STRING
    - - ESCOPO: STRING
    - - DATAINICIO: DATE
    - - DATAFIM: DATE
    - - TIPO: STRING
    - - OBSERVACOES: STRING
    - - PARTIDAS: HASHMAP
  + CRIAR CLASSES MVC E DAO:
    - M-CAMPEONATO
      * - GET()
      * - SET()
    - M-CAMPEONATODAO
      * - INCLUIR()
      * - EXCLUIR()
      * - ALTERAR()
      * - SALVAR()

DURAÇÃO: 4 horas

SPRINT 2:

* ALTERAR INTERFACE DE CAMPEONATO – 2/3
  + CRIAR INTERFACE DE CADASTRO DE CAMPEONATO
    - CAMPOS:
      * - ID (ID, NÃO EDITAVEL)
      * - Nome (NOME, OBRIGATORIO
      * - Escopo (ESCOPO, OBRIGATORIO, [REGIONAL, ESTADUAL, NACIONAL, CONTINENTAL, MUNDIAL])
      * - Data Início (DATAINICIO, OBRIGATORIO)
      * - Data Fim (DATAFIM, OBRIGATORIO)
      * - Tipo (TIPO, OBRIGATORIO, [Pontos, Mata-Mata])
      * - Observações (OBSERVACOES)
    - BOTÕES:
      * - Confirmar (Grava os dados e retorna para a interface de consulta)
      * - Voltar (Cancela a operação e retorna para a interface de consulta)
  + CRIAR INTERFACE DE CONSULTA DE CAMPEONATO
    - CAMPOS:
      * - Buscar Campeonato (Filtro por ID/NOME)
      * - Grid (Colunas com ID/Nome/Escopo/Data Início/Data Fim/Tipo/Time)
    - BOTÕES
      * - Buscar (Executa o filtro)
      * - Novo Campeonato(Direciona para a interface de cadastro cadastro)

DURAÇÃO: 10horas

SPRINT 3:

* CRIAR DINÂMICA DE CAMPEONATO – 3/3
  + IMPLEMENTAR DINÂMICA DO USUÁRIO E VALIDAÇÕES:
    - INTERFACE CONSULTA:
      * - Interface disponível parnas para Administradores
      * - Ao clicar no botão “Novo Campeonato” direcionar para tela de Cadastro com todos os campos vazios
      * - Ao clicar no botão “Buscar” filtrar o grid por ID ou NOME do Campeonato
      * - Ao dar dois cliques em algum registro do Grid, direcionar para a tela de Cadastro com os respectivos dados do Campeonato
    - INTERFACE CADASTRO:
      * - Interface disponível parnas para Administradores
      * - Ao clicar em “Confirmar” validar os dados preenchidos pelo Usuário, caso contenha erro de preenchimento indicar, senão exibir mensagem de confirmação. Após confirmação voltar para a interface de Consulta
      * - Ao clicar em “Voltar” cancelar a operação e voltar para a interface de Consulta
      * - Caso o Usuário não responda por 10 minutos retornar para a interface de Consulta

DURAÇÃO: 2horas

**RF004 - Gerenciar cadastro de times (Lucio)**

Sprint 01:

* Criar CRUD de times (01/03)
  + Criar tabela LocalOrigemTime
    - cidade: String;
    - estado: String;
    - país: String;
    - tipo: String.
  + Criar tabela Time
    - Id: int(PK);
    - IdLocalOrigem: int(FK);
    - Tipo: String;
    - Nome: String.
  + Criar Classes MVC e DAO
    - M-Time
      * getTipo(): String;
      * setTipo(tipo: String): void;
      * setNome(nome: String): void;
      * getNome(): String.
    - M-Time-DAO
      * incluir();
      * excluir();
      * alterar();
      * salvar().

Duração: 5 horas.

Sprint 02:

* Alterar Interface de times (02/03)
  + Criar interface cadastro time
    - Campos:
      * Id: (Id, não editável);
      * Tipo: (Tipo, obrigatório);
      * Nome: (Nome, obrigatório);
      * LocalOrigem: (LocalOrigem, obrigatório).
    - Botões:
      * Confirmar(Gravar dados da equipe, retornar para a interface de consulta);
      * Voltar(Cancelar a operação, retornar para a interface anterior).
* Criar interface de consulta de times
  + Campos
    - Buscar: (Filtro Id ou Nome);
    - Grid: (Tabela possuindo Id, Nome, Tipo, País);
  + Botões
    - Buscar: (Executar filtro);
    - NovoTime: (Redirecionar para interface de cadastro de times);
    - Voltar: (Voltar para interface anterior).

Duração: 12 horas.

Sprint 03:

* Criar dinâmica de times (03/03)
  + Implementar dinâmica do usuário e validações
    - Interface consulta:
      * Interfaces disponíveis a adminstradores;
      * Ao clicar no botão “Novo time”, redirecionar para a tela de cadastro de novo time, com os campos vazios;
      * Ao clicar no botão “Buscar”, filtrar por Id ou Nome;
      * Ao clicar duplamente na em algum registro na grid, direcionar para interface de cadastro da equipe;
    - Interface cadastro:
      * Interface disponível a adminstradores;
      * Ao clicar em confirmar, validar dados inseridos pelo usuário, contendo algum erro no preenchimento alertá-lo, caso contrário exibir mensagem de confirmação. Após, retornar a interface anterior;
      * Ao clicar em “Voltar”, cancelar operação e retornar a interface anterior.

Duração: 5 horas.

**RF005 - Gerenciar cadastro de partidas (Danilo)**

SPRINT 1:

* Criar CRUD de partidas(1/3)
  + Criar tabela PARTIDAS
    - ID: INT (PK)
    - ID-TIME-CASA: INT (FK)
    - ID-TIME VISITANTE: INT (FK)
    - DATA: DATETIME
    - GOLS-TIME-CASA: INT
    - GOLS-TIME-VISITANTE: INT
    - PALPITES: HASMAP
  + Criar classes MVC e DAO
    - M-PARTIDA
      * GETS E SETS
    - M-PARTIDA-DAO
      * INCLUIR
      * EXCLUIR
      * ALTERAR
      * SALVAR

Duração: 4 horas

SPRINT 2:

* Alterar Interface de partidas(2/3)
  + Criar interface para consulta das partidas
    - Campos:
      * Consulta (por ID, data, time da casa ou time visitante)
      * Grid para exibição dos registros (uma coluna por campo do registro)
    - Botões
      * Buscar (executa o filtro)
      * Novo (direciona para a tela de cadastro de nova partida)
  + Criar interface para cadastro de partidas
    - Campos:
      * ID (gerado automaticamente e não editável)
      * ID-TIME-CASA (implementar buscar para todos os times já cadastrados e disponibilizar para o usuário utilizar).
      * ID-TIME-VISITANTE (implementar buscar para todos os times já cadastrados, exceto o já escolhido anteriormente, e disponibilizar para o usuário utilizar).
      * DATA
    - Botões
      * Confirmar (Grava os dados e retorna para a interface de consulta)
      * Voltar (Cancela a operação e retorna para a interface de consulta)

Duração: 10 horas

SPRINT 3:

* Criar Dinâmica de partidas(3/3)
  + Criar Implementar dinâmica do usuário e validações
    - Interface de consulta
      * Limitar acesso apenas a usuários administradores e operadores
      * Ao clicar no botão NOVO, direcionar para a interface de cadastro com todos os campos vazios.
      * Ao clicar em BUSCAR, aplicar o filtro definido na grid.
      * Ao dar dois cliques em algum registro do grid, direcionar para a tela de cadastro com os dados do registro selecionado.
    - Interface de cadastro
      * Ao clicar em “Confirmar” validar os dados preenchidos pelo Usuário, caso contenha erro de preenchimento indicar, senão exibir mensagem de confirmação. Após confirmação voltar para a interface de Consulta.
      * Ao clicar em “Voltar” cancelar a operação e voltar para a interface de Consulta.

Duração: 3 horas

RF006 – GERENCIAR CADASTRO DE APOSTADOR

SPRINT 1:

**+ JUAN – CRIAR CRUD DE APOSTADOR 1/3**

OBJETIVO:

CRIAR TABELA APOSTADORES:

- ID: INT(PK)

- NOME: STRING

- SOBRENOME: STRING

- APELIDO: STRING

- EMAIL:STRING

- DT.NASCIMENTO: DATE

CRIAR CLASSES MVC E DAO:

M-APOSTADORES

- GET()

- SET()

M-APOSTADOR DAO

- INCLUIR()

- CONSULTAR()

- EXCLUIR()

- ALTERAR()

DURAÇÃO: 4 horas

SPRINT 2:

**+ JUAN – ALTERAR INTERFACE DE APOSTADOR – 2/3**

OBJETIVO:

CRIAR INTERFACE DE CONSULTA DE APOSTADOR

CAMPOS:

- Buscar Apostador (Filtro por ID/NOME/APELIDO)

- Grid (Colunas com ID/Nome/Sobrenome/Apelido/e-mail/Dt. Nasc.)

BOTÕES

- Buscar (Executa o filtro)

- Novo Apostador(Direciona para a interface de cadastro cadastro)

DURAÇÃO: 6horas

SPRINT 3:

**+ JUAN – CRIAR DINÂMICA DE APOSTADOR – 3/3**

OBJETIVO:

IMPLEMENTAR DINÂMICA DO USUÁRIO E VALIDAÇÕES:

- Interface disponível apenas para Administradores

- Ao clicar no botão “Novo Apostador” direcionar para tela de Cadastro com todos os campos vazios

- Ao clicar no botão “Buscar” filtrar o grid por ID,NOME ou APELIDO do Apostador

- Ao dar dois cliques em algum registro do Grid, direcionar para a tela de Cadastro com os respectivos dados do Apostador

DURAÇÃO: 3horas

**RF007 – Gerenciar cadastro de apostador (Alison)**

SPRINT 1:

* CRIAR INTERFACE DA CONSULTA DE APOSTADOR – 1/2
  + CRIAR INTERFACE DE CONSULTA DE APOSTADOR
    - CAMPOS:
      * - Buscar Apostador (Filtro por ID/NOME)
      * - Grid (Colunas com ID/Nome/Sobrenome/Apelido/e-mail/Dt. Nasc.)
    - BOTÕES
      * - Buscar (Executa o filtro)
      * - Novo Apostador(Direciona para a interface de cadastro)

DURAÇÃO: 5horas

SPRINT 2:

* CRIAR DINÂMICA DA CONSULTA DE APOSTADOR – 2/2
  + IMPLEMENTAR DINÂMICA DO USUÁRIO E VALIDAÇÕES:
    - - Interface disponível apenas para Administradores
    - - Ao clicar no botão “Novo Apostador” direcionar para tela de Cadastro com todos os campos vazios
    - - Ao clicar no botão “Buscar” filtrar o grid por ID ou NOME do Apostador
    - - Ao dar dois cliques em algum registro do Grid, direcionar para a tela de Cadastro com os respectivos dados do Apostador

DURAÇÃO: 2horas

**RF008 - Gerenciar cadastro de palpite (Lucio)**

Sprint 01:

* Criar CRUD de palpites (01/03)
  + Criar tabela Palpite
    - Id: int(PK);
    - IdApostador: int(FK);
    - IdPartida: int(FK);
    - DataCadastro: Date;
    - GolsCasa: Int;
    - GolsVisitante: Int.
  + Criar Classes MVC e DAO
    - M-Palpite
      * getApostador(): apostador;
      * setPartida(partida: Partida): void;
      * getPartida(): Partida;
      * setData(data: Date): void;
      * getData(): Date;
      * setGolsCasa(golsCasa: int): void;
      * getGolsCasa(): int;
      * setGolsVisitante(golsCasa: int): void;
      * getGolsVisitante(): int;
    - M-Palpite-DAO
      * incluir();
      * excluir();
      * alterar();
      * salvar().

Duração: 5 horas.

Sprint 02:

* Alterar Interface de palpites (02/03)
  + Criar interface cadastro palpite
    - Campos:
      * Id: (Id, não editável);
      * Campeonato: (Campeonato, não editável);
      * TimeCasa: (Time mandante, não editável);
      * TimeVisitante: (Time visitante, não editável);
      * GolsCasa: (Gols do time mandante, obrigatório);
      * GolsVisitante: (Gols do time visitante, obrigatório);
      * DataPartida: (Data da partida, não editável);
      * Status: (Status da partida, não editável).
    - Botões:
      * Confirmar(Gravar dados do palpite, retornar para a interface de consulta);
      * Voltar(Cancelar a operação, retornar para a interface anterior).
  + Criar interface de consulta de palpites
    - Campos
      * Buscar: (Filtro Id, nome do time ou data da partida);
      * Grid: (Tabela possuindo Id, TimeCasa, GolsCasa, GolsVisitante, TimeVisitante, DataPartida, Status);
      * Campeonato: (Lista dos campeonatos)
    - Botões
      * Buscar: (Executar filtro);

Duração: 12 horas.

Sprint 03:

* Criar Dinâmica de palpites (03/03)
  + Implementar dinâmica do usuário e validações
    - Interface consulta:
      * Interfaces disponíveis a adminstradores e usuários;
      * Ao clicar no botão “Buscar”, filtrar por Id, nome do time ou data da partida;
      * Ao clicar duplamente na em algum registro na grid, direcionar para interface de cadastro do palpite;
    - Interface cadastro:
      * Interface disponível a adminstradores e usuários;
      * Ao clicar em confirmar, validar dados inseridos pelo usuário, contendo algum erro no preenchimento alertá-lo, caso contrário exibir mensagem de confirmação. Após, retornar a interface anterior;
      * Ao clicar em “Voltar”, cancelar operação e retornar a interface anterior.

Duração: 5 horas.

**RF009 - Cadastrar placar e distribuir pontos (Danilo)**

PRÉ-REQUISITOS:

Cadastro de partidas.

SPRINT 3:

* Implementar robô que monitore as partidas e informe aos operadores caso exista alguma já ocorrida sem o cadastro do placar.
  + Criar classe C-DISTRIBUIDOR-PONTOS
    - VerificarPartidasSemPlacar();

Duração: 5 horas

SPRINT 4:

* Implementar robô que monitora cadastro de placares e atribui os pontos para os apostadores
  + Implementar método DistribuirPontos() na classe C-DISTRIBUIDOR-PONTOS
    - 2 pontos caso acerte o placar exato.
    - 1 ponto caso acerte o time vencedor da partida.

Duração: 5 horas

**RF010 – Ranking de apostadores (Alison)**

SPRINT 1:

* CRIAR RANKING DE APOSTADOR – 1/2
  + CRIAR INTERFACE DE RANKING DE APOSTADOR
    - CAMPOS:
      * - Selecionar Campeonato
      * - Buscar Apostador (Filtro por ID/NOME)
      * - Grid (Colunas com Posição/Nome(Apostador)/Pontos Marcados/Evolução)
    - BOTÕES
      * - Buscar (Executa o filtro)

DURAÇÃO: 5horas

SPRINT 2:

* ALTERAR INTERFACE CONSULTA DE APOSTADOR – 2/2
  + IMPLEMENTAR DINÂMICA DO USUÁRIO E VALIDAÇÕES:
    - - Interface aberto
    - - Carregar os dados apenas se o Campeonato estiver selecionado
    - - Ao clicar no botão “Buscar” filtrar o grid por NOME do Apostador
    - - Caso o Usuário Apostador estiver logado, destacar seu nome no Ranking

DURAÇÃO: 2horas

**RF011 - Consulta fácil de palpites (Lucio)**

Sprint 01:

* Criar consulta fácil de palpites (01/02)
  + Criar interface de consulta fácil de palpites
    - Campos
      * Buscar: (Filtro Id, nome do time ou data da partida);
      * Grid: (Tabela possuindo Id, TimeCasa, GolsCasa, GolsVisitante, TimeVisitante, DataPartida, Status);
    - Botões
      * Voltar: (Retornar a interface anterior);

Duração: 12 horas.

Sprint 02:

* Alterar Interface da consulta fácil de palpites (02/02)
  + Implementar dinâmica do usuário e validações
    - Interface consulta:
      * Interfaces disponíveis a administradores e usuários;
      * Ao clicar em “Voltar”, retornar a interface anterior.
      * Ao clicar duplamente na em algum registro no grid, direcionar para interface de cadastro do palpite;

Duração: 2 horas.

**RF012 – Tela de aceeso (login) (Alison)**

* Sprint 01: Criar tela de acesso

Campos: Usuário, Senha;

Botões: Entrar;

Duração: 3 horas.

* 1. **Product backlog e Planejamento das sprints**

|  |  |  |
| --- | --- | --- |
| **SPRINT 1** | | |
| **Estória** | **Prioridade** | **Duração(hr)** |
| DANILO - Criar CRUD de administradores (1/3) | 1 | 4 |
| JUAN - Criar CRUD de Operadores(1/3) | 2 | 4 |
| ALISON – Criar CRUD de campeonato – (1/3) | 3 | 4 |
| LUCIO - Criar CRUD de times (01/03) | 4 | 4 |
| DANILO - Criar CRUD de partidas(1/3) | 5 | 4 |
| JUAN - Criar CRUD de Apostador(1/3) | 6 | 4 |
| LUCIO - Criar CRUD de palpites (01/03) | 7 | 4 |
| ALISON – Criar INTERFACE da consulta apostador – (1/2) | 8 | 2 |
| **SPRINT 2** | | |
| **Estória** | **Prioridade** | **Duração(hr)** |
| DANILO - Alterar INTERFACE de administradores (2/3) | 1 | 10 |
| JUAN - Alterar INTERFACE de Operadores(2/3) | 2 | 10 |
| ALISON – Alterar INTERFACE de campeonato – (2/3) | 3 | 10 |
| LUCIO - Alterar INTERFACE de times (02/03) | 4 | 10 |
| DANILO - Alterar INTERFACE de partidas(2/3) | 5 | 10 |
| JUAN - Alterar INTERFACE de apostadores (2/3) | 6 | 10 |
| LUCIO - Alterar INTERFACE de palpites (02/03) | 7 | 10 |
| ALISON – Criar DINÂMICA da consulta de apostador – 2/2 | 8 | 2 |
| **SPRINT 3** | | |
| **Estória** | **Prioridade** | **Duração(hr)** |
| DANILO - Criar DINÂMICA de administradores (3/3) | 1 | 3 |
| JUAN - Criar DINÂMICA de Operadores (3/3) | 2 | 3 |
| ALISON – Criar DINÂMICA de campeonato – 3/3 | 3 | 3 |
| LUCIO - Criar DINÂMICA de times (03/03) | 4 | 3 |
| DANILO - Criar DINÂMICA de partidas(3/3) | 5 | 3 |
| JUAN - Criar DINÂMICA de apostadores (3/3) | 6 | 3 |
| LUCIO - Criar DINÂMICA de palpites (03/03) | 7 | 3 |
| **SPRINT 4** | | |
| **Estória** | **Prioridade** | **Duração(hr)** |
| DANILO - Implementar robô que monitore as partidas | 1 | 5 |
| DANILO - Implementar robô que monitora cadastro de placares | 2 | 5 |
| ALISON – Criar ranking de apostador – 1/2 | 3 | 5 |
| ALISON – Alterar INTERFACE ranking de apostador – 2/2 | 4 | 2 |
| LUCIO - Criar consulta fácil de palpites (01/02) | 5 | 12 |
| LUCIO - Alterar INTERFACE consulta fácil de palpites (02/02) | 6 | 2 |
| ALISON – Criar tela de LOGIN | 7 | 3 |

1. **Planejamento das Sprints**

**10.1 Detalhamento das estórias e tarefas**

**TESTE 1 – Validação do LOGIN (Alison)**

* Prazo: 1 hora
* Recursos:

- Técnica: Caixa Preta, Caixa Branca

- Ferramentas: JUnit

* Teste de unidade:

- Classe do JUnit: TesteLoginBEAN.java

- Classe a ser testada: LoginBEAN.java

- Validação: testValidarLogin() - verificar se o usuário e a senha estão corretos, se positivo retornar a página index.xhtml.

* Caixa Preta (Particionamento de equivalência):
* Variável:

- usuario;

- senha;

* Classes de Equivalência:

- usuario E {“Admin”; “Alison”};

- senha E {“Admin”; “Orita”};

* Casos de Teste:

1 – usuario=”Admin” e senha=”Admin”

2 – usuario=”Admin” e senha=”Orita”

3 – usuario=”Alison” e senha=”Admin”

4 – usuario=”Alison” e senha=”Orita”

* Resultados Esperados:

1 – Login validado com sucesso (redirecionamento para a página index.xhtml)

2 – Login inválido (manter na página login.xhtml)

3 – Login inválido (manter na página login.xhtml)

4 – Login validado com sucesso (redirecionamento para a página index.xhtml)

* Resultados Apresentados:

- Resultado de acordo com o esperado para os itens 1, 2, 3 e 4

* Depuração:

- JUnit