**Documento de Requisitos**

**FrenteCaixaEPS**

**Versão1.00**

|  |  |
| --- | --- |
| Integrantes | R.A. |
| Rafael Salin Soares | 62449 |
| Guilherme Sperandio dos Santos | 67192 |
| Flávio Augusto Leopoldino Pereira | 64358 |

**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

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)

9.2 Product backlog e Planejamento das sprints

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** |
| *Funcionário de Frente de Caixa* | *Usuário do sistema que deverá executar todas as funções disponíveis pelo sistema..* |

* 1. **Requisitos Funcionais**

**[RF001]** **Cadastrar Produtos**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário visualiza os pedidos de venda | | | | | |
| **Entrada:** | Nome; Preço | | | | | |
| **Processo:** | O sistema lista os pedidos de venda cadastrados | | | | | |
| **Saída:** | Listagem dos pedidos de venda | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF002] Alterar Produto**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário executa a alteração do campo desejado. Operação executada pelo atendente de balcão | | | | | |
| **Entrada:** | Nome | | | | | |
| **Processo:** | Atualização do banco de dados | | | | | |
| **Saída:** | Lista de produtos atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF003] Excluir Produto**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário executa a exclusão do campo desejado | | | | | |
| **Entrada:** | Nome | | | | | |
| **Processo:** | Atualização do banco de dados | | | | | |
| **Saída:** | Lista de produtos atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF004] Listar Produtos**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário visualiza os dados do produto | | | | | |
| **Entrada:** | Nome | | | | | |
| **Processo:** | O sistema lista os produtos cadastrados | | | | | |
| **Saída:** | Listagem dos produtos | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF005]** **Cadastrar Fornecedor**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário faz a inclusão de um fornecedor no sistema | | | | | |
| **Entrada:** | Nome; CNPJ; e-mail | | | | | |
| **Processo:** | O fornecedor será incluido no banco de dados | | | | | |
| **Saída:** | Lista de produtos atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF006] Alterar Fornecedor**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário executa a alteração do fornecedor | | | | | |
| **Entrada:** | Nome | | | | | |
| **Processo:** | Atualização do banco de dados | | | | | |
| **Saída:** | Lista de produtos atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF007] Excluir Fornecedor**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário executa a exclusão do fornecedor | | | | | |
| **Entrada:** | Nome | | | | | |
| **Processo:** | Atualização do banco de dados | | | | | |
| **Saída:** | Lista de produtos atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF008] Listar Fornecedores**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário visualiza os dados dos fornecedores | | | | | |
| **Entrada:** | Nome | | | | | |
| **Processo:** | O sistema lista os fornecedores cadastrados | | | | | |
| **Saída:** | Listagem dos fornecedores | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF009]** **Cadastrar Cliente**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário faz a inclusão de um cliente no sistema | | | | | |
| **Entrada:** | Nome; CPF; e-mail | | | | | |
| **Processo:** | O clientes será incluido no banco de dados | | | | | |
| **Saída:** | Lista de clientes atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF010] Alterar Cliente**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário executa a alteração do cliente | | | | | |
| **Entrada:** | Nome | | | | | |
| **Processo:** | Atualização do banco de dados | | | | | |
| **Saída:** | Lista de clientes atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF011] Excluir Cliente**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário executa a exclusão do cliente | | | | | |
| **Entrada:** | Nome | | | | | |
| **Processo:** | Atualização do banco de dados | | | | | |
| **Saída:** | Lista de clientes atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF012] Listar Clientes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário visualiza os clientes | | | | | |
| **Entrada:** | Nome | | | | | |
| **Processo:** | O sistema lista os clientes cadastrados | | | | | |
| **Saída:** | Listagem dos clientes | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF013]** **Cadastrar Pedido de Compra**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário faz a inclusão de um pedido de compra no sistema | | | | | |
| **Entrada:** | Nome do Produto; Quantidade; Preço Unitário; Status; Data; Nome do fornecedor; Valor total | | | | | |
| **Processo:** | O pedido de compra será incluido no banco de dados | | | | | |
| **Saída:** | Lista de pedidos de compra atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF014] Alterar Pedido de Compra**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário executa a alteração do pedidos de compra | | | | | |
| **Entrada:** | Código do pedido de compra | | | | | |
| **Processo:** | O pedido de compra será incluído no banco de dados | | | | | |
| **Saída:** | Lista de pedidos de compra atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF015] Excluir Pedido de Compra**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário executa a exclusão do pedido de compra | | | | | |
| **Entrada:** | Código do pedido de compra | | | | | |
| **Processo:** | Atualização do banco de dados | | | | | |
| **Saída:** | Lista de pedidos de compra atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF016] Listar Pedido de Compra**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário visualiza os cliente | | | | | |
| **Entrada:** | Código do pedido de compra | | | | | |
| **Processo:** | O sistema lista os pedidos de compra cadastrados | | | | | |
| **Saída:** | Listagem dos pedidos de compra | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF017]** **Cadastrar Pedido de Venda**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário faz a inclusão de um pedido de venda no sistema | | | | | |
| **Entrada:** | Nome do Cliente; Produtos; Data; Valor total | | | | | |
| **Processo:** | O pedido de venda será incluido no banco de dados | | | | | |
| **Saída:** | Lista de pedidos de venda atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF018] Alterar Pedido de Venda**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário executa a alteração do pedidos de venda | | | | | |
| **Entrada:** | Código do pedido de venda | | | | | |
| **Processo:** | O pedido de venda será incluído no banco de dados | | | | | |
| **Saída:** | Lista de pedidos de venda atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF019] Excluir Pedido de Venda**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário executa a exclusão do pedido de venda | | | | | |
| **Entrada:** | Código do pedido de venda | | | | | |
| **Processo:** | Atualização do banco de dados | | | | | |
| **Saída:** | Lista de pedidos de venda atualizada | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

**[RF020] Listar Pedido de Venda**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade:** | **X** | Essencial |  | Importante |  | Desejável |
| **Descrição:** | O usuário visualiza os pedidos de venda | | | | | |
| **Entrada:** | Código do pedido de venda | | | | | |
| **Processo:** | O sistema lista os pedidos de venda cadastrados | | | | | |
| **Saída:** | Listagem dos pedidos de venda | | | | | |
| **Ator (es):** | Atendente de Balcão | | | | | |

1. **Arquitetura de software**

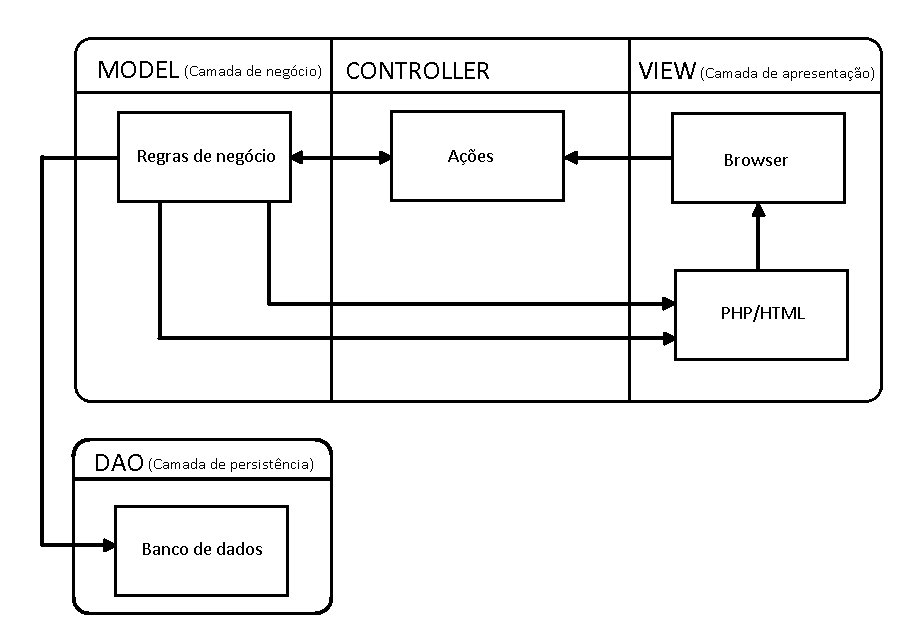
Definimos que o tipo de arquitetura de sistema que utilizaremos para o desenvolvimento do sistema web é o *Cliente x Servidor*. Onde especificamos que:

* **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.

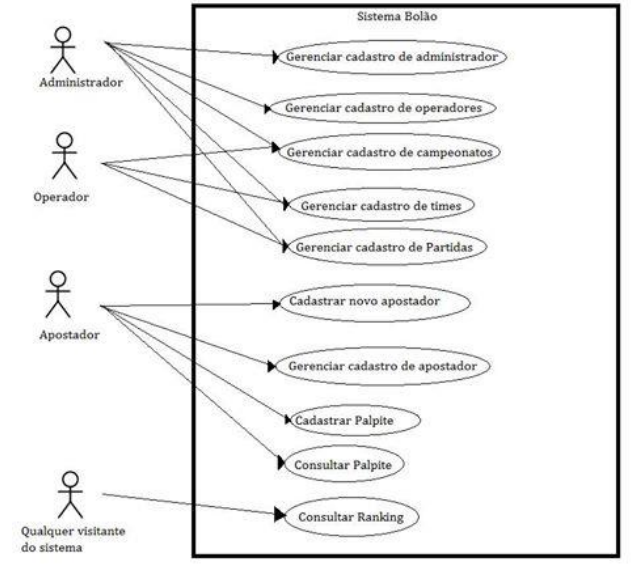
A seguir ilustraremos, com um diagrama, como será a arquitetura que visamos para desenvolver o projeto web:

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**

**Descrição do Caso de Uso Gerenciar Produto**

**Objetivo:**

Permite o gerenciamento e manipulação (incluir, excluir, alterar e consultar) de produtos no banco de dados do sistema.

**Pré-condições:**

No cadastro de um novo produto, ele não deve estar cadastrado no banco de dados. No cadastro de um novo produto, ele não deve estar cadastrado no banco de dados. Para editar ou excluir, o produto já deve estar incluso no banco de dados. Na opção de listar produtos, ao menos um produto deve estar cadastrado no banco de dados.

**Iniciado por:**

Usuário do computador de frente de caixa.

**Fluxo principal:**

1. O sistema mostra a tela para escolha da rotina que será manipulada (inclusão, alteração ou consulta)
2. O sistema abre a tela com as opções incluir, alterar, excluir e consultar produtos.
3. O usuário seleciona a opção para incluir novo produto.
4. O sistema exibe o formulário de cadastro.
5. O usuário informa nome e preço e seleciona a opção para gravação do novo produto.
6. O sistema gera o identificador do produto
7. O sistema salva as informações no banco de dados.

**Fluxo alternativo: Alterar produto**

No item 3, caso o usuário selecione a opção de alterar um produto, 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 produto**

No item 3, caso o usuário selecione a opção de excluir um produto, 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.

**Fluxo alternativo: Listar produto**

No item 3, caso o usuário selecione a opção de listar um produto, todos os produtos são listados contendo todos os atributos, e ao fim, uma opção de voltar ao menu principal é apresentada.

**Pós-condições:**

Um produto é criado, excluído, alterado ou todos os produtos são listados, caso o usuário não tenha abortado o processo.

**Descrição do Caso de Uso Gerenciar Cliente**

**Objetivo:**

Permite o gerenciamento e manipulação (incluir, excluir, alterar e consultar) de clientes no banco de dados do sistema.

**Pré-condições:**

No cadastro de um novo cliente, ele não deve estar cadastrado no banco de dados. Para editar ou excluir, o cliente já deve estar incluso no banco de dados. Na opção de listar clientes, ao menos um cliente deve estar cadastrado no banco de dados.

**Iniciado por:**

Usuário do computador de frente de caixa.

**Fluxo principal:**

1. O sistema mostra a tela para escolha da rotina que será manipulada (inclusão, alteração ou consulta)
2. O sistema abre a tela com as opções incluir, alterar, excluir e consultar clientes.
3. O usuário seleciona a opção para incluir novo cliente.
4. O sistema exibe o formulário de cadastro.
5. O usuário informa nome, CPF e e-mail e seleciona a opção para gravação do novo cliente.
6. O sistema gera o identificador do cliente.
7. O sistema salva as informações no banco de dados.

**Fluxo alternativo: Alterar Cliente**

No item 3, caso o usuário selecione a opção de alterar um cliente, 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 Cliente**

No item 3, caso o usuário selecione a opção de excluir um cliente, 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.

**Fluxo alternativo: Listar Clientes**

No item 3, caso o usuário selecione a opção de listar clientes, todos os cliente são listados contendo todos os atributos, e ao fim, uma opção de voltar ao menu principal é apresentada.

**Pós-condições:**

Um cliente é criado, excluído, alterado ou todos os cliente são listados, caso o usuário não tenha abortado o processo.

**Descrição do Caso de Uso Gerenciar Fornecedor**

**Objetivo:**

Permite o gerenciamento e manipulação (incluir, excluir, alterar e consultar) de clientes no banco de dados do sistema.

**Pré-condições:**

No cadastro de um novo fornecedor, ele não deve estar cadastrado no banco de dados. Para editar ou excluir, o fornecedor já deve estar incluso no banco de dados. Na opção de listar fornecedores, ao menos um fornecedor deve estar cadastrado no banco de dados.

**Iniciado por:**

Usuário do computador de frente de caixa.

**Fluxo principal:**

1. O sistema mostra a tela para escolha da rotina que será manipulada (inclusão, alteração ou consulta)
2. O sistema abre a tela com as opções incluir, alterar, excluir e consultar fornecedores.
3. O usuário seleciona a opção para incluir novo fornecedores.
4. O sistema exibe o formulário de cadastro.
5. O usuário informa nome, CPF e e-mail e seleciona a opção para gravação do novo fornecedor.
6. O sistema gera o identificador do fornecedor.
7. O sistema salva as informações no banco de dados.

**Fluxo alternativo: Alterar Fornecedor**

No item 3, caso o usuário selecione a opção de alterar um fornecedor, 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 Fornecedor**

No item 3, caso o usuário selecione a opção de excluir um fornecedor, 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.

**Fluxo alternativo: Listar Fornecedor**

No item 3, caso o usuário selecione a opção de listar fornecedores, todos os fornecedores são listados contendo todos os atributos, e ao fim, uma opção de voltar ao menu principal é apresentada.

**Pós-condições:**

Um fornecedor é criado, excluído, alterado ou todos os fornecedores são listados, caso o usuário não tenha abortado o processo.

**Descrição do Caso de Uso Gerenciar Pedido de Compra**

**Objetivo:**

Permite o gerenciamento e manipulação (incluir, excluir, alterar e consultar) de pedidos de compra no banco de dados do sistema.

**Pré-condições:**

No cadastro de um novo pedido de compra, ele não deve estar cadastrado no banco de dados. Para editar ou excluir, o pedido de compra já deve estar incluso no banco de dados. Na opção de listar pedidos de compra, ao menos um pedido de compra deve estar cadastrado no banco de dados.

**Iniciado por:**

Usuário do computador de frente de caixa.

**Fluxo principal:**

1. O sistema mostra a tela para escolha da rotina que será manipulada (inclusão, alteração ou consulta)
2. O sistema abre a tela com as opções incluir, alterar, excluir e consultar pedidos de compra.
3. O usuário seleciona a opção para incluir novo pedidos de compra.
4. O sistema exibe o formulário de cadastro.
5. O usuário informa nome do produto, quantidade, preço unitário, status, data, nome do fornecedor e valor total e seleciona a opção para gravação do novo pedido de compra.
6. O sistema gera o identificador do pedido de compra.
7. O sistema salva as informações no banco de dados.

**Fluxo alternativo: Alterar Pedido de Compra**

No item 3, caso o usuário selecione a opção de alterar um pedido de compra, 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 Pedido de Compra**

No item 3, caso o usuário selecione a opção de excluir um pedido de compra, 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.

**Fluxo alternativo: Listar Pedido de Compra**

No item 3, caso o usuário selecione a opção de listar pedidos de compra, todos os pedidos de compra são listados contendo todos os atributos, e ao fim, uma opção de voltar ao menu principal é apresentada.

**Pós-condições:**

Um pedido de compra é criado, excluído, alterado ou todos os pedidos de compra são listados, caso o usuário não tenha abortado o processo.

**Descrição do Caso de Uso Gerenciar Pedidos de Venda**

**Objetivo:**

Permite o gerenciamento e manipulação (incluir, excluir, alterar e consultar) de pedidos de venda no banco de dados do sistema.

**Pré-condições:**

No cadastro de um novo pedido de venda, ele não deve estar cadastrado no banco de dados. Para editar ou excluir, o pedido de venda já deve estar incluso no banco de dados. Na opção de listar pedidos de venda, ao menos um pedido de venda deve estar cadastrado no banco de dados.

**Iniciado por:**

Usuário do computador de frente de caixa.

**Fluxo principal:**

1. O sistema mostra a tela para escolha da rotina que será manipulada (inclusão, alteração ou consulta)
2. O sistema abre a tela com as opções incluir, alterar, excluir e consultar pedidos de venda.
3. O usuário seleciona a opção para incluir novo pedido de venda.
4. O sistema exibe o formulário de cadastro.
5. O usuário informa nome, CPF e e-mail e seleciona a opção para gravação do novo pedido de venda.
6. O sistema gera o identificador do pedido de venda.
7. O sistema salva as informações no banco de dados.

**Fluxo alternativo: Alterar Pedido de Venda**

No item 3, caso o usuário selecione a opção de alterar um pedido de venda, 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 Pedido de Venda**

No item 3, caso o usuário selecione a opção de excluir um pedido de venda, 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.

**Fluxo alternativo: Listar Pedido de Venda**

No item 3, caso o usuário selecione a opção de listar pedidos de venda, todos os pedidos de venda são listados contendo todos os atributos, e ao fim, uma opção de voltar ao menu principal é apresentada.

**Pós-condições:**

Um pedido de venda é criado, excluído, alterado ou todos os pedidos de venda são listados, caso o usuário não tenha abortado o processo.

* 1. **Modelo de análise**

* 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**

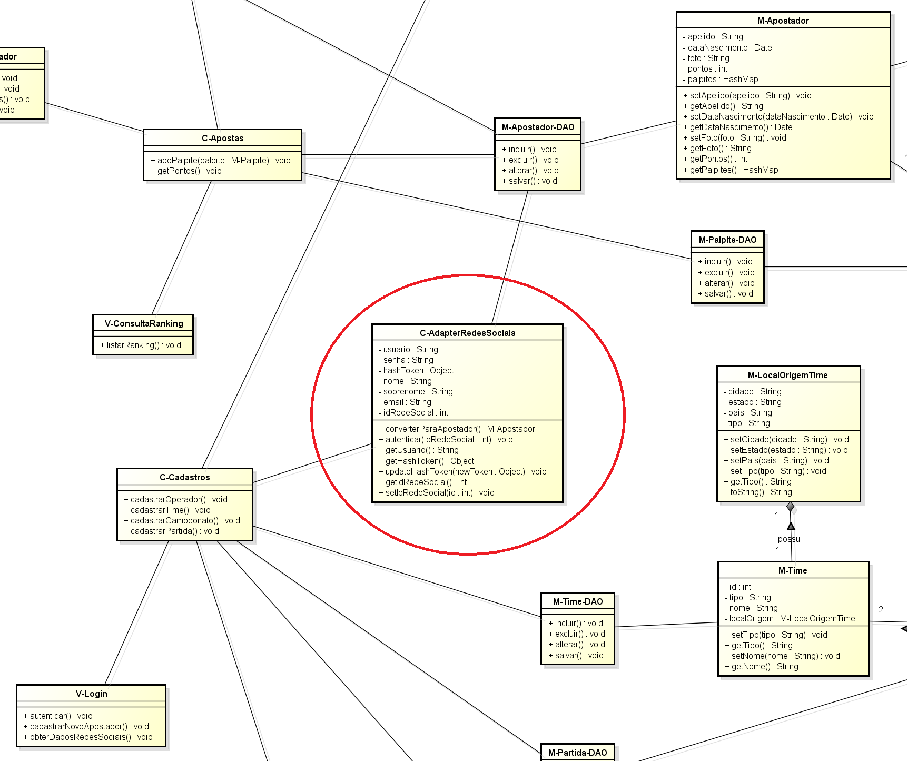
1. **Modelo de projeto**
   1. **Padrões de projeto**

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.

**Spring**

O Spring é um framework não intrusivo, baseado nos padrões de projeto inversão de controle e injeção de independência.

No Spring, o container se encarrega de instanciar classes de uma aplicação Java e definir as dependências entre elas através de um arquivo de configuração em formato XML, inferências do framework (auto-wiring ou anotações de classes), métodos e propriedades. Desta forma o Spring permite o baixo acoplamento entre classes de uma aplicação orientada a objetos.

O Spring possui uma arquitetura baseada em interface e POJOs (classes básicas), oferecendo ao POJOs características como mecanismos de segurança e controle de transações. Também facilita testes unitários e surge como uma alternativa à complexidade existente no uso de EJBs (componente da plataforma Java). Com Spring, pode-se ter um alto desempenho da aplicação.

Esse framework oferece diversos módulos que podem ser utilizados de acordo com as necessidades do projeto, como módulos voltados para desenvolvimento Web, persistência, acesso remoto e programação orientada a aspectos.

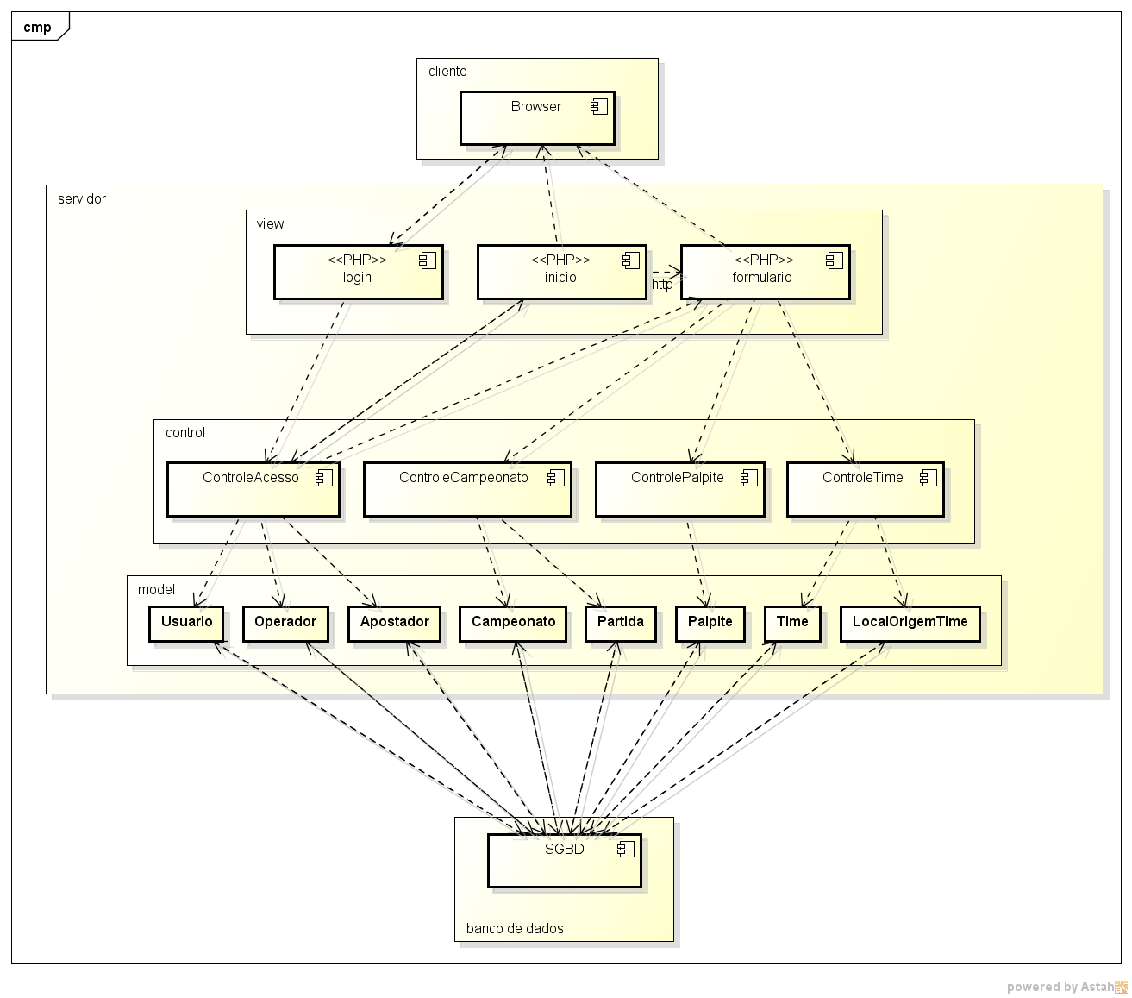
**Hibernate**

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.

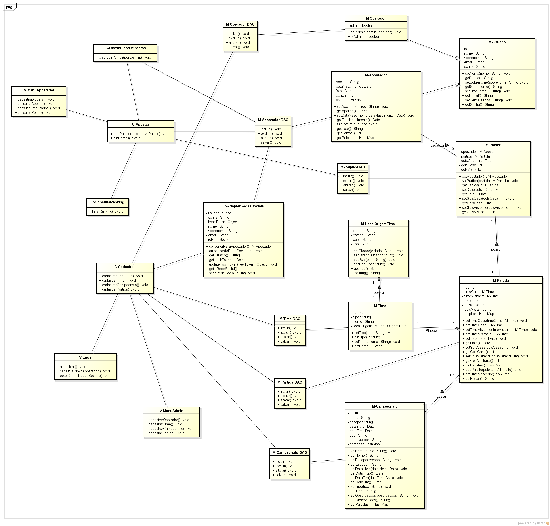
O objetivo do Hibernate é diminuir a complexidade entre os programas Java, baseado no modelo orientado a objeto, que precisam trabalhar com um banco de dados do modelo relacional. Em especial, no desenvolvimento de consultas e atualizações dos dados.

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.

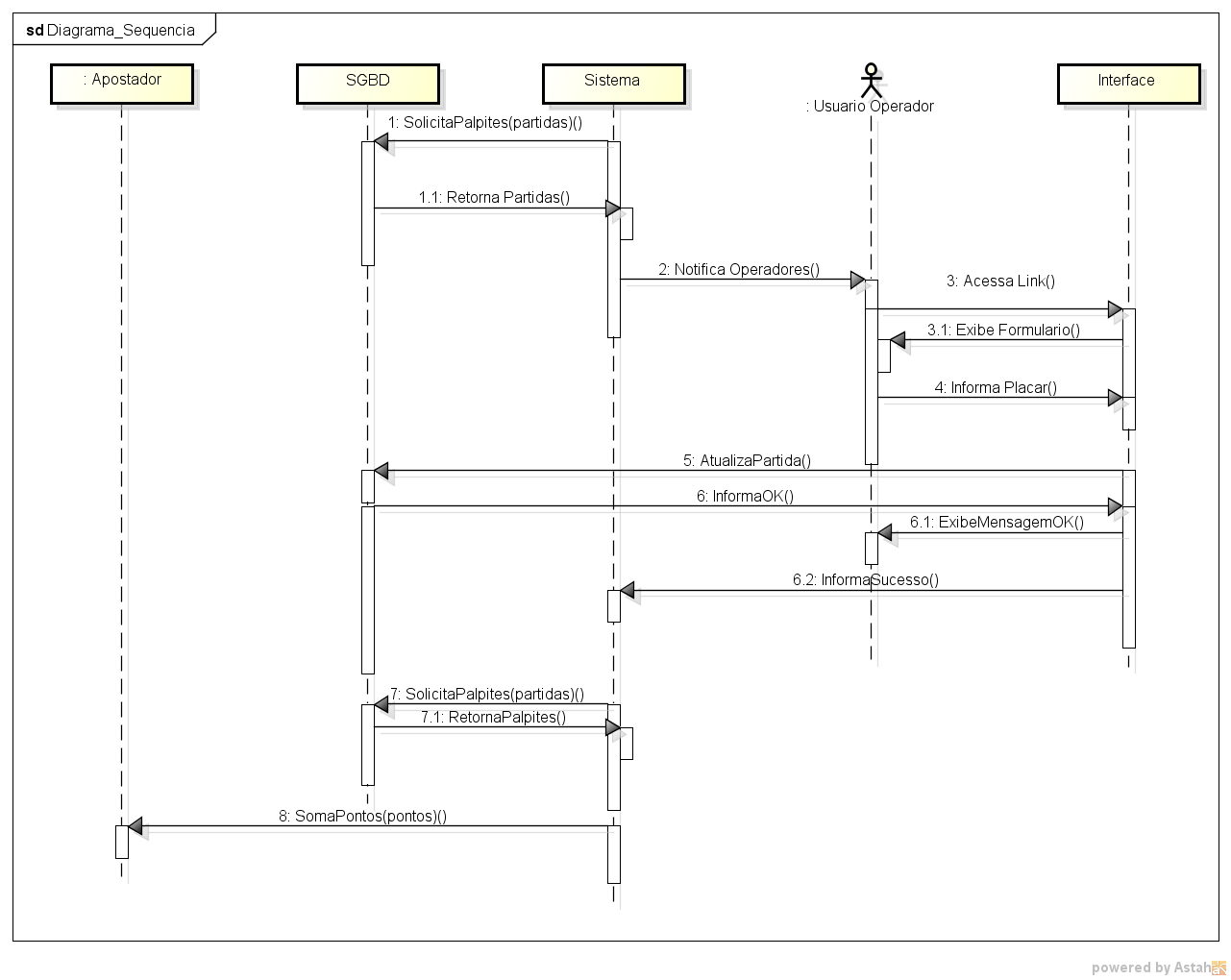
* 1. **Diagrama de componentes**



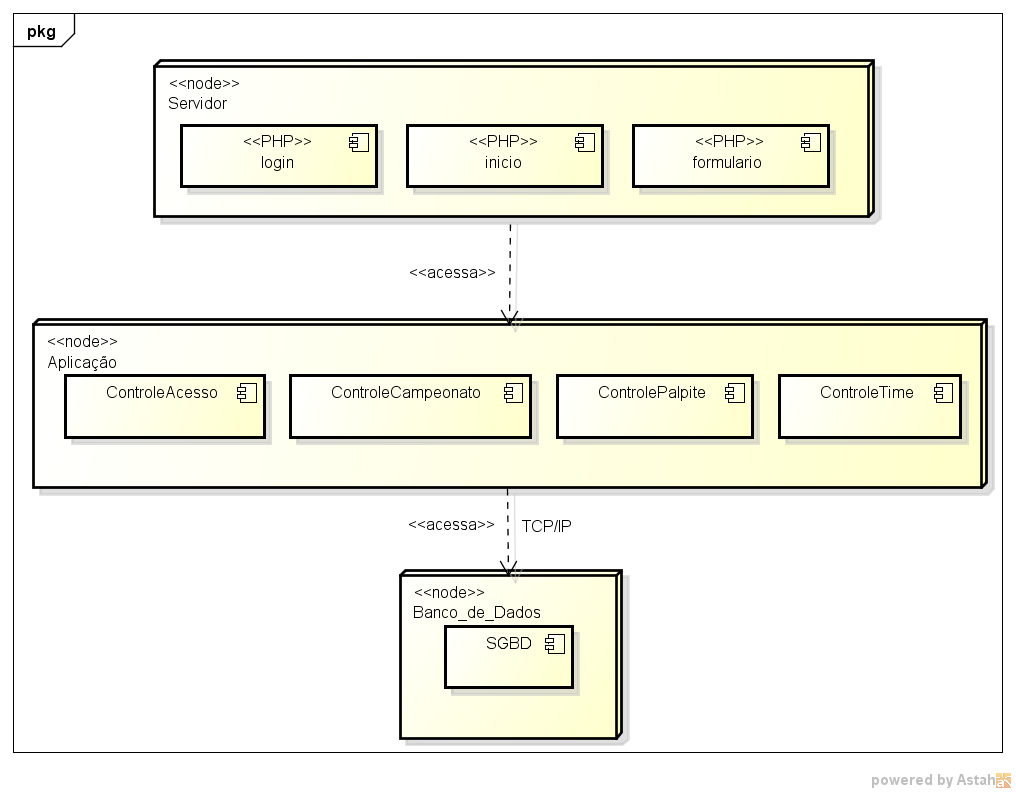
* 1. **Diagrama de classes de projeto**



* 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.

* 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 |