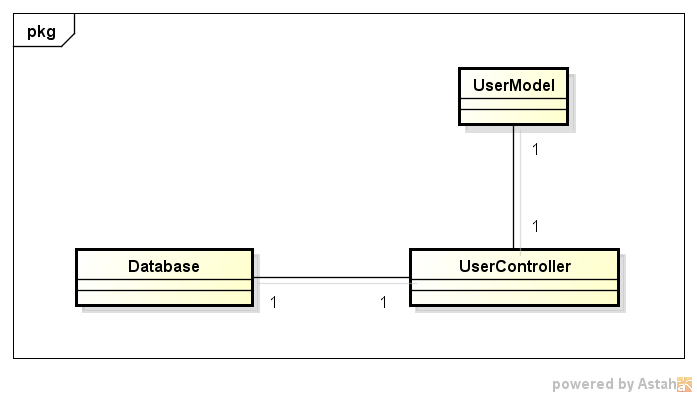
3.3 Diagrama de Classe do *WebService*

O diagrama de classe é responsável por oferecer uma representação da estrutura e relações das classes como também as operações solicitadas pelos atores que servem de modelo para os objetos.

*UserModel*: Esta classe tem por objetivo fornecer a classe *UserController* os atributos necessários para as regras de negócio (Figura 5). Não há métodos nesta classe.

Figura 5 - Diagrama de classe de usuário.Fonte: Elaborado pelo autor

*Database*: Esta classe tem por objetivo realizar a conexão com o Banco de Dados (Figura 5). Os métodos identificados para esta classe são apresentados na Tabela 20:

Tabela 20 - Diagrama de classe de usuários.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| Conexão | Abre conexão com o banco de dados |

Fonte: Elaborado pelo autor

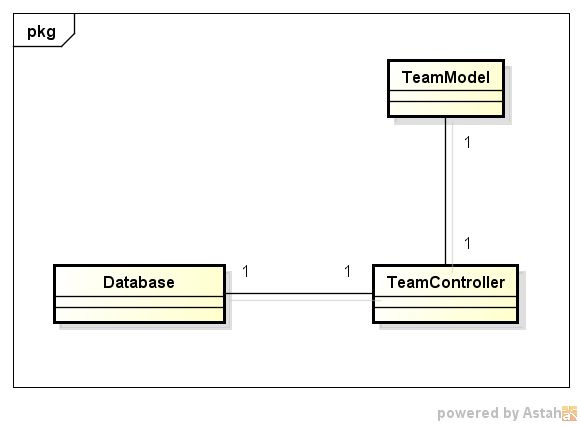
*UserController*: Esta classe tem por objetivo realizar todas as ações referentes aos dados do usuário (Figura 5). Os métodos identificados para esta classe são apresentados na Tabela 21:

Tabela 21 - Diagrama de classe de usuários.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *login* | Realiza uma validação de usuário no sistema. |
| *get* | Busca um usuário. |
| *getAll* | Lista uma lista de usuários ordenando-os por pontuação. |
| *post* | Cria um novo usuário. |
| *put* | Atualiza os dados de um usuário. |
| *delete* | Deleta a conta de um usuário. |

Fonte: Elaborado pelo autor

*TeamModel*: Esta classe tem por objetivo fornecer a classe *TeamController* os atributos necessários para as regras de negócio (Figura 6). Não há métodos nesta classe.

Figura 6 - Diagrama de classe de Times.Fonte: Elaborado pelo autor

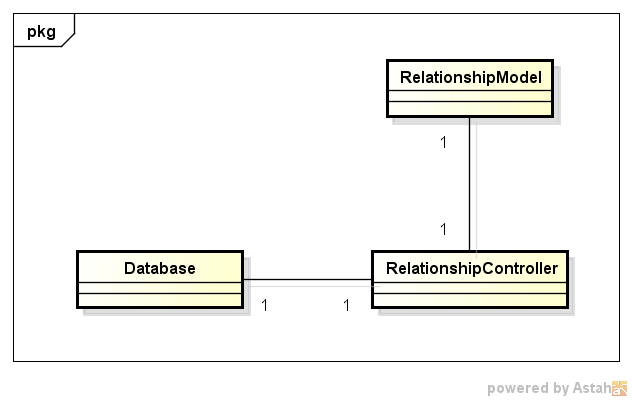
*TeamController*: Esta classe tem por objetivo realizar as ações em um time (Figura 6). Os métodos identificados para esta classe são apresentados na Tabela 22:

Tabela 22 - Diagrama de classe de times.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *get* | Busca um time. |
| *getAll* | Busca todos os times ordenados por pontuação. |
| *post* | Cria um novo time. |
| *put* | Atualiza um time. |
| *delete* | Remove um time. |

Fonte: Elaborado pelo autor

*RelationshipModel*: Esta classe tem por objetivo fornecer a classe *RelationshipController* os atributos necessários para as regras de negócio (Figura 7). Não há métodos nesta classe.

Figura 7 - Diagrama de classe de amizade.Fonte: Elaborado pelo autor

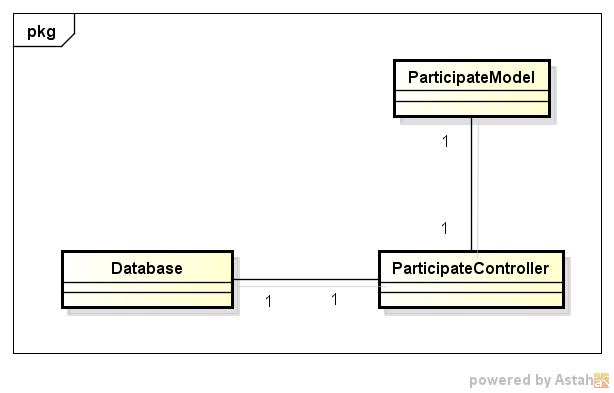
*RelationshipController*: Esta classe tem por objetivo formar uma relação entre usuários (Figura 7). Os métodos identificados para esta classe são apresentados na Tabela 23:

Tabela 23 - Diagrama de classe de amizade.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *getAll* | Busca todas as amizades de um usuário. |
| *get* | Busca relação de amizade com outro usuário |
| *post* | Cria uma nova amizade. |
| *delete* | Remove uma amizade entre usuários. |

Fonte: Elaborado pelo autor

*ParticipateModel*: Esta classe tem por objetivo fornecer a classe *ParticipateController* os atributos necessários para as regras de negócio (Figura 8). Não há métodos nesta classe.

Figura 8 - Diagrama de classe de participação do usuário no time.Fonte: Elaborado pelo autor

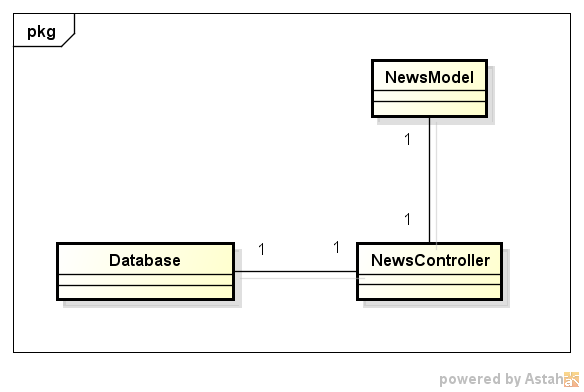
*ParticipateController*: Esta classe tem por objetivo realizar a conexão entre um usuário e um time (Figura 8). Os métodos identificados para esta classe são apresentados na Tabela 24:

Tabela 24 - Diagrama de classe de participação do usuário no time.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *get* | Busca um time de um usuário. |
| *post* | Cria uma nova participação de um usuário em um time. |
| *put* | Atualiza o usuário Capitão do time, caso o anterior saia dele. |
| *delete* | Remove a participação de um usuário em um time. |

Fonte: Elaborado pelo autor

*NewsModel*: Esta classe tem por objetivo fornecer a classe *NewsController* os atributos necessários para as regras de negócio (Figura 9). Não há métodos nesta classe.

Figura 9 - Diagrama de classe de Notícias.Fonte: Elaborado pelo autor

*NewsController*: Esta classe tem por objetivo realizar as ações das notícias para os usuários (Figura 9). Os métodos identificados para esta classe são apresentados na Tabela 25:

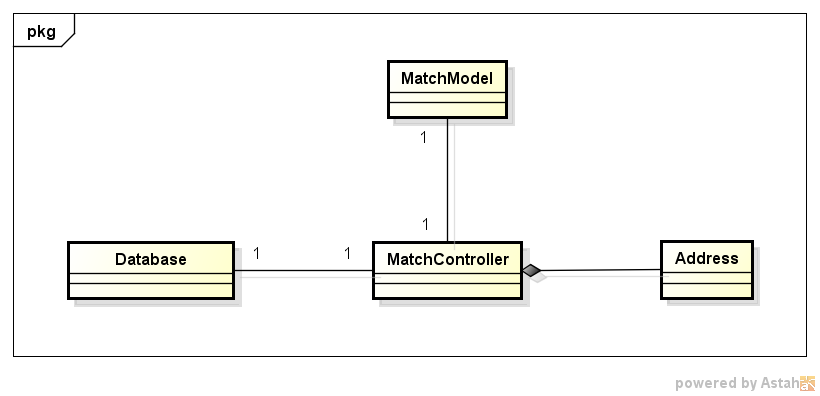
Tabela 25 - Diagrama de classe de Notícias.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *getAll* | Busca todas as notícias de futebol retornadas pela API de Bing *News*. |
| *get* | Busca uma notícia. |
| *post* | Cria uma nova notícia. |
| *put* | Atualiza uma notícia. |
| *delete* | Remove uma notícia. |

Fonte: Elaborado pelo autor

*MatchModel*: Esta classe tem por objetivo fornecer a classe *MatchController* os atributos necessários para as regras de negócio (Figura 10). Não há métodos nesta classe.

*Address*: Esta classe é uma classe dependente que existe apenas para disponibilizar os dados de endereço para a classe MatchController (Figura 10)

Figura 10 - Diagrama de classe de Partida.Fonte: Elaborado pelo autor

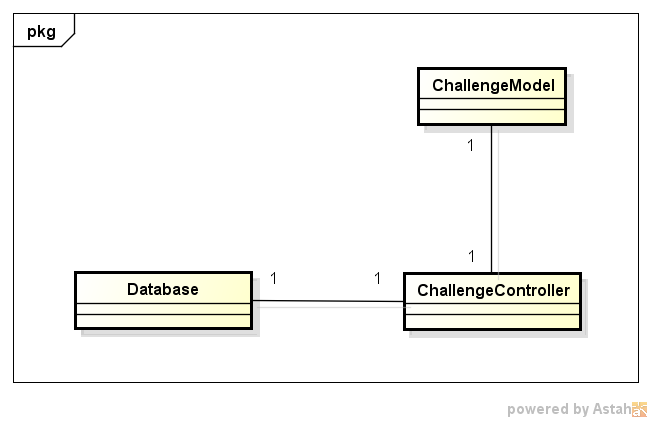
*MatchController*: Esta classe tem por objetivo realizar as ações em uma partida marcada entre times (Figura 10). Os métodos identificados para esta classe são apresentados na Tabela 26:

Tabela 26 - Diagrama de classe de Partida.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *get* | Busca os dados de uma partida. |
| *post* | Cria uma nova partida. |
| *delete* | Remove uma partida. |

Fonte: Elaborado pelo autor

*ChallengeModel*: Esta classe tem por objetivo fornecer a classe *ChallengeController* os atributos necessários para as regras de negócio (Figura 11). Não há métodos nesta classe.

Figura 11 - Diagrama de classe de Desafio.Fonte: Elaborado pelo autor

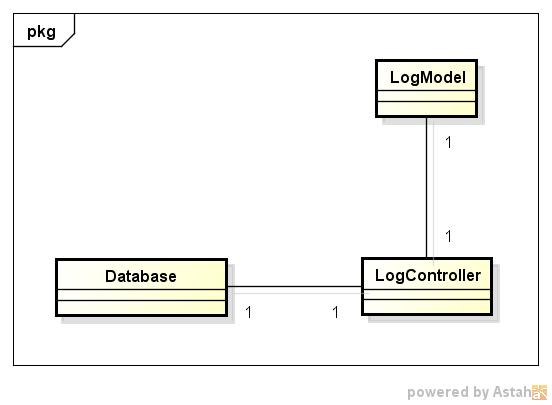
*ChallengeController*: Esta classe tem por objetivo realizar as ações referentes a um desafio entre times (Figura 11). Os métodos identificados para esta classe são apresentados na Tabela 27:

Tabela 27 - Diagrama de classe de Desafio.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *getAll* | Busca os Desafios de um time. |
| *get* | Busca as informações de um Desafio em específico. |
| *post* | Cria um novo Desafio. |
| *delete* | Remove um desafio. |

Fonte: Elaborado pelo autor

*LogModel*: Esta classe tem por objetivo fornecer a classe *LogController* os atributos necessários para as regras de negócio (Figura 12). Não há métodos nesta classe.

Figura 12 - Diagrama de classe de Log.Fonte: Elaborado pelo autor

*LogController*: Esta classe tem por objetivo realizar as ações referentes a um desafio entre times (Figura 12). Os métodos identificados para esta classe são apresentados na Tabela 28:

Tabela 28 - Diagrama de classe de Log.

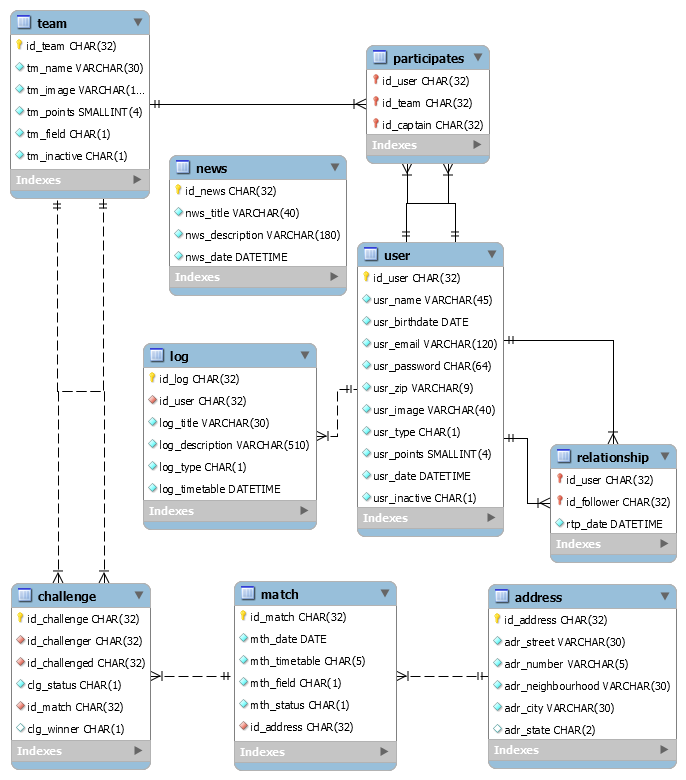
|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *getAll* | Busca todos Log completo de acordo com número de página passada |
| *get* | Busca os dados de um Log específico. |
| *post*  *put* | Cria um novo log.  Atualiza um log. |
| *delete* | Remove um desafio. |

Fonte: Elaborado pelo autor

3.4 Diagrama de Entidade e Relacionamento

Diagrama Entidade Relacionamento (DER) é um [modelo diagramático](http://pt.wikipedia.org/wiki/Diagrama) que descreve o modelo de dados de um sistema com alto nível de [abstração](http://pt.wikipedia.org/wiki/Abstra%C3%A7%C3%A3o). Ele é a principal representação do [Modelo de Entidades e Relacionamentos](http://pt.wikipedia.org/wiki/Modelo_de_Entidades_e_Relacionamentos). Sua maior aplicação é visualizar o relacionamento entre tabelas de um [banco de dados](http://pt.wikipedia.org/wiki/Banco_de_dados), no qual as relações são construídas através da associação de um ou mais atributos destas tabelas (SOMMERVILLE, 2011). A Figura 13 apresenta o DER do sistema proposto.

Figura 13 - Diagrama de Entidade e Relacionamento.



**Fonte: Elaborado pelo autor**

3.4.1 Dicionário de Dados

O Dicionário de Dados (DD) consiste numa lista organizada de todos os elementos de dados que são pertinentes ao sistema. As tabelas devem conter os seguintes campos: Entidade, Atributo, Classe, Domínio, Tamanho e Descrição. A Tabela 29 apresenta o DD da entidade *User*.

Tabela 29 - Dicionário de Dados da entidade *User*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entidade: *User*** | | | | |
| **Atributo** | **Classe** | **Domínio** | **Tamanho** | **Descrição** |
| id\_user | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |
| usr\_name | Simples | Texto | 45 | Nome do usuário |
| usr\_birthdate | Simples | Data |  | Data de Nascimento |
| usr\_email | Simples | Texto | 120 | E-mail |
| usr\_password | Simples | Texto | 64 | Senha |
| usr\_zip | Simples | Texto | 9 | CEP |
| usr\_image | Simples | Texto | 40 | Endereço da imagem de perfil |
| usr\_type | Simples | Texto | 1 | Identifica o tipo de usuário (Administrador, proprietário ou jogador) |
| usr\_points | Simples | Numérico | 4 | Pontos do time |
| usr\_date | Simples | Data |  | Data de cadastro do usuário. |
| usr\_inactive | Simples | Texto | 1 | Conta está ativa ou inativa. |

**Fonte: Elaborado pelo autor**

A Tabela 30 apresenta o DD da entidade *Relationship*.

Tabela 30 - Dicionário de Dados da entidade *Relationship*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entidade: *Relationship*** | | | | |
| **Atributo** | **Classe** | **Domínio** | **Tamanho** | **Descrição** |
| id\_user | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |
| id\_follower | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |
| rtp\_date | Simples | Data |  | Data que amizade iniciou. |

**Fonte: Elaborado pelo autor**

A Tabela 31 apresenta o DD da entidade *Participates*.

Tabela 31 - Dicionário de Dados da entidade *Participates*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entidade: *Participates*** | | | | |
| **Atributo** | **Classe** | **Domínio** | **Tamanho** | **Descrição** |
| id\_user | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |
| id\_team | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |
| id\_captain | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |

**Fonte: Elaborado pelo autor**

A Tabela 32 apresenta o DD da entidade *News*.

Tabela 32 - Dicionário de Dados da entidade *News*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entidade: *News*** | | | | |
| **Atributo** | **Classe** | **Domínio** | **Tamanho** | **Descrição** |
| id\_news | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |
| nws\_title | Simples | Texto | 40 | Título da notícia |
| nws\_description | Simples | Texto | 180 | Descrição da notícia |
| **Atributo** | **Classe** | **Domínio** | **Tamanho** | **Descrição** |
| nws\_date | Simples | Data |  | Data de publicação |

**Fonte: Elaborado pelo autor**

A Tabela 33 apresenta o DD da entidade *Log*.

Tabela 33 - Dicionário de Dados da entidade *Log*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entidade: *Log*** | | | | |
| **Atributo** | **Classe** | **Domínio** | **Tamanho** | **Descrição** |
| id\_log | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |
| id\_user | Simples | Texto | 32 | Identificador universal composto por 32 caracteres |
| log\_title | Simples | Texto | 30 | Título do log |
| log\_description | Simples | Texto | 510 | Descrição do log |
| log\_type | Simples | Texto | 1 | Se é operação interna ou *Exception* |

**Fonte: Elaborado pelo autor**

A Tabela 34 apresenta o DD da entidade *Team*.

Tabela 34 - Dicionário de Dados da entidade *Team*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entidade: *Team*** | | | | | |
| **Atributo** | **Classe** | **Domínio** | **Tamanho** | | **Descrição** | |
| id\_team | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres | |
| tm\_name | Simples | Texto | 30 | Nome do time | |
| tm\_image | Simples | Texto | 19 | Imagem do perfil do time | |
| tm\_points | Simples | Numérico | 4 | Pontos do time | |
| tm\_field | Simples | Texto | 1 | Tipo de campo do time | |
| tm\_inactive | Simples | Texto | 1 | Se time está ativo ou inativo | |

**Fonte: Elaborado pelo autor**

A Tabela 35 apresenta o DD da entidade *Challenge*.

Tabela 35 - Dicionário de Dados da entidade *Challenge*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entidade: *Challenge*** | | | | |
| **Atributo** | **Classe** | **Domínio** | **Tamanho** | **Descrição** |
| id\_challenge | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |
| id\_chellenger | Simples | Texto | 32 | Identificador universal composto por 32 caracteres |
| id\_challenged | Simples | Texto | 32 | Identificador universal composto por 32 caracteres |
| clg\_status | Simples | Texto | 1 | Partida aconteceu. |
| id\_mach | Simples | Texto | 32 | Identificador universal composto por 32 caracteres |
| clg\_winner | Simples | Texto | 1 | Define quem ganhou. |

**Fonte: Elaborado pelo autor**

A Tabela 36 apresenta o DD da entidade *Match*.

Tabela 36 - Dicionário de Dados da entidade *Match*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entidade: *Match*** | | | | |
| **Atributo** | **Classe** | **Domínio** | **Tamanho** | **Descrição** |
| id\_match | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |
| mth\_date | Simples | Data |  | Data da partida |
| mth\_timetable | Simples | Texto | 5 | Horário da partida |
| mth\_field | Simples | Texto | 1 | Tipo de campo |
| mth\_status | Simples | Texto | 1 | Se já ocorreu ou não |
| id\_address | Simples | Texto | 32 | Identificador universal composto por 32 caracteres |

**Fonte: Elaborado pelo autor**

A Tabela 37 apresenta o DD da entidade *Address*.

Tabela 37 - Dicionário de Dados da entidade *Address*.

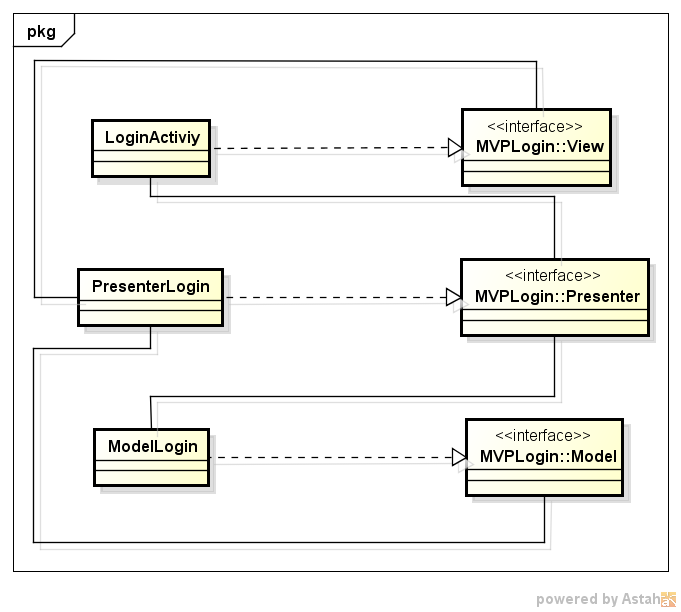
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entidade: *Address*** | | | | |
| **Atributo** | **Classe** | **Domínio** | **Tamanho** | **Descrição** |
| id\_address | Determinante | Texto | 32 | Identificador universal composto por 32 caracteres |
| adr\_street | Simples | Texto | 30 | Rua |
| adr\_number | Simples | Texto | 5 | Número |
| adr\_neighborhood | Simples | Texto | 30 | Bairro |
| adr\_city | Simples | Texto | 30 | Cidade |
| adr\_estate | Simples | Texto | 2 | Estado |

**Fonte: Elaborado pelo autor**

3.5 Diagrama de Classe do Aplicativo

Os diagramas de classe seguintes são representações da estrutura, das relações das classes e também das operações solicitadas pelos atores que servem de modelo para os objetos referentes ao aplicativo *mobile*.

A Figura 14 apresenta o diagrama de classe de login.

Figura 14 - Diagrama de classe de login.Fonte: Elaborado pelo autor

*LoginActivity*: Esta classe tem por objetivo receber os dados do usuário (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 38:

Tabela 38 – Métodos do Diagrama de classe de login.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *Cadastrar*  *Error*  *login\_invalido*  *login\_valido*  *validar*  *onCreate* | Carrega a tela de cadastrar  Emite uma janela de diálogo para informar de erro  Emite uma janela de diálogo para informar de falha nos dados inseridos  Abre a tela de *Splash*  Valida os campos  Estabelece contato com o Layout da *Activity*, inflando componente XML para transforma-lo em objeto do tipo *Views* |

Fonte: Elaborado pelo autor

*PresenterLogin*: Esta classe tem por objetivo representar a camada *Presenter* da arquitetura MVP (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 39:

Tabela 39 - Métodos do Diagrama de classe de login.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *checkLogin*  *error*  *getContext*  *login\_invalido*  *login\_valido*  *PresenterLogin*  *setView* | *presenter* do método da classe *Model*  *presenter* do método da classe *LoginActivity*  *presenter* do método que retorna o contexto para a classe que o chamar  *presenter* do método da classe *LoginActivity*  *presenter* do método da classe *LoginActivity*  construtor  *presenter* para construção da janela de diálogo |

Fonte: Elaborado pelo autor

*ModelLogin*: Esta é classe de modelagem (regra de negócio) (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 40:

Tabela 40- Métodos do Diagrama de classe de login.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *checkLogin*  *computeHash*  *ModelLogin* | realiza requisição API para realizar Login  gerador de criptografia SHA-256  construtor da classe |

Fonte: Elaborado pelo autor

*Interface MVPLogin::View* : Esta classe tem por objetivo realizar a interface para a view (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 41:

Tabela 41 - Métodos do Diagrama de classe de login.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *Error*  *login\_invalido*  *login\_valido* | Interface para o método *error*  Interface para o método *login\_invalido*  Interface para o método *login\_valido* |

Fonte: Elaborado pelo autor

*Inteface MVPLogin::* *Presenter*: Esta classe tem por objetivo realizar a interface para o *presenter* (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 42:

Tabela 42 - Métodos do Diagrama de classe de login.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *checkLogin*  *error*  *getContext*  *login\_invalido*  *login\_valido*  *setView* | Interface para o método *checkLogin*  Interface para o método *error*  Interface para o método *getContext*  Interface para o método *login\_invalido*  Interface para o método *login\_valido*  Interface para o método *setView* |

**Fonte: Elaborado pelo autor**

*Inteface MVPLogin:: Model*: Esta classe tem por objetivo realizar a interface para o *model* (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 43:

Tabela 43 - Métodos do Diagrama de classe de login.

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| *checkLogin* | Interface para o método *checkLogin* |

**Fonte: Elaborado pelo autor**