

## Relatório: Etapa 2

André Meneghelli Vale - 4898948

andredalton@gmail.com

Taís Pinheiro - 7580421

tais.aparecida.pinheiro@usp.br

17 de Novembro

# 1 Introdução

O projeto da disciplina *Laboratório de Programação 2* consiste no desenvolvimento de um sistema de batalhas entre dois jogadores de pokemom baseado no jogo homônimo.

Para tanto foi escolhida a linguagem Python versão 3 e o desenvolvimento orientado a objetos.

Foram determinadas inicialmente três etapas:

- **Etapla 1:** Modelagem da batalha em modo texto;
- **Etapla 2:** Comunicação ponto a ponto via rede pra batalhas;
- **Etapla 3:** Inteligência artificial para as batalhas;

Este relatório se foca apenas na *etapa 2*.

## 1.1 Desenvolvimento

Nesta etapa foram realizadas melhorias de forma a padronizar as classes ao padrão descrito no enunciado e a facilitar o desenvolvimento dos módulos de servidor e cliente.

Algumas rotinas comuns ao servidor e ao cliente foram mantidas no módulo `battle.py`. Permitindo assim a reutilização de código pelos `battle_client.py` e `battle_server.py`.

### 1.1.1 `battle.py`

Este modulo contém 5 métodos:

- `usage()`: Imprime o módo de utilização da linha de comando;
- `command_line(argv)`: Realiza o tratamento adequado da linha de comando;
- `validate(s)`: Faz a validação de um `battle_state`;
- `make_battle_state(pk1, pk2)`: Cria um novo `battle_state` para um ou dois pokemons;
- `simple_duel(patt, pdef)`: Realiza um ataque de `patt` em `pdef`.

### 1.1.2 `battle_server.py`

Contém o servidor, utiliza o tratamento da linha de comando contido no módulo *battle.py*. Não foi implementada uma validação de usuário adequada. Permitindo que qualquer acesso indevido a página de ataque modifique a batalha. Uma vez que a batalha atual tenha terminado, o pokémon do servidor é recarregado e aguarda um novo desafiante.

### 1.1.3 `battle_client.py`

Contém o cliente, também utiliza o tratamento de linha de comando contido no módulo *battle.py*.

Caso o servidor não esteja respondendo, avisa o usuário e termina a execução.

Caso o servidor responda com código diferente de 200, aguarda 10 segundos e tenta novamente até que consiga uma resposta adequada.

## 1.2 Modo de execução

Tanto o cliente como o servidor realizam o mesmo tratamento da linha de comando.

- **sem argumentos:** Modo default de execução, recebe os dados pela entrada padrão do sistema.
- **-h:** Imprime uma ajuda, **-h** pode ser substituído por **--help**;
- **-f:** Abre o arquivo passado, **-f** pode ser substituído por **--file**;
- **-x:** Abre o arquivo XML passado, **-x** pode ser substituído por **--xml**;

## 1.3 Teste

Também foi desenvolvido um teste, que verifica se dois pokémons originados do mesmo arquivo selecionado aleatoriamente de `./billpc/` realmente montam objetos distintos mas possuem alguns atributos instanciados no mesmo lugar.

Para executá-lo basta digitar `./teste_pokemon.py`.

## 2 Descrição das Classes

Algumas simplificações foram realizadas nesta etapa, como pode ser verificado no novo diagrama de classes. Iremos apenas descrever as alterações, portanto não haverá descrição da classe *Kind* e nem de suas especializações.

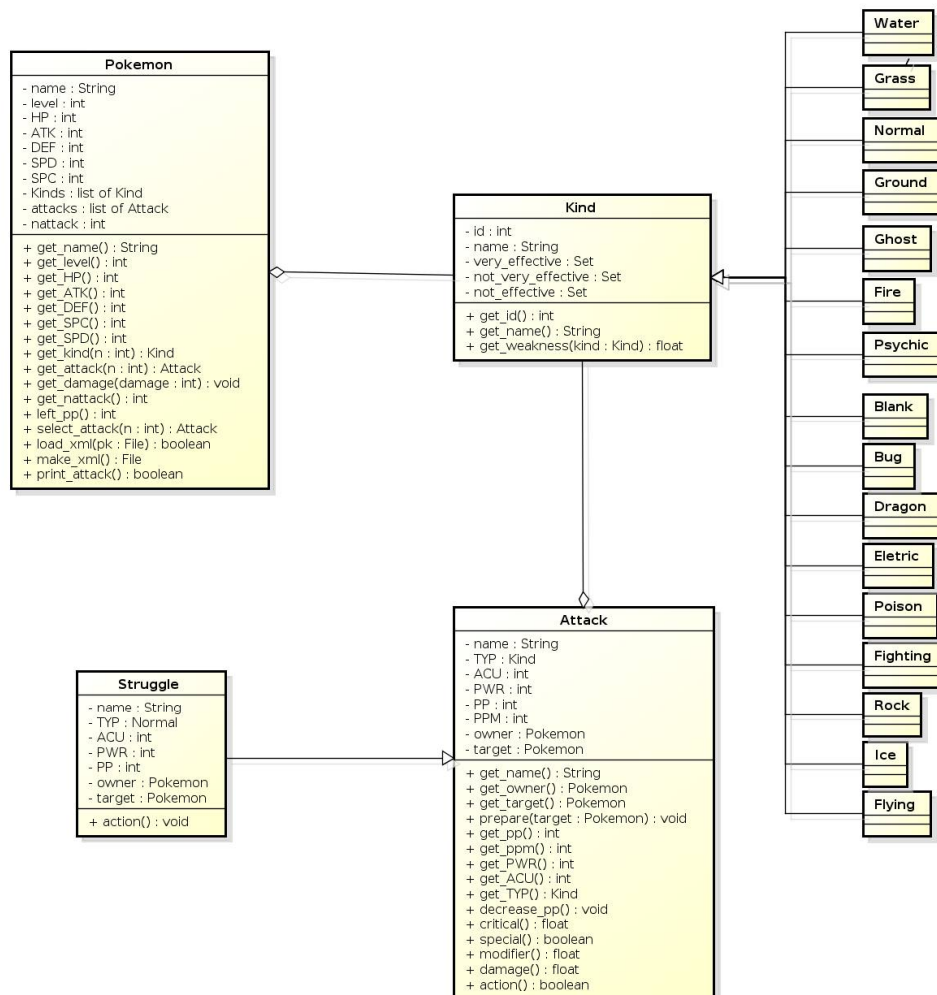


Diagrama de classes

## 2.1 Pokémon

É a classe principal do módulo pokémon e a única instanciada em `battle_server` e em `battle_client`.

Os atributos *action1*, *action2*, *action3* e *action4* foram substituídos pela lista *attacks*. O *struggle* agora está em *attacks[0]*.

Os atributos *kind1* e *kind2* foram substituídos pela lista *kinds*.

Os atributos *last\_hit* e *nhits* e seus métodos getters foram removidos já que eram utilizados apenas em *Counter*.

Os métodos *get\_kind1()*, *get\_kind2()*, *get\_action1()*, *get\_action2()*, *get\_action3()* e *get\_action4()* foram substituídos por novos métodos.

Foram desenvolvidos novos métodos:

- **get\_kind(n)**: Retorna o *n*-ésimo *Kind*, substituiu os antigos métodos *get\_kind1()* e *get\_kind2()*;
- **get\_attack(n)**: Retorna *n*-ésimo *Attack*, substituiu os antigos métodos *get\_action1()*, *get\_action2()*, *get\_action3()* e *get\_action4()*;
- **load\_xml(pk)**: Recebe um *lxml.etree* com um pokémon já validado e inicializa o pokémon.
- **make\_xml()**: Retorna um *lxml.etree.Element* contendo os dados deste pokémon.

## 2.2 Attack

Esta classe define um ataque genérico de um turno direto ao alvo. Anteriormente era uma especialização da classe *Skill*, mas como não será necessário o uso de qualquer outra habilidade que não seja um ataque simples a classe *Skill* foi removida. A especialização *Counter* também se mostrou desnecessária e foi removida.

### 3 Conclusão

Nesta etapa foi desenvolvido um método de batalha entre dois pokémons via rede. Para tanto foram desenvolvidos dois módulos em python que utilizam as classes desenvolvidas na *Etapa1*. Um módulo para servidor e outro para cliente.

Foi possível verificar que a manutenção do código orientado a objeto foi bastante fácil, mesmo tendo que realizar algumas modificações entre as duas etapas.

Embora ainda não exista uma *IA* para controlar os pokémons sozinha, é possível automatizar a resposta do servidor carregando um pokémon que não possui nenhum *pp*. O que facilitou bastante os testes.

## Referências

- [1] Enunciado
- [2] Bulbapedia
- [3] Serebii
- [4] Pokémon Red e Blue - Wikipedia
- [5] Modules - Python
- [6] Flask - Python