Pokemon Battle - 2ª Etapa

Bruno Sesso	8536002
César Yapunari Nontol Rodríguez	9137902
Gustavo Estrela de Matos	8536051
Rafael Mendonça Miller	7581818

25 de outubro de 2014

INTRODUÇÃO

Nessa etapa adicionamos funcionalidade de servidor ao programa que tivemos na fase 1. Para isso, usamos as mesmas ferramentas. A principio tinhamos um código que funcionava offline e adicionamos funcionalidade online que utiliza a *API RESTful.* Além disso para realizar a comunicação entre servidor e cliente tivemos que transformar as instancias de Pokemon em um arquivo xml.

DESENVOLVIMENTO

Problemas

A princípio precisamos resolver alguns problemas que tinham sobrado da primeira fase. Dentre eles, podemos citar: testes, o ataque struggle, ler pela *stdin, entre outros.*

Um grande problema foi aprender sobre como funciona o servidor e a *RESTful API.* Um grande problema foi a documentação dos módulos que nos auxiliariam, como o Flask e o Bottle.

Realizar testes em certas classes foi uma tarefa muito difícil. Classes como a *BattleIO* que tratam entrada e saída através da stdin e stdout, ou a classe *ServerBattle* que deixa o servidor rodando, são classes difíceis de serem testadas.

O maior problema dessa fase foi adicionar novas funcionalidades a um código já pronto e ainda assim manter a expansibilidade do programa.

Soluções

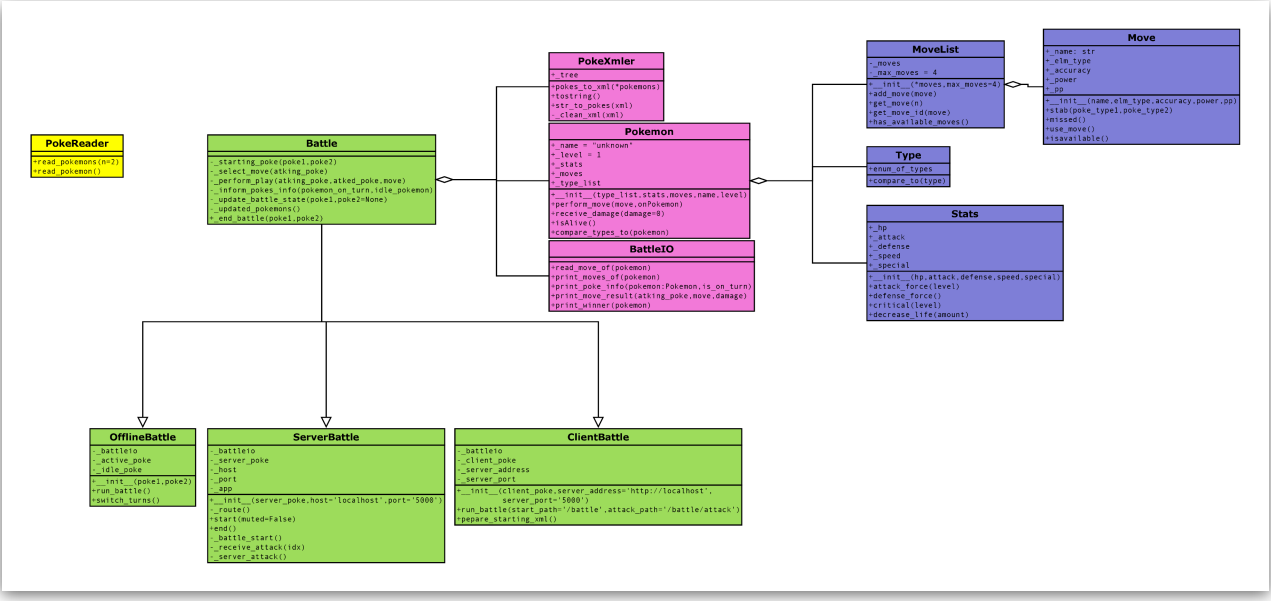
Primeiramente solucionamos os problemas restantes da fase anterior. Simultaneamente, começamos a pensar na estrutura do código na nova fase, porém como ainda não tinhamos conhecimento suficiente de servidores não conseguimos ter boas ideias.

Primeiramente tentamos utilizar o módulo Flask, porém encontramos algumas dificuldades na utilização desse, principalmente em sua documentação. Encontramos, então o módulo Bottle, que atendeu as nossas necessidades. A partir daí, passou-se a se preocupar com o XML e a comunicação. Que foi feita utilizando o módulo de xml do python.

Notamos que a realização dos testes paralelamente enquanto fazemos o código é muito importante. Isso ajuda bastante e notamos a importância de faze-los. Se fizéssemos testes melhores ainda poderíamos ter mais segurança ainda nos testes. Porém cada método tem apenas poucos testes. Ainda assim isso nos dá confiança nas mudanças que fazemos ao longo do desenvolvimento. Para testar, também fizemos um script que pega a lista de pokemons num site e converte para o formato usado no projeto. Portanto geramos diversos pokemons para usarmos de entrada. Testes nas classes difíceis de serem testadas foram feitos, porém não foi possível realiza-los com tão boa qualidade.

Por conta de não termos pensado muito na estrutura do código, passamos por muitas dificuldades. Não conseguimos estruturar de uma forma que parecesse natural, e tivemos que refatorar o código durante o desenvolvimento dele. Isso combinado com o prazo e a necessidade de fazer o programa funcionar, causou algumas partes do código estarem feitas para funcionarem. Gostaríamos de organizar melhor algumas variáveis que poderiam ser substituídas por objetos, para permitir maior controle e modularização. A escolha de nomes para elas também é algo crucial que faltou em alguns momentos. Apesar disso ainda assim o programa tem uma boa organização, e com a ajuda de um diagrama UML podemos ter uma boa visualização do código em geral.

DIAGRAMA DE CLASSES FINAL



DOCUMENTAÇÃO

A documentação do código foi feita usando o programa *sphinx* e pode ser encontrada no endereço:

http://www.ime.usp.br/~bsesso/PokemonBattle