Universidade Federal do Rio Grande do Sul
Departamento de Informática Aplicada
INF01121 - Modelos de Linguagens de Programação
Orientador: Prof. Dr. Leandro Krug Wives

Elyphoot
Trabalho Final - Relatório

Germano de Mello Andersson – 137719.
Marius Fontes – 172308.
Raphael Lopes Baldi – 143756.

Porto Alegre, 21 de Junho de 2012.

Sumário

Introdução

Elifoot é um jogo de futebol para PC da década de 90, estilo manager, que fez muito sucesso entre os brasileiros. Decidimos implementá-lo pois os componentes do grupo pertencem ao grupo de jovens que usufruíram de muitas horas diante deste clássico.

O Elifoot

O Elifoot simula o campeonato brasileiro, dividido em 4 divisões com 8 equipes. O fluxo do jogo é o seguinte: o usuário assume aleatoriamente um time da 4ª divisão, organiza a escalação e a formação de seu time e acompanha os jogos do seu campeonato. Cada campeonato possui rodadas de turno e returno, onde todos times de uma divisão jogam entre si. Ao final de uma temporada, os 2 primeiros colocados sobem para a divisão seguinte, enquanto que os 2 últimos são rebaixados. Uma nova temporada é então iniciada.

Requisitos do Projeto

Funcionais

- Sistema multiusuário.
- Gerenciar 4 campeonatos, separados em divisões, com rebaixamento.
- Fornecer um time da série D para o usuário gerenciar, aleatoriamente.
- Exibir tabela de classificação dos 4 campeonatos, rodada a rodada.
- Permitir a gerência da escalação do seu time, alterando formação e jogadores escalados para próxima partida.
- Exibir, minuto a minuto, o placar parcial de todos jogos de uma rodada.
- Permitir que o usuário interrompa a rodada e retome-a posteriormente, seguindo de onde abandonou o jogo.

Não funcionais

- Utilização de princípios de programação Orientada a Objetos.
- Utilização de princípios de programação Funcional.

Tecnologias

Linguagens

Optamos pela linguagem Python, versão 2.6. O motivo da escolha entre as disponíveis foi o interesse dos integrantes do grupo em desenvolver conhecimento sobre esta linguagem.

Para maior riqueza na interface disponibilizada via browser, utilizamos a linguagem JavaScript com manipulação de dados via JSON (AJAX).

Frameworks

Utilizamos o framework Django, versão 1.4. Ele é um framework web para Python que utiliza o modelo de desenvolvimento MVC. É um dos frameworks com maior destaque no principal motor de busca da internet.

Bibliotecas

- jQuery, versão 1.7.2.
- jQuery Effects.

Banco de Dados

- SQLite, versão 3.

Ferramentas

Para o ambiente de desenvolvimento, optamos por cada integrante utilizar a sua ferramenta preferida. As IDEs utilizadas foram: Eclipse, Vim e TextWrangler.

Como repositório de código utilizamos o Subversion, através do serviço gratuito oferecido pelo Google. A página do nosso projeto está disponível em http://code.google.com/p/elyphoot/.

Avaliação da Linguagem Python 2

Python é uma linguagem interpretada, desenvolvida com o intuito de trazer a experiência de escrita de um código mais limpo e simples, inclusive em detrimento de outras características, como performance.

Sistema de Tipos

Quanto ao sistema de tipos, o Python é uma linguagem de tipagem dinâmica forte. Isso significa que, durante a execução, uma variável possui um único tipo, é permitido, porém, que esta expressão altere seu tipo, implicitamente. O grande benefício desta característica é permitir implementações genéricas e alto nível de polimorfismo.

Análise das Características e Critérios da Linguagem

Característica	Nota	Justificativa
Legibilidade	9	Utiliza indentação como delimitador de bloco. Polimorfismo universal para diversas operações básicas. Não permite sobrecarga dos operadores básicos.
Redigibilidade	9	Sintaxe intuitiva. Polimorfismo universal permite maior abstração para construção dos TADs. Alto grau de expressividade em comandos de iteração.
Confiabilidade	7	Fortemente tipado. Possui STE. As palavras chave da linguagem são reservadas (o interpretador gera um alerta de sobrescrita caso você tente utilizá-las). Tipagem implícita e dinâmica pode levar o programador ao erro mais facilmente.
Simplicidade	10	Utiliza indentação como delimitador de bloco.
Ortogonalidade	7	Polimorfismo universal para diversas operações básicas. Não permite sobrecarga dos operadores básicos.
Portabilidade	9	Por ser linguagem interpretada, o porte é dependente das bibliotecas utilizadas.
Expressividade	9	Alto grau de expressividade, especialmente em comandos de iteração.
Reusabilidade	9	Polimorfismo universal; composição. Como Python permite múltipla herança, o efeito de interface em Java pode ser obtido via classes abstratas.
Estruturas de Controle	9	Possui pacote de maioria dos tipos de dado modelados para serem manipulados por iteradores. Fornece ao programador ferramentas para iterar sobre seus TADs.
Tipos de Dados	9	Básicos, listas, dicionários, coleções, tuplas de dados.
Estruturas de Dados	9	Classes.

Tratamento de Exceções	9	Possui. É possível estender a classe de exceções, permitindo ao programador criar suas próprias exceções.
Restrições de Aliasing	7	Atribuição de variável para variável é por referência. O interpretador Python é escrito em C e compilado usando *gcc* com a opção *-fno-strict-aliasing*, que evita conflitos na utilização de alias com otimizações realizadas pelo compilador.
Checagem de Tipos	10	Tipagem forte e dinâmica.

Tabela 1: Avaliação da Linguagem

Implementação

Modelagem

Este é o diagrama de classes, simplificado, de nossa aplicação:

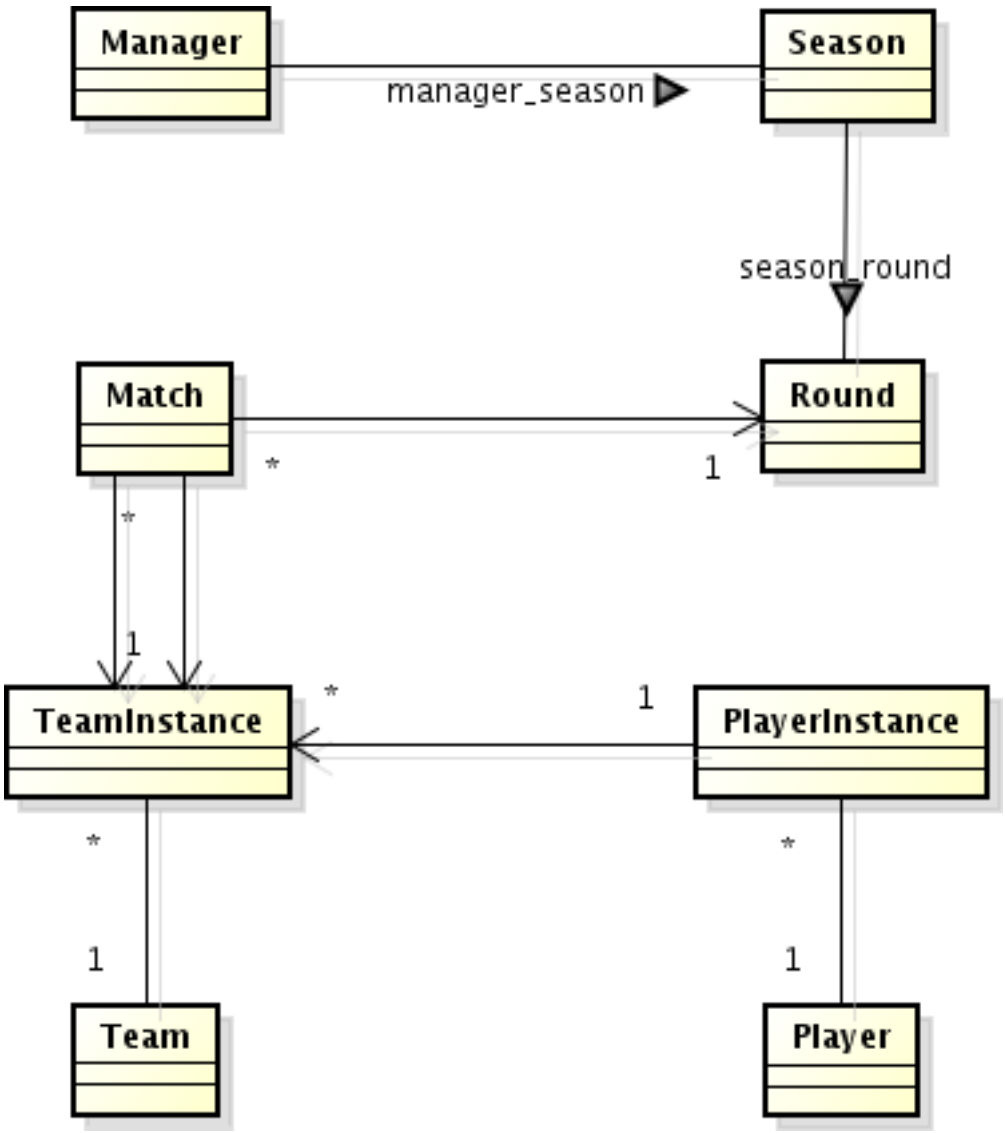

Figura 2: Diagrama de Classes

Manager

Esta classe representa o usuário do jogo, que exerce papel de treinador de uma equipe. Mantivemos o isolamento dos dados do usuário, necessários para o jogo, daqueles necessários para controle de sessão e autenticação.

Season

Esta classe representa uma temporada, ou seja, um campeonato das 4 divisões do início ao fim. Um manager está vinculado a uma temporada. O time que o manager gerencia é mapeado através desta representação. Uma temporada possui a menor quantidade de rounds possível, respeitando o fato de que todos times devem jogar contra todos em turno e returno, e que todos os times devem jogar em todas rodadas.

Round

Esta classe representa uma rodada dos 4 campeonatos. Uma rodada possui partidas entre todas equipes de todas divisões.

Match

Esta classe representa uma partida entre dois times de uma mesma divisão. Possui dos times associados e as estatísticas do jogo.

TeamInstance, Team, PlayerInstance, Player

Estas classes representam os times e os jogadores. Um time possui diversos jogadores, enquanto que um jogador pode estar associado a apenas um time.

Inicialmente, havíamos optado por criar uma tabela de relacionamento entre matches-players-managers, porém tal configuração atrapalharia a implementação da funcionalidade multiusuário. Para solucionar esta questão, decidimos modelar times e jogadores em duas classes cada: uma base, representando os dados originais e uma instância (representadas pelas classes 'Instance'), representando os dados alterados para uma sessão específica de um usuário.

Os dados iniciais populados foram importados do jogo Elifoot original, com algumas personalizações. Tal procedimento de importação também está presente no código, no pacote gameapp.database.

Pacotes e Estrutura

Quanto a sua estrutura física, nosso software foi organizado da seguinte maneira, de acordo com as características do framework escolhido:

- Projeto ‘elyphoot’.
    - Aplicação ‘gameapp’.
        - Pacote database.
        - Pacote manager.

        - Pacote match.
        - Pacote round.
        - Pacote season.
        - Modelos.
        - Visões.
        - Mapeamentos (urls).
- Database.
- Documentação.

O framework Django, apesar de permitir que os modelos sejam separados em pacotes, não permite que isso seja feita de forma simples e elegante, por isso optamos por manter todos modelos na raiz da aplicação, em um único arquivo models.py, e utilizamos os pacotes para armazenar o código de controle referente a cada modelo. O código relacionado as visões também foi mantido em um único arquivo, uma vez que ele foi simplificado (concentrado nos controladores) e o framework trabalha melhor desta forma.

Cabe ressaltar que as visões, no framework, são divididas em duas partes: uma responsável por tratar as requisições do usuário e outra por tratar a exibição (representada por modelos escritos em HTML, contendo marcações e estruturas de exibição de dados). Isso permite grande flexibilidade no desenvolvimento da aplicação.

Detalhes de Implementação

Algumas técnicas apresentadas nos laboratórios da nossa disciplina foram utilizadas em nossa implementação.

Utilizamos função como elemento de 1a ordem para marcação de passos do tempo das rodadas através da função nativa setTimeout do JavaScript:

```
setTimeout(runRoundStep, 1000);
```

Utilizamos funções de ordem maior para carregar a execução das partidas de uma rodada através da função nativa map do Python:

```
results = map(gameapp.match.controller.run_match, game_round.matches.all())
```

Para analisar se algum gol havia acontecido e poder gerar a devida alteração na interface utilizamos a função de ordem maior jQuery.each:

```
jQuery.each(matches, function(i, value) {
    ...
```

Conclusões

Foi uma ótima experiência alinhar aplicação de técnicas de programação com a modelagem de um jogo que tanto nos encantou, há mais de uma década, do ponto de vista do usuário. Conseguimos colocar em prática conceitos apresentados durante o semestre em uma linguagem de programação que não era de nosso domínio, mas que se mostrou poderosa e de uma simplicidade singular. Aproveitamos o desafio e optamos por trabalhar com um framework web que exigiu esforço inicial para conhecê-lo, recompensado pelas facilidades que ele trouxe posteriormente, especialmente quanto a sua API para mapeamento objeto-relacional.

Um marco em nosso projeto foi quando nos deparamos com o desafio de implementarmos o requisito multiusuário. Descobrimos que nosso modelo original dificultaria bastante tal implementação. Fizemos algumas reuniões de alinhamento e optamos por manter o requisito, sendo necessário algumas alterações no modelo do projeto para tal.

Também pensamos, no meio do projeto, em mudar a linguagem e bibliotecas utilizadas, pois a curva de aprendizado do framework escolhido estava se mostrando bastante íngreme. Após realivação decidimos continuar com as nossas primeiras decisões e implementar o jogo em Python utilizando o framework Django.

Bibliografia

ELIAS, A. **Elifoot - Site Oficial**, 2012. Disponível em: <http://www.elifoot.net/>. Acesso em: 17 jun. 2012.

GOOGLE INC. Google's Python Class. **Google Code University**. Disponível em: <http://code.google.com/intl/pt-BR/edu/languages/google-python-class/>. Acesso em: 22 maio 2012.

HOLOVATY, A.; KAPLAN-MOSS, J. The Django Book: Version 2.0. **The Django Book**, 16 mar. 2009. Disponível em: <http://www.djangobook.com/en/2.0/>. Acesso em: 01 jun. 2012.

LARMAN, C. **Applying UML and Patterns:** An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3rd Edition. ed. New Jersey: Pearson Education, Inc., 2004.

PYTHON SOFTWARE FOUNDATION. Python v2.6.8 documentation. **Python Docs**, 12 abr. 2012. Disponível em: <http://docs.python.org/release/2.6.8/>. Acesso em: 29 maio 2012.

SEBESTA, R. **Conceitos de Linguagens de Programação**. Tradução de Eduardo Kessler Piveta. Nona Edição. ed. Porto Alegre: Bookman Companhia Editora LTDA, 2010.

WIKIPEDIA. Elifoot. **Wikipedia**, 20 abr. 2012. Disponível em: <http://pt.wikipedia.org/wiki/Elifoot>. Acesso em: 15 jun. 2012.