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 – XXX
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 sí. 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 multi-usuário.
- 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
- jQuery Effects

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

Característica	Nota	Justificativa
Legibilidade		
Redigibilidade		
Confiabilidade		
Simplicidade		
Ortogonalidade		
Portabilidade		
Expressividade		
Reusabilidade		
Estruturas de Controle		
Tipos de Dados		
Estruturas de Dados		
Tratamento de Exceções		
Restrições de Aliasing		
Checagem de Tipos		

Implementação

Modelagem

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

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 multi-usuário. Para solucionar esta questão, decidimos modelar times e jogadores em duas classes cada: uma base, representando os dados originais e uma '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 templates html, contendo marcações e estruturas de exibição de dados). Isso permite grande flexibilidade no desenvolvimento da aplicação.

Composição, herança
Classes, atributos e métodos;
Visão
Nosso software possui
Funções como elemento de 1a ordem

Currying (ou funções não nomeadas, ou lambda, com mais de um parâmetro)
Casamento de padrões

Controle

Para fins de controle, modularizamos nosso software em 5 packages: database
Encapsulamentos (proteção, separação de interface da implementação)
Funções de ordem maior (map, reduce, foldr/foldl...)
Python 2 - Análise Critica
Análise crítica da linguagem, envolvendo uma tabela com os critérios e propriedades estudados em aula, com notas/valores justificados.

Conclusões

conclusoes e dificuldades encontradas

Bibliografia

na norma abnt2