Programação Orientada aos Objetos

Projeto Prático – FitnessUM

Braga, 07 de Junho de 2014

|  |  |  |  |
| --- | --- | --- | --- |
| Macintosh HD:Users:andreramos:Dropbox:Trabalho SO:Pedro.jpg |  | C:\Users\Pedro\Dropbox\Projeto POO 14-15\PVF\diogofoto.jpg | C:\Users\Pedro\Dropbox\Projeto SO 14-15\Universidade do Minho - Portal Académico.jpeg |
| Pedro Vieira Fortes  Nº 64309 |  | Diogo Ramos Constâncio  Nº 70719 | João Pedro Pereira Fontes  Nº71184 |



Escola de Engenharia

Departamento de Informática

Licenciatura em Engenharia Informática

Índice

[Introdução 2](#_Toc421371314)

[Objetivos 2](#_Toc421371315)

[Organização do Relatório 2](#_Toc421371316)

[Arquitetura e Estruturação de Classes 4](#_Toc421371317)

[Estruturas de Dados 6](#_Toc421371318)

[Login 6](#_Toc421371319)

[User 6](#_Toc421371320)

[CacheStorage 6](#_Toc421371321)

[Cache 6](#_Toc421371322)

[Funcionamento da Aplicação 8](#_Toc421371323)

[Conclusão e Análise Crítica 9](#_Toc421371324)

# Introdução

O presente projeto enquadra-se na unidade curricular de Programação Orientada a Objetos do curso de Licenciatura em Engenharia Informática da Universidade do Minho. O projeto pretende implementar uma aplicação, designada GeocachingPOO, para registar e simular atividades de geocaching. A aplicação foi desenvolvida na linguagem java e pretende simular um ambiente de rede social volta ao geocaching.

## Objetivos

O presente projeto tem como objetivo principal consolidar e colocar em prática os conceitos fundamentais transmitidos durante a UC de Programação Orientada a Objetos. De acordo com o enunciado, os principais objetivos defenidos para a aplicação GeocachingPOO são os seguintes:

Requisitos básicos

* Aceder à aplicação utilizando as credenciais (email e password);
* Visualizar a informação das últimas 10 atividades, do próprio e dos amigos, com possibilidade de aceder aos seus detalhes;
* Registar a criação de uma cache nova (de um dos vários tipos que for considerado);
* Registar a informação de uma atividade de descoberta de uma cache; Consultar, por ordem cronológica, e remover actividades;
* Fazer "report abuse" de uma cache
* Consultar e remover atividades de descoberta. A consulta deverá obedecer a uma ordem cronológica, num formato conhecido de "timeline" utilizado pelas redes sociais;
* Aceder às estatísticas (mensais e anuais).

Requisitos de valorização

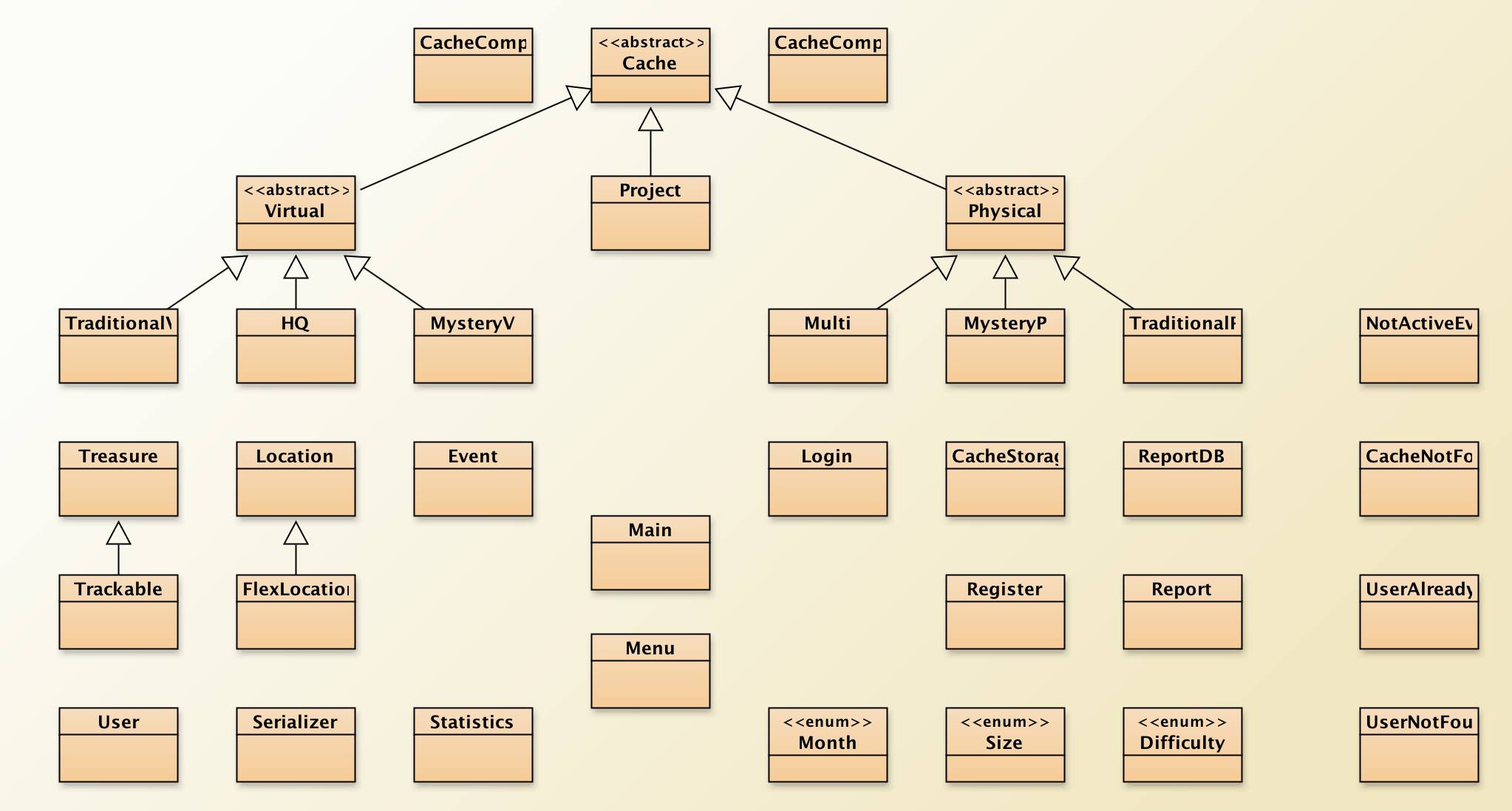
* Definir eventos e proceder à sua simulação a partir dos administradores da aplicação.

## Organização do Relatório

O presente relatório serve de suporte à aplicação desenvolvida, apresentando uma descrição geral dos aspetos mais importantes. Com exceção do presente capítulo introdutório, o relatório é composto por dois capítulos iniciais e nucleares – Arquitetura e Estruturação de Classes e Estruturas de Dados – e por outros 2 capítulos complementares relacionados com funcionalidades particulares da aplicação e estratégias usadas no seu desenvolvimento. No final são tecidas algumas conclusões sobre o projeto.

# Arquitetura e Estruturação de Classes

No IDE BlueJ, é formado um diagrama que representa as ligações e comunicações entre as diversas classes. O nosso resultado final foi o seguinte: **(mudar imagem!!!)**



A classe Main é responsável simplesmente pelo arranque do programa chamando o menu de início. Quando o programa é encerrado, este chama o método responsável pela gravação de estado do programa.

A classe Menu é responsável pela comunicação com o utilizador, criando os diferentes tipos de menus usados na aplicação e realizando algumas das tarefas com o auxílio das outras classes. Também é realizado o tratamento de exceções e guarda e carrega o estado do programa.

A classe CacheStorage contém toda a informação de caches registadas na aplicação.

A classe User foi desenvolvida com o intuito de armazenar toda a informação relativa ao mesmo tal como o email do utilizador (String), a sua password (String encriptada), o seu nome, género e morada (Strings), a sua data de nascimento (Calendar), as atividades que realizou (TreeSet<Cache>), as caches que criou (TreeSet<Cache>), a sua lista de amigos (ArrayList<String>) e a identificação se é administrador ou não (boolean).

A classe Statistics é responsável por calcular as estatísticas do utilizador no momento, ao longo do mês ou do ano, para que este possa visualizar a sua performance ao longo do tempo.

A classe Cache guarda toda a informação de uma cache que qualquer utilizador faça ou crie. É guardada nela o nome atribuído à cache, o código, a descrição e as dicas (String), o registo de utilizadores que fizeram a cache, o dia em que a cache foi realizada ou criada (Calendar), a localização da cache (Location) e a dificuldade (Difficulty). Esta classe é abstrata, sendo então super classe de várias outras classes, que neste caso são os diferentes tipos de cache. Neste caso, no 2º nível de herança temos as Classes Físicas e Virtuais que herdam tudo da Cache e no 3º nível de herança temos os tipos de cache que herdam tudo o que corresponder ao contexto em que foi criada (Física ou Virtual).

A classe Evento é responsável por criar um evento em que utilizadores se podem inscrever e também realiza uma simulação para ver quem ganharia dos utilizadores inscritos.

A classe Login é responsável por autenticar os utilizadores e guardar os salts resultantes da hash da password do mesmo. Também toda a informação de utilizadores registados na aplicação é gerida por esta classe.

A classe report é responsável pelo "report abuse" de caches.

# Estruturas de Dados

## Login

* HashMap<String,User> users;

Os utilizadores são guardados num HashMap onde a chave é o email do utilizador. Achamos esta a melhor estrutura para guardar os utilizadores porque eles podem ser identificados pelo seu email visto que este é único para cada um. Também com HashMap é fácil obter o utilizador através do email e como não é necessário estarem ordenados o HashMap é melhor do que um TreeMap.

## User

* private TreeSet<Cache> activities;
* private TreeSet<Cache> createdCaches;
* private ArrayList<User> friends;

As atividades e as caches criadas são armazenadas num TreeSet para que estejam ordenados. Já os amigos são armazenados num ArrayList, pois não é necessário eles estarem ordenados e pensamos que eles podem ser simplesmente armazenados numa lista de users.

### CacheStorage

* TreeMap<String,Cache> db

As caches criadas na aplicação são guardadas num TreeMap que tem como chave o código da cache que é único para cada um.

## Cache

* private TreeMap<String,Register> regBook;

As pessoas que realizam as caches são guardads num TreeMap de registos sendo a chave o seu email e o value a classe Register. Usamos esta estrutura para conseguirmos organizar os utilizadores mais facilmente e conseguir aceder a eles com facilidade se necessário.

**Estratégias Adotadas**

Da maneira como este projeto foi construído é possível acrescentar mais atividades apenas criando novas classes.

**(Não tou com ideias aqui)**

# Funcionamento da Aplicação

**(Aqui serão colocadas os menus da aplicação e como funcionam)**

# Conclusão e Análise Crítica

O presente relatório descreveu, de forma não exaustiva, as principais componentes da aplicação desenvolvida para a avaliação do projeto GeocachingPOO da unidade curricular de POO. Considera-se que os objetivos básicos terão sido todos atingidos, tendo-se ainda conseguido responder a alguns dos objetivos complementares, de que é exemplo a simulação de eventos. O trabalho foi desenvolvido num verdadeiro ambiente de trabalho de equipa, através de uma discussão permanente dos aspetos essenciais para a conceção e de um rigoroso e robusto controlo de versões.