4
LearnAds: um *Framework* de Recomendação de Anúncios

O **LearnAds** é um *framework* de recomendação de anúncios baseado em Aprendizado de Máquina que está sendo desenvolvido[5] no LEARN[6]. Iniciado no segundo semestre de 2007, o LearnAds se encontra em melhoria contínua através de refatorações.

O objetivo do *framework* é oferecer suporte ao emprego de técnicas de Aprendizado de Máquina na publicidade direcionada, mais especificamente tratando-a como um problema de **recomendação de anúncios**. Quando falamos em recomendação, estamos nos referindo ao procedimento de seleção (em inglês, *matching*) e ordenação (em inglês, *ranking*) de anúncios, dada uma consulta – palavra-chave submetida à máquina de busca – ou uma página de conteúdo.

A seguir, apresentamos o projeto e especificação do LearnAds. Na seção 4.1, descrevemos os requisitos funcionais do *framework*; na seção 4.2, os casos de uso que atendem a tais requisitos; na seção 4.3, os diagramas de seqüência correspondentes a esses casos de uso; e na seção 4.4, os diagramas de classes das camadas de modelo e de controle do *framework*.

4.1.
Requisitos Funcionais

Os requisitos funcionais do *framework* são os seguintes:

- Migrar dados de *query logs*, que obedeçam a um formato especificado pelo *framework*, para um banco de dados;
- Oferecer ferramentas para a avaliação e seleção das estratégias mais apropriadas para a recomendação de anúncios; e

[5] A equipe de desenvolvimento do LearnAds é composta por Ruy Luiz Milidiú, Roberto Cavalcante, Diogo Mendonça, Tulio Anibolete e Leonardo Majowka.

[6] Laboratório de Engenharia de Algoritmos e Redes Neurais, PUC-Rio.

- Recomendar anúncios baseados em uma estratégia previamente selecionada.

Este trabalho aborda apenas o segundo e o terceiro requisitos funcionais do *framework*, pois o primeiro requisito encontra-se fora de seu escopo.

4.2. Casos de Uso

A partir dos requisitos funcionais, definem-se os seguintes casos de uso:

- UC01 – Migração de dados (fora do escopo, conforme seção 4.1);
- UC02 – Avaliação de estratégias de recomendação de anúncios;
- UC03 – Geração de um modelo baseado em uma estratégia definida; e
- UC04 – Recomendação de anúncios baseada num modelo existente.

Os casos de uso UC02, UC03 e UC04 são descritos respectivamente no Quadro 2, Quadro 3 e Quadro 4.

Quadro 2: UC02 – Avaliação de estratégias de recomendação de anúncios.

Ator Primário: Programador
Pré-Condições: banco de dados preenchido com dados de *query logs*
Fluxo Normal:
1) Programador seleciona parâmetros da validação cruzada.
2) Programador seleciona a métrica de avaliação.
3) Programador seleciona algoritmo e/ou parâmetros que representam a estratégia a ser testada.
4) *Framework* retorna o resultado da avaliação.

Quadro 3: UC03 – Geração de um modelo baseado em uma estratégia definida.

Ator Primário: Programador
Pré-Condições: banco de dados preenchido com dados de *query logs*
Fluxo Normal:
1) Programador seleciona algoritmo e/ou parâmetros que representam a estratégia desejada para a geração do modelo preditivo.
2) *Framework* armazena o modelo gerado num arquivo.

Quadro 4: UC04 – Recomendação de anúncios baseada num modelo existente.

Ator Primário: Usuário Final Pré-Condições: modelo preditivo de recomendação previamente gerado. Fluxo Normal: 1) Usuário seleciona o modelo pretendido. 2) Usuário entra com a consulta ou a página de conteúdo. 3) *Framework* retorna uma lista ordenada de anúncios relacionados com a consulta ou palavra-chave.

4.3. Diagramas de Seqüência

Os diagramas de seqüência referentes aos três casos de uso desenvolvidos neste trabalho, conforme descrito na seção 4.2, são apresentados na Figura 3, Figura 4 e Figura 5. Como exemplo de instanciação, é utilizado o algoritmo de Fatoração de Matrizes, apresentado na seção 3.2. O modelo abstrai detalhes físicos de implementação, como a leitura de arquivos de configuração, ou chamadas à rotina de gravação dos *logs*. Além disso, os objetos são criados imediatamente antes de receberem a primeira mensagem, pois as mensagens de criação foram omitidas para manter a simplicidade dos diagramas.

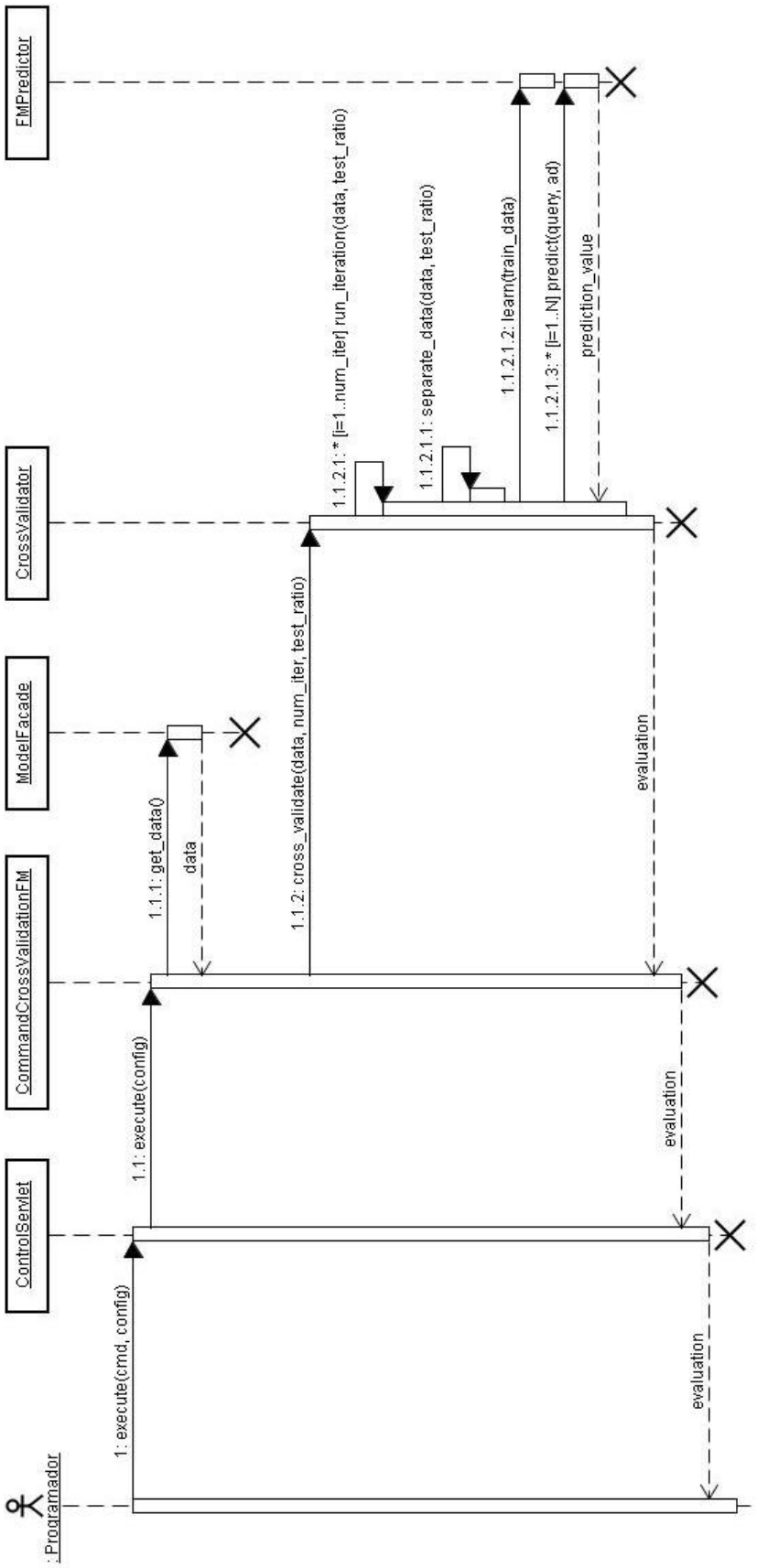

Figura 3: Diagrama de seqüências do caso de uso UC02 – Avaliação de estratégias de recomendação de anúncios.

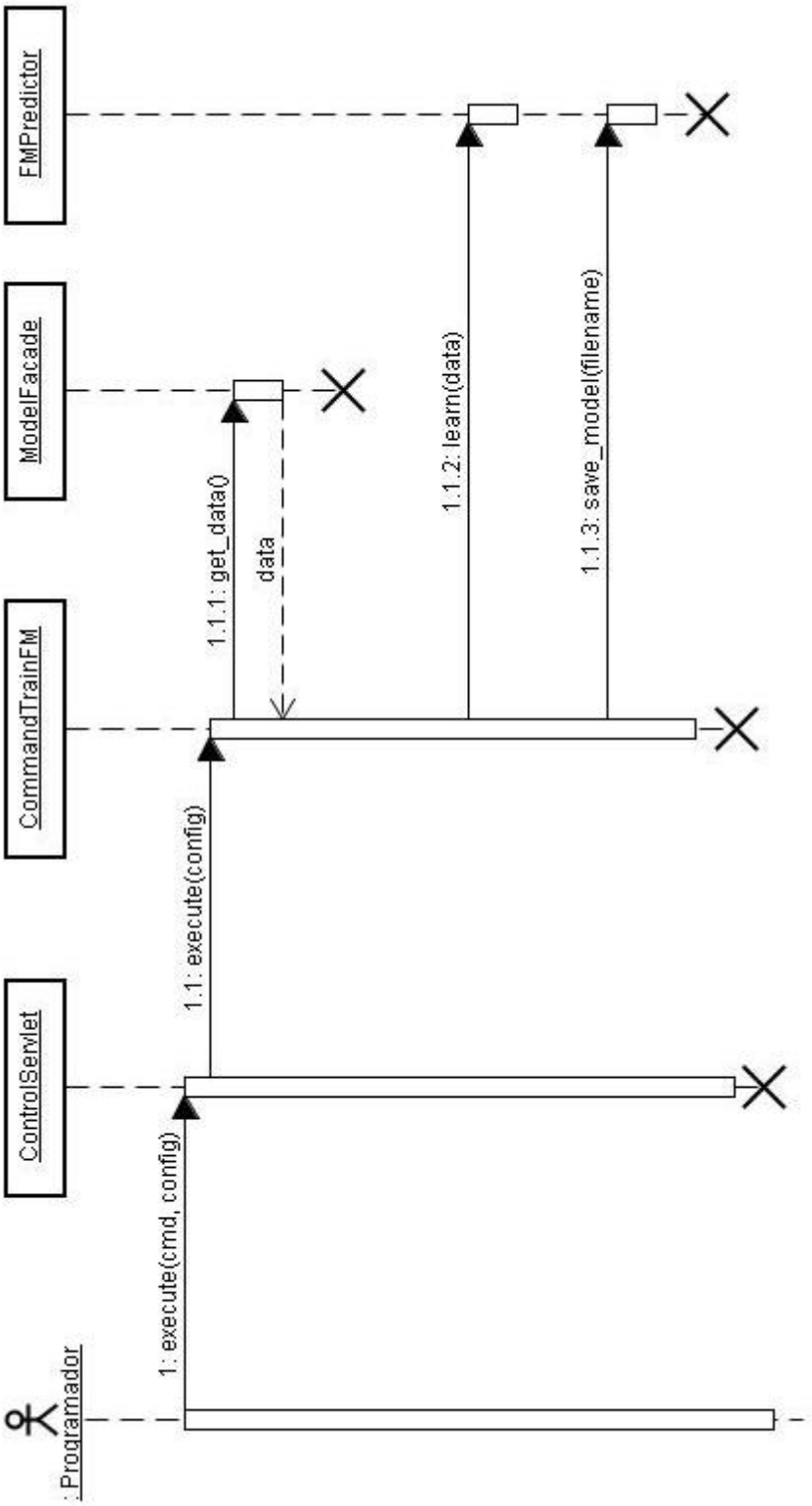

Figura 4: Diagrama de seqüências do caso de uso UC03 – Geração de um modelo baseado em uma estratégia definida.

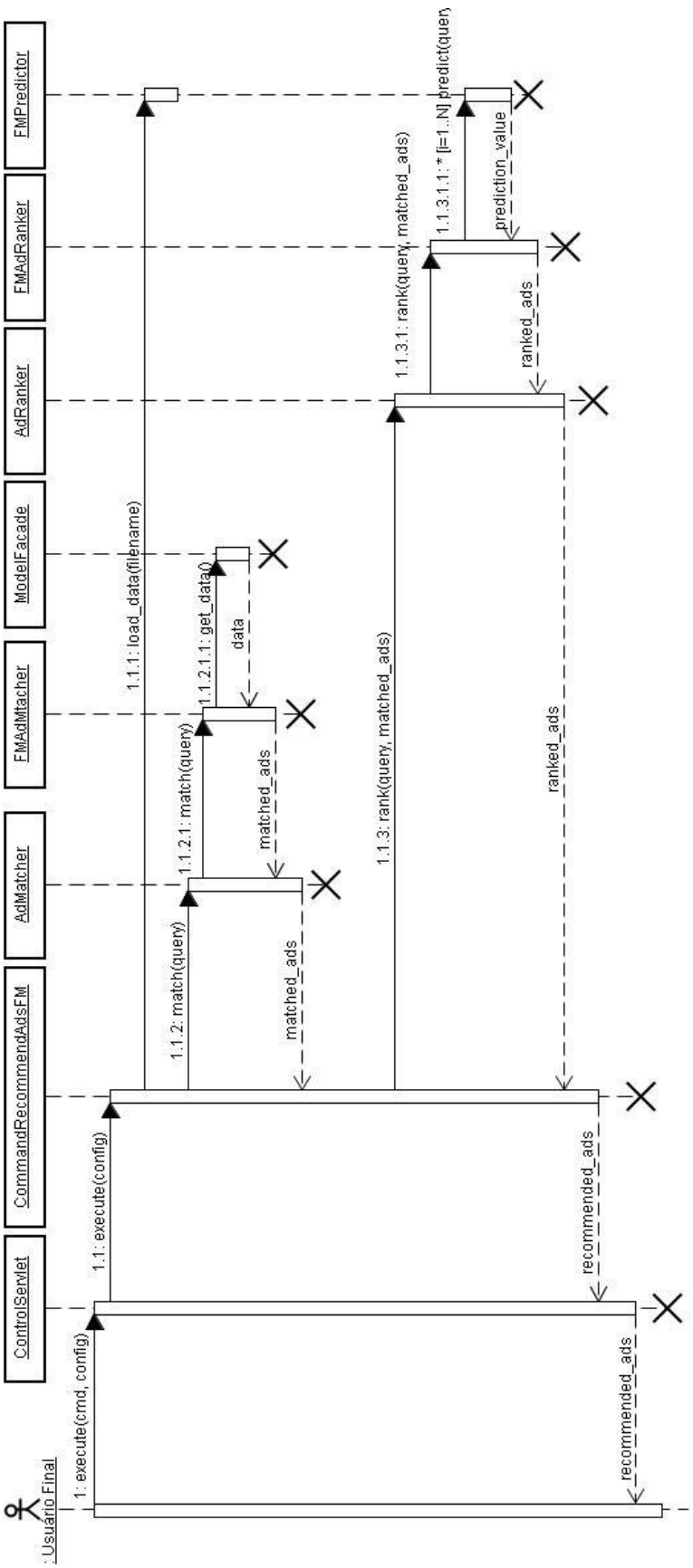

Figura 5: Diagrama de seqüências do caso de uso UC04 – Recomendação de anúncios baseada num modelo existente.

4.4. Diagramas de Classes

A seguir, apresentamos os diagramas de classes que descrevem o *framework*. Na seção 4.4.1, está representada sua a camada de modelo, enquanto na seção 4.4.2, está representada sua camada de controle.

4.4.1. Camada de Modelo

O diagrama de classes da Figura 6 representa objetos relacionados ao negócio e objetos auxiliares.

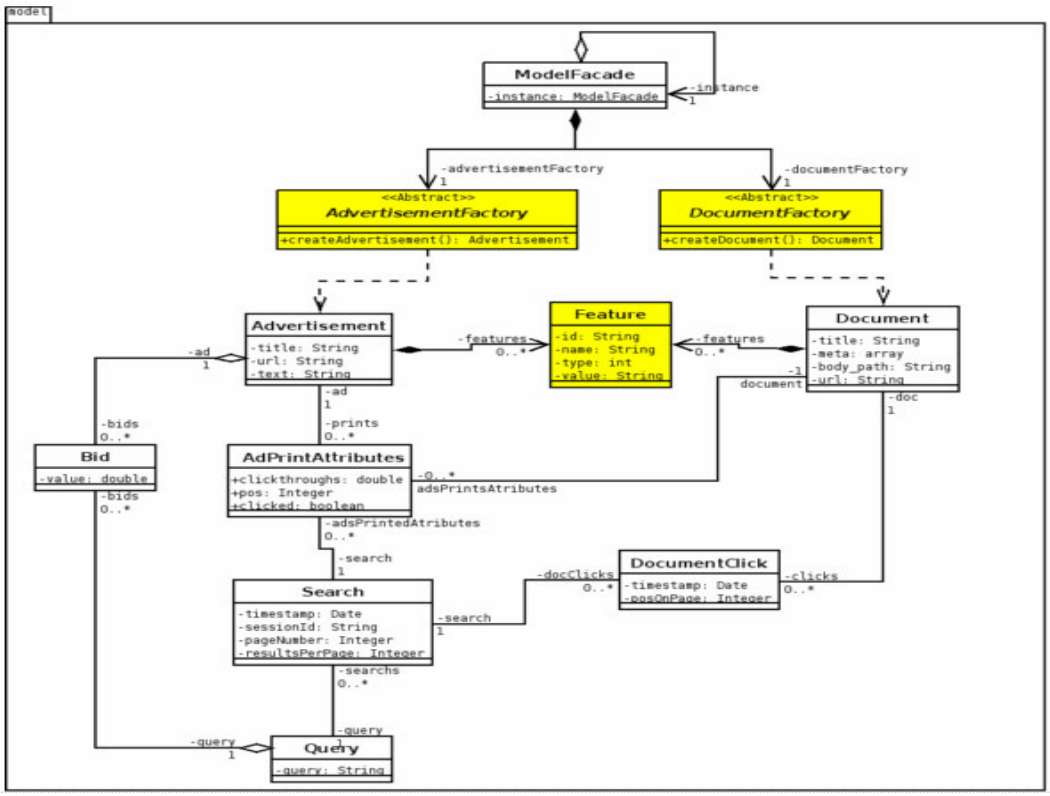

Figura 6: Diagrama de classes da camada de modelo.

A seguir é apresentada uma sucinta descrição de cada classe de negócio.

- *Query*: representa a consulta submetida à máquina de busca.
- *Advertisement*: representa um anúncio que pode ser exibido quando uma consulta é submetida à máquina de busca ou quando uma página de conteúdo é exibida. Esta classe é um *abstract product*, portanto, na instanciação, haverá um *concrete product* relacionado a ela.

- *Bid*: representa um lance que relaciona um anúncio e uma consulta, a partir da qual é gerado um valor que indica o grau de relevância do relacionamento. No ato da recomendação, o grau de relevância é avaliado para decidir quais e em que ordem os anúncios serão exibidos.
- *Search*: representa a submissão de uma consulta à máquina de busca. Uma seqüência contígua de buscas realizadas pelo mesmo usuário pode ser agrupada através do atributo *Query Session*.
- *Document*: representa os documentos que são retornados quando a consulta é submetida à máquina de busca. Esta classe é um *abstract product*.
- *DocumentClick*: representa os cliques efetuados nos documentos que são retornados pela máquina de busca após a submissão uma consulta.
- *AdPrintAttributes*: representa as características de exibição do anúncio.

A seguir é apresentada uma sucinta descrição de cada classe auxiliar.

- *AdvertisementFactory*: representa a *abstract factory* de *Advertisement*, ponto flexível do *framework*. Na instanciação, possuiremos uma *concrete factory* relacionada a essa classe.
- *DocumentFactory*: representa a *abstract factory* de *Document*, ponto flexível do *framework*.
- *Feature*: representa quaisquer propriedades relevantes para a recomendação de anúncio para um documento.
- *ModelFacade*: representa a implementação do padrão *Facade*. O objetivo é realizar a comunicação entre a Camada de Controle e a Camada de Modelo, o que reduz o acoplamento entre as classes dessas camadas.

4.4.2. Camada de Controle

O diagrama de classes da Figura 7 representa objetos relacionados ao tratamento de requisições e objetos que executam a recomendação.

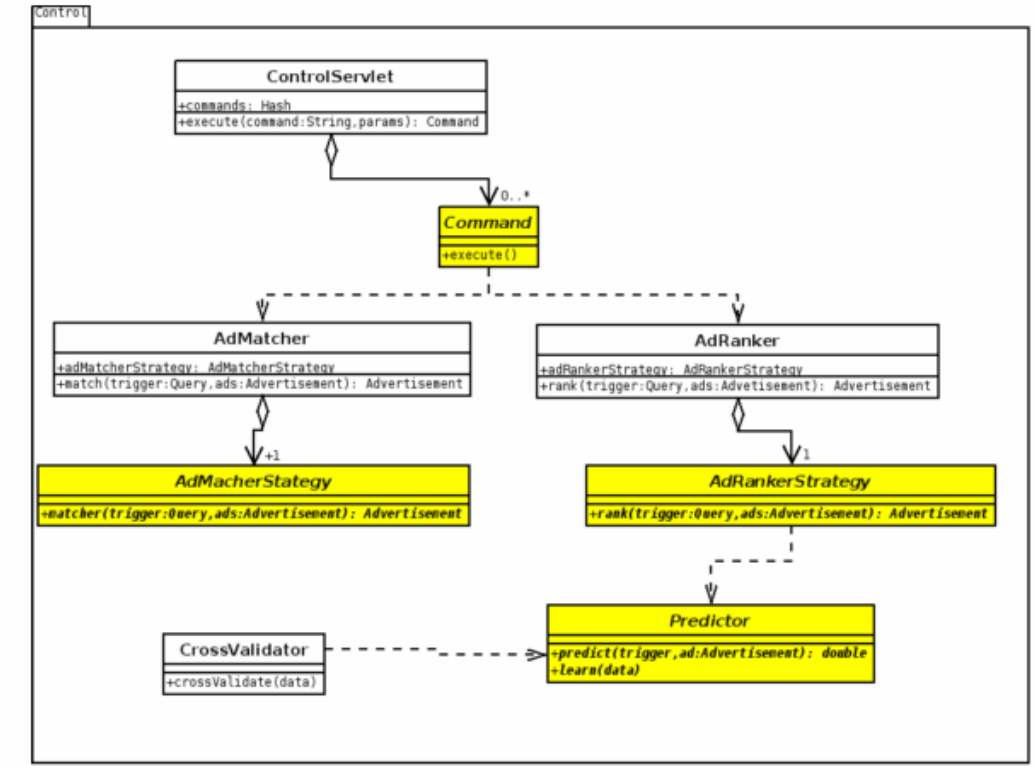

Figura 7: Diagrama de classes da camada de controle.

A seguir é apresentada uma sucinta descrição de cada classe de tratamento de requisições.

- *ControlServlet*: representa a etapa de recepção da requisição do usuário, ou seja, o comando a ser executado e seus parâmetros.
- *Command*: representa a implementação do padrão *command* para endereçamento dos parâmetros. Na instanciação, uma classe específica estende esta classe e trata os parâmetros da recomendação.

A seguir é apresentada uma sucinta descrição de cada classe de execução da recomendação.

- *AdMatcher* e *AdMatcherStrategy*: representam a implementação do padrão *strategy*, ponto flexível do *framework*. Na instanciação, haverá

uma classe concreta, associada a esta classe, responsável pela seleção dos anúncios apropriados.

- *AdRanker* e *AdRankerStrategy*: representam a implementação do padrão *strategy*, ponto flexível do *framework*. Na instanciação, haverá uma classe concreta, associada a esta classe, responsável pela ordenação do conjunto de anúncios selecionados.
- *Predictor*: preditor responsável por estimar um *quality score* que representa a relevância de um anúncio para uma determinada consulta. Na instanciação haverá uma classe concreta, associado a essa classe, que implementa um algoritmo de Aprendizado de Máquina.
- *CrossValidator*: realiza o procedimento de validação cruzada para determinar, a partir de uma métrica de avaliação, a qualidade de um preditor.