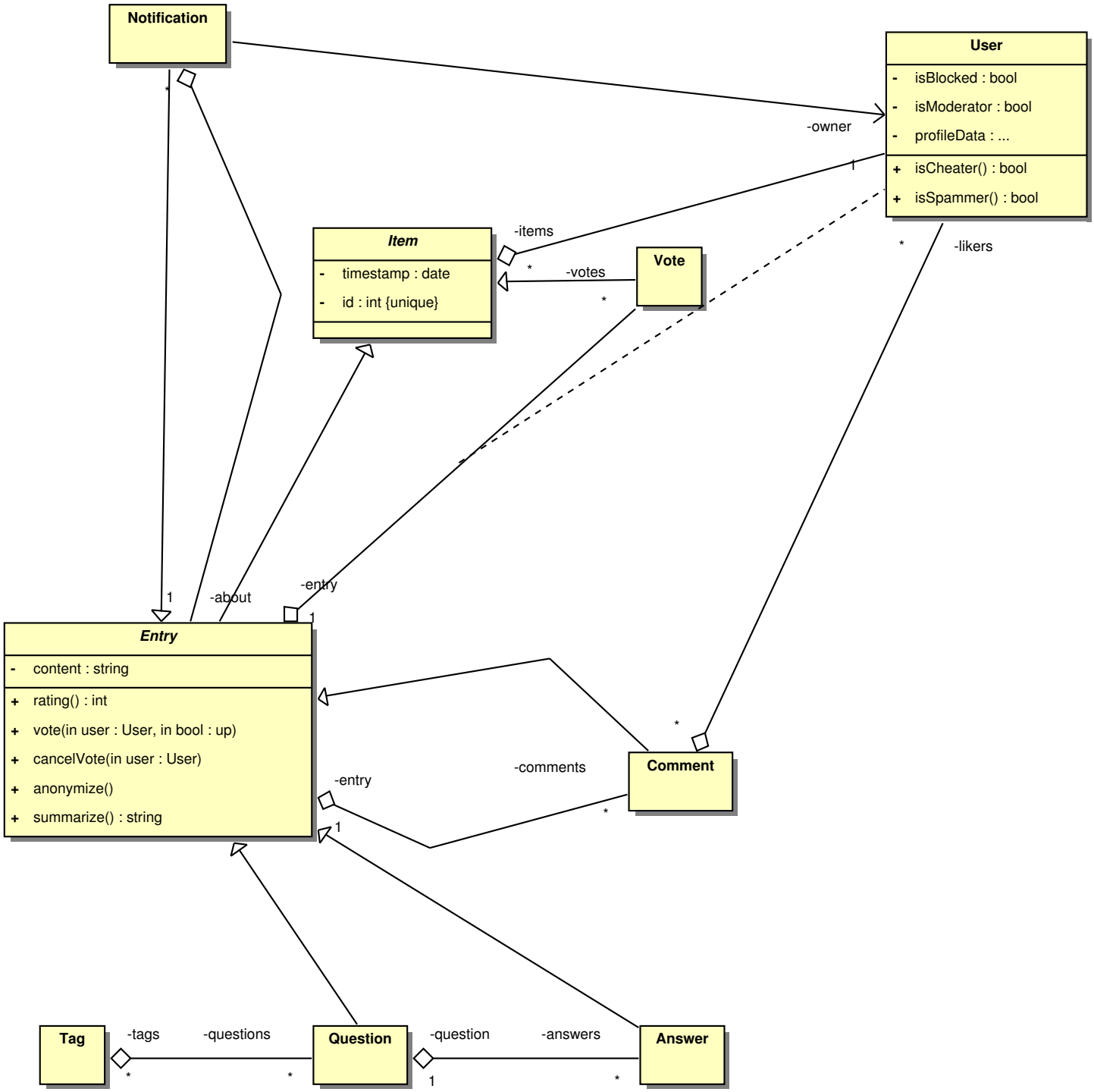
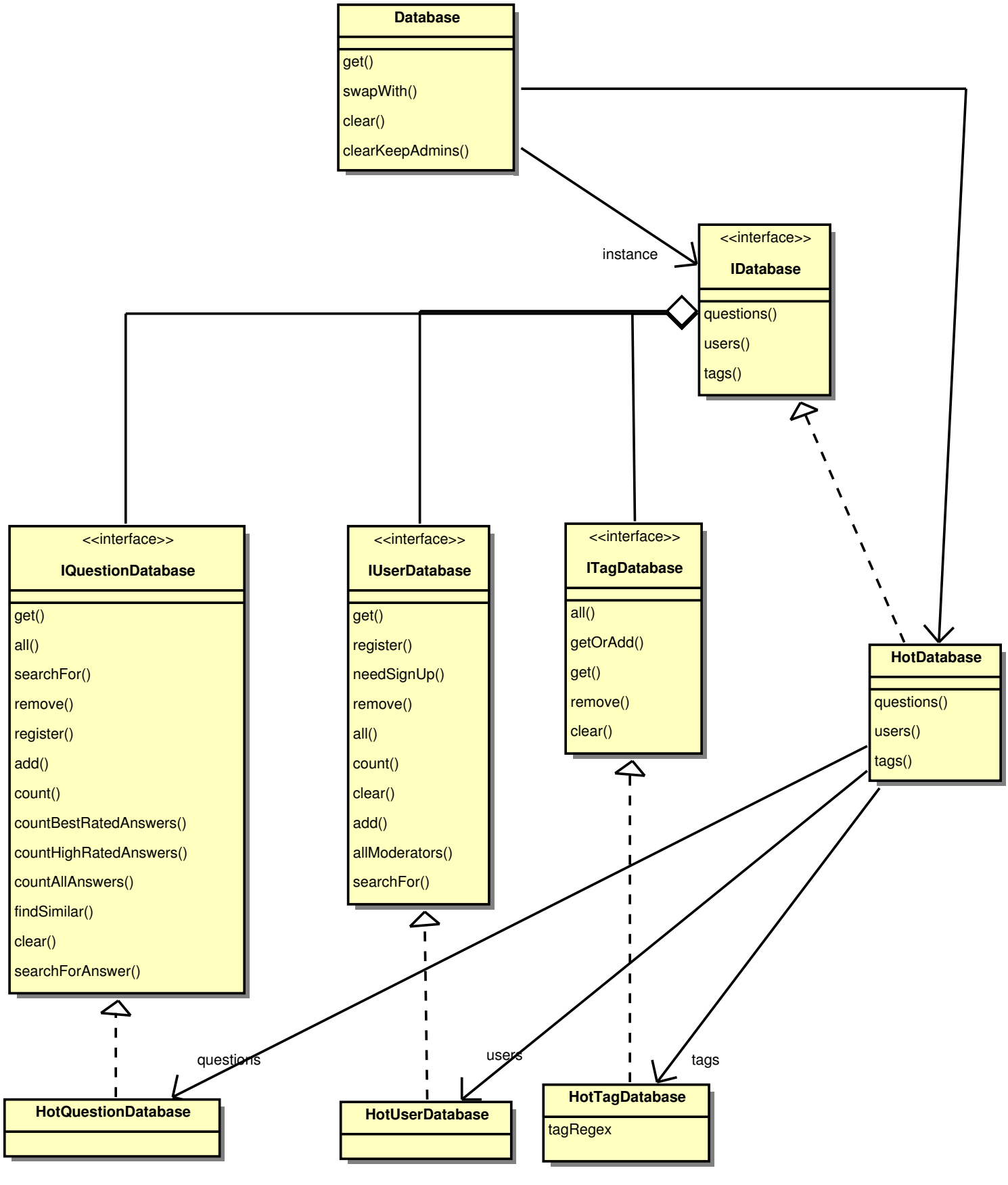

Notification
User
- isBlocked : bool
- isModerator : bool
- profileData : ...
+ isCheater() : bool
+ isSpammer() : bool
-owner
1
*
-likers
Item
- timestamp : date
- id : int {unique}
-items
-votes
Vote
*
1
-about
-entry
Entry
- content : string
+ rating() : int
+ vote(in user : User, in bool : up)
+ cancelVote(in user : User)
+ anonymize()
+ summarize() : string
Comment
-comments
-entry
Tag
-tags
-questions
Question
-question
-answers
Answer


Database
get()
swapWith()
clear()
clearKeepAdmins()
instance
<<interface>>
IDatabase
questions()
users()
tags()
<<interface>>
IQuestionDatabase
get()
all()
searchFor()
remove()
register()
add()
count()
countBestRatedAnswers()
countHighRatedAnswers()
countAllAnswers()
findSimilar()
clear()
searchForAnswer()
<<interface>>
IUserDatabase
get()
register()
needSignUp()
remove()
all()
count()
clear()
add()
allModerators()
searchFor()
<<interface>>
ITagDatabase
all()
getOrAdd()
get()
remove()
clear()
HotDatabase
questions()
users()
tags()
questions
users
tags
HotQuestionDatabase
HotUserDatabase
HotTagDatabase
tagRegex

Domain Model

The Item Hierarchy

The Item class signifies that an object is the content of the page.

Item

The common superclass ensures an unique id number for all objects and provide access to all instances over their primary key and are created by a user at some time. A creation timestamp is also available.

Entry

The entry provides the common functionality of all contents of the page - they can be rated (by voting it up or down), are created by an user, and can provide a short version of themselves in order to keep the interface clean.
All Entries can in principle be voted upon, but in practice, only Questions and Answers need this feature.
This is the base class for all entities in the application, that are conceptually text, eg Questions, Notifications, ...

Question

This is the central point of the application, but is a humble subclass of Entry. In addition to it, `Question`s can be tagged by the owner and answered by other users.

Answer

The Answer hierarchically belongs to a Question.

Comment

A Comment is in the hierarchy by importance below a Question or an Answer. Instead of being voted for, it can only be liked by users.

Notification

A notification is send to a user to inform them that something happened to an entry they are watching. It describes what happened and to what entry.

User

Represents the action, a user can perform if signed in and gives a unique representation for all of them.

Controllers

As there is little if any interaction between the controllers, there is no

gain in UML. The interaction with the model is very transparent.

Application

This controller handles all nonspecific, non-authentificated access to the site. Or in short: what isn't important enough to get an own controller, is here.

CAnswer/Question/User

The interaction with the corresponding models. Usually requires authentication for actions.

Session

Access to the session of the user to make it mockable.

Search

Handles differnet search requests by passing them to the DB.

Developement View

Playframework

The playframework sets the basis for the application and provides
- a webserver
- routing
- parsing for the views

Messages

The translations to all supported languages.

Routes

Links the requests to methods of the controllers.

Model

Package that is independend from Play and provides the modelling of the

Helpers

Package that provides functionality that doesn't quite fit into the responsibilities of the other classes.

Database

An interface for data management and maybe in the future persistancy.

SearchEngine

Filters the data and shows only the most useful by some standards.
problem domain.

Controller

Translates user actions into changes in the model.

Views

Shows the current state of the model.