Specs of the CADo Tool

1. Terminology

Sentence: One sentence in the API documentation. A sentence is the unit of analysis in the study. That is, each sentence will be tagged by a one tag.

Block: one API documentation block. That is, a text block that describe a particular API element.

Code: In our case this is the question that is answered by a quote in the API documentation. For example "What method sequence is allowed/not allowed on this element?". These sentences answer the question: "The methods in this interface can safely be called multiple times. The first invocation releases the connection, subsequent calls are ignored."

Element: A particular API element, such as the method "system.out.println()", the class "String" or the package "java.util".

Coder: the person who will read the sentences and tag them, i.e. assign the question that is answered by this sentence.

Sample: a selection (a subset) from all sentences and blocks of an API documentation, which will be studied by the coders.

2. Features

The app name is CADo: Content Analysis for API-Documentation. CADo has 2 modes: The admin mode and coder mode.

A. Admin Mode:

The following features are supported:

- Login (First and Lastname)
- Extract API Documentation (No need to have a UI for the first iteration)
- Select API
- Create Sample
    - Add a sample size
    - Select sampling scheme
- Select Sentences + Fetch
- Save sample
- Export sample (csv)
- Edit codes (i.e. add a bunch of code: [ID] codes [category]
- Add coder (First + Last name , email)
- Load codes (like excel)
- Export codes (csv)
- Add comment to code
- Calculate inter-coder's agreement

A. Coder's mode

- Login (First and Last name) The coder can then see all the sentences he needs to code (the sample).
- Select/Load sample
- Select sentence
  - Open link (real documentation)
  - Add code (select question)
- Save codes (A coder should be able to interrupt a coding session and store his codes sofar)
- Filter coded (yes/no)
- Submit codes (This means that the coding is done and checked. The status of the sample is updated)

3. Sentence Splitting and Quotes

- It seems that implementation of the sentence splitting is not correct: You split based on full stop and space. Many special cases are not considered (like "e.g. " or "i.e. " etc.). In order to avoid such mistakes, we should consider using standard natural language parsers.
- We should store the sentence-position in the block (i.e. the orders of the sentences).
- Package descriptions are missing (packages can have documentation as well).
- The total number of the retrieved sample is not equal to the entered number in the UI. (I suppose rounding mistakes).

4. Sampling

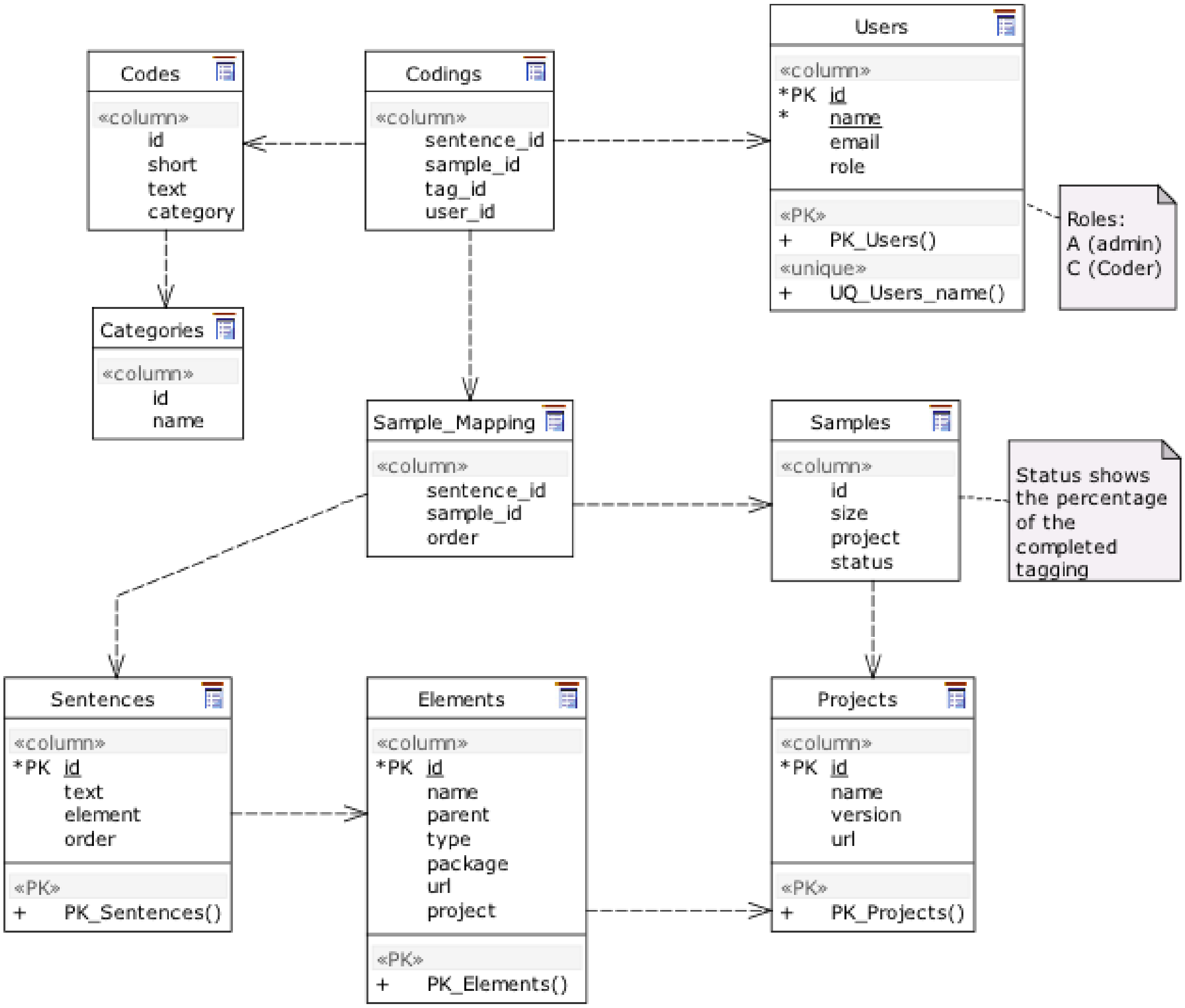
The sampling will be based on:

- Element types:
  - Packages
  - Types (i.e. classes, interfaces, abstract classes etc. no need for differentiation.)
  - Methods
  - Fields
- **Package**: The sampling will be also based to the package to which an element belongs. Therefore, we need to know what is the package of an element.

5. API Projects to be Coded

- Java API (Java 6 SDK + Java 6 Enterprise Edition)
- .Net

6. Database Scheme


Codes
«column»
id
short
text
category
Codings
«column»
sentence_id
sample_id
tag_id
user_id
Users
«column»
*PK id
* name
email
role
«PK»
+ PK_Users()
«unique»
+ UQ_Users_name()
Roles:
A (admin)
C (Coder)
Categories
«column»
id
name
Sample_Mapping
«column»
sentence_id
sample_id
order
Samples
«column»
id
size
project
status
Status shows the percentage of the completed tagging
Sentences
«column»
*PK id
text
element
order
«PK»
+ PK_Sentences()
Elements
«column»
*PK id
name
parent
type
package
url
project
«PK»
+ PK_Elements()
Projects
«column»
*PK id
name
version
url
«PK»
+ PK_Projects()