Documentation Trivial 1iB

Version 1.0

Ana Areces González (anareces)

Raquel Arrojo López (raquelarrojo)

David Casado Corona (dcasado)

María González González (meri294)

Álvaro Palanco Lorenzo (alvaropalanco)

Iván Sánchez Valdés (uo230364)

Project information

All the documentation files can be found inside the documentation folder in our Github repository

https://github.com/Arquisoft/Trivial_i1b

This is a direct access to the Readme that contains the general information of the group.

https://github.com/Arquisoft/Trivial_i1b/blob/master/README.md

This folder contains the Documentation information like manuals and all the diagrams, also the Visual Paradigm project.

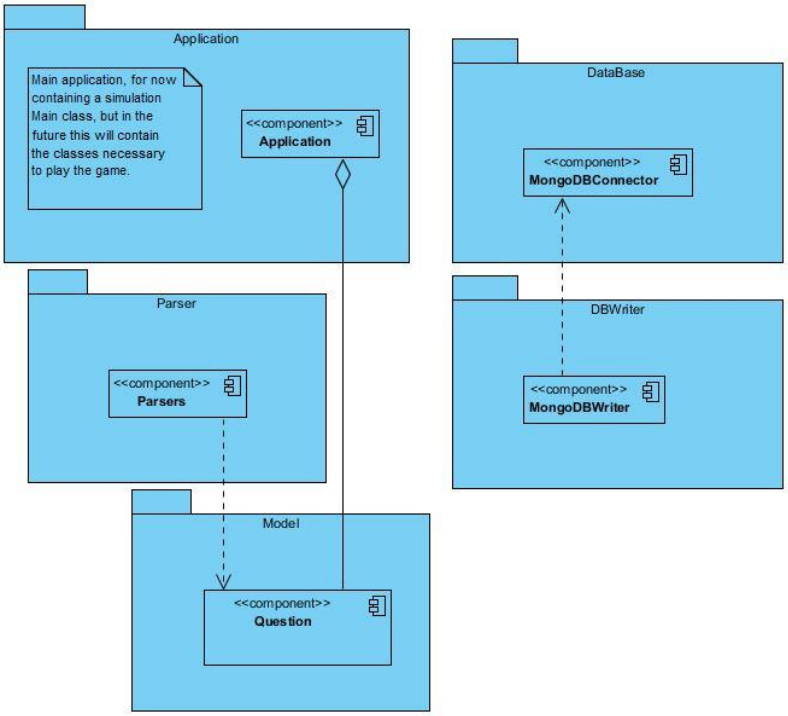
https://github.com/Arquisoft/Trivial_i1b/tree/master/Documentation

This one is the Eclipse project, where you could find the src folder with all the implementation.

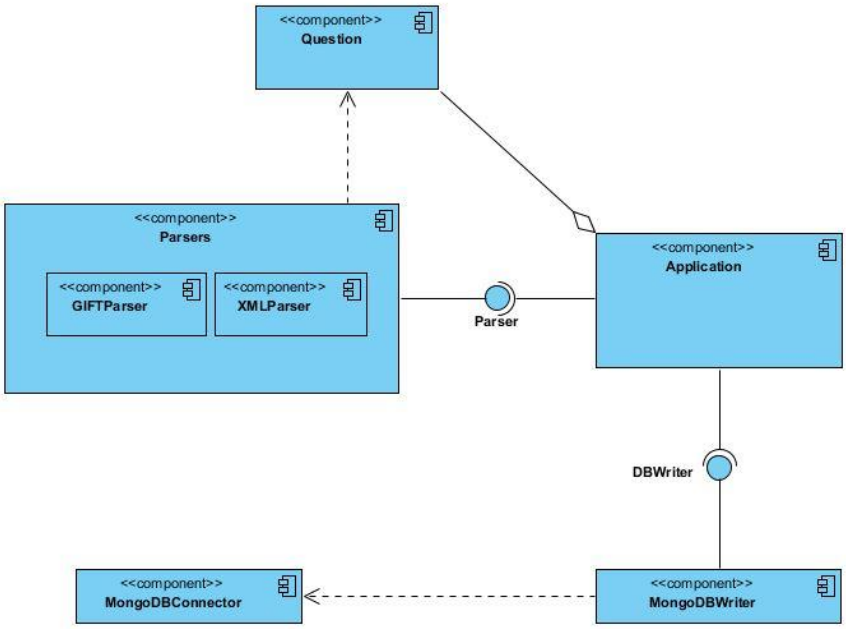
https://github.com/Arquisoft/Trivial_i1b/tree/master/Trivial_i1b

Architecture diagrams

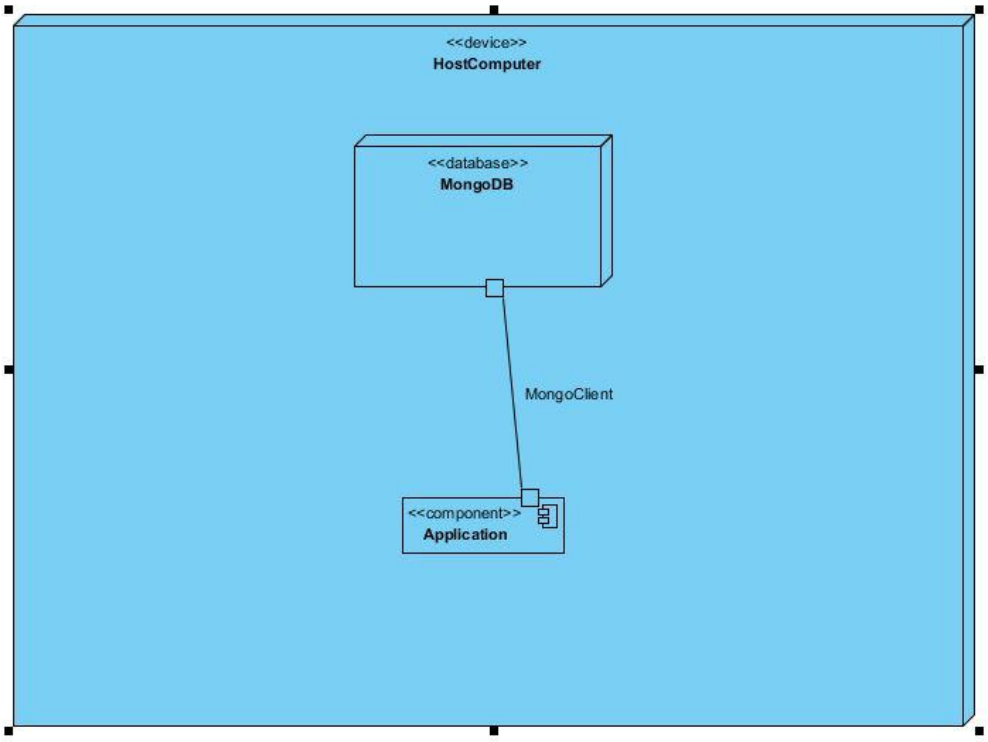
Package view



Component view



Deployment view



MANUAL

This manual is about the general information and how to deal with the module. This module receives a xml or gift file and makes the translation in order to store the information in the database.

There are two different manuals that are going to be described:

- **User manual** that explains the use that a normal user is going to do with it by using the interface.

- **System manual** that explains the code to other developers which are going to extend the app in the future, and is needed to do any changes in the source code. This manual is described to understand the functionality of the application in a short period of time.

User manual

This user manual describes the behaviour of the interface that will be used by an user (and administrator).

The main goal is to introduce in the database of the system the list with all the questions (including one right answer and two wrong ones for each question).

That will be useful to develop different types of games that can use this questions.

The application has to be started from the command line.

First, when the user starts the application to introduce the data on the database, he should write one of these three operations :

* Read + name of the file: the application reads the file with the specified name and shows its content on the command line (for example, “read test.xml” reads and shows the content of the test.xml file).

* Exit : The application is closed

* Help: It shows the help of the application with a brief explanation of each command.

When the read operation is executed, the application automatically detects the extension of the file that is going to be read. Until now, the application can read just two extensions: xml and gift, but it will be implemented to accept more formats.

System manual

In this manual is explained the different hierarchies and their relations. The implemented classes are explained by packages.

Main

Main

This class only has the task to launch the application. It reads the name of the input and calls to the appropriate class to parser the file.

Model

Question

This is an interface that generates the model to the Question objects. In this case the attribute has two abstract methods to add to the main fields that are:

- getQuestion()
- getAnswers()

TrivialQuestion

This a class that implements the Question interface. It has three fields a String of the question, a List of Strings to the answers and an integer to record the position of the True answer in the List. Then it has getters and setters to the fields and also it override the toString of the Object class.

Parser

Parser

This is an interface to use the Parser to generalize the Parser and be able to use the different classes to the different styles that are needed to be implemented. It only has one abstract method getQuestions() that returns a List of questions.

GIFTParser

GIFTParser is a class inherited from the interface Parser, consists in two methods that are parse and getQuestions, private and public respectively, as well as the constructor.

The file that will be parsed must be passed to the constructor.

The method parse is in charge of parsing the file dividing it into lines and analyzing one by one getting the useful information and storing it in an object Question, after that the object is stored in a list.

The method get Question is in charge of returning the list of objects Question previously filled by the parse method.

XMLParser

XMLParser inherits from the interface Parser. It has two methods that are a public method getQuestions and a public one, also a constructor.

The file that will be parsed must be passed to the constructor.

The parse method is uncharged of reading the file and creating the Question objects.

The method get Question returns the list of objects Question previously filled by the parse method.

DataBase

DBConnector

Interface in which we determine the method that is shared between all the different types of connectors (getConnection()).

MongoDBConnector

Specific connector for Mongo DB. In this class there is an attribute called mongo that is the client used for saving the data. It has a private method in order to create the connection used in the getConnection method in case there is not already an established connection. In case the connection cannot be established a message is shown.

DBWriter

DBWriter

Interface used in order to provide a common method for all the writers for the different databases which allows to insert questions.

MongoDBWriter

Class used to save the different questions in MongoDB. It is composed by several private methods, all of them used in order be used in the public method insertQuestions. The private methods are:

-createEntries()

-addQuestions()

We have used the Mongo Java driver (version 2.10.1) in order to build this class.