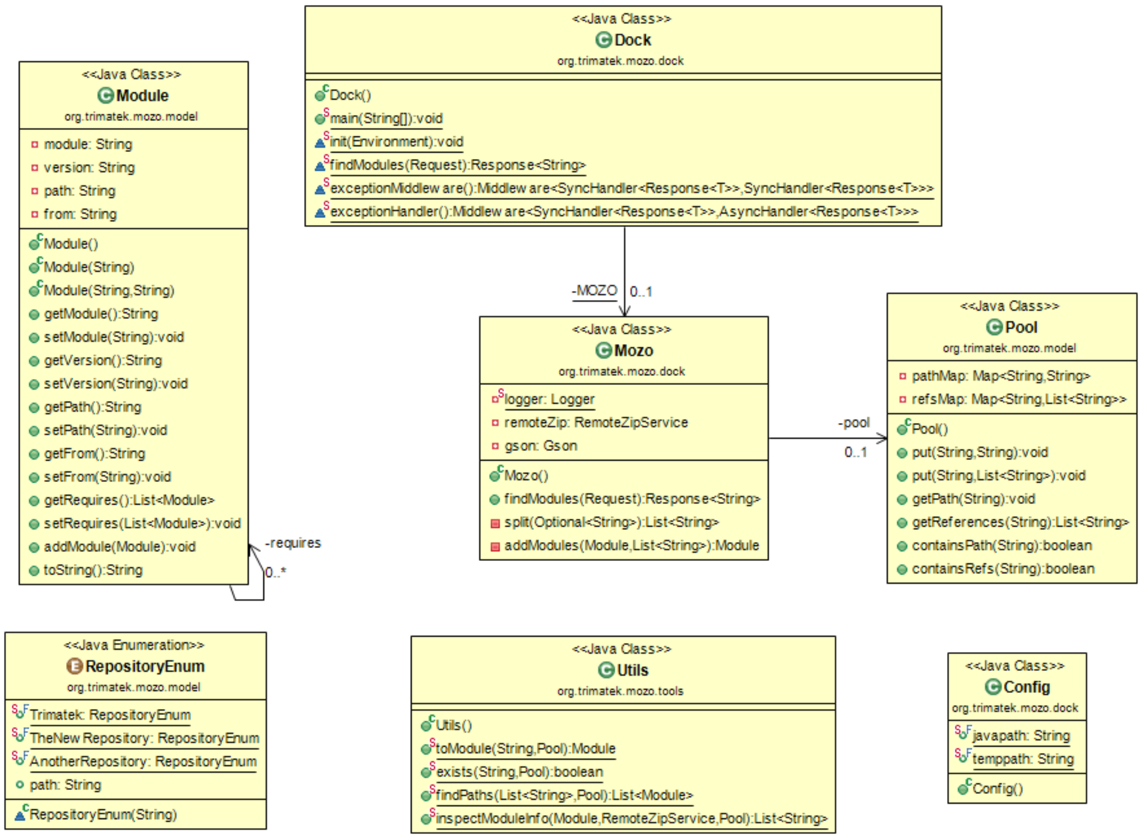
Anexo I

Mozo: Intermediario

Diagrama de Clases:



Clase Dock

Es el punto de entrada al intermediario. Inicia el servicio REST y delega en la clase Mozo la atención a cada solicitud.

```
package org.trimatek.mozo.dock;

import com.spotify.apollo.Environment;
import com.spotify.apollo.Request;
import com.spotify.apollo.Response;
import com.spotify.apollo.Status;
import com.spotify.apollo.httpservice.HttpService;
import com.spotify.apollo.httpservice.LoadingException;
import com.spotify.apollo.route.AsyncHandler;
import com.spotify.apollo.route.Middleware;
import com.spotify.apollo.route.Route;
import com.spotify.apollo.route.SyncHandler;

public final class Dock {
	private static Mozo MOZO;

	public static void main(String... args) throws LoadingException {
		HttpService.boot(Dock::init, "mozo-dock", args);
	}

	static void init(Environment environment) {
		SyncHandler<Response<String>>    addHandler    =    context    ->
findModules(context.request());

	environment.routingEngine().registerAutoRoute(Route.with(exceptionHandle
r(), "GET", "/mozo/find", addHandler));
		if (MOZO == null) {
			MOZO = new Mozo();
		}
	}

	static Response<String> findModules(Request request) {
		return MOZO.findModules(request);
	}

	/**
	 * A generic middleware that maps uncaught exceptions to error code 418
	 */
	static             <T>             Middleware<SyncHandler<Response<T>>,
SyncHandler<Response<T>>> exceptionMiddleware() {
		return handler -> requestContext -> {
			try {
				return handler.invoke(requestContext);
			} catch (RuntimeException e) {
				return Response.forStatus(Status.IM_A_TEAPOT);
			}
		};
	}

	/**
	 * Async version of {@link #exceptionMiddleware()}
	 */
	static             <T>             Middleware<SyncHandler<Response<T>>,
AsyncHandler<Response<T>>> exceptionHandler() {
		return                                                   Dock.<T>
exceptionMiddleware().and(Middleware::syncToAsync);
	}

}
```

Clase Mozo

Es donde se concentra la lógica de resolución de dependencias. Accede los repositorios (RepositoryEnum) y extrae los descriptores de módulos con RemoteZip.

```
package org.trimatek.mozo.dock;

import java.util.Arrays;
import java.util.List;
import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;

import org.trimatek.mozo.model.Module;
import org.trimatek.mozo.model.Pool;
import org.trimatek.mozo.model.RepositoryEnum;
import org.trimatek.mozo.tools.Utils;
import org.trimatek.remotezip.service.RemoteZipService;
import org.trimatek.remotezip.service.impl.RemoteZipServiceImpl;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.spotify.apollo.Request;
import com.spotify.apollo.Response;
import com.spotify.apollo.Status;

public class Mozo {

	private static Logger logger = Logger.getLogger(Mozo.class.getName());
	private RemoteZipService remoteZip;
	private Gson gson;
	private Pool pool;

	public Mozo() {
		remoteZip = new RemoteZipServiceImpl();
		gson                           =                           new
GsonBuilder().setPrettyPrinting().disableHtmlEscaping().create();
		pool = new Pool();
	}

	public Response<String> findModules(Request request) {
		try {
			List<String> targets = split(request.parameter("modules"));
			Module module = new Module();
			module.setFrom("user-request: " + request.uri());
			module = addModules(module, targets);
			return Response.forPayload(gson.toJson(module));
		} catch (Exception e) {
			return Response.forStatus(Status.BAD_REQUEST);
		}
	}

	private List<String> split(Optional<String> modules) {
		return (List<String>) Arrays.asList(modules.get().split(","));
	}

	private  Module  addModules(Module  module,  List<String>  targets)  throws
Exception {
		List<Module> requires = null;
		String path;
		if (targets == null) {
			if (module.getPath() == null) {
				for	(RepositoryEnum	repository	:
RepositoryEnum.values()) {
					path = repository.path + module.toString() +
".jar";
					if (Utils.exists(path, pool)) {
						module.setPath(path);
						break;
					}
				}
			}
			targets = Utils.inspectModuleInfo(module, remoteZip, pool);
		}
		if (module.getModule() != null) {
			pool.put(module.getModule(), targets);
		}
		if (targets != null && !targets.isEmpty()) {
			requires = Utils.findPaths(targets, pool);
			for (Module aModule : requires) {
				logger.log(Level.INFO, "\t\t >>>>> Module Added: " +
aModule.toString());
				pool.put(aModule.getModule(), aModule.getPath());
				addModules(aModule, null);
			}
			module.setRequires(requires);
		}
		return module;
	}

}
```

Clase Pool

Es el caché de descriptores. Mozo consulta a Pool si cuenta con ese descriptor antes de “salir a buscarlo” por los repositorios.

```
package org.trimatek.mozo.model;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Pool {

	private Map<String, String> pathMap;
	private Map<String, List<String>> refsMap;

	public Pool() {
		pathMap = new HashMap<String, String>();
		refsMap = new HashMap<String, List<String>>();
	}

	public void put(String moduleName, String path) {
		pathMap.put(moduleName, path);
	}

	public void put(String moduleName, List<String> refs) {
		refsMap.put(moduleName, refs);
	}

	public void getPath(String moduleName) {
		pathMap.get(moduleName);
	}

	public List<String> getReferences(String moduleName) {
		return refsMap.get(moduleName);
	}

	public boolean containsPath(String path){
		return pathMap.containsValue(path);
	}

	public boolean containsRefs(String moduleName){
		return refsMap.containsKey(moduleName);
	}

}
```

Clase Module

Es la entidad creada a partir de los descriptores de módulos. Se organiza como una composición de Módulos que luego es convertida a una estructura JSON.

```
package org.trimatek.mozo.model;

import java.util.ArrayList;
import java.util.List;

public class Module {

	private String module;
	private String version;
	private String path;
	private String from;
	private List<Module> requires;

	public Module() {
	}

	public Module(String name) {
		module = name;
	}

	public Module(String name, String version) {
		module = name;
		this.version = version;
	}

	public String getModule() {
		return module;
	}

	public void setModule(String module) {
		this.module = module;
	}

	public String getVersion() {
		return version;
	}

	public void setVersion(String version) {
		this.version = version;
	}

	public String getPath() {
		return path;
	}

	public void setPath(String path) {
		this.path = path;
	}

	public String getFrom() {
		return from;
	}

	public void setFrom(String from) {
		this.from = from;
	}

	public List<Module> getRequires() {
		return requires;
	}

	public void setRequires(List<Module> requires) {
		this.requires = requires;
	}

	public void addModule(Module module) {
		if (requires == null) {
			requires = new ArrayList<Module>();
		}
		requires.add(module);
	}

	public String toString() {
		if (module != null && version != null && path != null) {
			return module + "@" + version + "=" + path;
		} else if (module != null && version != null) {
			return module + "@" + version;
		} else if (module != null && path != null) {
			return module + "=" + path;
		} else if (module != null) {
			return module;
		}
		return null;
	}
}
```

RepositoryEnum

Es la colección de repositorios conocidos por el intermediario y que visitará al buscar descriptores.

```
package org.trimatek.mozo.model;

public enum RepositoryEnum {
	Trimatek("http://www.trimatek.org/repository/"),
	TheNewRepository("https://thenewrepository.000webhostapp.com/"),
	AnotherRepository("https://anotherrepository.000webhostapp.com/");

	public String path;

	RepositoryEnum(String path) {
		this.path = path;
	}
}
```

Clase Utils

Concentra métodos estáticos con herramientas de soporte al análisis y conversión de datos.

```
package org.trimatek.mozo.tools;

import static org.trimatek.mozo.dock.Config.javapath;
import static org.trimatek.mozo.dock.Config.temppath;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;
import org.trimatek.mozo.model.Module;
import org.trimatek.mozo.model.Pool;
import org.trimatek.mozo.model.RepositoryEnum;
import org.trimatek.remotezip.model.RemoteZipEntry;
import org.trimatek.remotezip.service.RemoteZipService;
import org.trimatek.remotezip.tools.RemoteZipFile;

public class Utils {

	public static Module toModule(String target, Pool pool) throws Exception
{
		Module module = null;
		String path = null;
		if (target.contains("@")) {
			String[] splitted = target.split("@");
			module = new Module(splitted[0], splitted[1]);
		} else {
			module = new Module(target);
		}
		for (RepositoryEnum repository : RepositoryEnum.values()) {
			path = repository.path + module.toString() + ".jar";
			if (exists(path, pool)) {
				module.setPath(path);
				break;
			}
		}
		return module;
	}

	public static boolean exists(String URLName, Pool pool) throws
MalformedURLException, IOException {
		if (pool.containsPath(URLName)) {
			return true;
		}
		HttpURLConnection.setFollowRedirects(false);
		// note : you may also need
		// HttpURLConnection.setInstanceFollowRedirects(false)
		HttpURLConnection con = (HttpURLConnection) new
URL(URLName).openConnection();
		con.setRequestMethod("HEAD");
		return (con.getResponseCode() == HttpURLConnection.HTTP_OK);
	}

	public static List<Module> findPaths(List<String> targets, Pool pool)
throws Exception {
		List<Module> modules = new ArrayList<Module>();
		for (String target : targets) {
			modules.add(Utils.toModule(target, pool));
		}
		return modules;
	}

	public static List<String> inspectModuleInfo(Module module,
RemoteZipService remoteZip, Pool pool)
			throws IOException {
		if (pool.containsRefs(module.getModule())) {
			return pool.getReferences(module.getModule());
		}
		List<String> targets = new ArrayList<String>();
		RemoteZipFile zip = remoteZip.load(module.getPath(), null);
		for (RemoteZipEntry e : zip.getEntries()) {
			if (e.getName().equals("module-info.class")) {
				InputStream inputStream =
remoteZip.getEntryStream(e, zip);
				File file = new File(temppath + e.getName());
				OutputStream outputStream = new
FileOutputStream(file);
```

```
                        IOUtils.copy(inputStream, outputStream);
                        outputStream.close();
                        break;
                    }
                }
                Runtime rt = Runtime.getRuntime();
                Process pr = rt.exec(javapath + "javap " + temppath + "module-
info.class");
                BufferedReader br = new BufferedReader(new
InputStreamReader(pr.getInputStream()));
                String line;
                while ((line = br.readLine()) != null) {
                    if (line.contains("requires")) {
                        String words[] = line.trim().split(" ");
                        if (!words[1].trim().startsWith("java.")) {
                            line = line.replace("requires", "");
                            line = line.replace(";", "");
                            targets.add(line.trim());
                        }
                    }
                }
                return targets;
        }

}
```

Clase Config

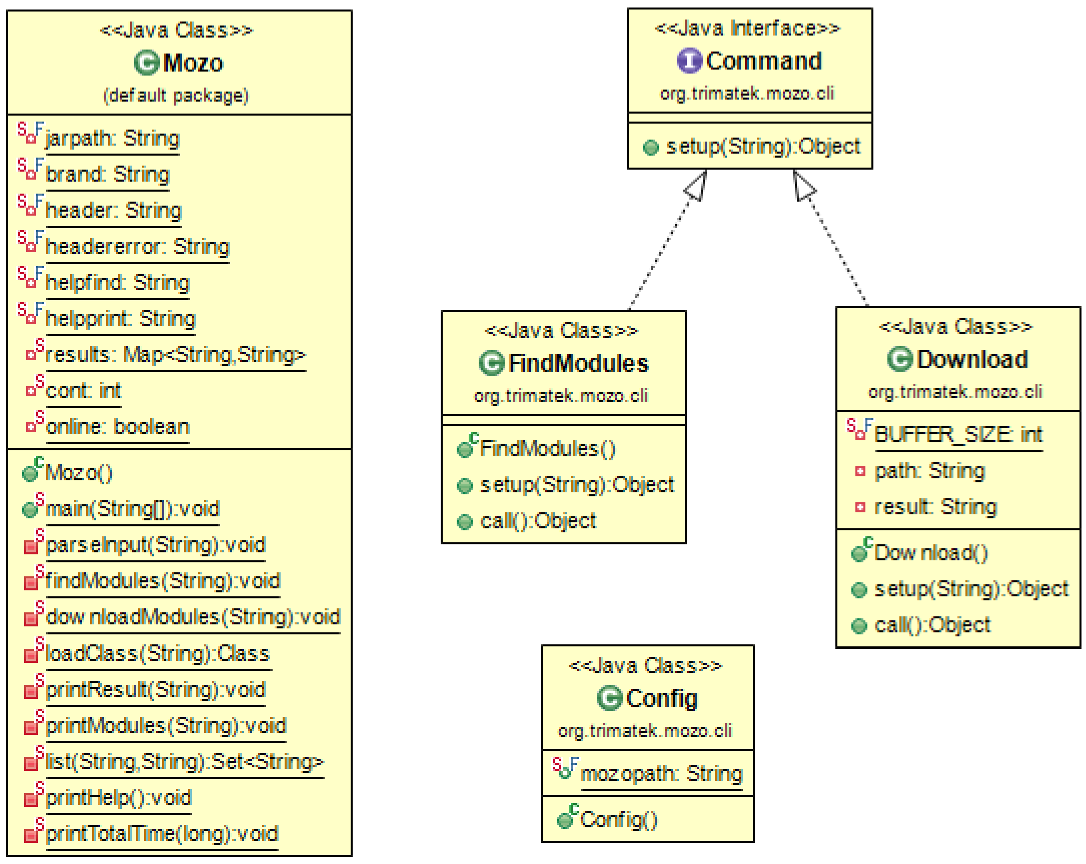
Define las rutas a un directorio temporal y al programa javap.

```
package org.trimatek.mozo.dock;

public class Config {

        public static final String javapath = "/opt/jdk-9/bin/";
        public static final String temppath = "/tmp/";

}
```

Anexo II

Mozo: Cliente

Diagrama de Clases:



Clase Mozo

Es la clase que se descargar desde una URL y activa la interfaz de comandos. Carga clases remotas que implementan la interfaz Command.

```
import java.lang.reflect.Method;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLClassLoader;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Scanner;
import java.util.Set;
import java.util.concurrent.Callable;
import java.util.concurrent.Executors;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;

public class Mozo {
	private final static String jarpath = 
"http://www.trimatek.org/mozo/org.trimatek.mozo.cli.jar";
	private final static String brand = "Mozo 0.3";
	private final static String header = "mozo> ";
	private final static String headererror = "Error: ";
	private final static String helpfind = "[List of modules names separated 
by commas (e.g.: com.mod1,org.mod2)]";
	private final static String helpprint = "[Variable with result (e.g.: 
res0)]";
	private static Map<String, String> results = new HashMap<String, 
String>();
	private static int cont;
	private static boolean online = true;

	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.println(brand);
		while (online) {
			System.out.print(header);
			parseInput(scanner.nextLine());
		}
		scanner.close();
	}

	private static void parseInput(String input) {
		long startTime = System.nanoTime();
		String[] args = input.split(" ");
		for (String arg : args) {
			if (arg.equals("find-modules") || arg.equals("fm")) {
				if (args.length < 2) {
					break;
				}
				findModules(args[1]);
				printTotalTime(startTime);
				return;
			} else if (arg.equals("download-modules") || 
arg.equals("dm")) {
				if (args.length < 2) {
					break;
				}
				downloadModules(args[1]);
				printTotalTime(startTime);
				return;
			} else if (arg.equals("print") || arg.equals("p")) {
				if (args.length < 2) {
					break;
				}
				return;
			} else if (arg.equals("list-modules") || arg.equals("lm")) 
{
				if (args.length < 2) {
					break;
				}
				printModules(args[1]);
				return;
			} else if (arg.equals("help") || arg.equals("ayuda") || 
arg.equals("man")) {
```

```
                        printHelp();
                        return;
                  } else if (arg.equals("quit") || arg.equals("exit") ||
arg.equals("salir")) {
                        System.out.println("bye");
                        online = false;
                        return;
                  }
            }
            System.out.println(headererror + "command not found");
            printHelp();
      }

      private static void findModules(String target) {
            try {
                  Class c = loadClass("org.trimatek.mozo.cli.FindModules");
                  Method m = c.getMethod("setup", String.class);
                  String key = "res" + cont++;
                  results.put(key, (String) m.invoke(c.newInstance(),
target));
                  printResult(key);
                  System.out.println("Result stored in: " + key);
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      private static void downloadModules(String key) {
            try {
                  ThreadPoolExecutor exec = (ThreadPoolExecutor)
Executors.newFixedThreadPool(5);
                  Class c = loadClass("org.trimatek.mozo.cli.Download");
                  if (results.containsKey(key)) {
                        for (String path : list(key, "path")) {
                              Method m = c.getMethod("setup",
String.class);
                              Object d = c.newInstance();
                              m.invoke(d, path);
                              exec.submit((Callable)d);
                        }
                  }
                  while (exec.getActiveCount() > 0) {
                  }
                  System.out.println("Total downloaded: " +
exec.getCompletedTaskCount());
                  exec.shutdown();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      private static Class loadClass(String className) throws
ClassNotFoundException, MalformedURLException {
            URL[] classLoaderUrls = new URL[] { new URL(jarpath) };
            URLClassLoader loader = new URLClassLoader(classLoaderUrls);
            return Class.forName(className, true, loader);
      }
```

```
	private static void printResult(String key) {
		System.out.println(results.get(key));
	}

	private static void printModules(String key) {
		int c = 0;
		if (results.containsKey(key)) {
			for (String module : list(key, "module")) {
				System.out.println(module);
				c++;
			}
		}
		System.out.println("Total: " + c);
	}

	// TODO Display version
	private static Set<String> list(String key, String field) {
		Set<String> elements = new HashSet<String>();
		for (String line : results.get(key).split("\n")) {
			if (line.contains("\"" + field + "\":")) {
				elements.add((line.replaceAll("\"",
"").replaceAll(field + ":", "").replaceAll(",", "")).trim());
			}
		}
		return elements;
	}

	private static void printHelp() {
		System.out.println("Syntax:");
		System.out.println("\tfind-modules " + helpfind);
		System.out.println("\tfm " + helpfind);
		System.out.println("\tdownload-modules " + helpprint);
		System.out.println("\tdm " + helpprint);
		System.out.println("\tprint " + helpprint);
		System.out.println("\tp " + helpprint);
		System.out.println("\tlist-modules " + helpprint);
		System.out.println("\tlm " + helpprint);
	}

	private static void printTotalTime(long startTime) {
		System.out.println("Elapsed time: "
				+ TimeUnit.MILLISECONDS.convert(System.nanoTime() -
startTime, TimeUnit.NANOSECONDS) / 1000.0
				+ " seconds");
	}

}
```

Interfaz Command

Es la interfaz que deben implementar las clases que se implementan los comandos que se ejecutan desde la clase Mozo.

```
package org.trimatek.mozo.cli;

import java.util.concurrent.Callable;
```

```
public interface Command extends Callable {

	public Object setup(String arg) throws Exception;

}
```

Clase FindModules

Es la clase que se carga remotamente al ejecutar el comando find-modules desde la interfaz de comandos. Implementa la interfaz Command.

```
package org.trimatek.mozo.cli;

import static org.trimatek.mozo.cli.Config.mozopath;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class FindModules implements Command {

	@Override
	public Object setup(String arg) throws Exception {
		URL url = new URL(mozopath + "find?modules=" + arg);
		HttpURLConnection con = (HttpURLConnection) url.openConnection();
		int responseCode = con.getResponseCode();
		System.out.println("GET request: " + url);
		System.out.println("Response code: " + responseCode);
		BufferedReader in = new BufferedReader(new
InputStreamReader(con.getInputStream()));
		String output;
		StringBuffer response = new StringBuffer();
		while ((output = in.readLine()) != null) {
			response.append(output + "\n");
		}
		in.close();
		con.disconnect();
		return response.toString();
	}

	@Override
	public Object call() throws Exception {
		// TODO Auto-generated method stub
		return null;
	}

}
```

Clase Download

Es la clase que implementa la descarga de módulos desde los repositorios y hacia el cliente. La clase Mozo del cliente ejecuta hasta 5 threads de esta clase en simultáneo de forma tal de optimizar el ancho de banda para la descarga de módulo.

```
package org.trimatek.mozo.cli;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.concurrent.Callable;

public class Download implements Command, Callable {

	private static final int BUFFER_SIZE = 3072;
	private String path;
	private String result = "Ready";

	public Object setup(String path) throws IOException {
		this.path = path;
		return result;
	}

	@Override
	public Object call() {
		try {
			result = "The file could not be downloaded";
			URL url = new URL(path);
			HttpURLConnection httpConn = (HttpURLConnection)
url.openConnection();
			int responseCode = httpConn.getResponseCode();
			if (responseCode == HttpURLConnection.HTTP_OK) {
				String fileName = "";
				fileName = path.substring(path.lastIndexOf("/") + 1,
path.length());
				System.out.println("Downloading: " + fileName);
				InputStream inputStream = httpConn.getInputStream();
				String saveFilePath = System.getProperty("user.dir")
+ File.separator + fileName;
				FileOutputStream outputStream = new
FileOutputStream(saveFilePath);
				int bytesRead = -1;
				byte[] buffer = new byte[BUFFER_SIZE];
				while ((bytesRead = inputStream.read(buffer)) != -1)
{
					outputStream.write(buffer, 0, bytesRead);
				}
				outputStream.close();
				inputStream.close();
			}
			httpConn.disconnect();
		} catch (IOException e) {
			System.out.println(e);
		}
		return result;
```

```
        }

}
```

Clase Config

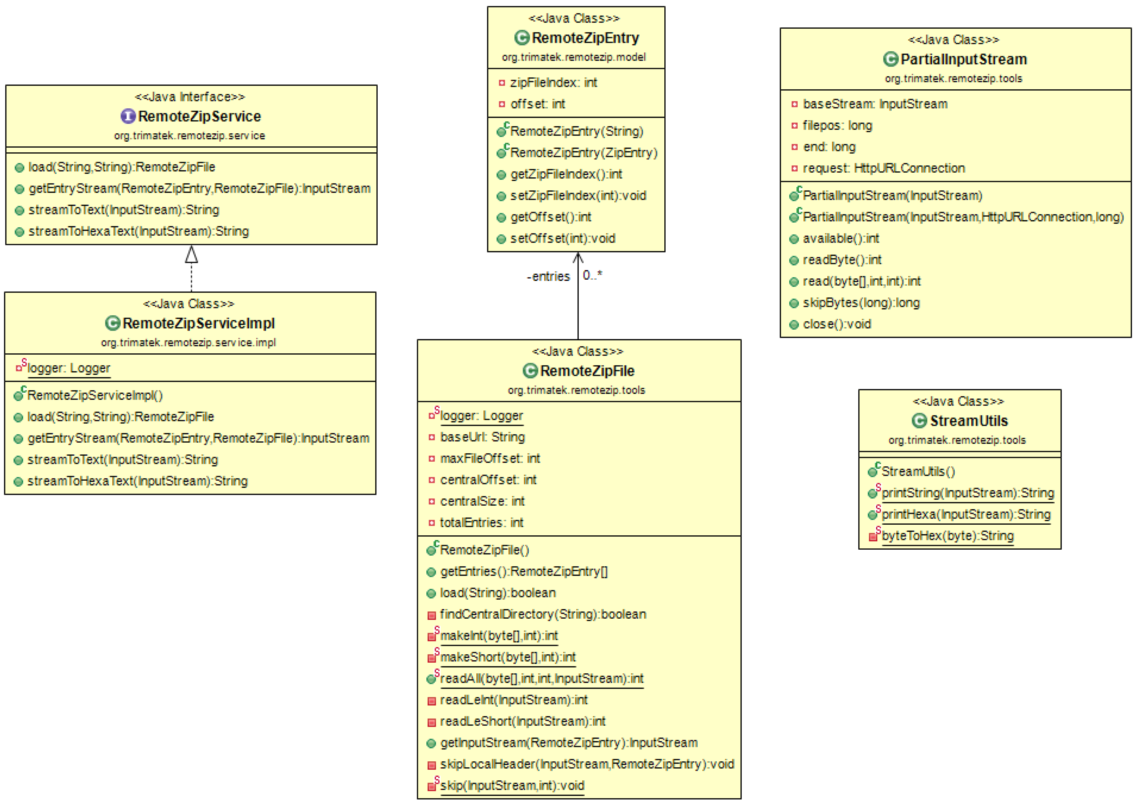
Define la URL al servicio en la nube.

```
package org.trimatek.mozo.cli;

public class Config {

        public static final String mozopath = "http://trimatek.org:8080/mozo/";

}
```

Anexo III

Remote Zip

Diagrama de Clases:



<<Java Interface>>
RemoteZipService
org.trimatek.remotezip.service
load(String,String):RemoteZipFile
getEntryStream(RemoteZipEntry,RemoteZipFile):InputStream
streamToText(InputStream):String
streamToHexaText(InputStream):String
<<Java Class>>
RemoteZipServiceImpl
org.trimatek.remotezip.service.impl
logger: Logger
RemoteZipServiceImpl()
load(String,String):RemoteZipFile
getEntryStream(RemoteZipEntry,RemoteZipFile):InputStream
streamToText(InputStream):String
streamToHexaText(InputStream):String
<<Java Class>>
RemoteZipEntry
org.trimatek.remotezip.model
zipFileIndex: int
offset: int
RemoteZipEntry(String)
RemoteZipEntry(ZipEntry)
getZipFileIndex():int
setZipFileIndex(int):void
getOffset():int
setOffset(int):void
-entries
0..*
<<Java Class>>
RemoteZipFile
org.trimatek.remotezip.tools
logger: Logger
baseUrl: String
maxFileOffset: int
centralOffset: int
centralSize: int
totalEntries: int
RemoteZipFile()
getEntries():RemoteZipEntry[]
load(String):boolean
findCentralDirectory(String):boolean
makeInt(byte[],int):int
makeShort(byte[],int):int
readAll(byte[],int,int,InputStream):int
readLeInt(InputStream):int
readLeShort(InputStream):int
getInputStream(RemoteZipEntry):InputStream
skipLocalHeader(InputStream,RemoteZipEntry):void
skip(InputStream,int):void
<<Java Class>>
PartialInputStream
org.trimatek.remotezip.tools
baseStream: InputStream
filepos: long
end: long
request: HttpURLConnection
PartialInputStream(InputStream)
PartialInputStream(InputStream,HttpURLConnection,long)
available():int
readByte():int
read(byte[],int,int):int
skipBytes(long):long
close():void
<<Java Class>>
StreamUtils
org.trimatek.remotezip.tools
StreamUtils()
printString(InputStream):String
printHexa(InputStream):String
byteToHex(byte):String

Clase RemoteZipServiceImpl

Es la implementación de la interfaz RemoteZipService y es el punto de entrada a todas las funcionalidades de la biblioteca.

```
package org.trimatek.remotezip.service.impl;

import java.io.IOException;
import java.io.InputStream;
import java.security.InvalidParameterException;
import java.util.logging.Level;
import java.util.logging.Logger;

import org.trimatek.remotezip.Config;
import org.trimatek.remotezip.model.RemoteZipEntry;
import org.trimatek.remotezip.service.RemoteZipService;
import org.trimatek.remotezip.tools.RemoteZipFile;
import org.trimatek.remotezip.tools.StreamUtils;

public class RemoteZipServiceImpl implements RemoteZipService {

	private static Logger logger =
Logger.getLogger(RemoteZipServiceImpl.class.getName());
```

```
        public RemoteZipFile load(String path, String proxy) throws IOException
{
                RemoteZipFile rf = null;
                if (proxy != null) {
                        String[] splitted = proxy.split(":");
                        if (splitted.length != 2) {
                                throw new InvalidParameterException("Proxy address
not valid");
                        } else {
                                Config.PROXY_URL = splitted[0];
                                Config.PROXY_PORT = splitted[1];
                        }
                }
                rf = new RemoteZipFile();
                if (rf.load(path)) {
                        return rf;
                }
                return null;
        }

        @Override
        public InputStream getEntryStream(RemoteZipEntry entry, RemoteZipFile
remoteZip) {
                try {
                        return remoteZip.getInputStream(entry);
                } catch (IOException e) {
                        logger.log(Level.SEVERE, "Error while retrieving entry",
e);
                }
                return null;
        }

        @Override
        public String streamToText(InputStream inputStream) {
                return StreamUtils.printString(inputStream);
        }

        @Override
        public String streamToHexaText(InputStream inputStream) {
                try {
                        return StreamUtils.printHexa(inputStream);
                } catch (IOException e) {
                        logger.log(Level.SEVERE, "Error while converting stream to
hexadecimal", e);
                }
                return null;
        }

}
```

Clase RemoteZipEntry

Es la clase que representa a un archivo del Zip remoto.

```java
package org.trimatek.remotezip.model;

import java.util.zip.ZipEntry;

public class RemoteZipEntry extends ZipEntry {

	private int zipFileIndex;
	private int offset;

	public RemoteZipEntry(String name) {
		super(name);
	}

	public RemoteZipEntry(ZipEntry e) {
		super(e);
	}

	public int getZipFileIndex() {
		return zipFileIndex;
	}

	public void setZipFileIndex(int zipFileIndex) {
		this.zipFileIndex = zipFileIndex;
	}

	public int getOffset() {
		return offset;
	}

	public void setOffset(int offset) {
		this.offset = offset;
	}

}
```

Clase RemoteZipFile

Es la clase que concentra la mayor parte de la funcionalidad de la solución. Sus métodos localizan el directorio central y extraen el archivo objetivo.

```java
package org.trimatek.remotezip.tools;

import static org.trimatek.remotezip.Config.PROXY_PORT;
import static org.trimatek.remotezip.Config.PROXY_URL;

import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.zip.Inflater;
import java.util.zip.InflaterInputStream;
import java.util.zip.ZipEntry;
import java.util.zip.ZipException;
import java.util.zip.ZipInputStream;

import
org.apache.commons.compress.compressors.bzip2.BZip2CompressorInputStream;
import org.trimatek.remotezip.model.RemoteZipEntry;
```

```
public class RemoteZipFile {

	private static Logger logger =
Logger.getLogger(RemoteZipFile.class.getName());
	private RemoteZipEntry[] entries;
	private String baseUrl;
	private int maxFileOffset;
	private int centralOffset, centralSize;
	private int totalEntries;

	public RemoteZipFile() {
		if (PROXY_URL != null && PROXY_PORT != null) {
			System.setProperty("https.proxyHost", PROXY_URL);
			System.setProperty("https.proxyPort", PROXY_PORT);
		}
	}

	public RemoteZipEntry[] getEntries() {
		return entries;
	}

	public boolean load(String path) throws IOException {

		if (!findCentralDirectory(path)) {
			return false;
		}
		maxFileOffset = centralOffset;
		baseUrl = path;
		entries = new RemoteZipEntry[totalEntries];
		URL url = new URL(path);
		HttpURLConnection req = (HttpURLConnection)
url.openConnection();
		req.setRequestProperty("Range", "bytes=" + centralOffset + "-"
				+ centralOffset + centralSize);
		req.connect();
		logger.log(Level.INFO, "Response Code: " +
req.getResponseCode());
		logger.log(Level.INFO, "Content-Length: " +
req.getContentLengthLong());
		logger.log(Level.INFO, "Total entries: " + totalEntries);

		InputStream s = req.getInputStream();
		try {
			for (int i = 0; i < totalEntries; i++) {
				if (readLeInt(s) != ZipInputStream.CENSIG) {
					throw new ZipException("Wrong Central
Directory signature");
				}
				readLeInt(s);
				readLeShort(s);
				int method = readLeShort(s);
				int dostime = readLeInt(s);
				int crc = readLeInt(s);
				int csize = readLeInt(s);
				int size = readLeInt(s);
				int nameLen = readLeShort(s);
				int extraLen = readLeShort(s);
				int commentLen = readLeShort(s);
				readLeInt(s);
				readLeInt(s);
				int offset = readLeInt(s);
				byte[] buffer = new byte[Math.max(nameLen,
commentLen)];
				readAll(buffer, 0, nameLen, s);
				String name = new String(buffer, "UTF-8");
				RemoteZipEntry entry = new RemoteZipEntry(name);
```

```
                        entry.setMethod((int) (method & 0xffffffffL));
                        entry.setCrc(crc & 0xffffffffL);
                        entry.setSize(size & 0xffffffffL);
                        entry.setCompressedSize(csize & 0xffffffffL);
                        // TODO check time data
                        entry.setTime(dostime);
                        if (extraLen > 0) {
                              byte[] extra = new byte[extraLen];
                              readAll(extra, 0, extraLen, s);
                              entry.setExtra(extra);
                        }
                        if (commentLen > 0) {
                              readAll(buffer, 0, commentLen, s);
                              entry.setComment(new String(buffer, "UTF-
8"));
                        }
                        entry.setZipFileIndex(i);
                        entry.setOffset(offset);
                        entries[i] = entry;
                  }
            } finally {
                  s.close();
                  req.disconnect();
            }
            return true;
      }

      private boolean findCentralDirectory(String path) throws IOException {

            URL url = new URL(path);
            int currentLength = 256;
            int entries = 0;
            int size = 0;
            int offset = -1;

            while (true) {

                  HttpURLConnection req = (HttpURLConnection)
url.openConnection();
                  req.setRequestProperty("Range", "bytes=" + "-"
                              + (currentLength + 22));
                  req.connect();
                  logger.log(Level.INFO, "Response Code: " +
req.getResponseCode());
                  logger.log(Level.INFO, "Content-Length: " +
req.getContentLength());

                  InputStream is = req.getInputStream();
                  byte[] bb = new byte[req.getContentLength()];

                  int endSize = readAll(bb, 0, req.getContentLength(), is);

                  req.disconnect();

                  int pos = endSize - 22;
                  int state = 0;
                  while (pos >= 0) {
                        if (bb[pos] == 0x50) {
                              if (bb[pos + 1] == 0x4b && bb[pos + 2] ==
0x05
                                          && bb[pos + 3] == 0x06) {
                                    logger.log(Level.INFO, "Central
directory found!");
                                    break;
                              }
                              pos -= 4;
                        } else
```

```
                            pos--;
                    }

                    if (pos < 0) {
                          if (currentLength == 65536)
                                break;

                          if (currentLength == 1024)
                                currentLength = 65536;
                          else if (currentLength == 256)
                                currentLength = 1024;
                          else
                                break;
                    } else {
                          centralSize = makeInt(bb, pos + 12);
                          centralOffset = makeInt(bb, pos + 16);
                          totalEntries = makeShort(bb, pos + 10);
                          logger.log(Level.INFO, "TotalEntries: " +
totalEntries);
                          return true;
                    }

              }

              return false;
       }

       private static int makeInt(byte[] bb, int pos) {
              int zero = bb[pos + 0];
              if (zero < 0)
                    zero += 256;
              int one = bb[pos + 1];
              if (one < 0)
                    one += 256;
              int three = bb[pos + 2];
              if (three < 0)
                    three += 256;
              int four = bb[pos + 3];
              if (four < 0)
                    four += 256;
              return zero | one << 8 | three << 16 | four << 24;
       }

       private static int makeShort(byte[] bb, int pos) {
              int zero = bb[pos + 0];
              if (zero < 0)
                    zero += 256;
              int one = bb[pos + 1];
              if (one < 0)
                    one += 256;
              return zero | one << 8;
       }

       public static int readAll(byte[] bb, int p, int sst, InputStream s)
                    throws IOException {
              int ss = 0;
              while (ss < sst) {
                    int r = s.read(bb, p, sst - ss);
                    if (r <= 0)
                          return ss;
                    ss += r;
                    p += r;
              }
              return ss;
       }

       private int readLeInt(InputStream s) throws IOException {
```

```
		return readLeShort(s) | readLeShort(s) << 16;
	}

	private int readLeShort(InputStream s) throws IOException {
		int first = new DataInputStream(s).readByte();
		if (first < 0)
			first += 256;
		int second = new DataInputStream(s).readByte();
		if (second < 0)
			second += 256;
		return first | second << 8;
	}

	public InputStream getInputStream(RemoteZipEntry entry)
			throws MalformedURLException, IOException {

		if (entry.getSize() == 0) {
			return null;
		}

		if (entries == null) {
			throw new IllegalStateException("ZipFile has been
closed");
		}

		int index = entry.getZipFileIndex();
		if (index < 0 || index >= entries.length
				|| entries[index].getName() != entry.getName()) {
			throw new IndexOutOfBoundsException();
		}

		HttpURLConnection req = (HttpURLConnection) new URL(baseUrl)
				.openConnection();
		int limit = (int) (entry.getOffset() + entry.getCompressedSize()
+ 16 + 65536 * 2);
		if (limit >= maxFileOffset) {
			limit = maxFileOffset - 1;
		}

		req.setRequestProperty("Range", "bytes=" + entry.getOffset() +
"-"
				+ limit);

		InputStream baseStream = req.getInputStream();

		skipLocalHeader(baseStream, entries[index]);

		InputStream istr = new PartialInputStream(baseStream, req,
				entries[index].getCompressedSize());

		int method = entries[index].getMethod();

		switch (method) {
		case ZipEntry.STORED:
			return istr;
		case ZipEntry.DEFLATED:
			return new InflaterInputStream(istr, new Inflater(true));
		case 12:
			return new BZip2CompressorInputStream(istr);
		default:
			throw new ZipException("Unknown compression method: " +
method);
		}

	}

	private void skipLocalHeader(InputStream baseStream, RemoteZipEntry
entry)
                throws IOException {
            if (readLeInt(baseStream) != ZipEntry.LOCSIG) {
                throw new ZipException("Wrong local header signature");
            }

            skip(baseStream, 10 + 12);
            int namelen = readLeShort(baseStream);
            int extralen = readLeShort(baseStream);
            skip(baseStream, namelen + extralen);
        }

        private static void skip(InputStream s, int n) throws IOException {
            for (int i = 0; i < n; i++)
                new DataInputStream(s).readByte();
        }

}
```

Clase PartialInputStream

Esta clase gestiona la transmisión de flujos.

```
package org.trimatek.remotezip.tools;

import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.util.zip.InflaterInputStream;

public class PartialInputStream extends InflaterInputStream {

        private InputStream baseStream;
        private long filepos;
        private long end;
        private HttpURLConnection request;

        public PartialInputStream(InputStream in) {
            super(in);
        }

        public PartialInputStream(InputStream baseStream,
                HttpURLConnection request, long len) {
            super(baseStream);
            this.baseStream = baseStream;
            filepos = 0;
            end = len;
            this.request = request;
        }

        public int available() {
            long amount = end - filepos;
            if (amount > Integer.MAX_VALUE) {
                return Integer.MAX_VALUE;
            }
            return (int) amount;
        }

        public int readByte() throws IOException {
            if (filepos == end) {
```

```
                        return -1;
                }
                filepos++;
                return new DataInputStream(baseStream).readByte();
        }

        public int read(byte[] b, int off, int len) throws IOException {
                if (len > end - filepos) {
                        len = (int) (end - filepos);
                        if (len == 0) {
                                return 0;
                        }
                }
                int count = RemoteZipFile.readAll(b, off, len, baseStream);
                if (count > 0) {
                        filepos += len;
                }
                return count;
        }

        public long skipBytes(long amount) throws IOException {
                if (amount < 0) {
                        throw new IndexOutOfBoundsException();
                }
                if (amount > end - filepos) {
                        amount = end - filepos;
                }
                filepos += amount;
                for (int i = 0; i < amount; i++)
                        new DataInputStream(baseStream).readByte();
                return amount;
        }

        public void close() throws IOException {
                request.disconnect();
                baseStream.close();
        }

}
```

Clase StreamUtils

Posee métodos de soporte a la visualización de los flujos.

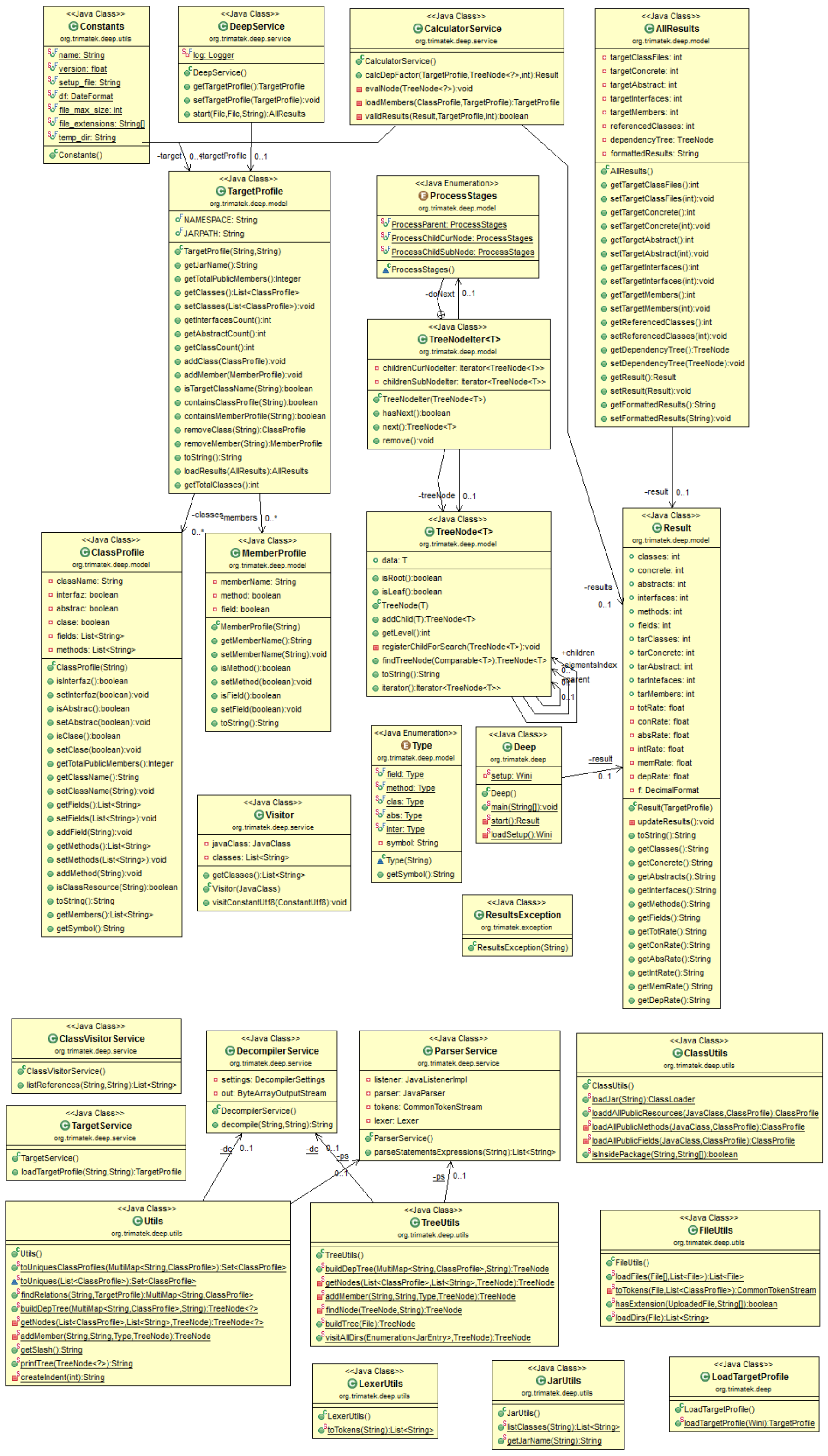
```
package org.trimatek.remotezip.tools;

import java.io.IOException;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

public class StreamUtils {

        public static String printString(java.io.InputStream stream) {
                java.util.Scanner s = new
java.util.Scanner(stream).useDelimiter("\\A");
                return s.hasNext() ? s.next() : "";
        }

        public static String printHexa(InputStream stream) throws IOException {
                StringBuffer sb = new StringBuffer();
```

```
            byte[] bytes = IOUtils.toByteArray(stream);
            for (byte b : bytes) {
                  sb.append(byteToHex(b) + " ");
            }
            return sb.toString();
      }

      private static String byteToHex(byte b) {
            return String.format("%02x", b & 0xff);
      }

}
```

Anexo IV

Deep

Diagrama de Clases:



<<Java Class>> Constants org.trimatek.deep.utils
name: String
version: float
setup_file: String
df: DateFormat
file_max_size: int
file_extensions: String[]
temp_dir: String
Constants()
<<Java Class>> DeepService org.trimatek.deep.service
log: Logger
DeepService()
getTargetProfile():TargetProfile
setTargetProfile(TargetProfile):void
start(File,File,String):AllResults
<<Java Class>> CalculatorService org.trimatek.deep.service
CalculatorService()
calcDepFactor(TargetProfile,TreeNode<?>,int):Result
evalNode(TreeNode<?>):void
loadMembers(ClassProfile,TargetProfile):TargetProfile
validResults(Result,TargetProfile,int):boolean
<<Java Class>> AllResults org.trimatek.deep.model
targetClassFiles: int
targetConcrete: int
targetAbstract: int
targetInterfaces: int
targetMembers: int
referencedClasses: int
dependencyTree: TreeNode
formattedResults: String
AllResults()
getTargetClassFiles():int
setTargetClassFiles(int):void
getTargetConcrete():int
setTargetConcrete(int):void
getTargetAbstract():int
setTargetAbstract(int):void
getTargetInterfaces():int
setTargetInterfaces(int):void
getTargetMembers():int
setTargetMembers(int):void
getReferencedClasses():int
setReferencedClasses(int):void
getDependencyTree():TreeNode
setDependencyTree(TreeNode):void
getResult():Result
setResult(Result):void
getFormattedResults():String
setFormattedResults(String):void
-target 0..1
-targetProfile 0..1
<<Java Class>> TargetProfile org.trimatek.deep.model
NAMESPACE: String
JARPATH: String
TargetProfile(String,String)
getJarName():String
getTotalPublicMembers():Integer
getClasses():List<ClassProfile>
setClasses(List<ClassProfile>):void
getInterfacesCount():int
getAbstractCount():int
getClassCount():int
addClass(ClassProfile):void
addMember(MemberProfile):void
isTargetClassName(String):boolean
containsClassProfile(String):boolean
containsMemberProfile(String):boolean
removeClass(String):ClassProfile
removeMember(String):MemberProfile
toString():String
loadResults(AllResults):AllResults
getTotalClasses():int
<<Java Enumeration>> ProcessStages org.trimatek.deep.model
ProcessParent: ProcessStages
ProcessChildCurNode: ProcessStages
ProcessChildSubNode: ProcessStages
ProcessStages()
-doNext 0..1
<<Java Class>> TreeNodeIter<T> org.trimatek.deep.model
childrenCurNodeIter: Iterator<TreeNode<T>>
childrenSubNodeIter: Iterator<TreeNode<T>>
TreeNodeIter(TreeNode<T>)
hasNext():boolean
next():TreeNode<T>
remove():void
-treeNode 0..1
-result 0..1
-classes 0..*
-members 0..*
<<Java Class>> ClassProfile org.trimatek.deep.model
className: String
interfaz: boolean
abstrac: boolean
clase: boolean
fields: List<String>
methods: List<String>
ClassProfile(String)
isInterfaz():boolean
setInterfaz(boolean):void
isAbstrac():boolean
setAbstrac(boolean):void
isClase():boolean
setClase(boolean):void
getTotalPublicMembers():Integer
getClassName():String
setClassName(String):void
getFields():List<String>
setFields(List<String>):void
addField(String):void
getMethods():List<String>
setMethods(List<String>):void
addMethod(String):void
isClassResource(String):boolean
toString():String
getMembers():List<String>
getSymbol():String
<<Java Class>> MemberProfile org.trimatek.deep.model
memberName: String
method: boolean
field: boolean
MemberProfile(String)
getMemberName():String
setMemberName(String):void
isMethod():boolean
setMethod(boolean):void
isField():boolean
setField(boolean):void
toString():String
<<Java Class>> TreeNode<T> org.trimatek.deep.model
data: T
isRoot():boolean
isLeaf():boolean
TreeNode(T)
addChild(T):TreeNode<T>
getLevel():int
registerChildForSearch(TreeNode<T>):void
findTreeNode(Comparable<T>):TreeNode<T>
toString():String
iterator():Iterator<TreeNode<T>>
-results 0..1
+children
-elementsIndex
0..*
parent
0..1
<<Java Class>> Result org.trimatek.deep.model
classes: int
concrete: int
abstracts: int
interfaces: int
methods: int
fields: int
tarClasses: int
tarConcrete: int
tarAbstract: int
tarInterfaces: int
tarMembers: int
totRate: float
conRate: float
absRate: float
intRate: float
memRate: float
depRate: float
f: DecimalFormat
Result(TargetProfile)
updateResults():void
toString():String
getClasses():String
getConcrete():String
getAbstracts():String
getInterfaces():String
getMethods():String
getFields():String
getTotRate():String
getConRate():String
getAbsRate():String
getIntRate():String
getMemRate():String
getDepRate():String
<<Java Enumeration>> Type org.trimatek.deep.model
field: Type
method: Type
clas: Type
abs: Type
inter: Type
symbol: String
Type(String)
getSymbol():String
<<Java Class>> Deep org.trimatek.deep
setup: Wini
Deep()
main(String[]):void
start():Result
loadSetup():Wini
-result 0..1
<<Java Class>> Visitor org.trimatek.deep.service
javaClass: JavaClass
classes: List<String>
getClasses():List<String>
Visitor(JavaClass)
visitConstantUtf8(ConstantUtf8):void
<<Java Class>> ResultsException org.trimatek.exception
ResultsException(String)
<<Java Class>> ClassVisitorService org.trimatek.deep.service
ClassVisitorService()
listReferences(String,String):List<String>
<<Java Class>> DecompilerService org.trimatek.deep.service
settings: DecompilerSettings
out: ByteArrayOutputStream
DecompilerService()
decompile(String,String):String
<<Java Class>> ParserService org.trimatek.deep.service
listener: JavaListenerImpl
parser: JavaParser
tokens: CommonTokenStream
lexer: Lexer
ParserService()
parseStatementsExpressions(String):List<String>
<<Java Class>> ClassUtils org.trimatek.deep.utils
ClassUtils()
loadJar(String):ClassLoader
loadAllPublicResources(JavaClass,ClassProfile):ClassProfile
loadAllPublicMethods(JavaClass,ClassProfile):ClassProfile
loadAllPublicFields(JavaClass,ClassProfile):ClassProfile
isInsidePackage(String,String[]):boolean
<<Java Class>> TargetService org.trimatek.deep.service
TargetService()
loadTargetProfile(String,String):TargetProfile
-dc 0..1
-dc 0..1
-ps 0..1
-ps 0..1
<<Java Class>> Utils org.trimatek.deep.utils
Utils()
toUniquesClassProfiles(MultiMap<String,ClassProfile>):Set<ClassProfile>
toUniques(List<ClassProfile>):Set<ClassProfile>
findRelations(String,TargetProfile):MultiMap<String,ClassProfile>
buildDepTree(MultiMap<String,ClassProfile>,String):TreeNode<?>
getNodes(List<ClassProfile>,List<String>,TreeNode):TreeNode<?>
addMember(String,String,Type,TreeNode):TreeNode
getSlash():String
printTree(TreeNode<?>):String
createIndent(int):String
<<Java Class>> TreeUtils org.trimatek.deep.utils
TreeUtils()
buildDepTree(MultiMap<String,ClassProfile>,String):TreeNode
getNodes(List<ClassProfile>,List<String>,TreeNode):TreeNode
addMember(String,String,Type,TreeNode):TreeNode
findNode(TreeNode,String):TreeNode
buildTree(File):TreeNode
visitAllDirs(Enumeration<JarEntry>,TreeNode):TreeNode
<<Java Class>> FileUtils org.trimatek.deep.utils
FileUtils()
loadFiles(File[],List<File>):List<File>
toTokens(File,List<ClassProfile>):CommonTokenStream
hasExtension(UploadedFile,String[]):boolean
loadDirs(File):List<String>
<<Java Class>> LexerUtils org.trimatek.deep.utils
LexerUtils()
toTokens(String):List<String>
<<Java Class>> JarUtils org.trimatek.deep.utils
JarUtils()
listClasses(String):List<String>
getJarName(String):String
<<Java Class>> LoadTargetProfile org.trimatek.deep
LoadTargetProfile()
loadTargetProfile(Wini):TargetProfile

A continuación, el código fuente de las clases principales.

Clase Deep

Es el punto de entrada desde la interfaz de comandos.

```
package org.trimatek.deep;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import java.util.Set;

import org.apache.commons.collections4.MultiMap;
import org.ini4j.InvalidFileFormatException;
import org.ini4j.Wini;
import org.trimatek.deep.model.ClassProfile;
import org.trimatek.deep.model.Result;
import org.trimatek.deep.model.TargetProfile;
import org.trimatek.deep.model.TreeNode;
import org.trimatek.deep.service.CalculatorService;
import org.trimatek.deep.utils.Constants;
import org.trimatek.deep.utils.TreeUtils;
import org.trimatek.deep.utils.Utils;

public class Deep {

	private static Wini setup;
	private static Result result;

	public static void main(String[] args) {

		try {
			setup = loadSetup();
			System.out.println(Constants.name + " " +
Constants.version);
			if (setup == null) {
				System.out.println("[Setup not found] Please fill "
						+ Constants.setup_file
						+ " file created at current directory
and retry");
			} else {
				System.out.println("Start " +
Constants.df.format(new Date()));
				result = start();
				System.out.println(result.toString());
				System.out.println("End " + Constants.df.format(new
Date()));
			}
		} catch (Exception e) {
			System.out.println("[Error] " + e.getMessage());
		}

	}

	private static Result start() throws Exception {
		/* Source */
		String sourceJarPath = setup.get("source", "path") +
Utils.getSlash();
		String jar = setup.get("source", "filename");
		sourceJarPath = sourceJarPath + jar;

		/* Target */
		TargetProfile target =
LoadTargetProfile.loadTargetProfile(setup);
             System.out.println(target.toString());

             /* Source -> Target */
             System.out.println("\nStart of [" + jar + " -> " +
target.getJarName()
                          + "] analysis");
             System.out.println("Please wait");
             MultiMap<String, ClassProfile> depMap = Utils.findRelations(
                          sourceJarPath, target);
             Set<ClassProfile> uniques = Utils.toUniquesClassProfiles(depMap);

             System.out.println("\n**Quick survey result:**");
             System.out
                          .println("Total of referenced
classes(concrete,abstract,interfaces) by "
                                       + jar + ": " + uniques.size() + "\n");

             /* Tree build */
             System.out.println("Building dependencies tree\nPlease wait");
             TreeNode<?> depTree = Utils.buildDepTree(depMap, sourceJarPath);
             System.out.println("\n\n**Deep survey:**");
             System.out.println(Utils.printTree(depTree));

             /* Calculate result */
             CalculatorService cs = new CalculatorService();
             return cs.calcDepFactor(target, depTree, uniques.size());
       }

       private static Wini loadSetup() throws InvalidFileFormatException,
                    IOException {
             File file = new File(Constants.setup_file);
             Wini ini = null;
             if (!file.exists()) {
                    PrintWriter writer = new PrintWriter(Constants.setup_file,
"UTF-8");
                    writer.println(Constants.name + " " + Constants.version);
                    writer.println("\n");
                    writer.println("# Windows path example: c:\\Temp");
                    writer.println("# Linux path example: /tmp");
                    writer.println("# source filename example: hibernate-core-
4.3.10.Final.jar");
                    writer.println("# library filename example: dom4j-
1.6.1.jar");
                    writer.println("# namespace example: org.dom4j");
                    writer.println("");
                    writer.println("[source]");
                    writer.println("path = ");
                    writer.println("filename = ");
                    writer.println("");
                    writer.println("[library]");
                    writer.println("path = ");
                    writer.println("filename = ");
                    writer.println("namespace = ");
                    writer.close();
             } else {
                    ini = new Wini(file);
             }
             return ini;
       }

}
```

Clase Result

Esta clase encapsula todos los resultados del análisis.

```java
package org.trimatek.deep.model;

import java.text.DecimalFormat;

public class Result {

	public int classes, concrete, abstracts, interfaces, methods, fields;
	public int tarClasses, tarConcrete, tarAbstract, tarIntefaces,
tarMembers;
	private float totRate, conRate, absRate, intRate, memRate, depRate;
	private DecimalFormat f = new DecimalFormat("#.####");

	public Result(TargetProfile target) {
		tarClasses = target.getClasses().size();
		tarConcrete = target.getClassCount();
		tarAbstract = target.getAbstractCount();
		tarIntefaces = target.getInterfacesCount();
		tarMembers = target.getTotalPublicMembers();
	}

	private void updateResults() {
		int divider = 0;
		if (tarClasses != 0)
			totRate = classes / (float) tarClasses;
		if (tarConcrete != 0)
			conRate = concrete / (float) tarConcrete;
			if (conRate > 0) divider++;
		if (tarAbstract != 0)
			absRate = abstracts / (float) tarAbstract;
			if (absRate > 0) divider++;
		if (tarIntefaces != 0)
			intRate = interfaces / (float) tarIntefaces;
			if (intRate > 0) divider++;
		if (tarMembers != 0)
			memRate = (fields + methods) / (float) tarMembers;
			if (memRate > 0) divider++;
		depRate = (conRate + absRate + intRate + memRate) / divider;
	}

	public String toString() {
		updateResults();
		StringBuffer sb = new StringBuffer();
		sb.append("Tree summary:" + "\n");
		sb.append("Total of referenced classes: " + classes + "\n");
		sb.append("Concrete: " + concrete + "\n");
		sb.append("Abstract: " + abstracts + "\n");
		sb.append("Interfaces: " + interfaces + "\n");
		sb.append("Referenced fields: " + fields + "\n");
		sb.append("Referenced methods: " + methods + "\n");
		sb.append("\n**Results**" + "\n");
		sb.append("Total classes ratio: " + getTotRate() + "\n");
		sb.append(">>Concrete ratio: " + getConRate() + "\n");
		sb.append(">>Abstract ratio: " + getAbsRate() + "\n");
		sb.append(">>Interfaces ratio: " + getIntRate() + "\n");
		sb.append(">>Members ratio: " + getMemRate() + "\n");
		sb.append("\n");
		sb.append(">>>> Dependency ratio: " + getDepRate());
		sb.append("\n");
		return sb.toString();
	}

	public String getClasses() {
		return classes + "";
	}
	public String getConcrete() {
		return concrete + "";
	}

	public String getAbstracts() {
		return abstracts + "";
	}

	public String getInterfaces() {
		return interfaces + "";
	}

	public String getMethods() {
		return methods + "";
	}

	public String getFields() {
		return fields + "";
	}

	public String getTotRate() {
		return f.format(totRate);
	}

	public String getConRate() {
		return f.format(conRate);
	}

	public String getAbsRate() {
		return f.format(absRate);
	}

	public String getIntRate() {
		return f.format(intRate);
	}

	public String getMemRate() {
		return f.format(memRate);
	}

	public String getDepRate() {
		return f.format(depRate);
	}

}
```

Clase TreeNode

Es una implementación del patrón de diseño composite y permite definir una estructura de árbol para luego representarla en la interfaz web.

```
package org.trimatek.deep.model;

import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

public class TreeNode<T> implements Iterable<TreeNode<T>> {

    public T data;
    public TreeNode<T> parent;
    public List<TreeNode<T>> children;
```

```
	public boolean isRoot() {
		return parent == null;
	}

	public boolean isLeaf() {
		return children.size() == 0;
	}

	private List<TreeNode<T>> elementsIndex;

	public TreeNode(T data) {
		this.data = data;
		this.children = new LinkedList<TreeNode<T>>();
		this.elementsIndex = new LinkedList<TreeNode<T>>();
		this.elementsIndex.add(this);
	}

	public TreeNode<T> addChild(T child) {
		TreeNode<T> childNode = new TreeNode<T>(child);
		childNode.parent = this;
		this.children.add(childNode);
		this.registerChildForSearch(childNode);
		return childNode;
	}

	public int getLevel() {
		if (this.isRoot())
			return 0;
		else
			return parent.getLevel() + 1;
	}

	private void registerChildForSearch(TreeNode<T> node) {
		elementsIndex.add(node);
		if (parent != null)
			parent.registerChildForSearch(node);
	}

	public TreeNode<T> findTreeNode(Comparable<T> cmp) {
		for (TreeNode<T> element : this.elementsIndex) {
			T elData = element.data;
			if (cmp.compareTo(elData) == 0)
				return element;
		}

		return null;
	}

	@Override
	public String toString() {
		return data != null ? data.toString() : "[data null]";
	}

	@Override
	public Iterator<TreeNode<T>> iterator() {
		TreeNodeIter<T> iter = new TreeNodeIter<T>(this);
		return iter;
	}

}
```

Clase DeepService

Esta clase sincroniza las operaciones necesarias para interpretar y relevar atributos de las clases analizadas.

```
package org.trimatek.deep.service;

import java.io.File;
import java.util.Set;

import org.apache.commons.collections4.MultiMap;
import org.apache.log4j.Logger;
import org.trimatek.deep.model.AllResults;
import org.trimatek.deep.model.ClassProfile;
import org.trimatek.deep.model.Result;
import org.trimatek.deep.model.TargetProfile;
import org.trimatek.deep.model.TreeNode;
import org.trimatek.deep.utils.TreeUtils;
import org.trimatek.deep.utils.Utils;

public class DeepService {

	private TargetProfile targetProfile;
	private final static Logger log = Logger.getLogger(DeepService.class);

	public TargetProfile getTargetProfile() {
		return targetProfile;
	}

	public void setTargetProfile(TargetProfile targetProfile) {
		this.targetProfile = targetProfile;
	}

	public AllResults start(File sourceFile, File targetFile, String
threshold) throws Exception {
		StringBuffer sb = new StringBuffer();
		String msg;
		log.info("Start of analysis");
		/* Target */
		TargetService targetService = new TargetService();
		AllResults allResults = new AllResults();
		log.info("Loading target profile with threshold: " + threshold);
		targetProfile = targetService.loadTargetProfile(
				targetFile.getAbsolutePath(), threshold);
		allResults = targetProfile.loadResults(allResults);
		sb.append(targetProfile.toString());
		/* Source -> Target */
		msg = "Start of [" + sourceFile.getName() + " >> "
				+ targetProfile.getJarName() + "] analysis";
		log.info(msg);
		sb.append(msg);
		sb.append("\nPlease wait");
		MultiMap<String, ClassProfile> depMap = Utils.findRelations(
				sourceFile.getAbsolutePath(), targetProfile);
		Set<ClassProfile> uniques = Utils.toUniquesClassProfiles(depMap);
		allResults.setReferencedClasses(uniques.size());
		log.info("Total of referenced
classes(concrete,abstract,interfaces) by "
				+ sourceFile.getName() + ": " + uniques.size());
		log.info("Building dependencies tree");
		log.info(msg);
		sb.append(msg);
		/* Tree build */
		TreeNode<?> depTree = Utils.buildDepTree(depMap,
				sourceFile.getAbsolutePath());
		org.primefaces.model.TreeNode primeTree =
TreeUtils.buildDepTree(depMap,
				sourceFile.getAbsolutePath());
```

```
             allResults.setDependencyTree(primeTree);
             sb.append("**Deep survey:**");
             sb.append(Utils.printTree(depTree));
             /* Calculate result */
             log.info("Calculating results");
             CalculatorService cs = new CalculatorService();
             Result result = cs.calcDepFactor(targetProfile, depTree,
uniques.size());
             sb.append(result.toString());
             allResults.setResult(result);
             allResults.setFormattedResults(sb.toString());
             log.info("Analysis completed");
             return allResults;
      }

}
```

Clase ParserService

Genera un AST luego de decompilar la clase.

```
package org.trimatek.deep.service;

import java.util.List;

import org.antlr.v4.runtime.ANTLRInputStream;
import org.antlr.v4.runtime.CommonTokenStream;
import org.antlr.v4.runtime.Lexer;
import org.antlr.v4.runtime.tree.ParseTreeWalker;
import org.trimatek.deep.lexer.JavaLexer;
import org.trimatek.deep.lexer.JavaListenerImpl;
import org.trimatek.deep.lexer.JavaParser;

public class ParserService {

      private JavaListenerImpl listener = new JavaListenerImpl();
      private JavaParser parser;
      private CommonTokenStream tokens;
      private Lexer lexer;

      public List<String> parseStatementsExpressions(String sourceCode) {
             lexer = new JavaLexer(new ANTLRInputStream(sourceCode));
             tokens = new CommonTokenStream(lexer);
             parser = new JavaParser(tokens);
             ParseTreeWalker.DEFAULT.walk(listener, parser.compilationUnit());
             return listener.getStatements();
      }

}
```

Clase DecompilerService

Descompila archivos class.

```
package org.trimatek.deep.service;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.util.jar.JarFile;
import com.strobel.assembler.metadata.CompositeTypeLoader;
import com.strobel.assembler.metadata.JarTypeLoader;
import com.strobel.decompiler.Decompiler;
import com.strobel.decompiler.DecompilerSettings;
import com.strobel.decompiler.PlainTextOutput;

public class DecompilerService {

	private DecompilerSettings settings = DecompilerSettings
			.javaDefaults();
	private ByteArrayOutputStream out;

	public String decompile(String className, String jarPath) throws
Exception {
		out = new ByteArrayOutputStream();
		JarFile jarFile = new JarFile(new File(jarPath));
		CompositeTypeLoader c = new CompositeTypeLoader(new
JarTypeLoader(
				jarFile));
		settings.setTypeLoader(c);
		try (final OutputStreamWriter writer = new
OutputStreamWriter(out);) {
			Decompiler.decompile(className.replace(".", "/"),
					new PlainTextOutput(writer), settings);
		} catch (final IOException e) {
			throw new Exception(e.getMessage());
		}
		return out.toString();
	}

}
```

Clase CalculatorService

Analiza cada nodo del árbol y cuenta atributos.

```
package org.trimatek.deep.service;

import java.util.List;

import org.trimatek.deep.model.ClassProfile;
import org.trimatek.deep.model.MemberProfile;
import org.trimatek.deep.model.Result;
import org.trimatek.deep.model.TargetProfile;
import org.trimatek.deep.model.TreeNode;
import org.trimatek.deep.model.Type;
import org.trimatek.exception.ResultsException;

public class CalculatorService {

	private TargetProfile target;
	private Result results;

	public Result calcDepFactor(TargetProfile target, TreeNode<?> depTree,
int ctrl)
			throws IllegalArgumentException, IllegalAccessException,
			ResultsException {
		results = new Result(target);
		this.target = target;
		for (TreeNode node : depTree) {
			evalNode(node);
		}
		if (!validResults(results, target, ctrl)) {
			throw new ResultsException("Invalid results");
			}
			return results;
		}

		private void evalNode(TreeNode<?> node) {
			String nodeData = (String) node.data;
			if (target.containsClassProfile(nodeData)) {
				ClassProfile cp = target.removeClass(nodeData);
				results.classes++;
				if (cp.isClase())
					results.concrete++;
				if (cp.isAbstrac())
					results.abstracts++;
				if (cp.isInterfaz())
					results.interfaces++;
				target = loadMembers(cp, target);
			} else {
				if (node.parent != null) {
					String className = (String) node.parent.data;
					if (target.containsMemberProfile(className + "." +
node.data)) {
						MemberProfile mp =
target.removeMember(node.parent.data
								+ "." + nodeData);
						if (mp.isField())
							results.fields++;
						if (mp.isMethod())
							results.methods++;
					}
				}
			}
		}

		private TargetProfile loadMembers(ClassProfile classProfile,
				TargetProfile target) {
			List<String> fields = classProfile.getFields();
			for (String fieldName : fields) {
				MemberProfile mp = new
MemberProfile(classProfile.getClassName()
						+ "." + fieldName + Type.field.getSymbol());
				mp.setField(true);
				this.target.addMember(mp);
			}
			List<String> methods = classProfile.getMethods();
			for (String methodName : methods) {
				MemberProfile mp = new
MemberProfile(classProfile.getClassName()
						+ "." + methodName +
Type.method.getSymbol());
				mp.setMethod(true);
				this.target.addMember(mp);
			}
			return target;
		}

		private boolean validResults(Result results, TargetProfile target, int
ctrl) {
			if (this.results.classes > ctrl)
				return false;
			return true;
		}

}
```

Clase TargetService

Itera sobre las clases de la biblioteca y selecciona las que cumplen con el criterio del estudio (que sean de visibilidad pública).

```
package org.trimatek.deep.service;

import java.io.IOException;
import java.util.List;

import org.apache.bcel.classfile.ClassParser;
import org.apache.bcel.classfile.JavaClass;
import org.trimatek.deep.model.ClassProfile;
import org.trimatek.deep.model.TargetProfile;
import org.trimatek.deep.utils.ClassUtils;
import org.trimatek.deep.utils.JarUtils;

public class TargetService {

	public TargetProfile loadTargetProfile(String path, String mask)
			throws ClassNotFoundException, IOException {
		List<String> classes = JarUtils.listClasses(path);
		TargetProfile target = new TargetProfile(path,mask);
		for (String className : classes) {
			if (ClassUtils.isInsidePackage(className,
					target.NAMESPACE.split("\\."))) {
				JavaClass jclass = new ClassParser(path,
className.replace(".",
						"/") + ".class").parse();
				if (jclass.isPublic()) {
					ClassProfile cp = new
ClassProfile(className);
					cp.setClassName(className);
					cp =
ClassUtils.loaddAllPublicResources(jclass, cp);
					target.addClass(cp);
				}
			}
		}
		return target;
	}

}
```

Clase ClassVisitorService

Es la subclase de la clase Visitor necesaria para recorrer los nodos del AST generado por el parser creado con ANTLR.

```
package org.trimatek.deep.service;

import java.util.ArrayList;
import java.util.List;

import org.apache.bcel.classfile.ClassParser;
import org.apache.bcel.classfile.ConstantPool;
import org.apache.bcel.classfile.ConstantUtf8;
import org.apache.bcel.classfile.DescendingVisitor;
import org.apache.bcel.classfile.EmptyVisitor;
import org.apache.bcel.classfile.JavaClass;

public class ClassVisitorService {

	public List<String> listReferences(String jarPath, String className)
			throws Exception {
```

```
             ClassParser parser = new ClassParser(jarPath,
className.replace(".",
                         "/") + ".class");
             JavaClass javaClass = parser.parse();
             Visitor visitor = new Visitor(javaClass);
             DescendingVisitor classWalker = new DescendingVisitor(javaClass,
                         visitor);
             classWalker.visit();
             return visitor.getClasses();
      }
}

class Visitor extends EmptyVisitor {

      private JavaClass javaClass;
      private List<String> classes = new ArrayList<String>();

      public List<String> getClasses() {
             return classes;
      }

      public Visitor(JavaClass javaClass) {
             this.javaClass = javaClass;
      }

      public void visitConstantUtf8(ConstantUtf8 obj) {
             ConstantPool cp = javaClass.getConstantPool();
             String s = obj.getBytes();
             classes.add(s);
      }

}
```

Clase ClassUtils

Agrupa métodos estáticos para analizar características de las clases.

```
package org.trimatek.deep.utils;

import java.io.File;
import java.lang.reflect.Modifier;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLClassLoader;

import org.apache.bcel.classfile.Field;
import org.apache.bcel.classfile.JavaClass;
import org.apache.bcel.classfile.Method;
import org.trimatek.deep.model.ClassProfile;

public class ClassUtils {

      public static ClassLoader loadJar(String path) throws
MalformedURLException {
             URL url = new File(path).toURI().toURL();
             URL[] urls = new URL[] { url };
             return new URLClassLoader(urls);
      }

      public static ClassProfile loaddAllPublicResources(JavaClass clase,
                   ClassProfile cp) {
             cp = loadAllPublicMethods(clase, cp);
             cp = loadAllPublicFields(clase, cp);
```

```
		if (clase.isClass() && !clase.isAbstract() &&
!clase.isInterface()) {
			cp.setClase(clase.isClass());
		} else if (clase.isAbstract() && !clase.isInterface()) {
			cp.setAbstrac(clase.isAbstract());
		} else if (clase.isInterface()) {
			cp.setInterfaz(clase.isInterface());
		}
		return cp;
	}

	private static ClassProfile loadAllPublicMethods(JavaClass jc,
			ClassProfile cp) {
		for (Method m : jc.getMethods()) {
			if (m.isPublic()) {
				cp.addMethod(m.getName());
			}
		}
		return cp;
	}

	private static ClassProfile loadAllPublicFields(JavaClass jc,
			ClassProfile cp) {
		for (Field f : jc.getFields()) {
			if (f.isPublic()) {
				cp.addField(f.getName());
			}
		}
		return cp;
	}

	public static boolean isInsidePackage(String className, String[] pcks) {
		String[] fqn = className.split("\\.");
		int i = 0;
		for (String pck : pcks) {
			if (pck.equals(fqn[i])) {
				i++;
			} else {
				return Boolean.FALSE;
			}
		}
		return Boolean.TRUE;
	}

}
```

Clase FileUtils

Agrupa métodos estáticos para acceder a archivos.

```
package org.trimatek.deep.utils;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.List;
import java.util.jar.JarEntry;
import java.util.jar.JarFile;

import org.antlr.v4.runtime.ANTLRFileStream;
import org.antlr.v4.runtime.CommonTokenStream;
import org.antlr.v4.runtime.Lexer;
import org.antlr.v4.runtime.Token;
import org.primefaces.model.UploadedFile;
import org.trimatek.deep.lexer.JavaLexer;
import org.trimatek.deep.model.ClassProfile;

public class FileUtils {

	public static List<File> loadFiles(File[] files, List<File> fList) {
		for (File file : files) {
			if (file.isDirectory()) {
				fList = loadFiles(file.listFiles(), fList);
			} else {
				fList.add(file);
			}
		}
		return fList;
	}

	private static CommonTokenStream toTokens(File file,
			List<ClassProfile> classes) throws IOException {
		Lexer lexer = new JavaLexer(new ANTLRFileStream(file.getName()));
		CommonTokenStream tokens = new CommonTokenStream(lexer);

		for (Token token = lexer.nextToken(); token.getType() !=
Token.EOF; token = lexer
				.nextToken()) {
			// TODO to be completed
			System.out.println(token.getText());
		}

		return null;
	}

	public static boolean hasExtension(UploadedFile file, String[]
extensions) {
		String fileExt = file.getFileName().substring(
				file.getFileName().lastIndexOf(".") + 1);
		for (String ext : extensions) {
			if (fileExt.equals(ext))
				return true;
		}
		return false;
	}

	public static List<String> loadDirs(File file) throws IOException {
		List<String> dList = new ArrayList<String>();
		Enumeration<JarEntry> entries = new JarFile(file).entries();
		while (entries.hasMoreElements()) {
			JarEntry je = entries.nextElement();
			if (je.isDirectory()) {
				dList.add(je.getName());
			}
		}
		return dList;
	}

}
```

Clase JarUtils

Agrupa métodos estáticos específicos para acceder a archivos Jar según los requerimientos de Deep.

```
package org.trimatek.deep.utils;
```

```
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.jar.JarEntry;
import java.util.jar.JarInputStream;

public class JarUtils {

	public static List<String> listClasses(String path) throws IOException {
		List<String> classes = new ArrayList<String>();
		JarInputStream jarFile = new JarInputStream(new
FileInputStream(path));
		JarEntry jarEntry;
		while (true) {
			jarEntry = jarFile.getNextJarEntry();
			if (jarEntry == null) {
				break;
			}
			if ((jarEntry.getName().endsWith(".class"))) {
				String className =
jarEntry.getName().replaceAll("/", "\\.");
				classes.add(className.substring(0,
className.lastIndexOf('.')));
			}
		}
		jarFile.close();
		return classes;
	}

	public static String getJarName(String jarPath) {
		return jarPath.substring(jarPath.lastIndexOf("\\") + 1,
				jarPath.lastIndexOf("."));
	}

}
```

Clase TreeUtils

Métodos específicos para construir e interactuar con el árbol de clases creados por Deep.

```
https://www.tuenti.com.ar/solicita-chip-sim-
gratis/?token=e5711cc0f814942f42b181256f17e5fd0349524d}
		}
		return root;
	}

}
```

Clase Utils

Agrupa métodos estáticos de uso general en Deep.

```
package org.trimatek.deep.utils;

import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

import org.apache.commons.collections4.MultiMap;
import org.apache.commons.collections4.map.MultiValueMap;
import org.trimatek.deep.model.ClassProfile;
import org.trimatek.deep.model.TargetProfile;
import org.trimatek.deep.model.TreeNode;
import org.trimatek.deep.model.Type;
import org.trimatek.deep.service.ClassVisitorService;
import org.trimatek.deep.service.DecompilerService;
import org.trimatek.deep.service.ParserService;

public class Utils {

	private static DecompilerService dc = new DecompilerService();
	private static ParserService ps = new ParserService();

	public static Set<ClassProfile> toUniquesClassProfiles(
			MultiMap<String, ClassProfile> fileClassMap) {
		Set<ClassProfile> uniques = new HashSet<ClassProfile>();
		Set<String> keySet = fileClassMap.keySet();
		Iterator<String> keyIterator = keySet.iterator();
		while (keyIterator.hasNext()) {
			String className = (String) keyIterator.next();
			List<ClassProfile> classes = (List<ClassProfile>)
fileClassMap
					.get(className);
			uniques.addAll(toUniques(classes));
		}
		return uniques;
	}

	static Set<ClassProfile> toUniques(List<ClassProfile> classes) {
		Set<ClassProfile> uniques = new HashSet<ClassProfile>();
		for (ClassProfile cp : classes) {
			uniques.add(cp);
		}
		return uniques;
	}

	public static MultiMap<String, ClassProfile> findRelations(
			String sourceJarPath, TargetProfile target) throws
Exception {
		MultiMap<String, ClassProfile> map = new MultiValueMap<String,
ClassProfile>();
		List<String> classes = JarUtils.listClasses(sourceJarPath);
		ClassVisitorService cv = new ClassVisitorService();
		for (String className : classes) {
			List<String> refList = cv.listReferences(sourceJarPath,
className);
			for (String ref : refList) {
				for (ClassProfile cp : target.getClasses()) {
					if (ref.replace("/",
".").contains(cp.getClassName())) {
						map.put(className, cp);
					}
				}
			}
		}
		return map;
	}

	public static TreeNode<?> buildDepTree(
			MultiMap<String, ClassProfile> sourceTargetmap, String
targetJarPath)
			throws Exception {
		TreeNode root = new TreeNode(JarUtils.getJarName(targetJarPath)
				+ ".jar");
		TreeNode classNode;
		String sourceCode;
		List<String> statements;
		Set<String> keySet = sourceTargetmap.keySet();
		Iterator<String> keyIterator = keySet.iterator();
		while (keyIterator.hasNext()) {
			String className = (String) keyIterator.next();
			classNode = root.addChild(className);
			List<ClassProfile> classes = (List<ClassProfile>)
sourceTargetmap
					.get(className);
			sourceCode = dc.decompile(className, targetJarPath);
			statements = ps.parseStatementsExpressions(sourceCode);
			getNodes(classes, statements, classNode);
		}
		return root;
	}

	private static TreeNode<?> getNodes(List<ClassProfile> classes,
			List<String> statements, TreeNode root) {
		TreeNode classNode;
		TreeNode memberNode;
		for (ClassProfile cp : toUniques(classes)) {
			classNode = root.addChild(cp.getClassName());
			for (String statement : statements) {
				for (String field : cp.getFields()) {
					addMember(statement, field, Type.field,
classNode);
				}
				for (String method : cp.getMethods()) {
					addMember(statement, method, Type.method,
classNode);
				}
			}
		}
		return root;
	}

	private static TreeNode addMember(String statement, String field,
			Type type, TreeNode classNode) {
		if (statement.contains(field)) {
			if (classNode.findTreeNode(field + type.getSymbol()) ==
null) {
				classNode.addChild(field + type.getSymbol());
			}
		}
		return classNode;
	}

	public static String getSlash() {
		return System.getProperty("os.name").startsWith("W") ? "\\" :
"/";
	}

	public static String printTree(TreeNode<?> depTree){
		StringBuffer sb = new StringBuffer();
		for (TreeNode treeNode : depTree) {
			String indent = createIndent(treeNode.getLevel());
			System.out.println(indent + treeNode.data);
			sb.append(indent + treeNode.data + "\r");
		}
		return sb.toString();
	}

	private static String createIndent(int depth) {
		StringBuilder sb = new StringBuilder();
		for (int i = 0; i < depth; i++) {
			sb.append("  ");
		}
		return sb.toString();
```

```
            }

  }
```