

API DE INTEGRAÇÃO

RESPONSYS – DATABASE

Software para automatização dos processos envolvendo
Oracle Database e Oracle Responsys

R2

ADIN
2019

1 VISÃO GERAL

O QUE É

O presente software tem por finalidade automatizar o processo de captação de dados do Oracle Database e enviá-los ao Oracle Responsys, por intermédio da tecnologia REST.


O mesmo foi escrito na linguagem de programação Java, lê e serializa arquivos JSON e emite requisições via protocolo HTTP.

Seu uso é bem simples e pode ser aplicável a qualquer projeto que envolva o Database em conjunto com Responsys.

COMO USAR

Seu uso, como dito antes, é bastante simples. Basta que o usuário entenda o básico de arquivos JSON para poder ajustar os parâmetros de busca da API.

Primeiro, você deve localizar os arquivos JSON

 file.json	01/03/2019 17:07	Arquivo JSON	1 KB
 jsonOracleDatabase.json	06/03/2019 13:37	Arquivo JSON	1 KB
 jsonResponsys.json	06/03/2019 13:38	Arquivo JSON	1 KB

Explicando:

- file.json : é o arquivo modelo de JSON requerido pelo Responsys, nele há os campos utilizados para as definições dos dados no próprio Responsys, por exemplo *"matchColumnName1" : "CUSTOMER_ID_"*.
- jsonOracleDatabase.json : é o arquivo que define os parâmetros de conexão, por exemplo *"user" : "system"* e uso do Database, por exemplo *"query" : "SELECT * FROM CRM.CLIENTE"*.
- JsonResponsys.json : é o arquivo que define os parâmetros de conexão com o Responsys, por exemplo *"user" : "adin_api"* e há um objeto que serve como referenciador de trocas para o usuário ajustar campos do Database que estejam com nomes diferentes no Responsys, assim o software aplicará uma alteração nos nomes dos campos mas conservando seus registros.

A explicação sobre a manipulação dos dados mais detalhada dos dados, será feita posteriormente.

Tendo feito isso, basta iniciar o software através de um **Console**. Independentemente do sistema operacional, o processo é o mesmo. Caso você esteja utilizando uma distribuição Windows (Windows 7,8,10,...) basta abrir o Prompt de Comando, caso esteja usando uma distribuição Linux (Ubuntu, Fedora, MAC OS,...) basta abrir o Terminal. Digite o comando **java -jar "caminho do arquivo.jar"**. Ao apertar ENTER, ele já dará início ao software e todas as mensagens emitidas pelo software, tanto *debug* quanto os avisos, aparecerão no console.

Observação: é obrigatório ter o Java instalado na máquina!

2 ARQUIVOS JSON

FILE

Como dito na visão geral, este arquivo é o modelo a ser seguido para o envio do JSON para o Responsys. Os campos dele deverem ser inalterados, porém os registros podem ser manipulados sem qualquer problema, respeitando os limites dos tipos de dados.

```
1  {
2      "recordData" : {
3          "fieldNames" : [],
4          "records" : [
5              []
6          ],
7          "mapTemplateName" : null
8      },
9      "mergeRule" : {
10         "htmlValue" : "H",
11         "optinValue" : "I",
12         "textValue" : "T",
13         "insertOnNoMatch" : true,
14         "updateOnMatch" : "REPLACE_ALL",
15         "matchColumnName1" : "CUSTOMER_ID_",
16         "matchColumnName2" : null,
17         "matchOperator" : "NONE",
18         "optoutValue" : "O",
19         "rejectRecordIfChannelEmpty" : null,
20         "defaultPermissionStatus" : "OPTIN"
21     }
22 }
```

- RecordData : é um objeto que contém os dados a serem enviados ao Responsys;
- fieldNames : array de String's que armazena quais campos serão manipulados no Responsys, aqui ele se encontra vazio apenas para o modelo mas o software preenche esses campos com os valores correspondentes;
- records : array de String's que armazena quais registros serão inseridos no Responsys, aqui ele se encontra vazio apenas para o modelo mas o software preenche esses campos com os valores correspondentes;
- mapTemplateName : String que contém o nome do template a ser seguido, o padrão usado é null;
- mergeRule : objeto que contém os dados de formatação do envio, são definidos de acordo com as normas do Responsys;
- htmlValue : String que define o formato de dados pretendido para o e-mail, por exemplo "H" representa a preferência pelo formato HTML;

- "optinValue" : String que define o opt-in dos dados, por exemplo "I" representa que os dados devem ser escritos com opt-in;
- "textValue" : String que define o formato de dados pretendido para o e-mail, por exemplo "T" representa a preferência pelo formato Texto;
- "insertOnNoMatch" : Boolean que define o que ocorrerá com um dado caso não haja um registro igual a ele, "true" para inserir o dado e "false" para não inserir;
- "updateOnMatch" : String que define o que ocorrerá com um dado caso haja um registro igual a ele, "REPLACE_ALL" para substituir o dado e "NO_UPDATE" para não substituir;
- "matchColumnName1" : String que define qual campo será usado na busca por registros idênticos no Responsys;
- "matchColumnName2" : (opcional) String que define outro campo a ser usado na busca por registros idênticos no Responsys ;
- "matchOperator" : String que define como será feita a comparação dos campos no momento da busca, pode ser "NONE" para ignorar o segundo campo, "AND" para avaliar se ambos os campos possuem registros idênticos no Responsys e "OR" para avaliar se um dos campos possui registros idênticos no Responsys;
- "optoutValue" : String que define o opt-out dos dados, por exemplo "O" representa que os dados devem ser escritos com opt-out;
- "rejectRecordIfChannelEmpty" : String que define quando um envio deve ser rejeitado caso um campo seja vazio, "E" para o canal de email, "P" para push e "M" para SMS;
- "defaultPermissionStatus" : String que define qual modo padrão, "OPTIN" para opt-in e "OPTOUT" para opt-out.

JSONRESPONSYS

Como dito na visão geral, este arquivo é o responsável por definir a conexão e alguns critérios usados pelo Responsys.

Conexão talvez seja um termo errôneo, tendo em vista que não há uma conexão de fato com o Responsys, mas apenas uma autenticação com chave de acesso.

```
{
  "user" : "adin_api",
  "password" : "Adin@2019_",
  "auth_type" : "password",
  "key": "",
  "urlPost" : "https://rest001.rsys9.net/rest/api/v1.3/lists/TESTE_API/members",
  "urlConnection" : "https://login.rsys9.net/rest/api/v1.3/auth/token",
  "formatter" : {
    "databaseFields" : ["CODIGO_CLIENTE", "EMAIL", "PRIMIERO_NOME"],
    "responsysFields" : ["CUSTOMER_ID_", "EMAIL_ADDRESS_", "NOME"]
  }
}
```

- user : String que define o nome de usuário que acessará o Responsys;
- password : String que define a senha utilizada por este usuário;
- auth_type : String que define qual tipo de autenticação será usada, "password" instrui que a autenticação será feita por usuário + senha;
- key : String que armazena a chave de acesso provisória concedida pelo Responsys, este campo deve sempre permanecer vazio pois a chave é obtida através do Responsys e inserida no corpo do programa ao decorrer dos envios;
- urlPost : String que armazena o endereço URL de onde será feito o envio;
- urlConnection : String que armazena o endereço URL de onde será feita a autenticação;
- formatter : Objeto que armazena duas Strings para formatação dos nomes dos campos;
- databaseFields : Array de String que armazena os nomes dos campos do Database que devem ser substituídos ao realizar o envio;
- responsysFields : Array de String que armazena os nomes dos campos do Responsys que serão inseridos no envio no lugar dos campos do Database.

JSONORACLEDATABASE

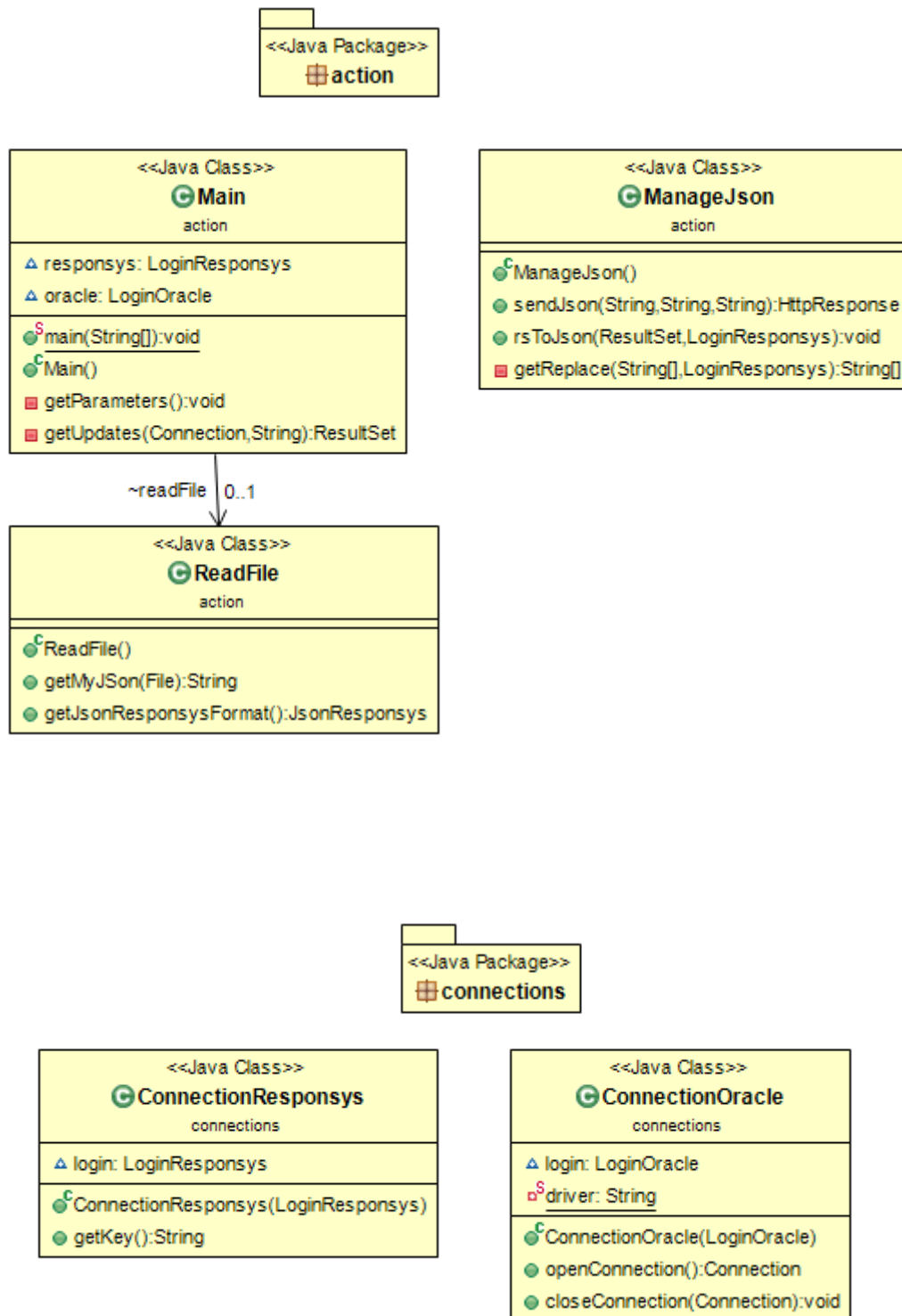
Como dito na visão geral, este arquivo é responsável por definir a conexão com o Database.

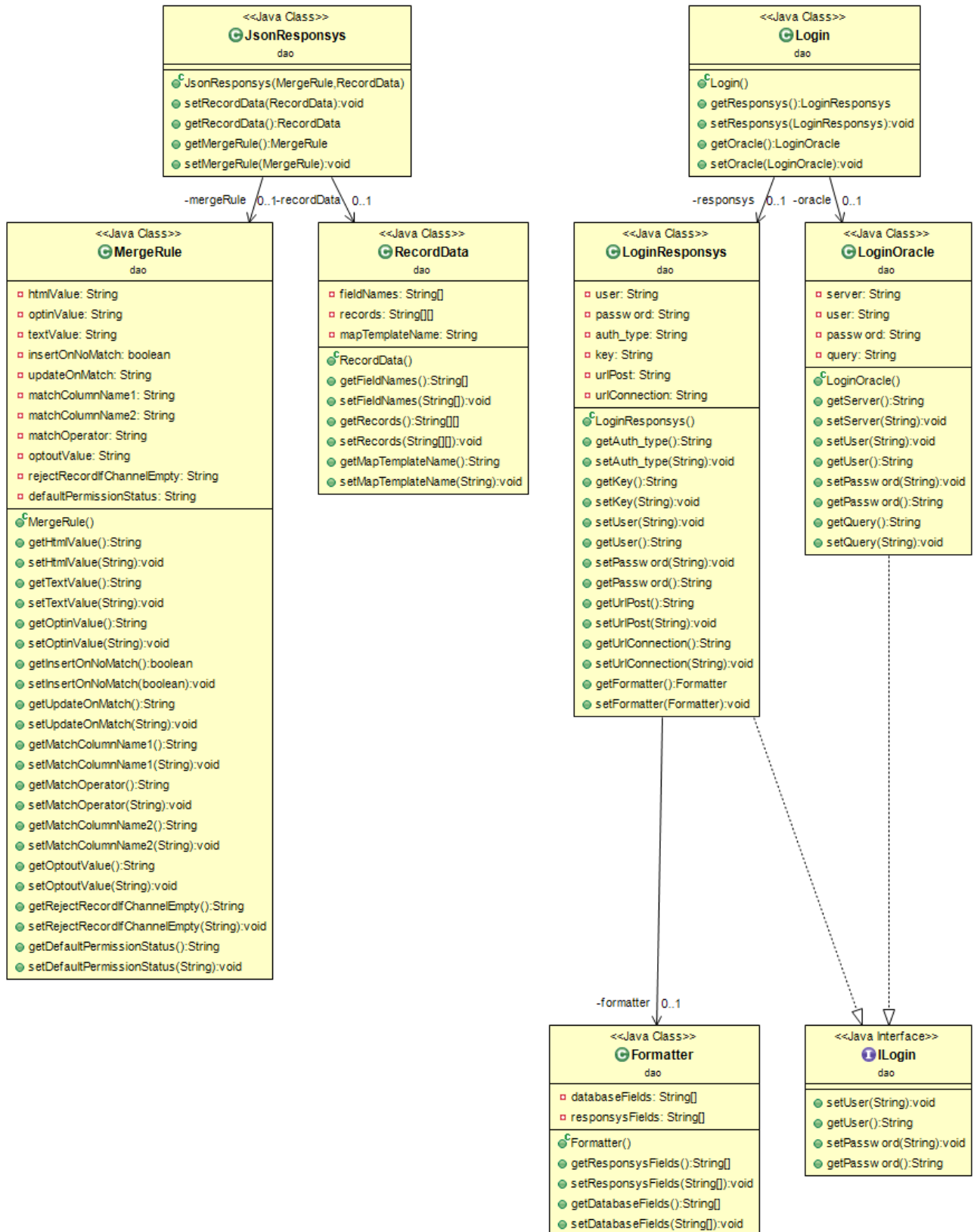
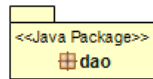
```
{
  "server" : "144.22.100.60",
  "user" : "system",
  "password" : "Bcd7z3#m",
  "query" : "SELECT CODIGO_CLIENTE,EMAIL,PRIMIERO_NOME FROM CRM.CLIENTE where rownum <= 50"
}
```

- server : String que armazena o endereço IP do servidor do database;
- user : String que armazena o usuário utilizado para acessar o database;
- password : String que armazena a senha utilizada para acessar o database;
- query : String que armazena a query SQL utilizada para fazer a consulta no database.

3 PROGRAMA JAVA

DIAGRAMA DE CLASSES





COMO FUNCIONA

Pacote connections: A ideia principal é a conexão entre ambos os serviços, portanto há a classe ConnectionOracle que faz a conexão com o banco de dados e a classe ConnectionResponsys que faz a conexão com o Responsys e recebe a chave de autenticação emitida pelo mesmo, que será usada para o envio de dados.

Pacote action: Promove toda a “ação” do programa, chama as classes que fazem conexão e aquelas que representam os dados em forma de objeto (DAO). Conta com a classe Main que rege todo o programa, a mesma chama a classe ReadFile para ler os arquivos JSON com as configurações do programa, e também a classe ManageJson que trata de todos os objetos formados a partir dos dados provindos dos arquivos JSON lidos na ReadFile, ManageJson também faz a conversão dos dados do database para o JSON que será gerado e enviado ao Responsys posteriormente, também cabe a ela montar o escopo da requisição alocando o JSON em seu corpo.

Pacote dao: É responsável pela representação de dados do banco em forma de objetos Java para melhor tratativa, faz o mesmo com os dados provindos dos arquivos JSON lidos, por exemplo, as classes MergeRule e RecordData. Login é a classe que representa um tipo de login, portanto ela possui 2 objetos, são eles: classe LoginOracle e LoginResponsys, elas representam e manuseiam as conexões com o Database e Responsys, respectivamente, ambas implementam a interface Login que possui os métodos necessários para qualquer conexão. JsonResponsys representa o JSON que será gerado e enviado ao Responsys, por isso obedece a estrutura padrão do Responsys, com os dados e objetos necessários. Formatter representa os campos que vieram do banco e serão submetidos ao Responsys, devido à diferença dos nomes dos campos do Responsys e do Database, há dois arrays de String, um contendo os nomes dos campos do banco e outro contendo os nomes dos campos do Responsys, estes serão substituídos quando o JSON for formado.

Librarys: foram usadas algumas bibliotecas para o software, para a conversão de arquivos JSON foi usada a Gson 2.8, para o manuseio do protocolo HTTP foi usada a HttpClient 4.5 e para a conexão com o Database foi usada a OJDBC 8. Algumas outras foram usadas para formatação dos dados e correção de falhas.

CÓDIGO FONTE

Main

```
package action;

/**
 * @author R2
 * @version 1.0
 * @lastUpdate 03/2019
 * Thanks GOD*/

import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import com.google.gson.Gson;

import connections.ConnectionOracle;
import dao.Login;
import dao.LoginOracle;
import dao.LoginResponsys;

public class Main{
    Login login;
    ReadFile readFile;

    public static void main(String[] args) {
        try {
            while(true) {
```

```

        new Main();

        long segundo = 1000, minuto = 60*segundo, hora = 60*minuto;
        Thread.sleep(hora);
    }
}

catch(InterruptedException e) {
    e.printStackTrace();
}

}

public Main() {
    try {
        System.out.print("INICIANDO SOFTWARE");
        for(int i = 0; i < 6;i++) {
            System.out.print(".");
            Thread.sleep(500);
        }
        System.out.println();
        readFile = new ReadFile();
        login = getParameters();
        Connection con = new
ConnectionOracle(login.getOracle()).openConnection();
        System.out.println("CONEXÃO ESTABELECIDADA COM O BANCO");
        ResultSet rs = getUpdates(con,login.getOracle().getQuery());
        System.out.println("DADOS OBTIDOS DO BANCO");
        new ManageJson().rsToJson(rs,login.getResponys());
        con.close();
    }
    catch(SQLException e) {
        e.printStackTrace();
    }
    catch (ClassNotFoundException e) {

```

```

        e.printStackTrace();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    catch (InterruptedException e) {
        e.printStackTrace();
    }
}

private Login getParameters() throws IOException {
    Gson gson = new Gson();
    Login login = new Login();
    login.setOracle(gson.fromJson(readFile.getMyJSoN(new
File(System.getProperty("user.home")+"//Desktop//my_api//jsonOracleDatabase.json")),
LoginOracle.class));
    login.setResponsys(gson.fromJson(readFile.getMyJSoN(new
File(System.getProperty("user.home")+"//Desktop//my_api//jsonResponsys.json")),
LoginResponsys.class));
    return login;
}

private ResultSet getUpdates(Connection con,String sql) throws SQLException{
    PreparedStatement ps = con.prepareStatement(sql);
    return ps.executeQuery();
}
}

```

ManageJson

```

package action;

import java.io.File;
import java.io.IOException;
import java.sql.ResultSet;

```

```
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.ArrayList;
```

```
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.HttpClientBuilder;
```

```
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonIOException;
import com.google.gson.JsonSyntaxException;
```

```
import connections.ConnectionResponsys;
import dao.JsonResponsys;
import dao.LoginResponsys;
```

```
public class ManageJson {
```

```
    public HttpResponse sendJson(String textJson,String urlPost, String key) throws
    ClientProtocolException, IOException {
```

```
        HttpClient httpClient = HttpClientBuilder.create().build();
        HttpPost post = new HttpPost(urlPost);
        post.setHeader("Authorization",key);
        post.setHeader("Charset","UTF-8");
        post.setHeader("Content-type", "application/json");
        HttpEntity body = new StringEntity(textJson,"UTF-8");
        post.setEntity(body);
```

```
        return httpClient.execute(post);
    }
}
```

```
public void rsToJson(ResultSet rs, LoginResponsys loginResponsys) throws SQLException,
JsonSyntaxException, JsonIOException, ClientProtocolException, IOException,
InterruptedException {
    JsonResponsys responsys = new ReadFile().getJsonResponsysFormat();
    ResultSetMetaData meta = rs.getMetaData();
    int columns = meta.getColumnCount();
    String[] column_names_array = new String[columns];
    for(int i = 0; i < columns; i++) {
        column_names_array[i] = meta.getColumnName(i+1);
    }
    ArrayList<String[]> arrayOfarray = new ArrayList<String[]>();
    int k = 1, m = 0;
    while (rs.next()) {
        k++;
        if(k < 200) {
            String[] records = new String[columns];
            for(int i = 0; i < columns; i++) {
                if(rs.getString(column_names_array[i]) == null)
                    records[i] = "";
                else
                    records[i] = rs.getString(column_names_array[i]);
            }
            arrayOfarray.add(records);
        }
    }
    else {
        String[][] arrayRecord = new String[arrayOfarray.size()][columns];
        for(int i = 0; i < arrayOfarray.size(); i++) {
            for(int j = 0; j < columns; j++) {
                arrayRecord[i][j] = arrayOfarray.get(i)[j];
            }
        }
    }
}
```

```
    }  
}
```

```
responsys.getRecordData().setFieldNames(getReplace(column_names_array,loginResponsys));
```

```
    responsys.getRecordData().setRecords(arrayRecord);
```

```
    sendJson(new GsonBuilder().serializeNulls().create().toJson(responsys),  
loginResponsys.getUrlPost(), new ConnectionResponsys(new Gson().fromJson(new  
ReadFile().getMyJJson(new  
File(System.getProperty("user.home")+"//Desktop//my_api//jsonResponsys.json")),  
LoginResponsys.class)).getKey());
```

```
    k = 0;
```

```
    arrayOfarray.clear();
```

```
    try {
```

```
        Thread.sleep(10000);
```

```
    }
```

```
    catch(InterruptedException e) {
```

```
        e.printStackTrace();
```

```
    }
```

```
}
```

```
    m++;
```

```
}
```

```
if(!arrayOfarray.isEmpty()) {
```

```
    String[][] arrayRecord = new String[arrayOfarray.size()][columns];
```

```
    for(int i = 0; i < arrayOfarray.size();i++) {
```

```
        for(int j = 0; j < columns;j++) {
```

```
            arrayRecord[i][j] = arrayOfarray.get(i)[j];
```

```
        }
```

```
    }
```

```
responsys.getRecordData().setFieldNames(getReplace(column_names_array,loginResponsys));
```

```
    responsys.getRecordData().setRecords(arrayRecord);
```

```
    sendJson(new GsonBuilder().serializeNulls().create().toJson(responsys),  
loginResponsys.getUrlPost(), new ConnectionResponsys(new Gson().fromJson(new
```



```

ReadFile().getMyJson(new
File(System.getProperty("user.home")+"//Desktop//my_api//jsonResponsys.json")),
LoginResponsys.class)).getKey());

```

```

        k = 0;

        arrayOfarray.clear();
    }
}

```

```

private String[] getReplace(String[] column_names_array, LoginResponsys responsys) {
    String[] aux = column_names_array.clone();
    for(int i = 0; i < aux.length;i++) {
        String coluna = aux[i];
        for(int j = 0; j < responsys.getFormatter().getDatabaseFields().length;j++) {
            if(coluna.equals(responsys.getFormatter().getDatabaseFields()[j])) {
                aux[i] = responsys.getFormatter().getResponsesysFields()[j];
            }
        }
    }
    return aux;
}

```

```

}

```

```

}

```

ReadFile

```

package action;

```

```

import java.io.BufferedReader;

```

```

import java.io.File;

```

```

import java.io.FileNotFoundException;

```

```

import java.io.FileReader;

```

```

import java.io.IOException;

```

```

import com.google.gson.Gson;
import com.google.gson.JsonIOException;
import com.google.gson.JsonSyntaxException;

import dao.JsonResponsys;

public class ReadFile {

    public String getMyJson(File file) throws IOException {

        FileReader reader = new FileReader(file);
        BufferedReader buff = new BufferedReader(reader);
        StringBuilder build = new StringBuilder();
        while(buff.ready()) {
            build.append(buff.readLine());
        }
        buff.close();
        reader.close();
        return build.toString();
    }

    public JsonResponsys getJsonResponsysFormat() throws JsonSyntaxException,
    JsonIOException, FileNotFoundException {

        File file = new
File(System.getProperty("user.home")+"//Desktop//my_api//file.json");

        return new Gson().fromJson(new FileReader(file), JsonResponsys.class);
    }
}

```

ConnectionOracle

```

package connections;

import java.sql.Connection;
import java.sql.DriverManager;

```

```
import java.sql.SQLException;
```

```
import dao.LoginOracle;
```

```
public class ConnectionOracle {
```

```
    LoginOracle login;
```

```
    private static String driver;
```

```
    public ConnectionOracle(LoginOracle login){
```

```
        driver = "oracle.jdbc.driver.OracleDriver";
```

```
        this.login = login;
```

```
    }
```

```
    public Connection openConnection() throws SQLException, ClassNotFoundException{
```

```
        Class.forName(driver);
```

```
        return DriverManager.getConnection("jdbc:oracle:thin:@" + login.getServer() + ":1521:orcl", login.getUser(), login.getPassword());
```

```
    }
```

```
    public void closeConnection(Connection con) throws SQLException {
```

```
        con.close();
```

```
    }
```

```
}
```

ConnectionResponsys

```
package connections;
```

```
import java.io.IOException;
```

```
import org.apache.http.client.ClientProtocolException;
```

```
import org.apache.http.client.HttpClient;
```

```
import org.apache.http.client.methods.HttpPost;
```

```
import org.apache.http.entity.StringEntity;
```

```

import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.util.EntityUtils;

import com.google.gson.Gson;
import com.google.gson.JsonObject;

import dao.LoginResponsys;

public class ConnectionResponsys {
    LoginResponsys login;

    public ConnectionResponsys(LoginResponsys login) {
        this.login = login;
    }

    public String getKey() throws ClientProtocolException, IOException {
        HttpClient httpClient = HttpClientBuilder.create().build();
        HttpPost post = new HttpPost(login.getUrlConnection());
        String body = "user_name="+login.getUser()+"&password="+login.getPassword()
        + "&auth_type="+login.getAuth_type();
        StringEntity params = new StringEntity(body);
        post.addHeader("content-type", "application/x-www-form-urlencoded");
        post.setEntity(params);
        JsonObject ob = new
        Gson().fromJson(EntityUtils.toString(httpClient.execute(post).getEntity()), JsonObject.class);
        return ob.get("authToken").getAsString();
    }
}

```

Formatter

```

package dao;

public class Formatter {

```

```

private String[] databaseFields, responsysFields;

public String[] getResponsysFields() {
    return responsysFields;
}

public void setResponsysFields(String[] responsysFields) {
    this.responsysFields = responsysFields;
}

public String[] getDatabaseFields() {
    return databaseFields;
}

public void setDatabaseFields(String[] databaseFields) {
    this.databaseFields = databaseFields;
}
}

```

JsonResponsys

```

package dao;

public class JsonResponsys {
    private RecordData recordData;
    private MergeRule mergeRule;

    public JsonResponsys(MergeRule mergeRule, RecordData recordData) {
        setMergeRule(mergeRule);
        setRecordData(recordData);
    }

    public void setRecordData(RecordData recordData) {

```

```

        this.recordData = recordData;
    }

    public RecordData getRecordData() {
        return recordData;
    }

    public MergeRule getMergeRule() {
        return mergeRule;
    }

    public void setMergeRule(MergeRule mergeRule) {
        this.mergeRule = mergeRule;
    }
}

```

Login

```

package dao;

public interface ILogin {

    public void setUser(String user);

    public String getUser();

    public void setPassword(String password);

    public String getPassword();
}

```

LoginOracle

```

package dao;

public class LoginOracle implements ILogin{

```

```
private String server, user, password, query;
```

```
public String getServer() {  
    return server;  
}
```

```
public void setServer(String server) {  
    this.server = server;  
}
```

```
@Override  
public void setUser(String user) {  
    this.password = user;  
}
```

```
@Override  
public String getUser() {  
    return this.user;  
}
```

```
@Override  
public void setPassword(String password) {  
    this.password = password;  
}
```

```
@Override  
public String getPassword() {  
    return this.password;  
}
```

```
public String getQuery() {
```

```
        return query;
    }

    public void setQuery(String query) {
        this.query = query;
    }
}
```

LoginResponsys

```
package dao;
```

```
public class LoginResponsys implements ILogin{

    private String user, password,auth_type, key, urlPost, urlConnection;
    private Formatter formatter;

    public String getAuth_type() {
        return auth_type;
    }

    public void setAuth_type(String auth_type) {
        this.auth_type = auth_type;
    }

    public String getKey() {
        return key;
    }

    public void setKey(String key) {

    }

    @Override
```



```
public void setUser(String user) {  
    this.user = user;  
}
```

```
@Override  
public String getUser() {  
    return this.user;  
}
```

```
@Override  
public void setPassword(String password) {  
    this.password = password;  
}
```

```
@Override  
public String getPassword() {  
    return this.password;  
}
```

```
public String getUrlPost() {  
    return urlPost;  
}
```

```
public void setUrlPost(String urlPost) {  
    this.urlPost = urlPost;  
}
```

```
public String getUrlConnection() {  
    return urlConnection;  
}
```

```
public void setUrlConnection(String urlConnection) {
```

```

        this.urlConnection = urlConnection;
    }

    public Formatter getFormatter() {
        return formatter;
    }

    public void setFormatter(Formatter formatter) {
        this.formatter = formatter;
    }
}

```

MergeRule

```

package dao;

public class MergeRule {
    private String htmlValue;
    private String optinValue;
    private String textValue;
    private boolean insertOnNoMatch;
    private String updateOnMatch;
    private String matchColumnName1;
    private String matchColumnName2;
    private String matchOperator;
    private String optoutValue;
    private String rejectRecordIfChannelEmpty;
    private String defaultPermissionStatus;

    public String getHtmlValue() {
        return htmlValue;
    }

    public void setHtmlValue(String htmlValue) {
        this.htmlValue = htmlValue;
    }
}

```

```
}  
  
public String getTextValue() {  
    return textValue;  
}  
  
public void setTextValue(String textValue) {  
    this.textValue = textValue;  
}  
  
public String getOptinValue() {  
    return optinValue;  
}  
  
public void setOptinValue(String optinValue) {  
    this.optinValue = optinValue;  
}  
  
public boolean getInsertOnNoMatch() {  
    return insertOnNoMatch;  
}  
  
public void setInsertOnNoMatch(boolean insertOnNoMatch) {  
    this.insertOnNoMatch = insertOnNoMatch;  
}  
  
public String getUpdateOnMatch() {  
    return updateOnMatch;  
}  
  
public void setUpdateOnMatch(String updateOnMatch) {  
    this.updateOnMatch = updateOnMatch;  
}  
  
public String getMatchColumnName1() {  
    return matchColumnName1;  
}  
  
public void setMatchColumnName1(String matchColumnName1) {  
    this.matchColumnName1 = matchColumnName1;  
}  
  
public String getMatchOperator() {
```

```

        return matchOperator;
    }

    public void setMatchOperator(String matchOperator) {
        this.matchOperator = matchOperator;
    }

    public String getMatchColumnName2() {
        return matchColumnName2;
    }

    public void setMatchColumnName2(String matchColumnName2) {
        this.matchColumnName2 = matchColumnName2;
    }

    public String getOptoutValue() {
        return optoutValue;
    }

    public void setOptoutValue(String optoutValue) {
        this.optoutValue = optoutValue;
    }

    public String getRejectRecordIfChannelEmpty() {
        return rejectRecordIfChannelEmpty;
    }

    public void setRejectRecordIfChannelEmpty(String rejectRecordIfChannelEmpty) {
        this.rejectRecordIfChannelEmpty = rejectRecordIfChannelEmpty;
    }

    public String getDefaultPermissionStatus() {
        return defaultPermissionStatus;
    }

    public void setDefaultPermissionStatus(String defaultPermissionStatus) {
        this.defaultPermissionStatus = defaultPermissionStatus;
    }
}

```

RecordData

```
package dao;
```

```

public class RecordData {
    private String[] fieldNames;
    private String[][] records;
    private String mapTemplateName;

    public String[] getFieldNames() {
        return fieldNames;
    }
    public void setFieldNames(String[] fieldNames) {
        this.fieldNames = fieldNames;
    }
    public String[][] getRecords() {
        return records;
    }
    public void setRecords(String[][] records) {
        this.records = records;
    }
    public String getMapTemplateName() {
        return mapTemplateName;
    }
    public void setMapTemplateName(String mapTemplateName) {
        this.mapTemplateName = mapTemplateName;
    }
}

```

Login

```

package dao;

```

```

public class Login {

    private LoginOracle oracle;
    private LoginResponsys responsys;

```

```
public LoginResponsys getResponsys() {  
    return responsys;  
}  
public void setResponsys(LoginResponsys responsys) {  
    this.responsys = responsys;  
}  
public LoginOracle getOracle() {  
    return oracle;  
}  
public void setOracle(LoginOracle oracle) {  
    this.oracle = oracle;  
}  
}
```

4 CONSIDERAÇÕES FINAIS

O presente documento visa explicar de maneira simples e eficaz a utilização e manutenção da API de integração Responsys – Database Oracle. O software foi pensado a fim de não necessitar de qualquer alteração em seu código fonte, pois cada vez que uma modificação for necessária, basta redigir os arquivos JSON de configuração e reiniciar o software, sem necessidade de recompilação ou coisa do gênero.

O software encontra-se na versão 1.0 e possivelmente sofrerá atualizações futuramente, caso ocorra serão escritos novos arquivos para documentar tais atualizações.

5 TESTES E RESULTADOS

Especificações da máquina:

CPU	AMD A10-8700P Raedon R6 1.8GHz
RAM	12 GB
SISTEMA OPERACIONAL	WINDOWS 10 x64
JAVA VERSION	1.8.0_201

Notas:

- Responsys: Durante os testes se constatou que o Responsys possui um limite de 200 registros por requisição. Também foi verificado que é necessário um tempo de intervalo entre as requisições, o tempo médio estipulado foi de 10 segundos. Para ambos os casos, já foram inseridas as devidas correções no software.
- Database: A conexão com o banco é encerrada após um tempo e o software acusa TIMEOUT, a mesma permanece estável até 1 hora (aproximadamente) e após isso é fechada pelo próprio database.

Resultados de testes:

- Em 1 hora de envio, a API submeteu, aproximadamente, 62 mil de registros.