UPskill

qpdc

INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL

Java

Client-side programming

Tutorial

Client-side programming assignment: Developing a web service client

Paulo Baltarejo Sousa, Joaquim Peixoto dos Santos
{pbs,jpe}@isep.ipp.pt
2021

P.PORTO

isep Instituto Superior de Engenharia do Porto

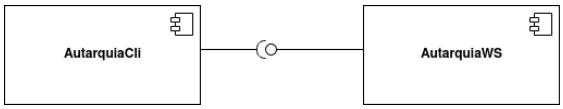
DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
Instituto Superior de Engenharia do Porto

1 Creating a Java Console Application Client RESTFul Web Service

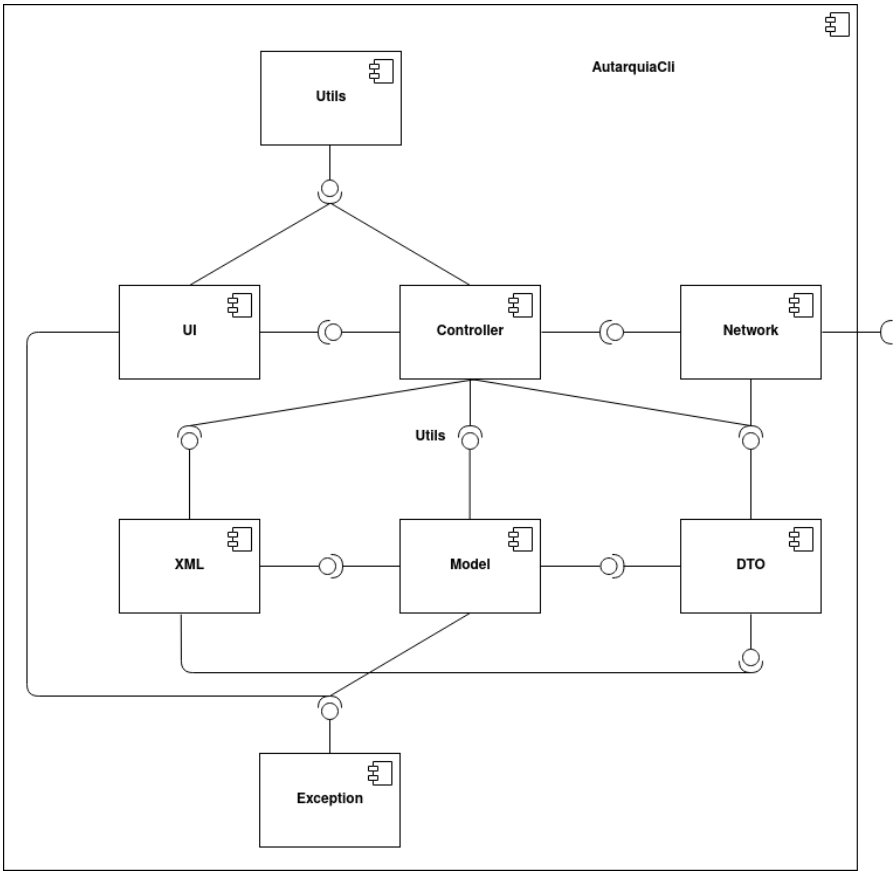
In this tutorial, we will create a web service called `AutarquiaCli`.

1.1 Software architecture (Component Diagram)

Next figure shows the system architecture.



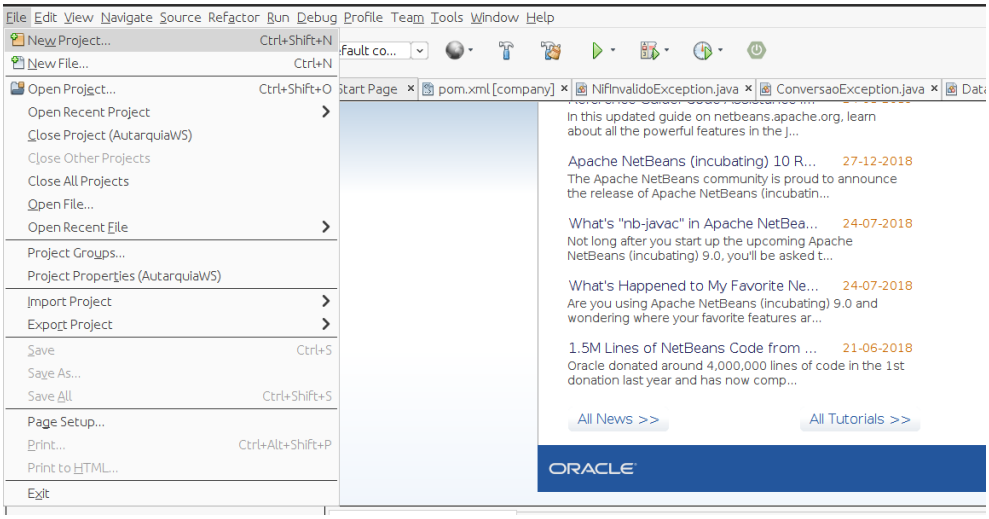
Next figure shows the `AutarquiaCli` component architecture.



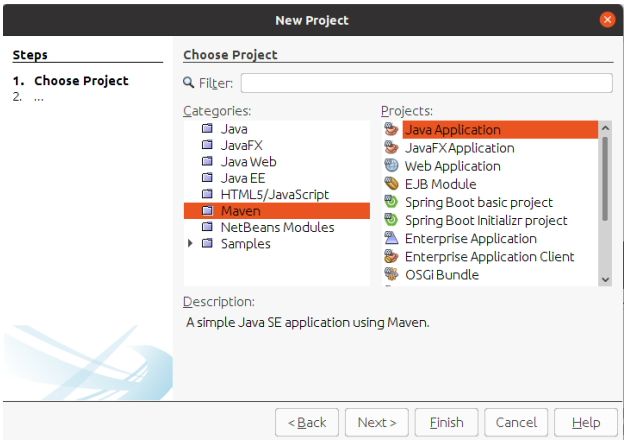
1.2 Create Project

1. Open the Netbeans IDE.

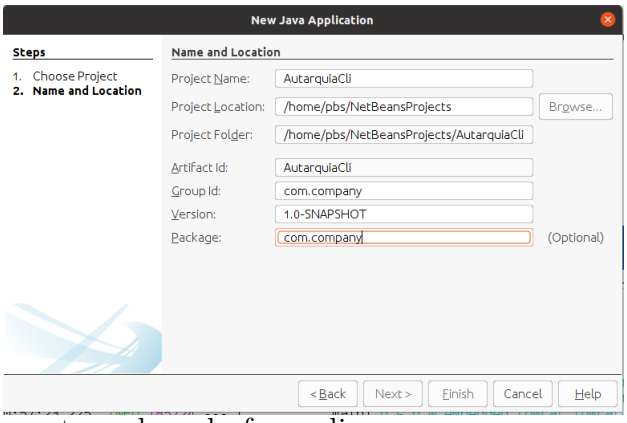
2. Select `File->New Project`.



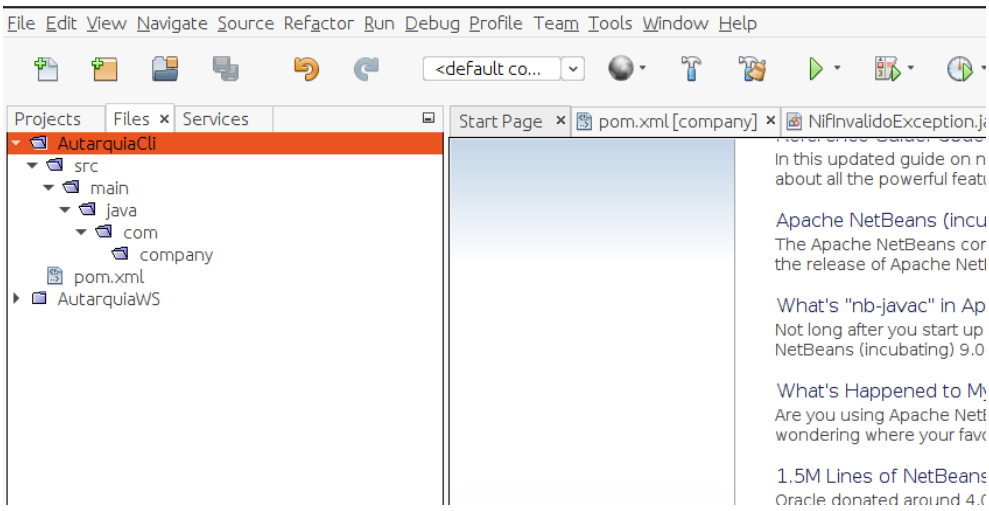
3. From `Categories`, select `Maven`. From `Projects`, select `Java Application`. Click the `Next` button.



4. Fill in the form "Name and Location" fields as shown in the following figure. Click the `Finish` button.



The project is create and ready for coding.



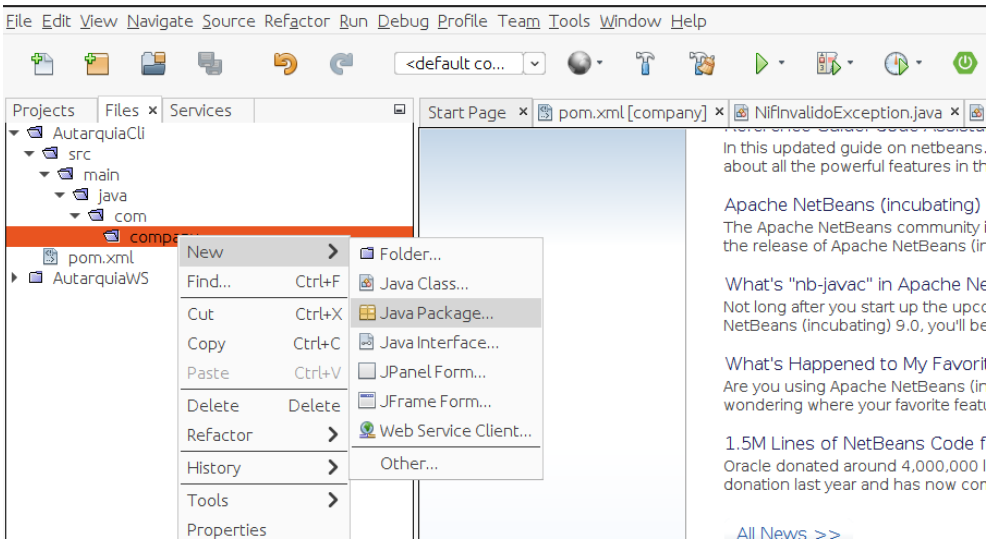
1.3 Coding Project

For coding the project, the appropriate view is the provided when selecting the `Files` tab in the project explore window.

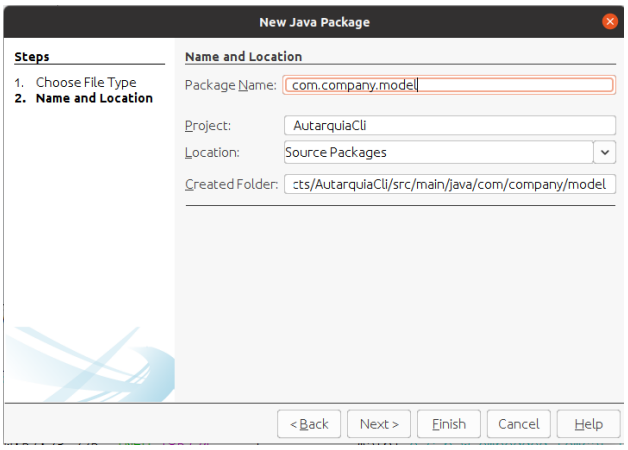
1.3.1 Organizing code

For organizing code, it will be created several packages into the main package `com.company`.

1. Rigth-click on the `company` package and select `New->Java Package`.



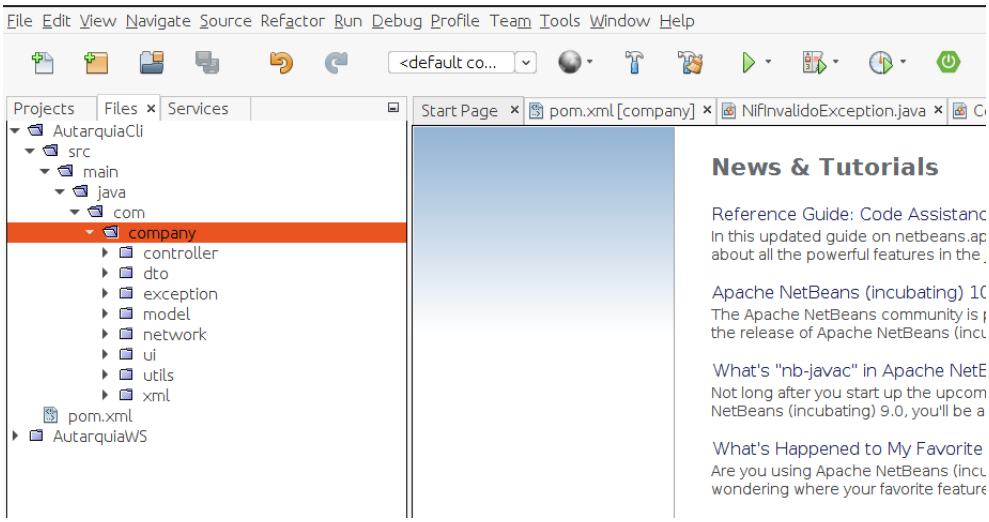
2. Enter name `model` and click the `Finish` button.



3. Repeat the procedure for creating the following packages:

   - `controller`
   - `network`
   - `ui`
   - `dto`
   - `exception`
   - `utils`

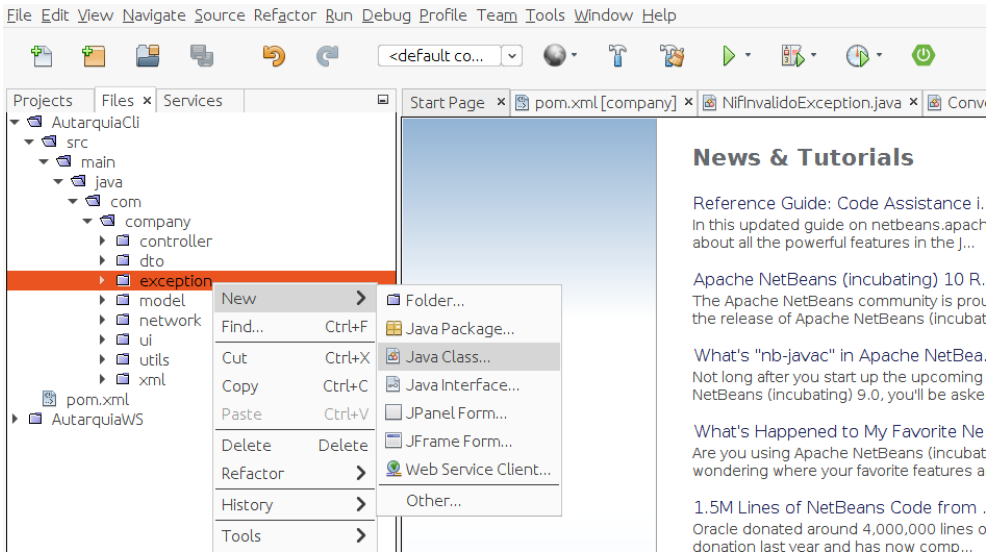
- `xml`



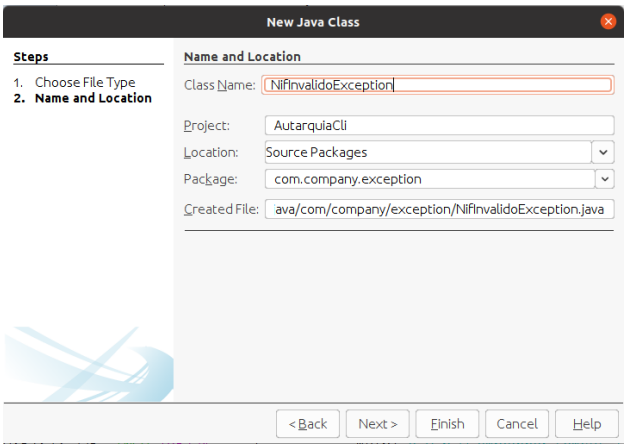
1.3.2 Coding `exception`

This package contains all classes used to manage the exceptions. To create a class you must follow the next steps:

1. Rigth-click on the `exception` package and select `New->Java Class`.



2. Name the class, `NifInvalidoException`, and click the `Finish` button.



3. Update the `NifInvalidoException` class code.

```
/*
* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package com.company.exception;

/**
*
* @author pbs
*/
public class NifInvalidoException extends RuntimeException {
  public NifInvalidoException(String s) {
    super(s);
  }
}
```

Repeat the procedure to create the following classes:

- `ConversaoException` class.

```
public class ConversaoException extends RuntimeException {

  public ConversaoException(String classe) {
    super("Erro a converter a classe:" + classe);
  }
}
```

- `DataInvalidaException` class.

```
public class DataInvalidaException extends RuntimeException {

  public DataInvalidaException(String s) {
    super(s);
  }
}
```

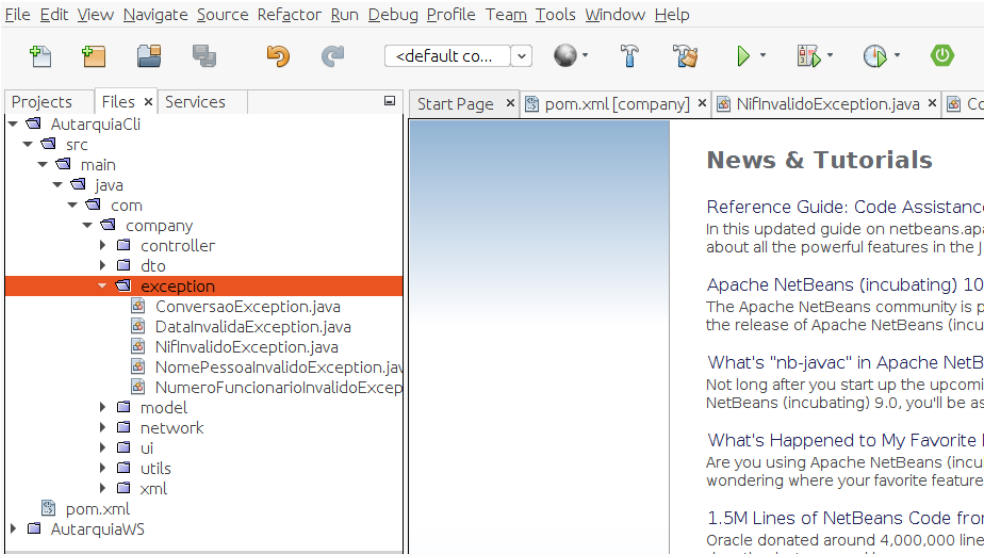
- `NomePessoaInvalidoException` class.

```
public class NomePessoaInvalidoException extends RuntimeException {

  public NomePessoaInvalidoException(String s) {
    super(s);
  }
}
```

- `NumeroFuncionarioInvalidoException` class.

```
public class NumeroFuncionarioInvalidoException extends RuntimeException {

  public NumeroFuncionarioInvalidoException(String s) {
    super(s);
  }

}
```

In the end, the content of the `exception` package must be similar to the shown in the following figure.



1.3.3 Coding `model`

This package implements all model classes:

- `Data` class.

```
import com.company.exception.DataInvalidaException;

import java.io.Serializable;

public class Data implements Serializable {
  private int dia;
  private int mes;
  private int ano;
```

```
public Data(int dia, int mes, int ano) {
  checkData(dia, mes, ano);
}

public Data(Data data) {
  checkData(data.dia, data.mes, data.ano);
}



public int getDia() {
  return dia;
}

public int getMes() {
  return mes;
}

public int getAno() {
  return ano;
}

public void setData(int dia, int mes, int ano) {
  checkData(dia, mes, ano);
}

private void checkData(int dia, int mes, int ano) throws DataInvalidaException{
  if (eValida(dia, mes, ano) == true) {
    this.dia = dia;
    this.mes = mes;
    this.ano = ano;
  } else {
    throw new DataInvalidaException(dia + "/" + mes + "/" + ano + ": data invalida");
  }
}

private boolean eBissexto(int ano){
  if(((ano % 4 == 0) && (ano % 100 != 0)) || (ano % 400 == 0)){
    return true;
  }
return false;
}

private boolean eValida(int dia, int mes, int ano){
  boolean f = false;
  switch(mes){
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
      if(dia > 0 && dia <= 31) {
        f = true;
      }
    break;
    case 4:
    case 6:
    case 9:
    case 11:
      if(dia > 0 && dia <= 30) {
        f = true;
      }
    break;
    case 2:
      if(eBissexto(ano) == true){
        if(dia > 0 && dia <=29) {
          f = true;
        }
      }else{
        if(dia > 0 && dia <=28) {
```

```
          f = true;
        }
      }
    break;
    default: break;
  }
  return f;
 }
}
```

- Pessoa class.

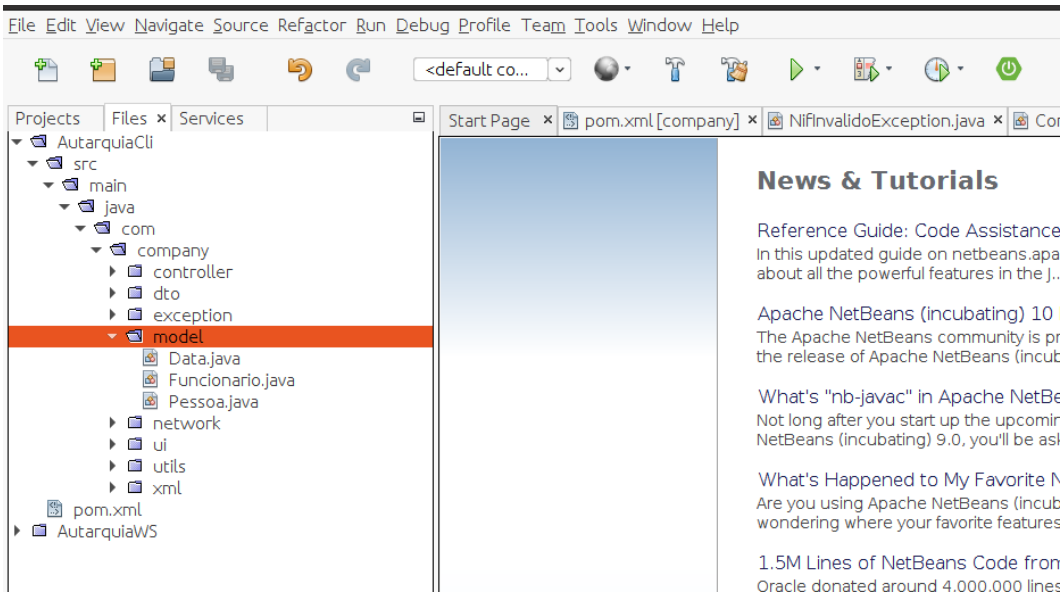
```
import com.company.exception.NifInvalidoException;
import com.company.exception.NomePessoaInvalidoException;

import java.io.Serializable;

public class Pessoa implements Serializable {
  private long nif;
  private String nome;
  private Data nascimento;

  public Pessoa() {
  }

  public Pessoa(long nif, String nome, Data nascimento) {
    setNif(nif);
    setNome(nome);
    this.nascimento = new Data(nascimento);
  }
  public Pessoa(Pessoa pessoa) {
    setNif(pessoa.nif);
    setNome(pessoa.nome);
    this.nascimento = new Data (pessoa.nascimento);
  }


  public long getNif() {
    return nif;
  }

  public String getNome() {
    return nome;
  }

  public Data getNascimento() {
    Data data = new Data(nascimento);
    return data;
  }

  public void setNif(long nif) throws NifInvalidoException{
    if (nif >= 100000000 && nif <= 999999999){
      this.nif = nif;
    }
    else {
      throw new NifInvalidoException(nif+ ": NIF inválido");
    }
  }

  public void setNome(String nome) throws NomePessoaInvalidoException{
    if(eNomeValido(nome)) {
      this.nome = nome;
    }else {
      throw new NomePessoaInvalidoException(nome+ ": Nome inválido");
    }
  }

  public void setNascimento(Data nascimento) {
    this.nascimento = nascimento;
  }
```

```
  private boolean eNomeValido(String nome){
    if(nome == null){
      return false;
    }
    if(nome.length() < 3){
      return false;
    }
    for(int i=0;i<nome.length();i++){
      if(nome.charAt(i) >= '0' && nome.charAt(i) <= '9')
      return false;
    }
    return true;
  }
}
```

- `Funcionario` class.

```
import com.company.exception.NumeroFuncionarioInvalidoException;

import java.io.Serializable;

public class Funcionario extends Pessoa implements Serializable {
  private int numeroFuncionario;
  private String cargo;

  public Funcionario(long nif, String nome, Data nascimento) {
    super(nif, nome, nascimento);
  }

  public Funcionario(long nif, String nome, Data nascimento, int numeroFuncionario, String cargo) {
    super(nif, nome, nascimento);
    setNumeroFuncionario(numeroFuncionario);
    this.cargo = cargo;
  }
  public Funcionario(Funcionario funcionario) {
    super(funcionario.getNif(),funcionario.getNome(), funcionario.getNascimento());
    setNumeroFuncionario(funcionario.getNumeroFuncionario());
    this.cargo = funcionario.getCargo();
  }

  public int getNumeroFuncionario() {
    return numeroFuncionario;
  }

  public String getCargo() {
    return cargo;
  }

  public void setNumeroFuncionario(int numeroFuncionario) throws NumeroFuncionarioInvalidoException{
    if(numeroFuncionario > 0){
      this.numeroFuncionario = numeroFuncionario;
    }else{
      throw new NumeroFuncionarioInvalidoException(numeroFuncionario+ ": Número inválido");
    }
  }

  public void setCargo(String cargo) {
    this.cargo = cargo;
  }

}
```

In the end, the content of the `model` package must be similar to the shown in the following figure.



1.3.4 Coding `dto`

The model classes cannot be exposed to outside. So, to avoid that it will be created a set of "model class clones". These classes have only attributes, at least one constructor (without parameter) and getters and setters methods. These classes will be used for serailization and deserialization using the Jackson library. In order to incluse such library into the project update the `pom.xml` file with the blue text.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
      schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.company</groupId>
  <artifactId>AutarquiaCli</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <jackson.version>2.8.1</jackson.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>com.fasterxml.jackson.dataformat</groupId>
      <artifactId>jackson-dataformat-xml</artifactId>
      <version>${jackson.version}</version>
    </dependency>
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-core</artifactId>
      <version>${jackson.version}</version>
    </dependency>
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>${jackson.version}</version>
```

```
    </dependency>
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-annotations</artifactId>
      <version>${jackson.version}</version>
    </dependency>
  </dependencies>
</project>
```

The `dto` package classes:

- `DataDTO` class.

```
import com.fasterxml.jackson.annotation.JsonPropertyOrder;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlProperty;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlRootElement;

@JsonPropertyOrder({"dia", "mes", "ano"})
@JacksonXmlRootElement(localName = "data")
public class DataDTO {
  @JacksonXmlProperty(localName = "dia")
  private int dia;
  @JacksonXmlProperty(localName = "mes")
  private int mes;
  @JacksonXmlProperty(localName = "ano")
  private int ano;

  public DataDTO() {

  }

  public DataDTO(int dia, int mes, int ano) {
    this.dia = dia;
    this.mes = mes;
    this.ano = ano;
  }

  public int getDia() {
    return dia;
  }

  public void setDia(int dia) {
    this.dia = dia;
  }

  public int getMes() {
    return mes;
  }

  public void setMes(int mes) {
    this.mes = mes;
  }

  public int getAno() {
    return ano;
  }

  public void setAno(int ano) {
    this.ano = ano;
  }
}
```

- `ErroDTO` class.

```
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlProperty;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlRootElement;
```

```
@JacksonXmlRootElement(localName = "erro")
public class ErroDTO {
  @JacksonXmlProperty(localName = "mensagem")
  private String mensagemErro;

  public ErroDTO(Exception e) {
    mensagemErro = e.getMessage();
    // e.printStackTrace();
  }

  public ErroDTO() {
  }

  public String getMensagemErro() {
    return mensagemErro;
  }

  public void setMensagemErro(String mensagemErro) {
    this.mensagemErro = mensagemErro;
  }
}
```

- `FuncionarioDTO` class.

```
import com.fasterxml.jackson.annotation.JsonPropertyOrder;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlProperty;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlRootElement;

@JsonPropertyOrder({"numeroFuncionario", "cargo"})
@JacksonXmlRootElement(localName = "funcionario")
public class FuncionarioDTO extends PessoaDTO{
  @JacksonXmlProperty(localName = "numero")
  private int numeroFuncionario;
  @JacksonXmlProperty(localName = "cargo")
  private String cargo;

  public FuncionarioDTO() {
    super();
  }

  public int getNumeroFuncionario() {
    return numeroFuncionario;
  }

  public void setNumeroFuncionario(int numeroFuncionario) {
    this.numeroFuncionario = numeroFuncionario;
  }

  public String getCargo() {
    return cargo;
  }

  public void setCargo(String cargo) {
    this.cargo = cargo;
  }
}
```

- `ListaFuncionarioDTO` class.

```
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlElementWrapper;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlProperty;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlRootElement;

import java.util.ArrayList;

@JacksonXmlRootElement(localName = "funcionarios")
public class ListaFuncionarioDTO {
  @JacksonXmlElementWrapper(useWrapping = false)
  @JacksonXmlProperty(localName = "funcionario")
```

```
  private ArrayList<FuncionarioDTO> funcionarios;

  public ListaFuncionarioDTO() {
  }

  public ArrayList<FuncionarioDTO> getFuncionarios() {
    return funcionarios;
  }

  public void setFuncionarios(ArrayList<FuncionarioDTO> funcionarios) {
    this.funcionarios = funcionarios;
  }
}
```

- `ListaPessoaDTO` class.

```
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlElementWrapper;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlProperty;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlRootElement;

import java.util.ArrayList;

@JacksonXmlRootElement(localName = "pessoas")
public class ListaPessoaDTO {
  @JacksonXmlElementWrapper(useWrapping = false)
  @JacksonXmlProperty(localName = "pessoa")
  private ArrayList<PessoaDTO> pessoas;

  public ListaPessoaDTO() {
  }

  public ArrayList<PessoaDTO> getPessoas() {
    return pessoas;
  }

  public void setPessoas(ArrayList<PessoaDTO> pessoas) {
    this.pessoas = pessoas;
  }

}
```

- `PessoaDTO` class.

```
import com.fasterxml.jackson.annotation.JsonPropertyOrder;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlProperty;
import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlRootElement;

@JsonPropertyOrder({"nif", "nome", "nascimento"})
@JacksonXmlRootElement(localName = "pessoa")
public class PessoaDTO {
  @JacksonXmlProperty(localName = "nif")
  private long nif;
  @JacksonXmlProperty(localName = "nome")
  private String nome;
  @JacksonXmlProperty(localName = "data_nascimento")
  private DataDTO nascimento;

  public PessoaDTO() {
  }

  public long getNif() {
    return nif;
  }

  public void setNif(long nif) {
    this.nif = nif;
  }

  public String getNome() {
```

```
    return nome;
  }

  public void setNome(String nome) {
    this.nome = nome;
  }

  public DataDTO getNascimento() {
    return nascimento;
  }

  public void setNascimento(DataDTO nascimento) {
    this.nascimento = nascimento;
  }
}
```

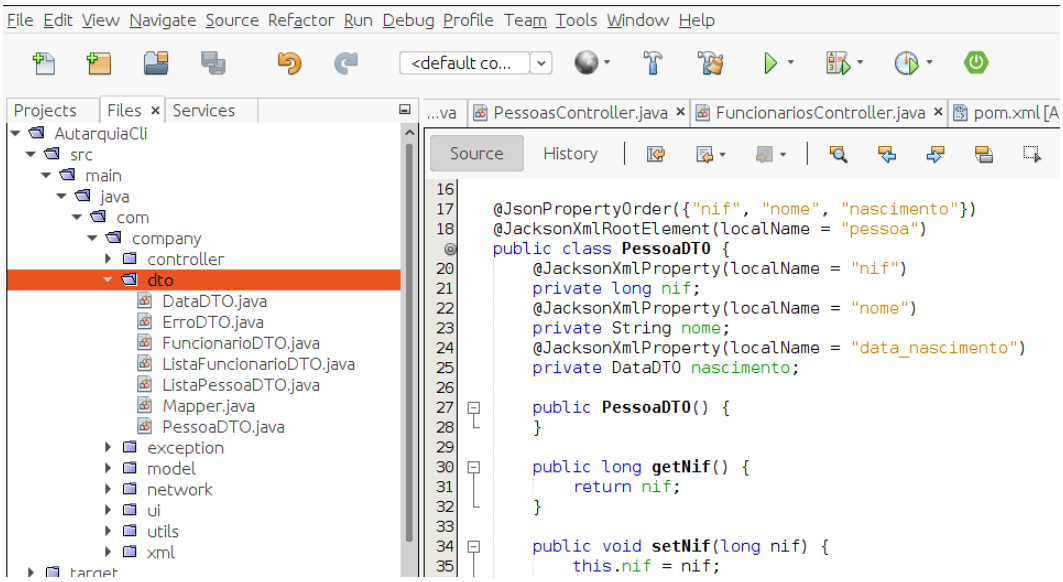
- `Mapper` class.

```
/*
* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package com.company.dto;

/**
*
* @author pbs
*/

import com.company.model.Data;
import com.company.model.Funcionario;
import com.company.model.Pessoa;

import java.util.ArrayList;

public class Mapper {

  public static DataDTO data2dataDTO(Data data) throws NullPointerException {
    DataDTO dataDTO = new DataDTO();
    dataDTO.setDia(data.getDia());
    dataDTO.setMes(data.getMes());
    dataDTO.setAno(data.getAno());
    return dataDTO;
  }

  public static Data dataDTO2data(DataDTO dataDTO) throws NullPointerException {
    Data data = null;

    data = new Data(dataDTO.getDia(), dataDTO.getMes(), dataDTO.getAno());

    return data;
  }

  public static PessoaDTO pessoa2PessoaDTO(Pessoa pessoa) throws NullPointerException {
    PessoaDTO pessoaDTO = new PessoaDTO();
    pessoaDTO.setNif(pessoa.getNif());
    pessoaDTO.setNome(pessoa.getNome());
    DataDTO dataDTO = data2dataDTO(pessoa.getNascimento());
    pessoaDTO.setNascimento(dataDTO);
    return pessoaDTO;
  }

  public static Pessoa pessoaDTO2Pessoa(PessoaDTO pessoaDTO) throws NullPointerException {
    Pessoa pessoa = null;

    Data data = dataDTO2data(pessoaDTO.getNascimento());
    pessoa = new Pessoa(pessoaDTO.getNif(), pessoaDTO.getNome(), data);

    return pessoa;
  }
```

```
public static ListaPessoaDTO listaPessoa2ListaPessoaDTO(ArrayList<Pessoa> pessoas) throws
      NullPointerException {
  ArrayList<PessoaDTO> pessoasDTO = new ArrayList<>();
  for (Pessoa pessoa : pessoas) {
    try {
      PessoaDTO pessoaDTO = pessoa2PessoaDTO(pessoa);
      pessoasDTO.add(pessoaDTO);
    } catch (NullPointerException e) {
      //does nothing. Actually, nothing is added to arraylist
    }
  }
  ListaPessoaDTO listaPessoaDTO = new ListaPessoaDTO();
  listaPessoaDTO.setPessoas(pessoasDTO);
  return listaPessoaDTO;
}
public static ArrayList<Pessoa> listaPessoaDTO2ListaPessoa( ListaPessoaDTO listaPessoasDTO) throws
      NullPointerException {
  ArrayList<Pessoa> pessoas = new ArrayList<>();
  ArrayList<PessoaDTO> pessoasDTO = listaPessoasDTO.getPessoas();

  for (PessoaDTO pessoaDTO : pessoasDTO) {
    try {
      Pessoa pessoa = pessoaDTO2Pessoa(pessoaDTO);
      pessoas.add(pessoa);
    } catch (NullPointerException e) {
      //does nothing. Actually, nothing is added to arraylist
    }
  }
  return pessoas;
}

public static FuncionarioDTO funcionario2FuncionarioDTO(Funcionario funcionario) throws
      NullPointerException {
  FuncionarioDTO funcionarioDTO = new FuncionarioDTO();
  funcionarioDTO.setNif(funcionario.getNif());
  funcionarioDTO.setNome(funcionario.getNome());
  DataDTO dataDTO = data2dataDTO(funcionario.getNascimento());
  funcionarioDTO.setNascimento(dataDTO);
  funcionarioDTO.setNumeroFuncionario(funcionario.getNumeroFuncionario());
  funcionarioDTO.setCargo(funcionario.getCargo());
  return funcionarioDTO;
}

public static Funcionario funcionarioDTO2Funcionario(FuncionarioDTO funcionarioDTO) throws
      NullPointerException {
  Funcionario funcionario = null;

  Data data = dataDTO2data(funcionarioDTO.getNascimento());
  funcionario = new Funcionario(funcionarioDTO.getNif(), funcionarioDTO.getNome(), data, funcionarioDTO.
        getNumeroFuncionario(), funcionarioDTO.getCargo());

  return funcionario;
}

public static ListaFuncionarioDTO listaFuncionario2ListaFuncionarioDTO(ArrayList<Funcionario>
      funcionarios) throws NullPointerException {
  ArrayList<FuncionarioDTO> funcionariosDTO = new ArrayList<>();
  for (Funcionario funcionario : funcionarios) {
    try {
      FuncionarioDTO funcionarioDTO = funcionario2FuncionarioDTO(funcionario);
      funcionariosDTO.add(funcionarioDTO);
    } catch (NullPointerException e) {
      //does nothing. Actually, nothing is added to arraylist
    }
  }
  ListaFuncionarioDTO listaFuncionarioDTO = new ListaFuncionarioDTO();
  listaFuncionarioDTO.setFuncionarios(funcionariosDTO);
  return listaFuncionarioDTO;
}
public static ArrayList<Funcionario> listaFuncionarioDTO2ListaFuncionario( ListaFuncionarioDTO
      listaFuncionariosDTO) throws NullPointerException {
  ArrayList<Funcionario> funcionarios = new ArrayList<>();
```

```
    ArrayList<FuncionarioDTO> funcionariosDTO = listaFuncionariosDTO.getFuncionarios();

    for (FuncionarioDTO funcionarioDTO : funcionariosDTO) {
      try {
        Funcionario funcionario = funcionarioDTO2Funcionario(funcionarioDTO);
        funcionarios.add(funcionario);
      } catch (NullPointerException e) {
        //does nothing. Actually, nothing is added to arraylist
      }
    }
    return funcionarios;
  }


}
```

In the end, the content of the `dto` package must be similar to the shown in the following figure.



1.3.5 Coding `network`

This package implements implements the component that is responsible for handling the HTTP connections (requests and responses). This package it is composed by following classes:

- `HttpRequestType`: this is a enumerator class, where are defined the identifiers for HTTP methods.

```
package com.company.network;

public enum HttpRequestType {
  GET,
  POST,
```

```
  PUT,
  DELETE
}
```

- `HttpStatusCode` this is a class composed by a set of constants for the HTTP status code.

```
package com.company.network;

public class HttpStatusCode {
  public static final int  Continue = 100 ;
  public static final int  SwitchingProtocols = 101 ;
  public static final int  Processing = 102 ;
  public static final int  EarlyHints = 103 ;
  public static final int  OK = 200 ;
  public static final int  Created = 201 ;
  public static final int  Accepted = 202 ;
  public static final int  NonAuthoritativeInformation = 203 ;
  public static final int  NoContent = 204 ;
  public static final int  ResetContent = 205 ;
  public static final int  PartialContent = 206 ;
  public static final int  MultiStatus = 207 ;
  public static final int  AlreadyReported = 208 ;
  public static final int  IMUsed = 226 ;
  public static final int  Ambiguous = 300 ;
  public static final int  MultipleChoices = 300 ;
  public static final int  Moved = 301 ;
  public static final int  MovedPermanently = 301 ;
  public static final int  Found = 302 ;
  public static final int  Redirect = 302 ;
  public static final int  RedirectMethod = 303 ;
  public static final int  SeeOther = 303 ;
  public static final int  NotModified = 304 ;
  public static final int  UseProxy = 305 ;
  public static final int  Unused = 306 ;
  public static final int  RedirectKeepVerb = 307 ;
  public static final int  TemporaryRedirect = 307 ;
  public static final int  PermanentRedirect = 308 ;
  public static final int  BadRequest = 400 ;
  public static final int  Unauthorized = 401 ;
  public static final int  PaymentRequired = 402 ;
  public static final int  Forbidden = 403 ;
  public static final int  NotFound = 404 ;
  public static final int  MethodNotAllowed = 405 ;
  public static final int  NotAcceptable = 406 ;
  public static final int  ProxyAuthenticationRequired = 407 ;
  public static final int  RequestTimeout = 408 ;
  public static final int  Conflict = 409 ;
  public static final int  Gone = 410 ;
  public static final int  LengthRequired = 411 ;
  public static final int  PreconditionFailed = 412 ;
  public static final int  RequestEntityTooLarge = 413 ;
  public static final int  RequestUriTooLong = 414 ;
  public static final int  UnsupportedMediaType = 415 ;
  public static final int  RequestedRangeNotSatisfiable = 416 ;
  public static final int  ExpectationFailed = 417 ;
  public static final int  MisdirectedRequest = 421 ;
  public static final int  UnprocessableEntity = 422 ;
  public static final int  Locked = 423 ;
  public static final int  FailedDependency = 424 ;
  public static final int  UpgradeRequired = 426 ;
  public static final int  PreconditionRequired = 428 ;
  public static final int  TooManyRequests = 429 ;
  public static final int  RequestHeaderFieldsTooLarge = 431 ;
  public static final int  UnavailableForLegalReasons = 451 ;
  public static final int  InternalServerError = 500 ;
  public static final int  NotImplemented = 501 ;
  public static final int  BadGateway = 502 ;
  public static final int  ServiceUnavailable = 503 ;
  public static final int  GatewayTimeout = 504 ;
```

```
  public static final int  HttpVersionNotSupported = 505 ;
  public static final int  VariantAlsoNegotiates = 506 ;
  public static final int  InsufficientStorage = 507 ;
  public static final int  LoopDetected = 508 ;
  public static final int  NotExtended = 510 ;
  public static final int  NetworkAuthenticationRequired = 511 ;
}
```

- `HttpRequest`: this is a class model the HTTP request.

```
package com.company.network;

public class HttpRequest {
  private HttpRequestType type;
  private String url;
  private String body;

  public HttpRequest(HttpRequestType type, String url, String body) {
    this.type = type;
    this.url = url;
    this.body = body;
  }

  public HttpRequestType getType() {
    return type;
  }

  public void setType(HttpRequestType type) {
    this.type = type;
  }

  public String getUrl() {
    return url;
  }

  public void setUrl(String url) {
    this.url = url;
  }

  public String getBody() {
    return body;
  }

  public void setBody(String body) {
    this.body = body;
  }
}
```

- `HttpResponse`: this is a class model the HTTP response.

```
package com.company.network;

public class HttpResponse {
  private int status;
  private String body;

  public HttpResponse(int status, String body) {
    this.status = status;
    this.body = body;
  }

  public int getStatus() {
    return status;
  }

  public void setStatus(int status) {
    this.status = status;
  }
```

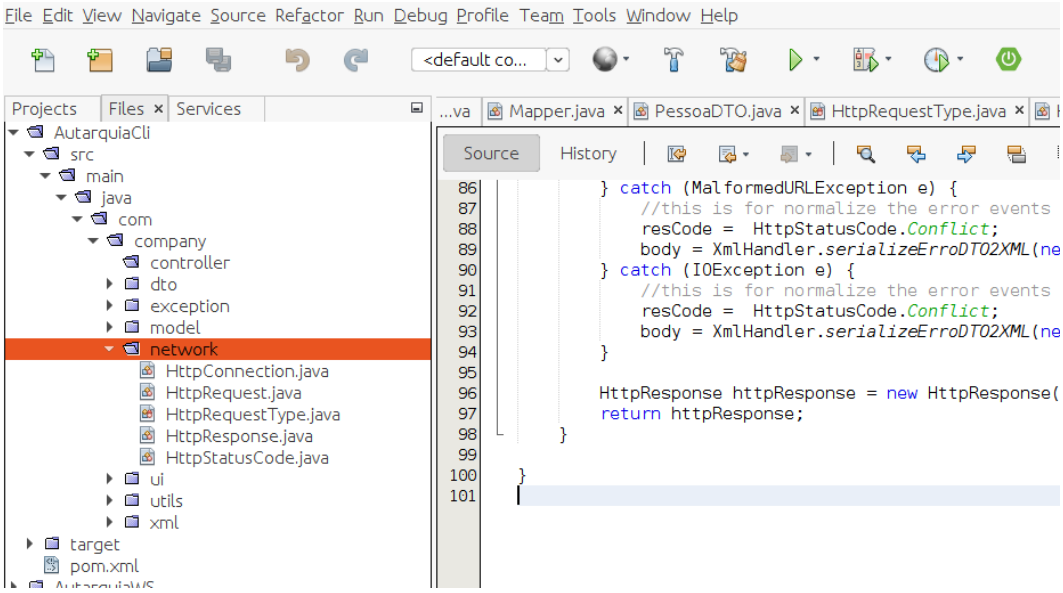
```
  public String getBody() {
    return body;
  }

  public void setBody(String body) {
    this.body = body;
  }
}
```

- `HttpConnection`: this is class is responsible for handling HTTP connections.
  - `public static HttpResponse makeRequest(HttpRequest httpRequest)`: this method is responsible for managing all HTTP connections. It receives an `HttpRequest` object and returns an `HttpResponse` object.

```
package com.company.network;

import com.company.dto.ErroDTO;
import com.company.xml.XmlHandler;
import java.io.*;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;

public class HttpConnection {

  private static String readBody(InputStream in){
    StringBuilder sb = new StringBuilder();
    BufferedReader br = new BufferedReader(new InputStreamReader(in));
    try {
      String read = br.readLine();
      while(read !=null){
        sb.append(read);
        read = br.readLine();
      }
    } catch (IOException e) {
      e.printStackTrace();
    }

    return sb.toString();
  }
  private static void writeBody(OutputStream writer, String body){
    try {
      byte[] dataBytes = body.getBytes("UTF-8");
      writer.write(dataBytes);
      writer.flush();
      writer.close();

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


  public static HttpResponse makeRequest(HttpRequest httpRequest) {
    HttpURLConnection httpConn = null;
    int resCode = -1;
    String body = "";
    try {
      URL url = new URL(httpRequest.getUrl());
      URLConnection urlConn = url.openConnection();

      if (!(urlConn instanceof HttpURLConnection)) {
        throw new IOException("URL is not an Http URL");
      }
```

```
      httpConn = (HttpURLConnection) urlConn;
      httpConn.setConnectTimeout(3000);
      httpConn.setRequestProperty("Content-Type", "application/xml");

      switch (httpRequest.getType()){
        case GET:
        httpConn.setRequestMethod("GET");
        httpConn.setDoInput(true);
        break;
        case POST:
        httpConn.setRequestMethod("POST");
        httpConn.setDoOutput(true);
        writeBody(httpConn.getOutputStream(), httpRequest.getBody());

        break;
        case PUT:
        httpConn.setRequestMethod("PUT");
        httpConn.setDoOutput(true);
        writeBody(httpConn.getOutputStream(), httpRequest.getBody());
        break;
        case DELETE:
        httpConn.setRequestMethod("DELETE");
        break;
      }

      httpConn.connect();

      resCode = httpConn.getResponseCode();
      body = readBody(httpConn.getInputStream());
    } catch (MalformedURLException e) {
      //this is for normalize the error events according to the way is handled by the WS
      resCode = HttpStatusCode.Conflict;
      body = XmlHandler.serializeErroDTO2XML(new ErroDTO(e));
    } catch (IOException e) {
      //this is for normalize the error events according to the way is handled by the WS
      resCode = HttpStatusCode.Conflict;
      body = XmlHandler.serializeErroDTO2XML(new ErroDTO(e));
    }

    HttpResponse httpResponse = new HttpResponse(resCode, body);
    return httpResponse;
  }

}
```

In the end, the content of the `network` package must be similar to the shown in the following figure.



1.3.6 Coding `xml`

This package is composed by a class, called `XmlHandler`, used for serializing and deserializing data.

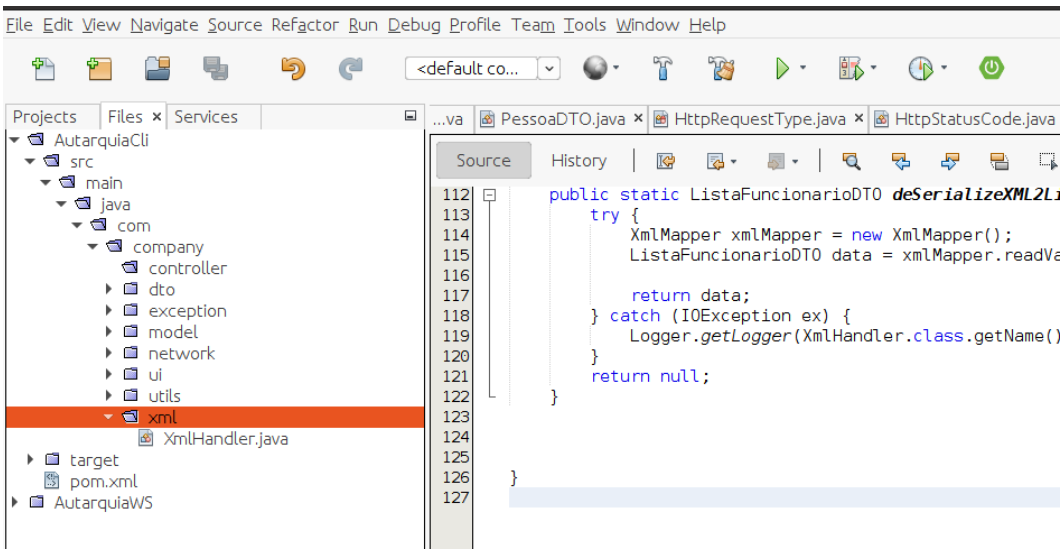
- `XmlHandler` class.

```
package com.company.xml;

import com.company.dto.ErroDTO;
import com.company.dto.FuncionarioDTO;
import com.company.dto.ListaFuncionarioDTO;
import com.company.dto.ListaPessoaDTO;
import com.company.dto.PessoaDTO;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.dataformat.xml.XmlMapper;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;


public class XmlHandler {


  public static String serializeErroDTO2XML(ErroDTO erroDTO) {
    try {
      XmlMapper xmlMapper = new XmlMapper();
      String xml = xmlMapper.writeValueAsString(erroDTO);
      return xml;
    } catch (JsonProcessingException ex) {
      Logger.getLogger(XmlHandler.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
  }
  public static ErroDTO deSerializeXML2ErroDTO(String xmlData) {
    try {
      XmlMapper xmlMapper = new XmlMapper();
      ErroDTO data = xmlMapper.readValue(xmlData, ErroDTO.class);
```

```
      return data;
    } catch (IOException ex) {
      Logger.getLogger(XmlHandler.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
  }
  public static String serializePessoaDTO2XML(PessoaDTO pessoaDTO) {
    try {
      XmlMapper xmlMapper = new XmlMapper();
      String xml = xmlMapper.writeValueAsString(pessoaDTO);
      return xml;
    } catch (JsonProcessingException ex) {
      Logger.getLogger(XmlHandler.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
  }
  public static PessoaDTO deSerializeXML2PessoaDTO(String xmlData) {
    try {
      XmlMapper xmlMapper = new XmlMapper();
      PessoaDTO data = xmlMapper.readValue(xmlData, PessoaDTO.class);

      return data;
    } catch (IOException ex) {
      Logger.getLogger(XmlHandler.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
  }
  public static String serializeListaPessoaDTO2XML(ListaPessoaDTO listaPessoaDTO) {
    try {
      XmlMapper xmlMapper = new XmlMapper();
      String xml = xmlMapper.writeValueAsString(listaPessoaDTO);
      return xml;
    } catch (JsonProcessingException ex) {
      Logger.getLogger(XmlHandler.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
  }
  public static ListaPessoaDTO deSerializeXML2ListaPessoaDTO(String xmlData) {
    try {
      XmlMapper xmlMapper = new XmlMapper();
      ListaPessoaDTO data = xmlMapper.readValue(xmlData, ListaPessoaDTO.class);

      return data;
    } catch (IOException ex) {
      Logger.getLogger(XmlHandler.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
  }
  public static String serializeFuncionarioDTO2XML(FuncionarioDTO funcionarioDTO) {
    try {
      XmlMapper xmlMapper = new XmlMapper();
      String xml = xmlMapper.writeValueAsString(funcionarioDTO);
      return xml;
    } catch (JsonProcessingException ex) {
      Logger.getLogger(XmlHandler.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
  }
  public static FuncionarioDTO deSerializeXML2FuncionarioDTO(String xmlData) {
    try {
      XmlMapper xmlMapper = new XmlMapper();
      FuncionarioDTO data = xmlMapper.readValue(xmlData, FuncionarioDTO.class);

      return data;
    } catch (IOException ex) {
      Logger.getLogger(XmlHandler.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
  }
  public static String serializeListaFuncionarioDTO2XML(ListaFuncionarioDTO listaFuncionarioDTO) {
    try {
      XmlMapper xmlMapper = new XmlMapper();
      String xml = xmlMapper.writeValueAsString(listaFuncionarioDTO);
```

```
        return xml;
      } catch (JsonProcessingException ex) {
        Logger.getLogger(XmlHandler.class.getName()).log(Level.SEVERE, null, ex);
      }
      return null;
    }
    public static ListaFuncionarioDTO deSerializeXML2ListaFuncionarioDTO(String xmlData) {
      try {
        XmlMapper xmlMapper = new XmlMapper();
        ListaFuncionarioDTO data = xmlMapper.readValue(xmlData, ListaFuncionarioDTO.class);

        return data;
      } catch (IOException ex) {
        Logger.getLogger(XmlHandler.class.getName()).log(Level.SEVERE, null, ex);
      }
      return null;
    }



  }
```

In the end, the content of the `xml` package must be similar to the shown in the following figure.



1.3.7 Coding `utils`

This package is composed by two classes:

- `Constants`: contains an constant that holds the web server ip address.

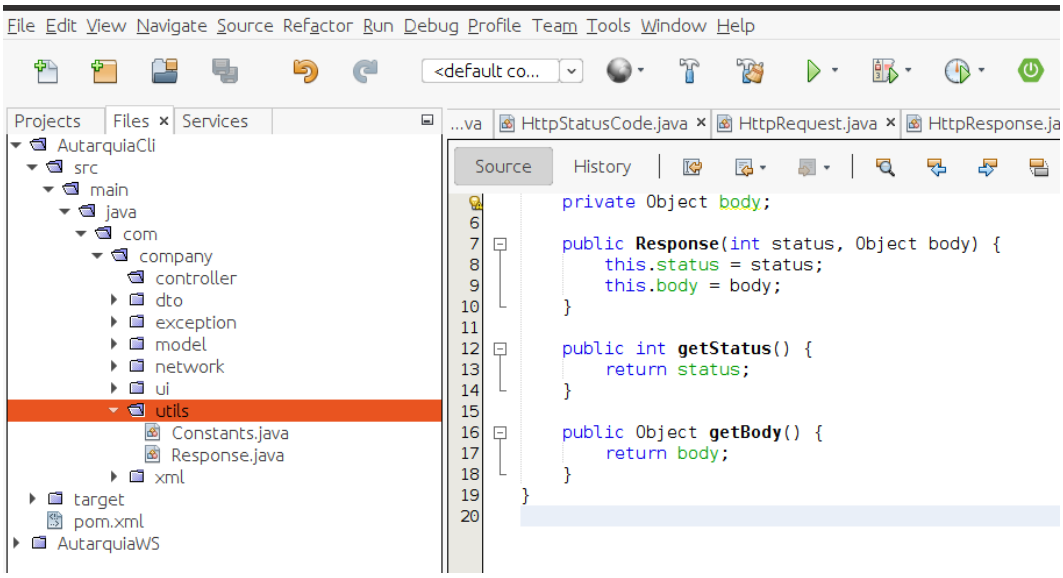
```
package com.company.utils;

/**
*
* @author pbs
*/
```

```
public class Constants {
  public static final String HOST="http://localhost:8080/";
}
```

- `Response`: this is a basic class (similar to a DTO) that is used to model a response. This is used to normalize the data transfer between the `controllers` and `ui` class methods

```
package com.company.utils;

public class Response {
  private int status;
  private Object body;

  public Response(int status, Object body) {
    this.status = status;
    this.body = body;
  }

  public int getStatus() {
    return status;
  }

  public Object getBody() {
    return body;
  }
}
```

In the end, the content of the `utils` package must be similar to the shown in the following figure.



1.3.8 Coding `controller`

Controllers classes manage the requests. They receive requests from UI and deliver to responseto UI. This package implements the following classes:

- `PessoasController` class.

```
package com.company.controller;

import com.company.dto.ErroDTO;
import com.company.dto.ListaPessoaDTO;
import com.company.dto.Mapper;
import com.company.dto.PessoaDTO;
import com.company.model.Pessoa;
import com.company.network.HttpConnection;
import com.company.network.HttpRequest;
import com.company.network.HttpRequestType;
import com.company.network.HttpResponse;
import com.company.network.HttpStatusCode;
import com.company.utils.Response;
import com.company.xml.XmlHandler;
import java.util.ArrayList;

/**
*
* @author pbs
*/
public class PessoasController {


  public static Response getPessoas(String uri) {
    Response response = null;
    HttpRequest httpRequest = new HttpRequest(HttpRequestType.GET, uri, "");
    HttpResponse httpResponse = HttpConnection.makeRequest(httpRequest);
    switch (httpResponse.getStatus()) {
      case HttpStatusCode.OK:
      ListaPessoaDTO listaPessoaDTO = XmlHandler.deSerializeXML2ListaPessoaDTO(httpResponse.getBody());
      ArrayList<Pessoa> pessoas = Mapper.listaPessoaDTO2ListaPessoa(listaPessoaDTO);
      response = new Response(HttpStatusCode.OK, pessoas);
      break;
      case HttpStatusCode.Conflict:
      ErroDTO erroDTO = XmlHandler.deSerializeXML2ErroDTO(httpResponse.getBody());
      response = new Response(HttpStatusCode.Conflict, erroDTO.getMensagemErro());
      break;
    }
    return response;
  }


  public static Response getPessoa(String uri) {
    Response response = null;
    HttpRequest httpRequest = new HttpRequest(HttpRequestType.GET, uri, "");
    HttpResponse httpResponse = HttpConnection.makeRequest(httpRequest);
    switch (httpResponse.getStatus()) {
      case HttpStatusCode.OK:
      PessoaDTO pessoaDTO = XmlHandler.deSerializeXML2PessoaDTO(httpResponse.getBody());
      Pessoa pessoa = Mapper.pessoaDTO2Pessoa(pessoaDTO);
      response = new Response(HttpStatusCode.OK, pessoa);
      break;
      case HttpStatusCode.Conflict:
      ErroDTO erroDTO = XmlHandler.deSerializeXML2ErroDTO(httpResponse.getBody());
      response = new Response(HttpStatusCode.Conflict, erroDTO.getMensagemErro());
      break;
    }
    return response;
  }

  public static Response addPessoa(String uri, PessoaDTO pessoaDTO) {
    Response response = null;
    final String body = XmlHandler.serializePessoaDTO2XML(pessoaDTO);
```

```
    HttpRequest httpRequest = new HttpRequest(HttpRequestType.POST, uri, body);
    HttpResponse httpResponse = HttpConnection.makeRequest(httpRequest);
    switch (httpResponse.getStatus()) {
      case HttpStatusCode.Created:
      response = new Response(HttpStatusCode.Created, null);
      break;
      case HttpStatusCode.Conflict:
      ErroDTO erroDTO = XmlHandler.deSerializeXML2ErroDTO(httpResponse.getBody());
      response = new Response(HttpStatusCode.Conflict, erroDTO.getMensagemErro());
      break;
    }
    return response;
  }
  public static Response updatePessoa(String uri, PessoaDTO pessoaDTO) {
    Response response = null;
    final String body = XmlHandler.serializePessoaDTO2XML(pessoaDTO);
    HttpRequest httpRequest = new HttpRequest(HttpRequestType.PUT, uri, body);
    HttpResponse httpResponse = HttpConnection.makeRequest(httpRequest);
    switch (httpResponse.getStatus()) {
      case HttpStatusCode.OK:
      response = new Response(HttpStatusCode.OK, null);
      break;
      case HttpStatusCode.Conflict:
      ErroDTO erroDTO = XmlHandler.deSerializeXML2ErroDTO(httpResponse.getBody());
      response = new Response(HttpStatusCode.Conflict, erroDTO.getMensagemErro());
      break;
    }
    return response;
  }


  public static Response deletePessoa(String uri) {
    Response response = null;
    HttpRequest httpRequest = new HttpRequest(HttpRequestType.DELETE, uri, "");
    HttpResponse httpResponse = HttpConnection.makeRequest(httpRequest);
    switch (httpResponse.getStatus()) {
      case HttpStatusCode.OK:
      response = new Response(HttpStatusCode.OK, null);
      break;
      case HttpStatusCode.Conflict:
      ErroDTO erroDTO = XmlHandler.deSerializeXML2ErroDTO(httpResponse.getBody());
      response = new Response(HttpStatusCode.Conflict, erroDTO.getMensagemErro());
      break;
    }
    return response;
  }



}
```

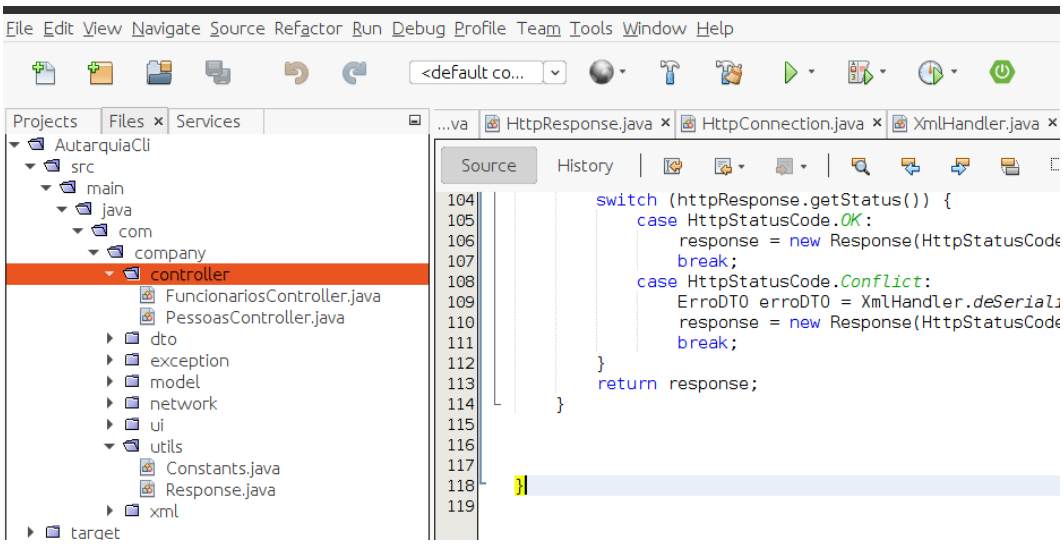
- `FuncionariosController` class.

```
package com.company.controller;

import com.company.dto.ErroDTO;
import com.company.dto.FuncionarioDTO;
import com.company.dto.ListaFuncionarioDTO;
import com.company.dto.Mapper;
import com.company.model.Funcionario;
import com.company.network.HttpConnection;
import com.company.network.HttpRequest;
import com.company.network.HttpRequestType;
import com.company.network.HttpResponse;
import com.company.network.HttpStatusCode;
import com.company.utils.Response;
import com.company.xml.XmlHandler;
import java.util.ArrayList;

/**
```

```
 *
 * @author pbs
 */
public class FuncionariosController {


  public static Response getFuncionarios(String uri) {
    Response response = null;
    HttpRequest httpRequest = new HttpRequest(HttpRequestType.GET, uri, "");
    HttpResponse httpResponse = HttpConnection.makeRequest(httpRequest);
    switch (httpResponse.getStatus()) {
      case HttpStatusCode.OK:
      ListaFuncionarioDTO listaFuncionarioDTO = XmlHandler.deSerializeXML2ListaFuncionarioDTO(httpResponse.
            getBody());
      ArrayList<Funcionario> funcionarios = Mapper.listaFuncionarioDTO2ListaFuncionario(listaFuncionarioDTO
            );
      response = new Response(HttpStatusCode.OK, funcionarios);
      break;
      case HttpStatusCode.Conflict:
      ErroDTO erroDTO = XmlHandler.deSerializeXML2ErroDTO(httpResponse.getBody());
      response = new Response(HttpStatusCode.Conflict, erroDTO.getMensagemErro());
      break;
    }
    return response;
  }


  public static Response getFuncionario(String uri) {
    Response response = null;
    HttpRequest httpRequest = new HttpRequest(HttpRequestType.GET, uri, "");
    HttpResponse httpResponse = HttpConnection.makeRequest(httpRequest);
    switch (httpResponse.getStatus()) {
      case HttpStatusCode.OK:
      FuncionarioDTO funcionarioDTO = XmlHandler.deSerializeXML2FuncionarioDTO(httpResponse.getBody());
      Funcionario funcionario = Mapper.funcionarioDTO2Funcionario(funcionarioDTO);
      response = new Response(HttpStatusCode.OK, funcionario);
      break;
      case HttpStatusCode.Conflict:
      ErroDTO erroDTO = XmlHandler.deSerializeXML2ErroDTO(httpResponse.getBody());
      response = new Response(HttpStatusCode.Conflict, erroDTO.getMensagemErro());
      break;
    }
    return response;
  }

  public static Response addFuncionario(String uri, FuncionarioDTO funcionarioDTO) {
    Response response = null;
    final String body = XmlHandler.serializeFuncionarioDTO2XML(funcionarioDTO);
    HttpRequest httpRequest = new HttpRequest(HttpRequestType.POST, uri, body);
    HttpResponse httpResponse = HttpConnection.makeRequest(httpRequest);
    switch (httpResponse.getStatus()) {
      case HttpStatusCode.Created:
      response = new Response(HttpStatusCode.Created, null);
      break;
      case HttpStatusCode.Conflict:
      ErroDTO erroDTO = XmlHandler.deSerializeXML2ErroDTO(httpResponse.getBody());
      response = new Response(HttpStatusCode.Conflict, erroDTO.getMensagemErro());
      break;
    }
    return response;
  }
  public static Response updateFuncionario(String uri, FuncionarioDTO funcionarioDTO) {
    Response response = null;
    final String body = XmlHandler.serializeFuncionarioDTO2XML(funcionarioDTO);
    HttpRequest httpRequest = new HttpRequest(HttpRequestType.PUT, uri, body);
    HttpResponse httpResponse = HttpConnection.makeRequest(httpRequest);
    switch (httpResponse.getStatus()) {
      case HttpStatusCode.OK:
      response = new Response(HttpStatusCode.OK, null);
      break;
      case HttpStatusCode.Conflict:
      ErroDTO erroDTO = XmlHandler.deSerializeXML2ErroDTO(httpResponse.getBody());
      response = new Response(HttpStatusCode.Conflict, erroDTO.getMensagemErro());
```

```
        break;
      }
      return response;
    }


    public static Response deleteFuncionario(String uri) {
      Response response = null;
      HttpRequest httpRequest = new HttpRequest(HttpRequestType.DELETE, uri, "");
      HttpResponse httpResponse = HttpConnection.makeRequest(httpRequest);
      switch (httpResponse.getStatus()) {
        case HttpStatusCode.OK:
        response = new Response(HttpStatusCode.OK, null);
        break;
        case HttpStatusCode.Conflict:
        ErroDTO erroDTO = XmlHandler.deSerializeXML2ErroDTO(httpResponse.getBody());
        response = new Response(HttpStatusCode.Conflict, erroDTO.getMensagemErro());
        break;
      }
      return response;
    }



  }
```

In the end, the content of the `controller` package must be similar to the shown in the following figure.



1.3.9 Coding `ui`

This package implements the user interface and is composed by the following classes:

- `UIGeral` class.

```java
package com.company.ui;

import java.util.InputMismatchException;
import java.util.Scanner;

public class UIGeral {

  public static long getNumber(String label) {
    Scanner kbd = new Scanner(System.in);
    boolean flag;
    long number = -1;
    do {
      System.out.print(label + ": ");
      try {
        flag = false;
        number = kbd.nextLong();
      } catch (InputMismatchException e) {
        flag = true;
      }
      kbd.nextLine();
    } while (flag);
    return number;
  }

  public static String getText(String label) {
    Scanner kbd = new Scanner(System.in);
    String text = "";
    System.out.print(label + ": ");
    text = kbd.nextLine();
    return text;
  }

}
```

- `UIData` class.

```java
package com.company.ui;

import com.company.exception.DataInvalidaException;
import com.company.model.Data;

public class UIData {

  public static Data getData() {
    boolean flag;
    Data data = null;
    do {
      try {
        flag = false;
        System.out.println("Data");

        int dia = (int) UIGeral.getNumber("Dia");
        int mes = (int) UIGeral.getNumber("Mes");
        int ano = (int) UIGeral.getNumber("Ano");

        data = new Data(dia, mes, ano);
      } catch (DataInvalidaException e) {
        flag = true;
        System.out.println("Atenção: "+ e.getMessage());
      }
    } while (flag);
    return data;
  }
}
```

- `UIPessoa` class.

```java
package com.company.ui;
```

```
import com.company.controller.PessoasController;

import com.company.dto.Mapper;
import com.company.dto.PessoaDTO;
import com.company.exception.NifInvalidoException;
import com.company.exception.NomePessoaInvalidoException;
import com.company.model.Data;
import com.company.model.Funcionario;
import com.company.model.Pessoa;
import com.company.network.HttpStatusCode;
import com.company.utils.Constants;
import com.company.utils.Response;
import java.util.ArrayList;
import java.util.List;

public class UIPessoa {

  public static void mainPessoa() {
    Pessoa pessoa = null;
    Response response = null;
    PessoaDTO pessoaDTO = null;
    long nif;
    int op;
    do {
      op = menuPessoa();
      switch (op) {
        case 0:
        System.out.println("Volta para o menu anterior.");
        break;
        case 1:
        System.out.println("\nInserir\n");
        pessoa = getPessoa();
        pessoaDTO = Mapper.pessoa2PessoaDTO(pessoa);
        response = PessoasController.addPessoa(Constants.HOST + "api/pessoas", pessoaDTO);
        if (response != null) {
          Object object = response.getBody();
          switch (response.getStatus()) {
            case HttpStatusCode.Created:
            System.out.println(HttpStatusCode.Created + "- Created");
            break;
            case HttpStatusCode.Conflict:
            if (object instanceof String) {
              String message = (String) object;
              System.out.println(HttpStatusCode.Conflict + "- Conflict:" + message);
            }
            default:
            break;
          }
        }

        break;
        case 2:
        System.out.println("\nPesquisar\n");
        nif = UIGeral.getNumber("NIF");
        response = PessoasController.getPessoa(Constants.HOST + "api/pessoas/" + nif);
        if (response != null) {
          Object object = response.getBody();
          switch (response.getStatus()) {
            case HttpStatusCode.OK:
            if (object instanceof Pessoa) {
              pessoa = (Pessoa) object;
              printPessoa(pessoa);
            }
            break;
            case HttpStatusCode.Conflict:
            if (object instanceof String) {
              String message = (String) object;
              System.out.println(HttpStatusCode.Conflict + "- Conflict:" + message);
            }
            default:
            break;
          }
```

```
        }
        break;
        case 3:

        System.out.println("\nRemover\n");

        nif = UIGeral.getNumber("NIF");
        response = PessoasController.deletePessoa(Constants.HOST + "api/pessoas/" + nif);
        if (response != null) {
          Object object = response.getBody();
          switch (response.getStatus()) {
            case HttpStatusCode.OK:
            System.out.println("Deleted");
            break;
            case HttpStatusCode.Conflict:
            if (object instanceof String) {
              String message = (String) object;
              System.out.println(HttpStatusCode.Conflict + "- Conflict:" + message);
            }
            default:
            break;
          }
        }

        break;
        case 4:

        System.out.println("\nAlterar\n");
        pessoa = getPessoa();
        pessoaDTO = Mapper.pessoa2PessoaDTO(pessoa);
        nif = pessoa.getNif();
        response = PessoasController.updatePessoa(Constants.HOST + "api/pessoas/" + nif, pessoaDTO);
        if (response != null) {
          Object object = response.getBody();
          switch (response.getStatus()) {
            case HttpStatusCode.OK:
            System.out.println("Updated");
            break;
            case HttpStatusCode.Conflict:
            if (object instanceof String) {
              String message = (String) object;
              System.out.println(HttpStatusCode.Conflict + "- Conflict:" + message);
            }
            default:
            break;
          }
        }
        break;

        case 5:
        System.out.println("\nListar (Todas)\n");
        response = PessoasController.getPessoas(Constants.HOST + "api/pessoas/");
        if (response != null) {
          Object object = response.getBody();
          switch (response.getStatus()) {
            case HttpStatusCode.OK:
            if (object instanceof List) {
              ArrayList<Pessoa> pessoas = (ArrayList) object;
              printPessoas(pessoas);
            }
            break;
            case HttpStatusCode.Conflict:
            if (object instanceof String) {
              String message = (String) object;
              System.out.println(HttpStatusCode.Conflict + "- Conflict:" + message);
            }
            default:
            break;
          }
        }
        break;
        default:
        System.out.println("Opção Errada");
```

```
        break;
      }

    } while (op != 0);
  }

  private static int menuPessoa() {
    int op;
    do {
      System.out.println("\nMenu Pessoas");
      System.out.println("1 - Inserir");
      System.out.println("2 - Listar");
      System.out.println("3 - Eliminar");
      System.out.println("4 - Alterar");
      System.out.println("5 - Listar (Todas)");
      System.out.println("\n0 - Voltar");
      op = (int) UIGeral.getNumber("--->");
    } while (op < 0 || op > 5);
    return op;
  }

  public static Pessoa getPessoa() {
    Pessoa pessoa = new Pessoa();
    boolean flag;
    do {
      try {
        flag = false;
        long nif = UIGeral.getNumber("NIF");
        pessoa.setNif(nif);
      } catch (NifInvalidoException e) {
        flag = true;
        System.out.println("Atenção: " + e.getMessage());
      }
    } while (flag);
    do {
      try {
        flag = false;
        String nome = UIGeral.getText("Nome");
        pessoa.setNome(nome);
      } catch (NomePessoaInvalidoException e) {
        flag = true;
        System.out.println("Atenção: " + e.getMessage());
      }
    } while (flag);
    Data data = UIData.getData();
    pessoa.setNascimento(data);
    return pessoa;
  }

  public static void printPessoa(Pessoa pessoa) {
    System.out.print("[" + pessoa.getNif() + "] " + pessoa.getNome() + ", " + pessoa.getNascimento().
          getDia() + "/" + pessoa.getNascimento().getMes() + "/" + pessoa.getNascimento().getAno());
  }

  public static void printPessoas(ArrayList<Pessoa> pessoas) {
    Pessoa pessoa = null;
    for (int i = 0; i < pessoas.size(); i++) {
      pessoa = pessoas.get(i);
      printPessoa(pessoa);
      System.out.print("\n");
    }
  }
}
```

- `UIFuncionario` class.

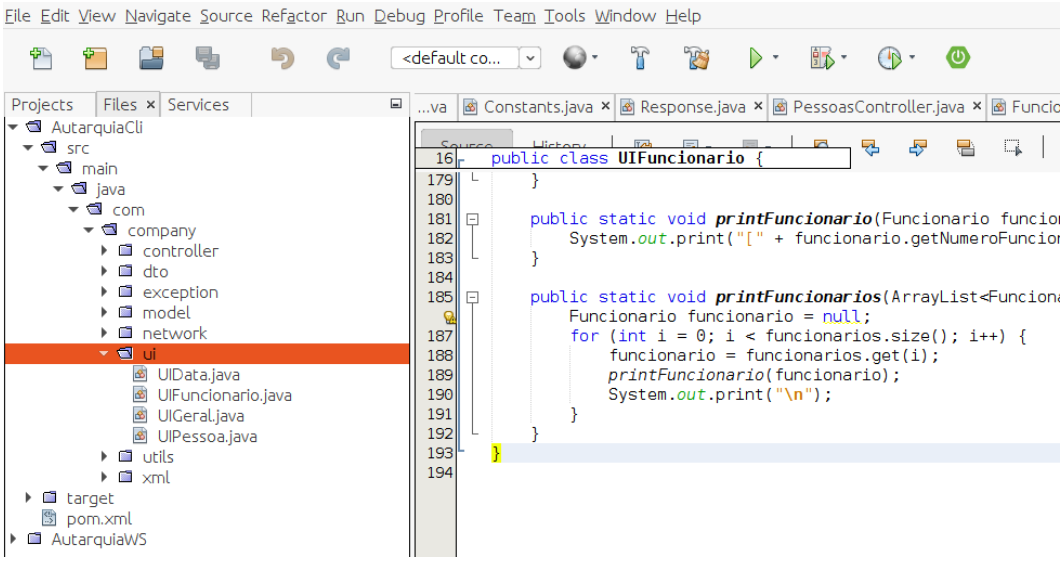
```
package com.company.ui;

import com.company.controller.FuncionariosController;
import com.company.dto.FuncionarioDTO;
import com.company.dto.Mapper;
```

```
import com.company.exception.NumeroFuncionarioInvalidoException;
import com.company.model.Funcionario;
import com.company.model.Pessoa;
import com.company.network.HttpStatusCode;
import com.company.utils.Constants;
import com.company.utils.Response;

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

public class UIFuncionario {

  public static void mainFuncionario() {
    Funcionario funcionario = null;
    Response response = null;
    FuncionarioDTO funcionarioDTO = null;
    int numero;
    int op;
    do {
      op = menuFuncionario();
      switch (op) {
        case 0:
        System.out.println("Volta para o menu anterior.");
        break;
        case 1:
        System.out.println("\nInserir\n");
        funcionario = getFuncionario();
        funcionarioDTO = Mapper.funcionario2FuncionarioDTO(funcionario);
        response = FuncionariosController.addFuncionario(Constants.HOST + "api/funcionarios",
              funcionarioDTO);
        if (response != null) {
          Object object = response.getBody();
          switch (response.getStatus()) {
            case HttpStatusCode.Created:
            System.out.println(HttpStatusCode.Created + "- Created");
            break;
            case HttpStatusCode.Conflict:
            if (object instanceof String) {
              String message = (String) object;
              System.out.println(HttpStatusCode.Conflict + "- Conflict:" + message);
            }
            default:
            break;
          }
        }
        break;
        case 2:
        System.out.println("\nPesquisar\n");
        numero = (int) UIGeral.getNumber("Numero");
        response = FuncionariosController.getFuncionario(Constants.HOST + "api/funcionarios/" + numero);
        if (response != null) {
          Object object = response.getBody();
          switch (response.getStatus()) {
            case HttpStatusCode.OK:
            if (object instanceof Funcionario) {
              funcionario = (Funcionario) object;
              printFuncionario(funcionario);
            }
            break;
            case HttpStatusCode.Conflict:
            if (object instanceof String) {
              String message = (String) object;
              System.out.println(HttpStatusCode.Conflict + "- Conflict:" + message);
            }
            default:
            break;
          }
        }
        break;
        case 3:
        System.out.println("\nRemover\n");
        numero = (int) UIGeral.getNumber("Numero");
        response = FuncionariosController.deleteFuncionario(Constants.HOST + "api/funcionarios/" + numero)
```

```
                ;
        if (response != null) {
          Object object = response.getBody();
          switch (response.getStatus()) {
            case HttpStatusCode.OK:
            System.out.println("Deleted");
            break;
            case HttpStatusCode.Conflict:
            if (object instanceof String) {
              String message = (String) object;
              System.out.println(HttpStatusCode.Conflict + "- Conflict:" + message);
            }
            default:
            break;
          }
        }
        break;
        case 4:
        System.out.println("\nAlterar\n");
        funcionario = getFuncionario();
        funcionarioDTO = Mapper.funcionario2FuncionarioDTO(funcionario);
        numero = funcionario.getNumeroFuncionario();
        response = FuncionariosController.updateFuncionario(Constants.HOST + "api/funcionarios/" + numero,
               funcionarioDTO);
        if (response != null) {
          Object object = response.getBody();
          switch (response.getStatus()) {
            case HttpStatusCode.OK:
            System.out.println("Updated");
            break;
            case HttpStatusCode.Conflict:
            if (object instanceof String) {
              String message = (String) object;
              System.out.println(HttpStatusCode.Conflict + "- Conflict:" + message);
            }
            default:
            break;
          }
        }
        break;
        case 5:
        System.out.println("\nListar (Todas)\n");
        response = FuncionariosController.getFuncionarios(Constants.HOST + "api/funcionarios/");
        if (response != null) {
          Object object = response.getBody();
          switch (response.getStatus()) {
            case HttpStatusCode.OK:
            if (object instanceof List) {
              ArrayList<Funcionario> funcionarios = (ArrayList<Funcionario>) object;
              printFuncionarios(funcionarios);
            }
            break;
            case HttpStatusCode.Conflict:
            if (object instanceof String) {
              String message = (String) object;
              System.out.println(HttpStatusCode.Conflict + "- Conflict:" + message);
            }
            default:
            break;
          }
        }
        break;
        default:
        System.out.println("Opção Errada");
        break;
      }
    } while (op != 0);
  }

  private static int menuFuncionario() {
    int op;
    do {
      System.out.println("\nMenu Funcionario");
```

```
        System.out.println("1 - Inserir ");
        System.out.println("2 - Listar ");
        System.out.println("3 - Eliminar ");
        System.out.println("4 - Alterar ");
        System.out.println("5 - Listar (Todos) ");
        System.out.println("\n0 - Voltar");
        op = (int) UIGeral.getNumber("--->");
      } while (op < 0 || op > 5);
      return op;
    }

    public static Funcionario getFuncionario() {
      Pessoa pessoa = UIPessoa.getPessoa();
      Funcionario funcionario = new Funcionario(pessoa.getNif(), pessoa.getNome(), pessoa.getNascimento());
      boolean flag;
      do {
        flag = false;
        try {
          int numeroFuncionario = (int) UIGeral.getNumber("Numero");
          funcionario.setNumeroFuncionario(numeroFuncionario);
        } catch (NumeroFuncionarioInvalidoException e) {
          flag = true;
          System.out.println("Atenção: " + e.getMessage());
        }
      } while (flag);

      String cargo = UIGeral.getText("Cargo");
      funcionario.setCargo(cargo);

      return funcionario;
    }

    public static void printFuncionario(Funcionario funcionario) {
      System.out.print("[" + funcionario.getNumeroFuncionario() + "]" + funcionario.getNome() + ", " +
            funcionario.getCargo() + ", " + funcionario.getNif() + "," + funcionario.getNascimento().getDia
            () + "/" + funcionario.getNascimento().getMes() + "/" + funcionario.getNascimento().getAno());
    }

    public static void printFuncionarios(ArrayList<Funcionario> funcionarios) {
      Funcionario funcionario = null;
      for (int i = 0; i < funcionarios.size(); i++) {
        funcionario = funcionarios.get(i);
        printFuncionario(funcionario);
        System.out.print("\n");
      }
    }
  }
```

In the end, the content of the `ui` package must be similar to the shown in the following figure.



1.3.10 Coding `AutarquiaCliApplication` class

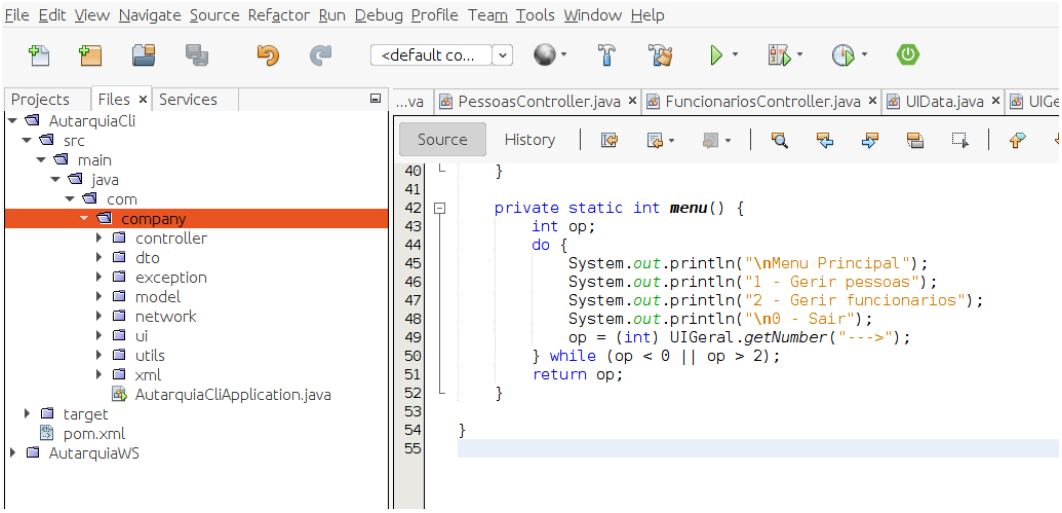
The codeAutarquiaCliApplication implements `main` method and must be placed in the main package, `company`.

- `UIGeral` class.

```
/*
* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package com.company;

import com.company.ui.UIFuncionario;
import com.company.ui.UIGeral;
import com.company.ui.UIPessoa;

/**
*
* @author pbs
*/
public class AutarquiaCliApplication {
  public static void main(String[] args) {
    int op;
    System.out.println("AutarquiaCliApplication iniciou.");
    do {
      op = menu();
      switch (op) {
        case 0:
        System.out.println("AutarquiaCliApplication terminou. Adeus.");
        break;
        case 1:
        UIPessoa.mainPessoa();
        break;
        case 2:
        UIFuncionario.mainFuncionario();
        break;
        default:
        System.out.println("Opção Errada");
```

```
        break;
      }
    } while (op != 0);



  }

  private static int menu() {
    int op;
    do {
      System.out.println("\nMenu Principal");
      System.out.println("1 - Gerir pessoas");
      System.out.println("2 - Gerir funcionarios");
      System.out.println("\n0 - Sair");
      op = (int) UIGeral.getNumber("--->");
    } while (op < 0 || op > 2);
    return op;
  }

}
```

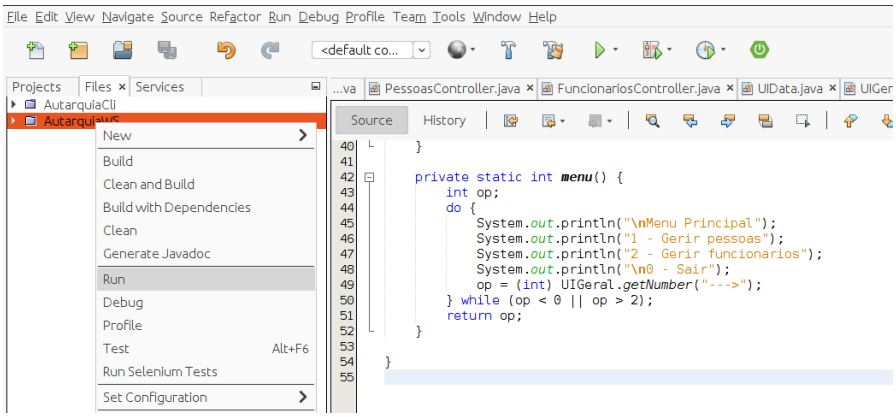
In the end, the content of the `company` package must be similar to the shown in the following figure.



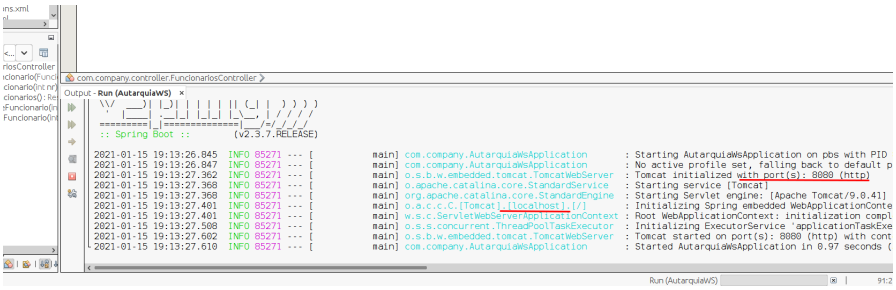
2 Deploying the `AutarquiaWS` web service

2.1 Deploying the `AutarquiaWS` web service

To do this, using NetBeans IDE, right-click on `AutarquiaWS` and select `Run`.

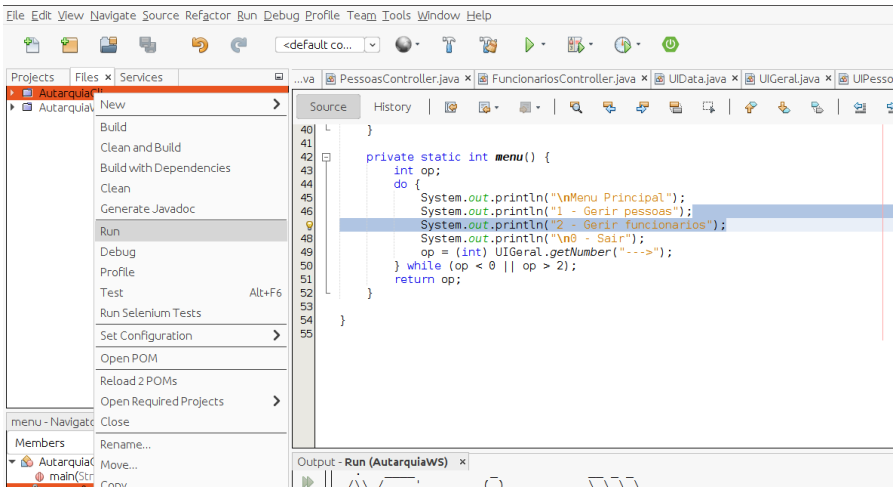


If everything goes well, the following messages appear into the IDE console.

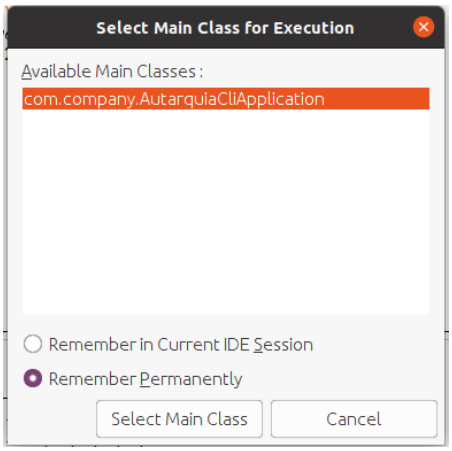


2.2 Deploying the `AutarquiaCli` console application

To do this, using NetBeans IDE, right-click on `AutarquiaCli` and select `Run`.



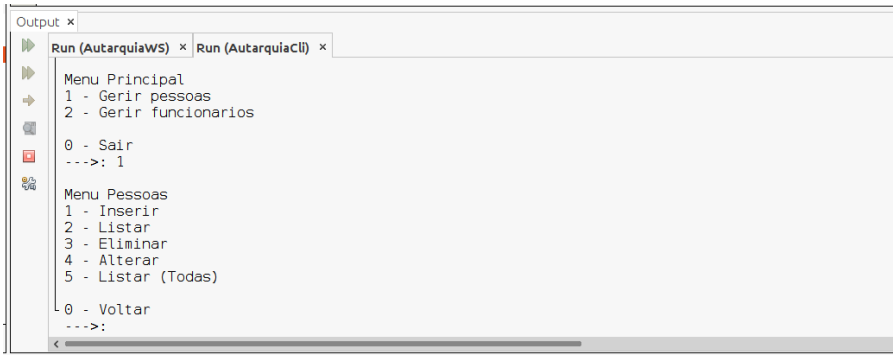
Select the Main Class.



If everything goes well, a console is open for each application, `AutarquiaWS` and `AutarquiaCli`.

3 Testing

In the `AutarquiaCli` console application select `1` to enter into the `Pessoas` functionalities.



Then interact with application following the menu messages. For instance, if

the `AutarquiaWS` has data, namely, `Pessoas` data, you can get a list of `Pessoas` selecting `5` option.