Java

Server-side programming

Tutorial

Server-side programming assignment: Developing a web service

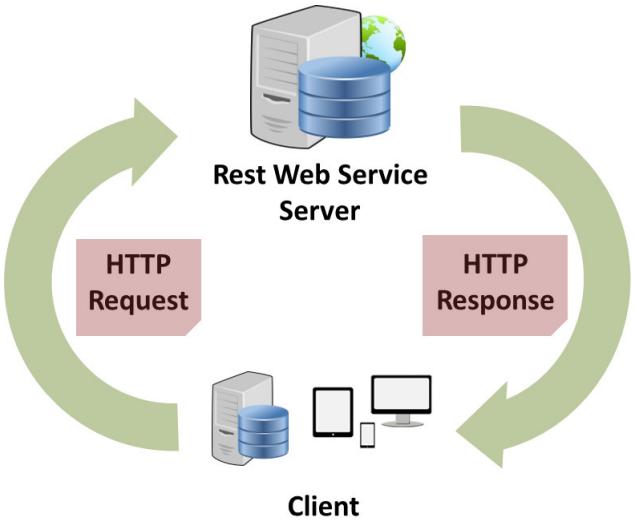
Paulo Baltarejo Sousa, Joaquim Peixoto dos Santos
{pbs,jpe}@isep.ipp.pt
2021

1 What is REST?

REST stands for **REpresentational State Transfer** and it is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web.

The REST architectural style constrains an architecture to a **client/server architecture** and is designed to use a stateless communication protocol, typically HTTP. Clients make **HTTP requests** and servers reply **HTTP responses**. It revolves around resources (i.e., images, files and others) and a resource is accessed by a common interface using **HTTP standard methods**.

In REST architecture, a REST Server simply provides access to resources and the REST client accesses and presents the resources. Each **resource is identified by Uniform Resource Identifier (URI)**.



2 HTTP Methods

The following HTTP methods are most commonly used in a REST based architecture.

- GET - Provides a read only access to a resource.
- PUT - Used to create a new resource.
- DELETE - Used to remove a resource.

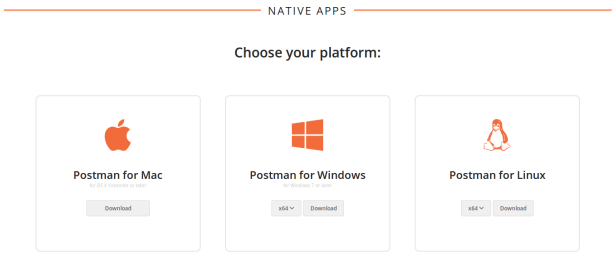
- POST - Used to update an existing resource or create a new resource.

3 Setting up Developing Environment

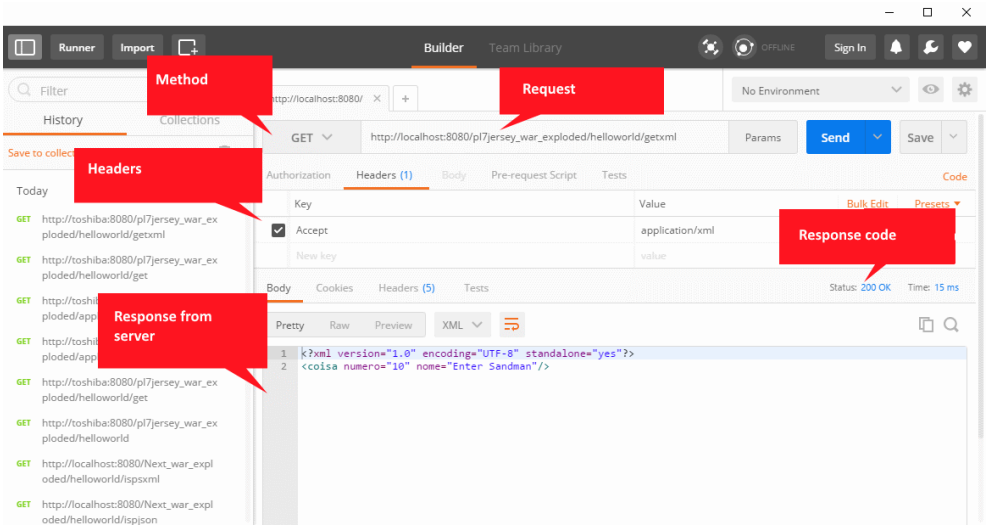
3.1 Postman

Postman is a powerful HTTP client for testing web services.

1. Download Postman app from `https://www.postman.com/downloads/`.

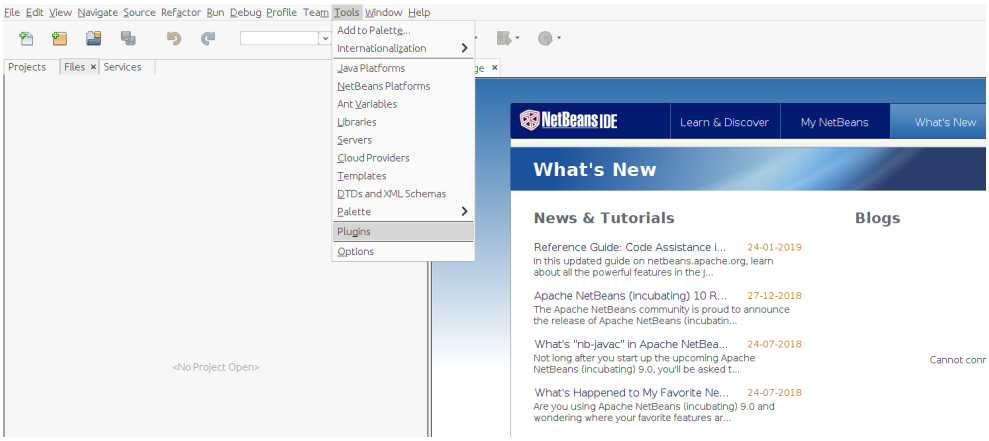


2. Install it
   - Installation procedure can be found at `https://learning.postman.com/docs/getting-started/installation-and-updates/`
3. Launch it

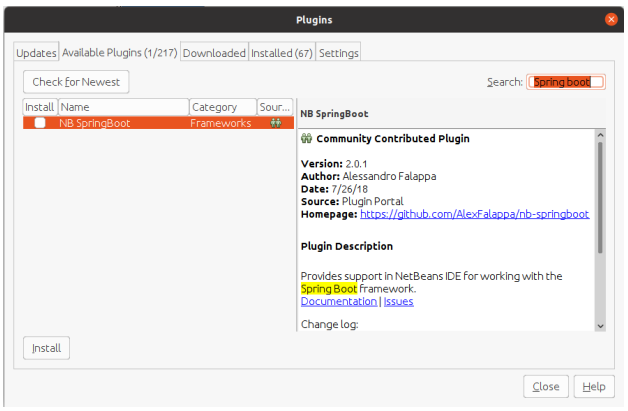


3.2 Spring boot plugin

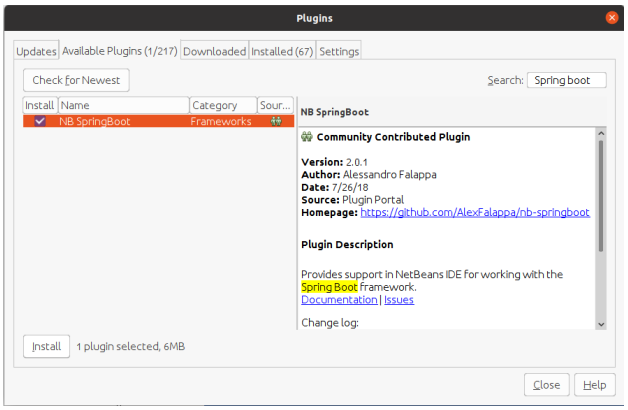
1. Open the Netbeans IDE.
2. On the Menu Bar choose `Tools` and further choose the `Plugins` menu option.



3. Select `Available Plugins` tab from Plugins dialog box.
4. Type in "Spring boot" in the search box.

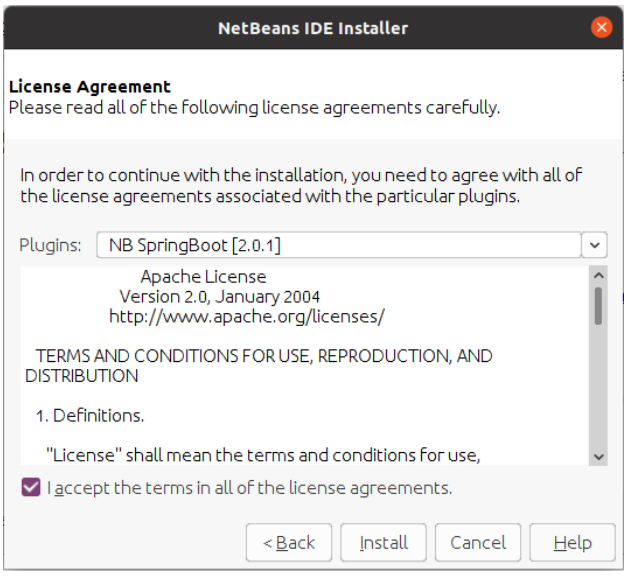


5. Check it and click the `Install` button

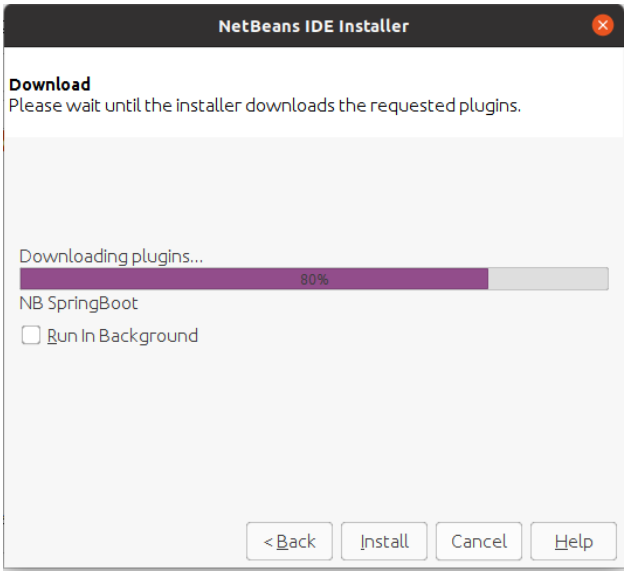


6. A window pops up; Click the `Next` button

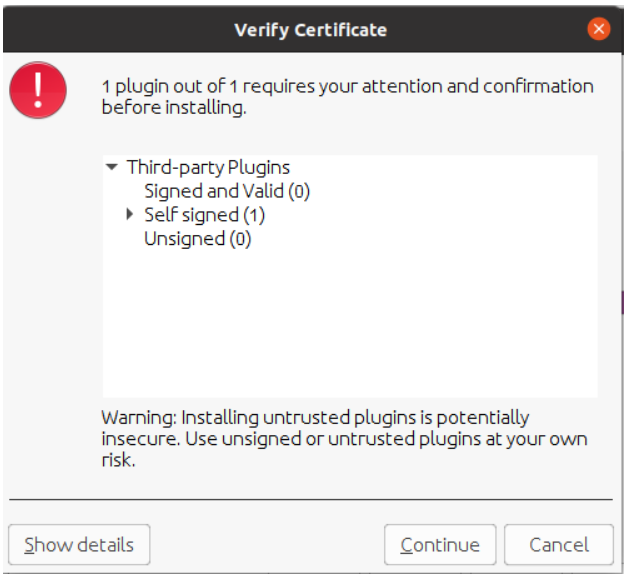
7. Check "I accept...." and click the `Install` button.



Downloading plugin.

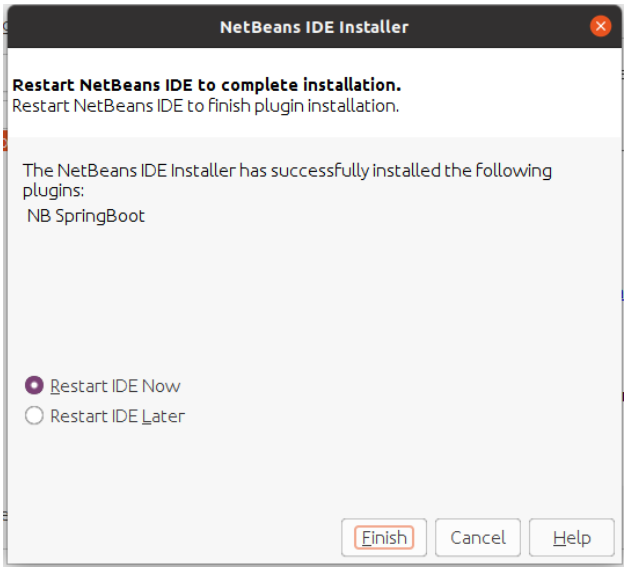


8. Since the plugin would be designed by a 3rd party developer it may not be digitally signed and you would need to ignore and continue. Click the `Continue` button.



9. The plugin will install automatically.

10. Once the installation is finished, you should restart the IDE. Click the `Finish` button.



The installation of the Plugin is complete and it can now be used with the IDE.

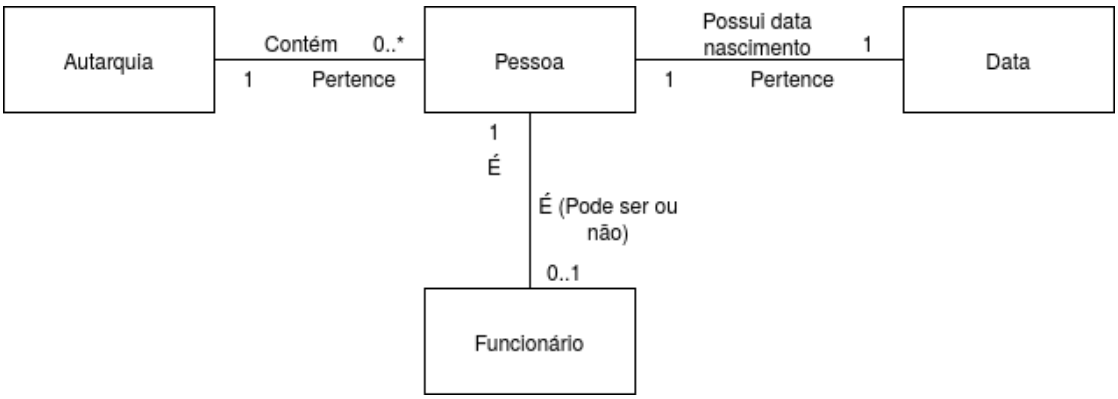
4 Creating a Java RESTFul Web Service

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

4.1 Domain Model

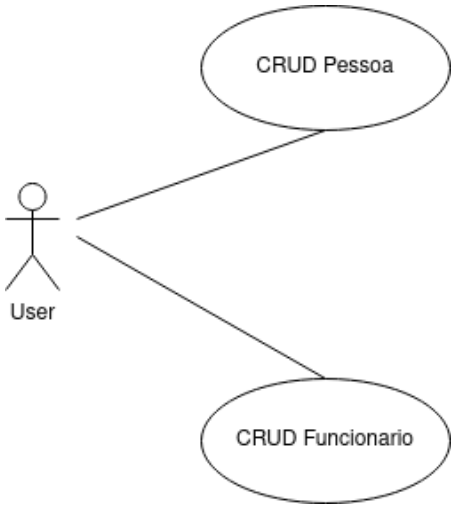
A domain model is a visual representation of conceptual classes or real-world objects (also called entities) in a domain of interest. In this particular case, there are four entities: `Autarquia`, `Pessoa`, `Data` and `Funcionario`. An `Autarquia` is an entity that models the local authority (autarchy). A `Pessoa` is an entity that models a person that reside in such a autarchy. `Data` is an entity to capture the concept of date. Finally, `Funcionario` is an entity that models the concept that a specific person could also be autarchy employee. More:

- An `Autarquia` is composed by an unlimited number of `Pessoa`.
- A `Data` is used to model a date, in this particular case, the birthday of a `Pessoa`.
- A `Funcionario` is a particular case of a `Pessoa`



4.2 Use Case Diagram

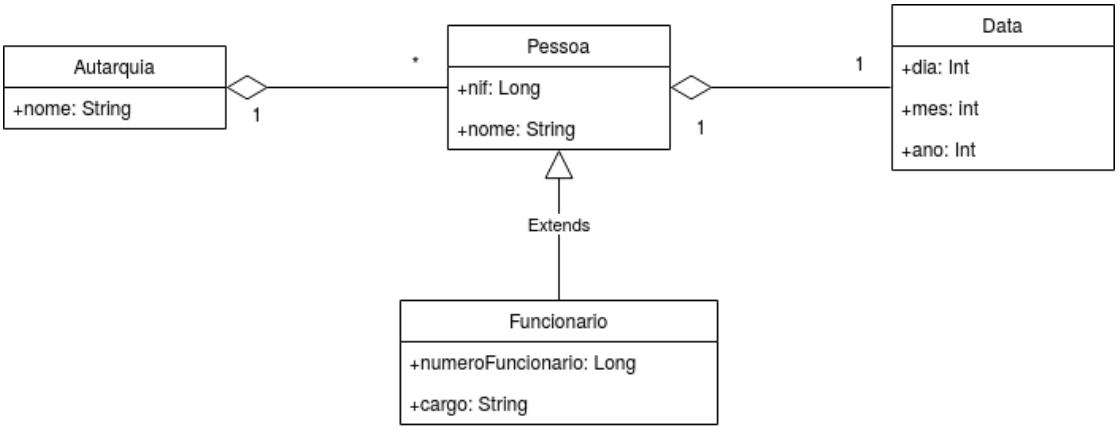
The `AutarquiaWS` has the following use cases.



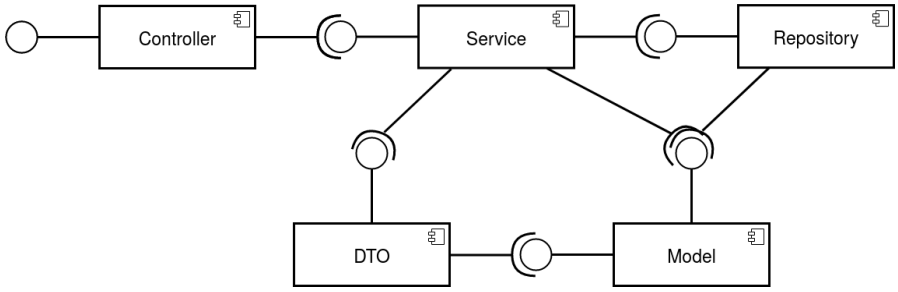
For the sake of simplicity, the functionalities were grouped by CRUD (Create, Read, Update and Delete) operations.

4.3 Model Class Diagram

From the domain model, we could derive the class diagram.



4.4 Software architecture (Component Diagram)



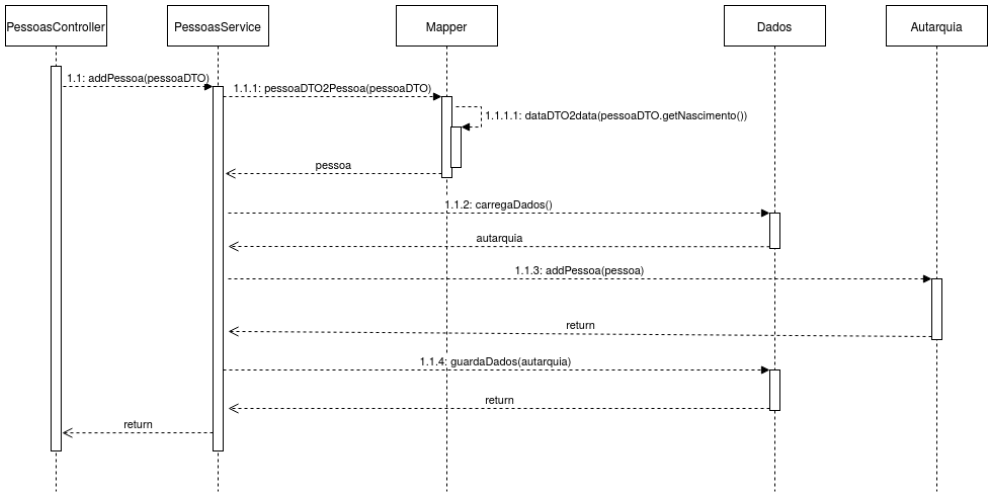
4.5 Endpoints

The Rest API provide the following endpoints:

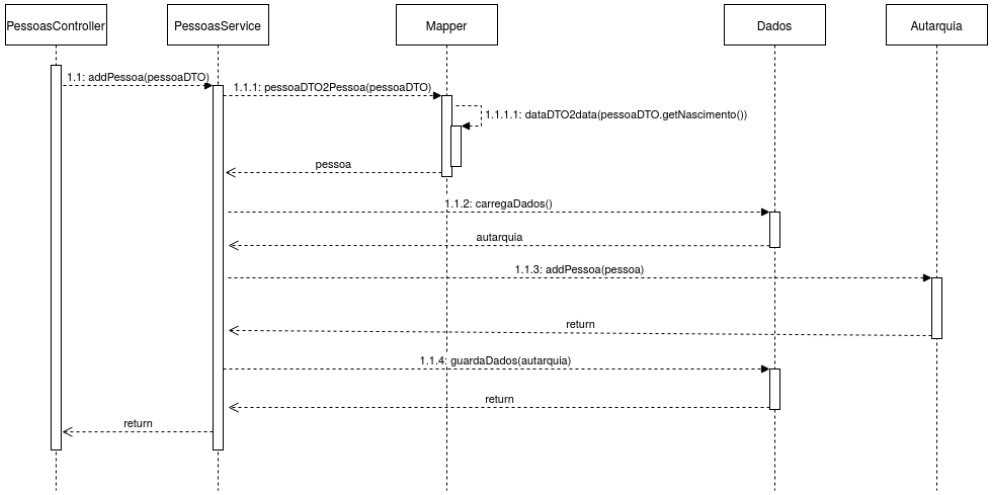
Method	URI	Operation
GET	`/api/pessoas`	Get list of people
GET	`/api/pessoas/1`	Get person with `nif 1`
PUT	`/api/pessoas/2`	Update person with `nif 2`
POST	`/api/pessoas`	Create a new person
DELETE	`/api/pessoas/1`	Delete Person with `nif 1`
GET	`/api/funcionarios`	Get list of employees
GET	`/api/pessoas/1`	Get employee with `nr 1`
PUT	`/api/pessoas/2`	Update employee with `nr 2`
POST	`/api/pessoas`	Create a new employee
DELETE	`/api/pessoas/1`	Delete employee with `nr 1`

4.6 Sequence Diagrams

- Create Person

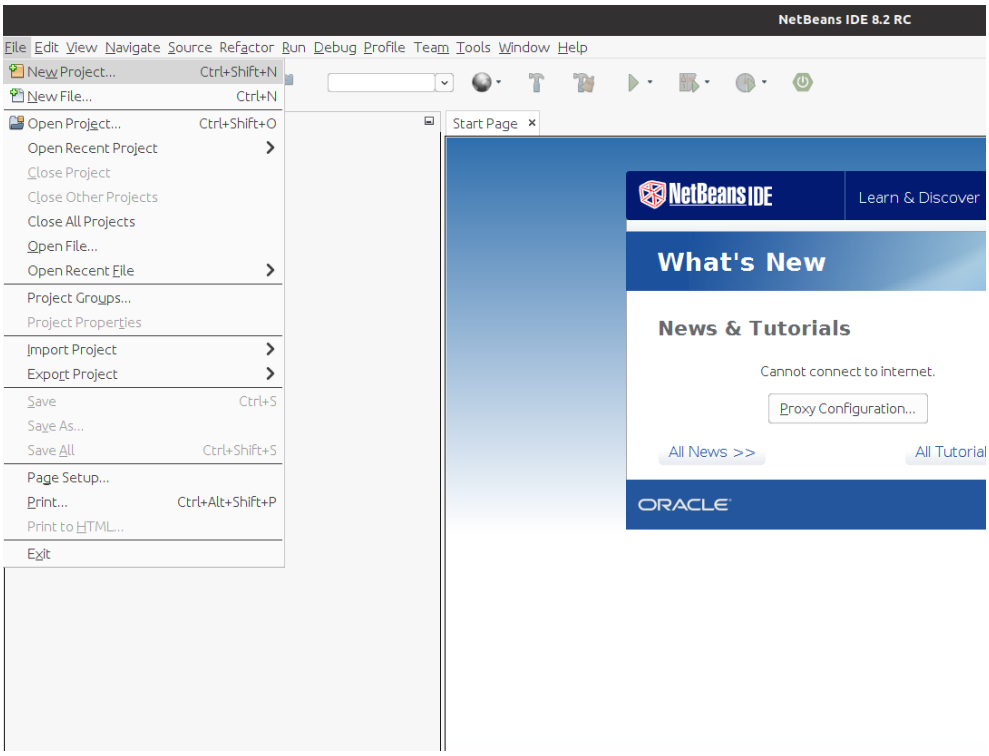


- Get Person

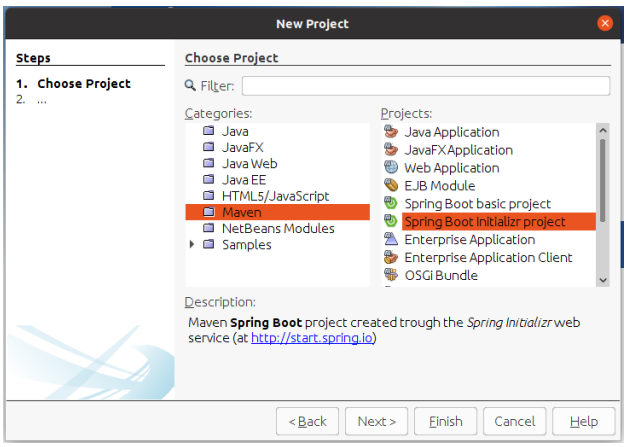


4.7 Create Project

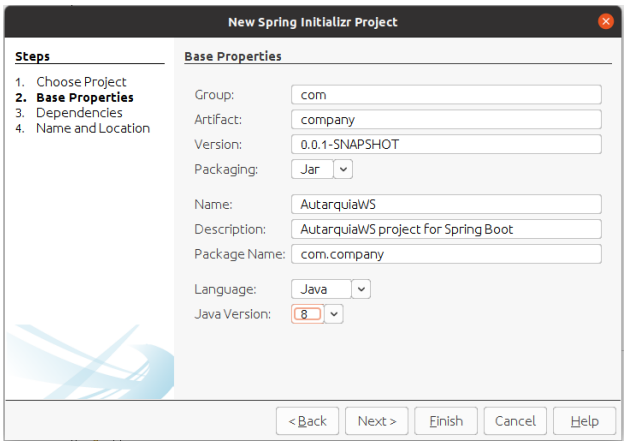
1. Open the Netbeans IDE.
2. Select `File->New Project`.



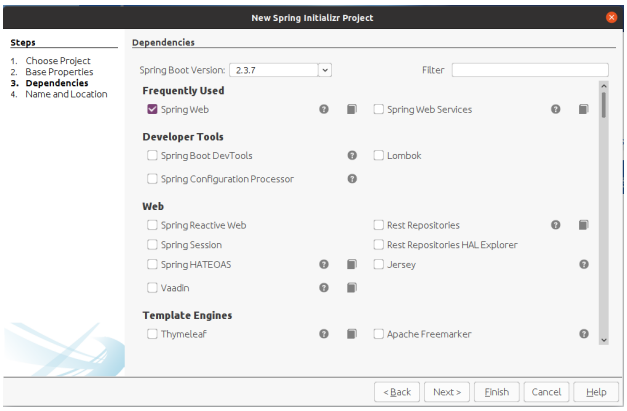
3. From `Categories`, select `Maven`. From `Projects`, select `Spring Boot Initializr project`. Click the `Next` button.



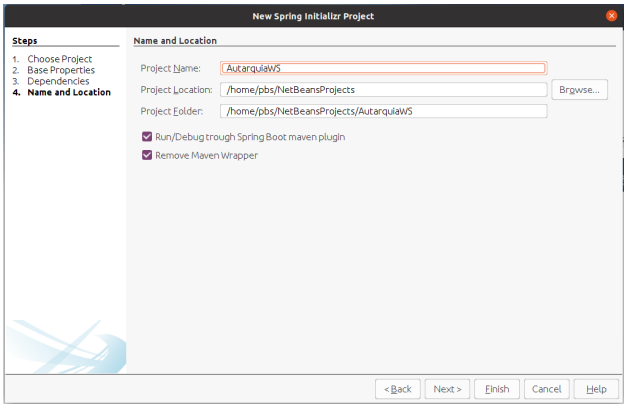
4. Fill in the form "Base Properties" fields as shown in the following figure. Click the `Next` button.



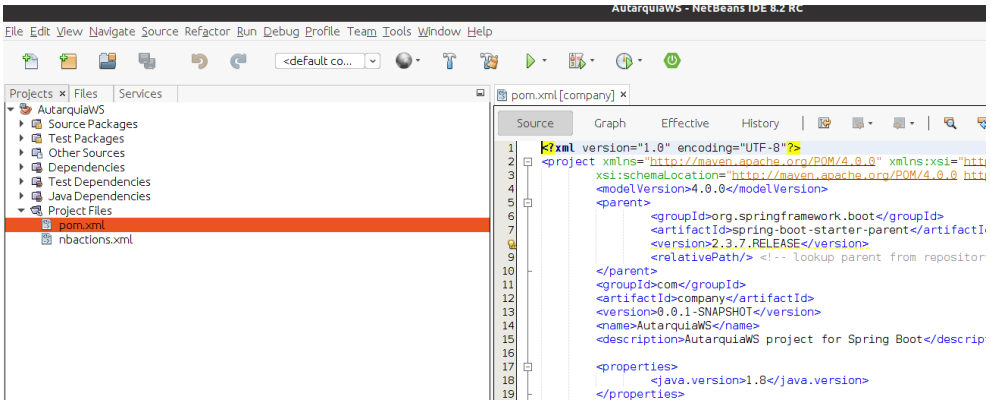
5. Check `Spring Web` option. Click the `Next` button.



6. Name the project. Click the `Finish` button.

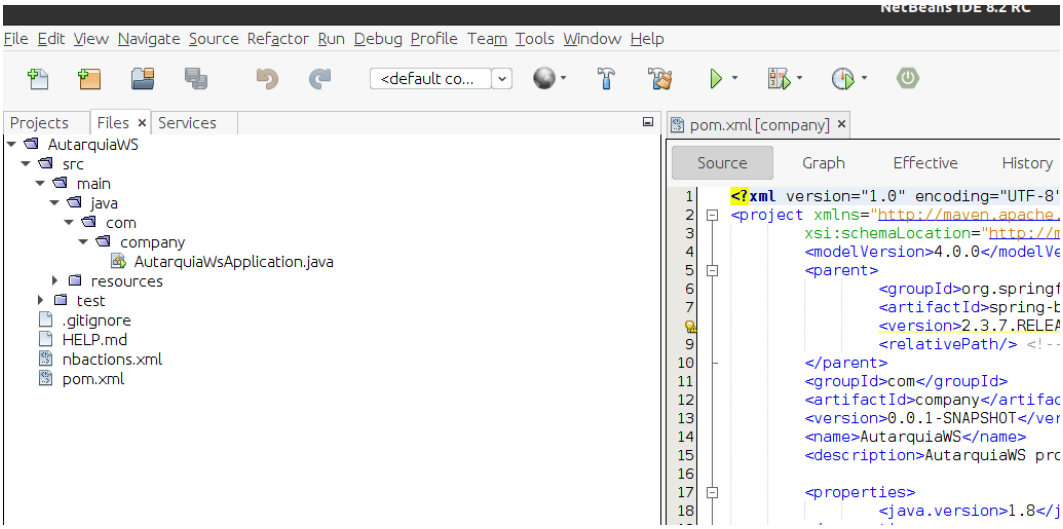


The project is create and ready for coding.



4.8 Coding Project

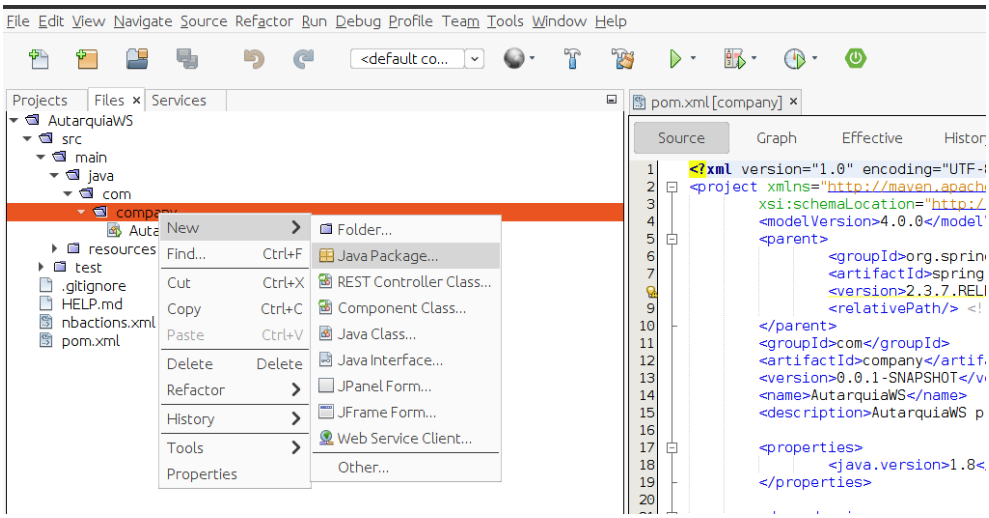
For coding the project, the appropriate view is the provided when selecting the `Files` tab in the project explore window.



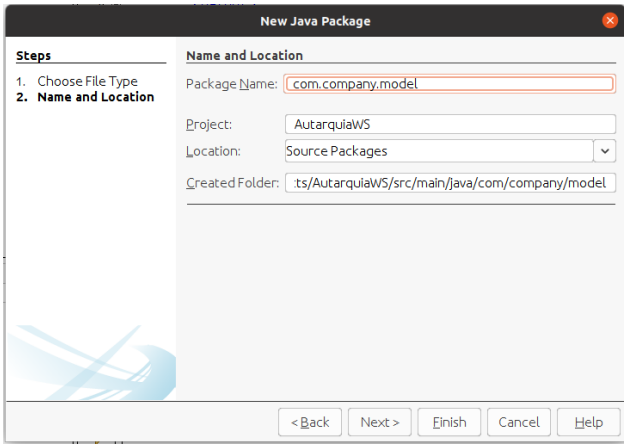
4.8.1 Organizing code

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`.

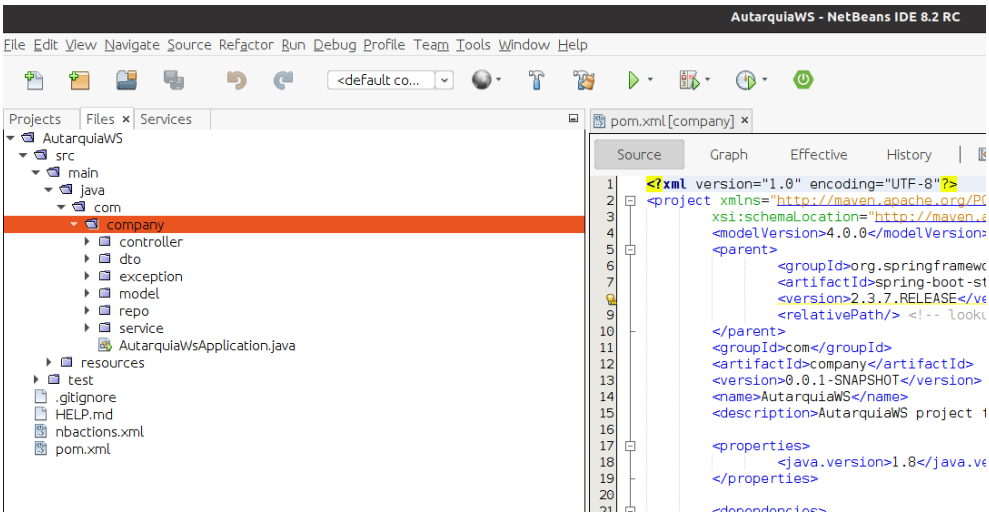


2. Enter name `model` and click the `Finish` button.



3. Repeat the procedure for creating the following packages:

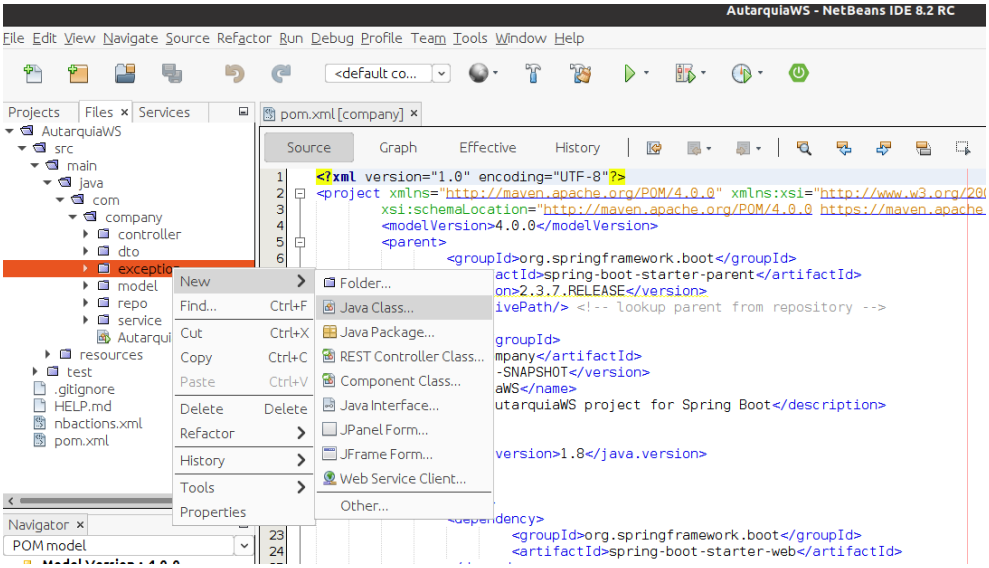
   - `controller`
   - `service`
   - `repo`
   - `dto`
   - `exception`



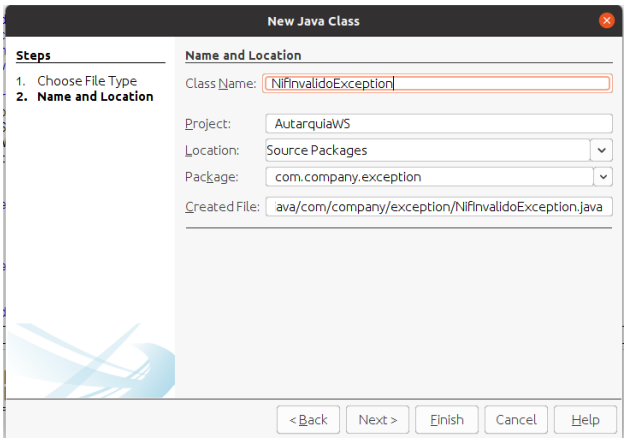
4.8.2 Coding `exception`

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);
  }
}
```

- `ElementoNaoExistenteException` class.

```
public class ElementoNaoExistenteException extends RuntimeException {

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

- `NifDuplicadoException` class.

```
public class NifDuplicadoException extends RuntimeException {

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

- `NomePessoaInvalidoException` class.

```
public class NomePessoaInvalidoException extends RuntimeException {

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

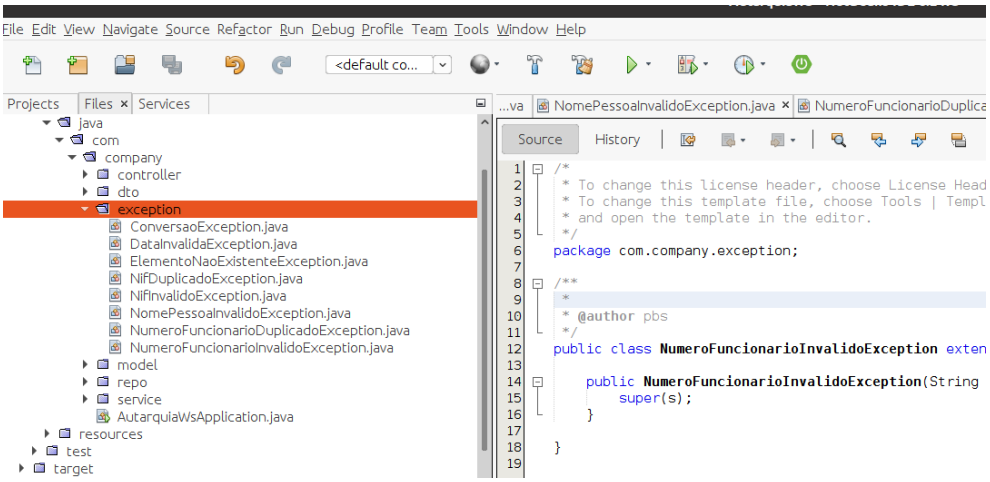
- `NumeroFuncionarioDuplicadoException` class.

```
public class NumeroFuncionarioDuplicadoException extends RuntimeException {

  public NumeroFuncionarioDuplicadoException(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.



4.8.3 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;
  }

}
```

- `Autarquia` class.

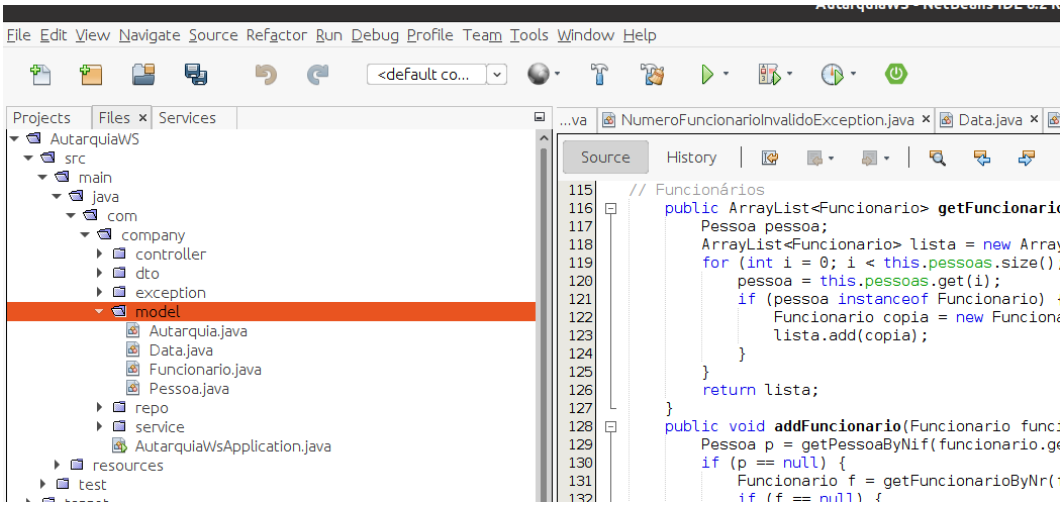
```
import com.company.exception.ElementoNaoExistenteException;
import com.company.exception.NifDuplicadoException;
import com.company.exception.NumeroFuncionarioDuplicadoException;

import java.io.Serializable;
import java.util.ArrayList;

public class Autarquia implements Serializable {
  private String nome;
  private ArrayList<Pessoa> pessoas;

  public Autarquia(String nome) {
    this.nome = nome;
    this.pessoas = new ArrayList<Pessoa>();
  }

  public ArrayList<Pessoa> getAllPessoas() {
    Pessoa pessoa;
    ArrayList<Pessoa> lista = new ArrayList<>();
    for (int i = 0; i < this.pessoas.size(); i++) {
      pessoa = this.pessoas.get(i);
      if (!(pessoa instanceof Funcionario)) {
        Pessoa copiaPessoa = new Pessoa(pessoa);
        lista.add(copiaPessoa);
      } else {
        Funcionario funcionarioFuncionario = new Funcionario((Funcionario) pessoa);
        lista.add(funcionarioFuncionario);
      }
    }
    return lista;
  }
  ////////////////////////////////////////////////
  public ArrayList<Pessoa> getPessoas() {
    Pessoa pessoa;
    ArrayList<Pessoa> lista = new ArrayList<>();
    for (int i = 0; i < this.pessoas.size(); i++) {
```

```
    pessoa = this.pessoas.get(i);
    if (!(pessoa instanceof Funcionario)) {
      Pessoa copia = new Pessoa(pessoa);
      lista.add(copia);
    }
  }
  return lista;
}

public void addPessoa(Pessoa pessoa) throws NifDuplicadoException {
  Pessoa p = getPessoaByNif(pessoa.getNif());
  if (p == null) {
    this.pessoas.add(pessoa);
  } else {
    throw new NifDuplicadoException(p.getNif() + ": NIF já existe");
  }

}

public Pessoa getPessoa(long nif) {
  return getPessoaByNif(nif);
}

public void removePessoa(long nif) throws ElementoNaoExistenteException {
  Pessoa pessoa = null;
  for (int i = 0; i < this.pessoas.size(); i++) {
    pessoa = this.pessoas.get(i);
    if (pessoa.getNif() == nif) {
      if (!(pessoa instanceof Funcionario)) {
        this.pessoas.remove(i);
        return;
      } else {
        throw new ElementoNaoExistenteException(nif + ": Não éuma pessoa, éum funcionário");
      }
    }
  }
  throw new ElementoNaoExistenteException(nif + ": Não existe essa pessoa");
}

public void updatePessoa(long nif, Pessoa p) throws ElementoNaoExistenteException {
  Pessoa pessoa = null;
  boolean updated = false;
  for (int i = 0; i < this.pessoas.size() && !updated; i++) {
    pessoa = this.pessoas.get(i);
    if (pessoa.getNif() == nif) {
      pessoa = p;
      updated = true;
    }
  }
  if(updated == false){
    throw new ElementoNaoExistenteException(nif + ": Não existe essa pessoa");
  }
}

private Pessoa getPessoaByNif(long nif) {
  Pessoa pessoa = null;
  for (int i = 0; i < this.pessoas.size(); i++) {
    pessoa = this.pessoas.get(i);
    if (pessoa.getNif() == nif) {
      Pessoa copia = new Pessoa(pessoa);
      return copia;
    }
  }
  return null;
}
////////////////////////////////////////////////
// Funcionários
public ArrayList<Funcionario> getFuncionarios() {
  Pessoa pessoa;
  ArrayList<Funcionario> lista = new ArrayList<>();
  for (int i = 0; i < this.pessoas.size(); i++) {
    pessoa = this.pessoas.get(i);
    if (pessoa instanceof Funcionario) {
```

```
      Funcionario copia = new Funcionario((Funcionario) pessoa);
      lista.add(copia);
    }
  }
  return lista;
}
public void addFuncionario(Funcionario funcionario) throws NumeroFuncionarioDuplicadoException,
      NifDuplicadoException{
  Pessoa p = getPessoaByNif(funcionario.getNif());
  if (p == null) {
    Funcionario f = getFuncionarioByNr(funcionario.getNumeroFuncionario());
    if (f == null) {
      addPessoa(funcionario);
    } else {
      throw new NumeroFuncionarioDuplicadoException(f.getNumeroFuncionario() + ": Número de Funcionário já
            existe");
    }
  } else {
    throw new NifDuplicadoException(p.getNif() + ": NIF já existe");
  }
}

public Funcionario getFuncionario(int nr) {
  Funcionario funcionario = getFuncionarioByNr(nr);
  if (funcionario != null) {
    Funcionario copia = new Funcionario(funcionario);
    return copia;
  }
  return null;
}

public void removeFuncionario(int nr) throws ElementoNaoExistenteException {
  Pessoa pessoa = null;
  Funcionario funcionario = null;
  for (int i = 0; i < this.pessoas.size(); i++) {
    pessoa = this.pessoas.get(i);
    if (pessoa instanceof Funcionario) {
      if (pessoa instanceof Funcionario) {
        funcionario = (Funcionario) pessoa;
        if (funcionario.getNumeroFuncionario() == nr) {
          this.pessoas.remove(i);
          return;
        }
      }
    }
  }
  throw new ElementoNaoExistenteException(nr + ": Não existe esse funcionário");
}

public void updateFuncionario(int nr, Funcionario f) throws ElementoNaoExistenteException {
  boolean updated = false;
  Pessoa pessoa = null;
  Funcionario funcionario = null;
  for (int i = 0; i < this.pessoas.size() && !updated; i++) {
    pessoa = this.pessoas.get(i);
    if (pessoa instanceof Funcionario) {
      funcionario = (Funcionario) pessoa;
      if (funcionario.getNumeroFuncionario() == nr) {
        funcionario = f;
        updated = true;
      }
    }
  }
  if(updated == false){
    throw new ElementoNaoExistenteException(nr + ": Não existe esse funcionario");
  }
}

private Funcionario getFuncionarioByNr(int nr) {
  Pessoa pessoa = null;
  Funcionario funcionario = null;
  for (int i = 0; i < this.pessoas.size(); i++) {
    pessoa = this.pessoas.get(i);
```

```
      if (pessoa instanceof Funcionario) {
        funcionario = (Funcionario) pessoa;
        if (funcionario.getNumeroFuncionario() == nr) {
          return funcionario;
        }
      }
    }
    return null;
  }
}
```

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



4.8.4 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=
...
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
      <exclusions>
        <exclusion>
```

```
        <groupId>org.junit.vintage</groupId>
        <artifactId>junit-vintage-engine</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</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;
```

```
  }
}
```

```
\item \code{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() {
```

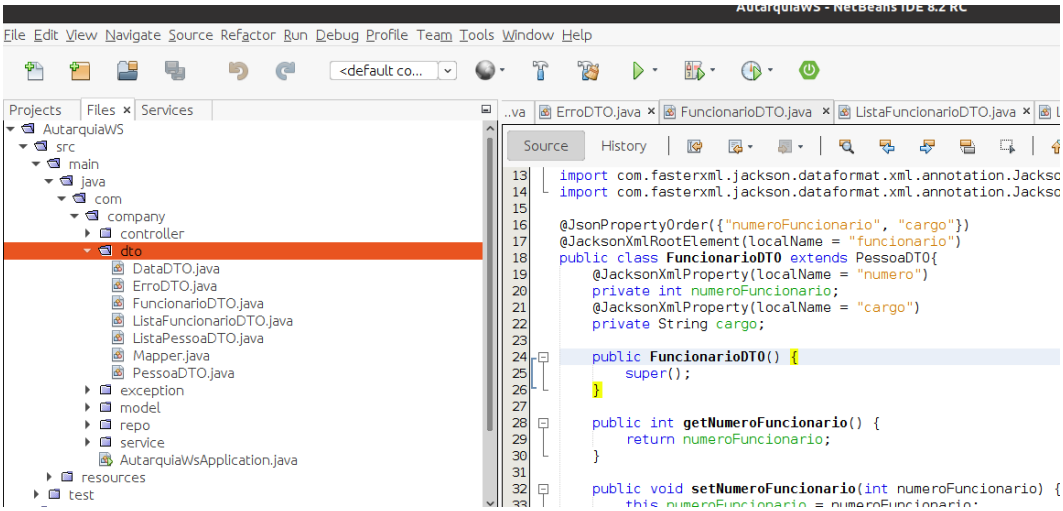
```
  }

  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.

```
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 listPessoa2PessoaDTO(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 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 listFuncionario2FuncionarioDTO(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;
  }
}
```

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



4.8.5 Coding `repo`

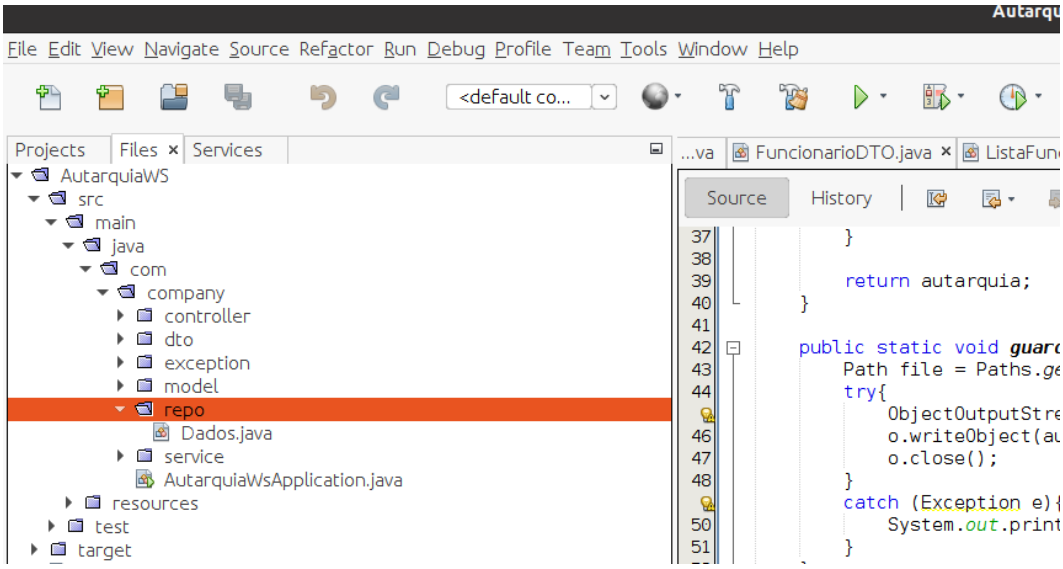
This package implements the functions required to persist the application data. In this case, the data is stored in a binary file, called `autarquia\_dados.dat`:

- `Dados` class.

```
import com.company.model.Autarquia;

import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

import static java.nio.file.StandardOpenOption.CREATE;

public class Dados {
  static final String AUTARQUIA_FILE ="autarquia_dados.dat";

  public static Autarquia carregarDados(){
    Autarquia autarquia = new Autarquia("Curral de Moinas");
    Path file = Paths.get(AUTARQUIA_FILE);
    try {
      ObjectInputStream o = new ObjectInputStream(new FileInputStream(file.toString()));
      autarquia = (Autarquia)o.readObject();
      o.close();
    }
    catch (Exception e) {
      System.out.println(e.getMessage());
    }

    return autarquia;
  }

  public static void guardarDados(Autarquia autarquia) {
    Path file = Paths.get(AUTARQUIA_FILE);
    try{
      ObjectOutputStream o = new ObjectOutputStream(Files.newOutputStream(file, CREATE));
      o.writeObject(autarquia);
      o.close();
    }
```

```
    catch (Exception e){
      System.out.println(e);
    }
  }
}
```

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



4.8.6 Coding `service`

Services classes will bridge the controllers and repository data. The `service` package classes:

- `PessoasService` class.

```
import com.company.dao.Dados;
import com.company.dto.ListaPessoaDTO;
import com.company.dto.Mapper;
import com.company.dto.PessoaDTO;
import com.company.exception.ConversaoException;
import com.company.model.Autarquia;
import com.company.model.Pessoa;

import java.util.ArrayList;

public class PessoasService {

  public static ListaPessoaDTO getPessoas() {
    ListaPessoaDTO listaPessoaDTO = null;
    Autarquia autarquia = Dados.carregarDados();
    ArrayList<Pessoa> pessoas = autarquia.getAllPessoas();
    listaPessoaDTO = Mapper.listPessoa2PessoaDTO(pessoas);
    return listaPessoaDTO;
  }
```

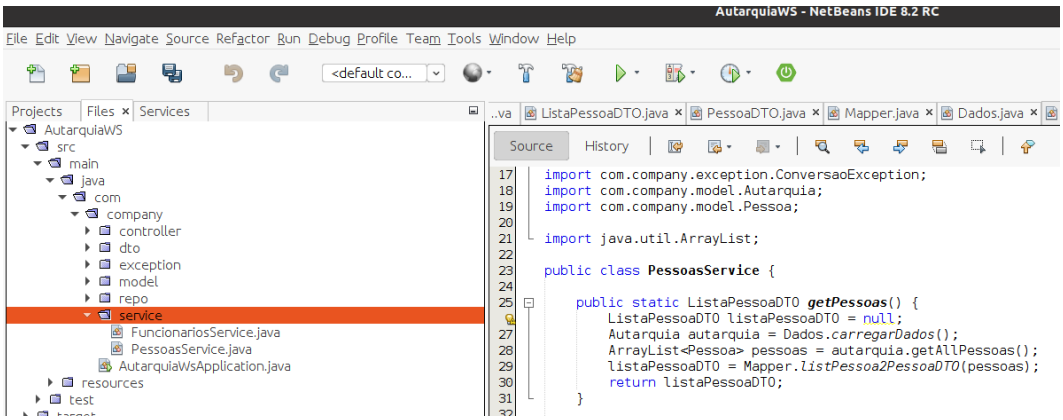
```
  public static PessoaDTO getPessoa(long nif) {
    Autarquia autarquia = Dados.carregarDados();
    Pessoa pessoa = autarquia.getPessoa(nif);
    if (pessoa == null) {
      return null;
    }
    PessoaDTO pessoaDTO = Mapper.pessoa2PessoaDTO(pessoa);
    if (pessoaDTO != null) {
      return pessoaDTO;
    } else {
      throw new ConversaoException("PessoaDTO");
    }
  }

  public static void addPessoa(PessoaDTO pessoaDTO) {
    Pessoa pessoa = Mapper.pessoaDTO2Pessoa(pessoaDTO);
    if (pessoa != null) {
      Autarquia autarquia = Dados.carregarDados();
      autarquia.addPessoa(pessoa);
      Dados.guardarDados(autarquia);
    } else {
      throw new ConversaoException("PessoaDTO");
    }
  }

  public static void updatePessoa(long nif, PessoaDTO pessoaDTO) {
    Pessoa pessoa = Mapper.pessoaDTO2Pessoa(pessoaDTO);
    if (pessoa != null) {
      Autarquia autarquia = Dados.carregarDados();
      autarquia.updatePessoa(nif, pessoa);
      Dados.guardarDados(autarquia);
    } else {
      throw new ConversaoException("PessoaDTO");
    }
  }

  public static void removePessoa(long nif) {
    Autarquia autarquia = Dados.carregarDados();
    autarquia.removePessoa(nif);
    Dados.guardarDados(autarquia);
  }

}
```

- `FuncionariosService` class.

```
import com.company.dao.Dados;
import com.company.dto.FuncionarioDTO;
import com.company.dto.ListaFuncionarioDTO;
import com.company.dto.Mapper;
import com.company.exception.ConversaoException;
import com.company.model.Autarquia;
import com.company.model.Funcionario;

import java.util.ArrayList;

public class FuncionariosService {

  public static ListaFuncionarioDTO getFuncionarios() {
    Autarquia autarquia = Dados.carregarDados();
    ArrayList<Funcionario> funcionarios = autarquia.getFuncionarios();
    ListaFuncionarioDTO listaFuncionarioaDTO = Mapper.listFuncionario2FuncionarioDTO(funcionarios);
    return listaFuncionarioaDTO;
  }

  public static FuncionarioDTO getFuncionario(int nr) {
    Autarquia autarquia = Dados.carregarDados();
    Funcionario funcionario = autarquia.getFuncionario(nr);
    if (funcionario == null) {
      return null;
    }
```

```
    FuncionarioDTO funcionarioDTO = Mapper.funcionario2FuncionarioDTO(funcionario);
    if(funcionarioDTO != null){
      return funcionarioDTO;
    }else{
      throw new ConversaoException("FuncionarioDTO");
    }

  }

  public static void addFuncionario(FuncionarioDTO funcionarioDTO) {

    Funcionario funcionario = Mapper.funcionarioDTO2Funcionario(funcionarioDTO);
    if (funcionario != null) {
      Autarquia autarquia = Dados.carregarDados();
      autarquia.addFuncionario(funcionario);
      Dados.guardarDados(autarquia);
    } else {
      throw new ConversaoException("FuncionarioDTO");
    }
  }

  public static void updateFuncionario(int nr, FuncionarioDTO funcionarioDTO) {
    Funcionario funcionario = Mapper.funcionarioDTO2Funcionario(funcionarioDTO);
    if (funcionario != null) {
      Autarquia autarquia = Dados.carregarDados();
      autarquia.updateFuncionario(nr, funcionario);
      Dados.guardarDados(autarquia);
    } else {
      throw new ConversaoException("FuncionarioDTO");
    }
  }

  public static void removeFuncionario(int nr) {
    Autarquia autarquia = Dados.carregarDados();
    autarquia.removeFuncionario(nr);
    Dados.guardarDados(autarquia);
  }
}
```

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



4.8.7 Coding `controller`

Controllers classes manage the HTTP requests. They receive HTTP requests from , they forward the request to service classes, receive the response from services classes, then build the HTTP response and send to client. This package implements the following classes:

- `PessoasController` class.

```
import com.company.dto.ErroDTO;
import com.company.dto.ListaPessoaDTO;
import com.company.dto.PessoaDTO;
import com.company.service.PessoasService;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api")
public class PessoasController {
  @RequestMapping(value = "/pessoas",
  method = RequestMethod.GET,
  produces = MediaType.APPLICATION_XML_VALUE)
  public ResponseEntity<Object> getPessoas() {
    try {
      ListaPessoaDTO listaPessoaDTO = PessoasService.getPessoas();
      if (listaPessoaDTO.getPessoas().size() > 0) {
        return new ResponseEntity<>(listaPessoaDTO, HttpStatus.OK);
      } else {
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
      }
    } catch (Exception e) {
      return new ResponseEntity<>(new ErroDTO(e), HttpStatus.CONFLICT);
    }
  }

  @RequestMapping(value = "/pessoas/{id}",
  method = RequestMethod.GET,
  produces = MediaType.APPLICATION_XML_VALUE)
  public ResponseEntity<Object> getPessoa(@PathVariable("id") long nif) {
    try {
      PessoaDTO pessoaDTO = PessoasService.getPessoa(nif);
      if (pessoaDTO != null) {
        return new ResponseEntity<>(pessoaDTO, HttpStatus.OK);
      } else {
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
      }
    } catch (Exception e) {
      return new ResponseEntity<>(new ErroDTO(e), HttpStatus.CONFLICT);
    }
  }

  @RequestMapping(value = "/pessoas",
  method = RequestMethod.POST,
  consumes = MediaType.APPLICATION_XML_VALUE,
  produces = MediaType.APPLICATION_XML_VALUE)
  public ResponseEntity<Object> addPessoa(@RequestBody PessoaDTO pessoaDTO) {
    try {
      PessoasService.addPessoa(pessoaDTO);
      return new ResponseEntity<>(HttpStatus.CREATED);
    } catch (Exception e) {
      return new ResponseEntity<>(new ErroDTO(e), HttpStatus.CONFLICT);
    }
  }

  @RequestMapping(value = "/pessoas/{id}",
  method = RequestMethod.PUT,
  consumes = MediaType.APPLICATION_XML_VALUE,
```

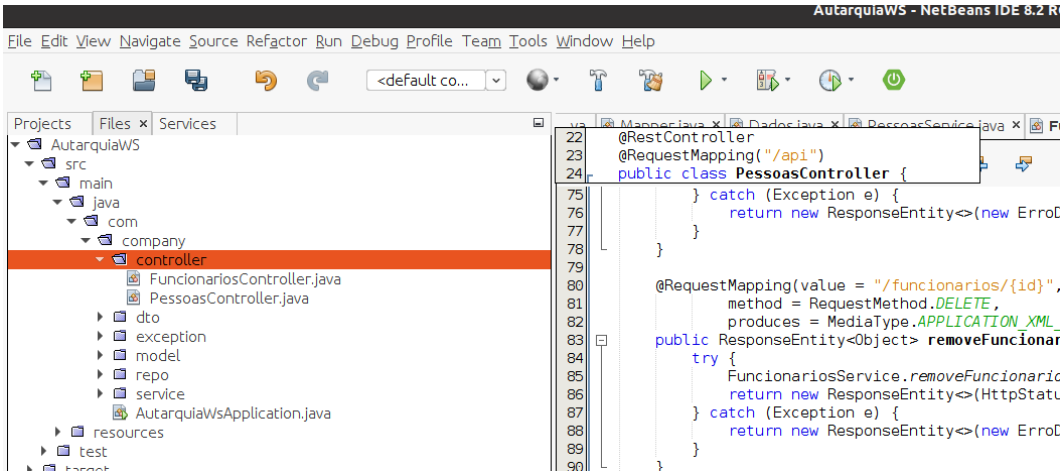
```
  produces = MediaType.APPLICATION_XML_VALUE)
  public ResponseEntity<Object> updatePessoa(@PathVariable("id") long nif, @RequestBody PessoaDTO pessoaDTO
        ) {
    try {
      PessoasService.updatePessoa(nif, pessoaDTO);
      return new ResponseEntity<>(HttpStatus.OK);
    } catch (Exception e) {
      return new ResponseEntity<>(new ErroDTO(e), HttpStatus.CONFLICT);
    }
  }

  @RequestMapping(value = "/pessoas/{id}",
  method = RequestMethod.DELETE,
  produces = MediaType.APPLICATION_XML_VALUE)
  public ResponseEntity<Object> removePessoa(@PathVariable("id") long nif) {
    try {
      PessoasService.removePessoa(nif);
      return new ResponseEntity<>(HttpStatus.OK);
    } catch (Exception e) {
      return new ResponseEntity<>(new ErroDTO(e), HttpStatus.CONFLICT);
    }
  }
}
```

- `FuncionariosController` class.

```
import com.company.dto.ErroDTO;
import com.company.dto.FuncionarioDTO;
import com.company.dto.ListaFuncionarioDTO;
import com.company.service.FuncionariosService;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api")
public class FuncionariosController {
  @RequestMapping(value = "/funcionarios",
  method = RequestMethod.GET,
  produces = MediaType.APPLICATION_XML_VALUE)
  public ResponseEntity<Object> getFuncionarios() {
    try {
      ListaFuncionarioDTO listaFuncionarioDTO = FuncionariosService.getFuncionarios();
      if (listaFuncionarioDTO != null) {
        return new ResponseEntity<>(listaFuncionarioDTO, HttpStatus.OK);
      } else {
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
      }
    } catch (Exception e) {
      return new ResponseEntity<>(new ErroDTO(e), HttpStatus.CONFLICT);
    }
  }
  @RequestMapping(value = "/funcionarios/{id}",
  method = RequestMethod.GET,
  produces = MediaType.APPLICATION_XML_VALUE)
  public ResponseEntity<Object> getFuncionario(@PathVariable("id") int nr) {
    try {
      FuncionarioDTO funcionarioDTO = FuncionariosService.getFuncionario(nr);
      if (funcionarioDTO != null) {
        return new ResponseEntity<>(funcionarioDTO, HttpStatus.OK);
      } else {
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
      }
    } catch (Exception e) {
      return new ResponseEntity<>(new ErroDTO(e), HttpStatus.CONFLICT);
    }
  }
  @RequestMapping(value = "/funcionarios",
  method = RequestMethod.POST,
  consumes = MediaType.APPLICATION_XML_VALUE,
```

```
    produces = MediaType.APPLICATION_XML_VALUE)
    public ResponseEntity<Object> addFuncionario(@RequestBody FuncionarioDTO funcionarioDTO) {
      try {
        FuncionariosService.addFuncionario(funcionarioDTO);
        return new ResponseEntity<>(HttpStatus.CREATED);
      } catch (Exception e) {
        return new ResponseEntity<>(new ErroDTO(e), HttpStatus.CONFLICT);
      }
    }

    @RequestMapping(value = "/funcionarios/{id}",
    method = RequestMethod.PUT,
    consumes = MediaType.APPLICATION_XML_VALUE,
    produces = MediaType.APPLICATION_XML_VALUE)
    public ResponseEntity<Object> updateFuncionario(@PathVariable("id") int nr, @RequestBody FuncionarioDTO
          funcionarioDTO) {
      try {
        FuncionariosService.updateFuncionario(nr, funcionarioDTO);
        return new ResponseEntity<>(HttpStatus.OK);
      } catch (Exception e) {
        return new ResponseEntity<>(new ErroDTO(e), HttpStatus.CONFLICT);
      }
    }

    @RequestMapping(value = "/funcionarios/{id}",
    method = RequestMethod.DELETE,
    produces = MediaType.APPLICATION_XML_VALUE)
    public ResponseEntity<Object> removeFuncionario(@PathVariable("id") int nr) {
      try {
        FuncionariosService.removeFuncionario(nr);
        return new ResponseEntity<>(HttpStatus.OK);
      } catch (Exception e) {
        return new ResponseEntity<>(new ErroDTO(e), HttpStatus.CONFLICT);
      }
    }
  }
```

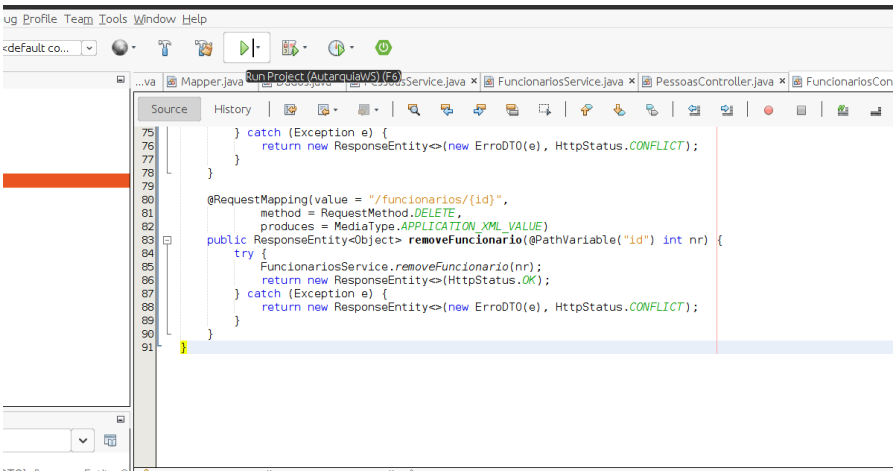
In the end, the content of the `controller` package must be similar to the shown in the following figure.



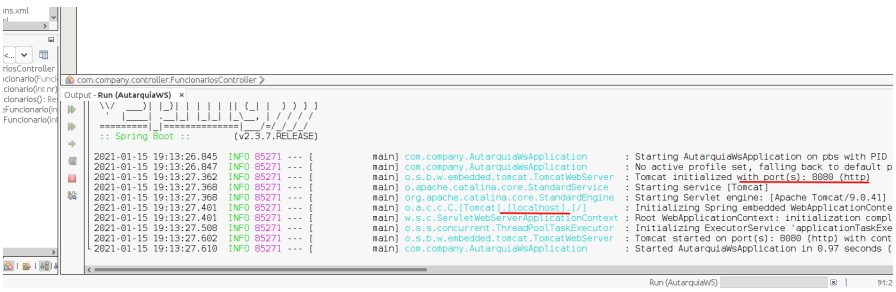
5 Deploying the AutarquiaWS web service

Once you are done with creating source and web configuration files, you are ready for this step which is compiling and running your program.

To do this, using NetBeans IDE, click on **Run**.



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



6 Testing

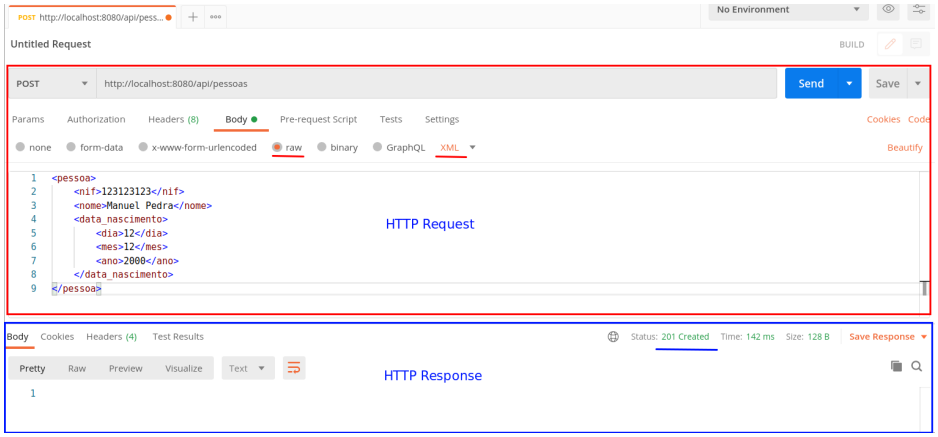
We will use Postman, to test our web service.

6.1 Making POST HTTP requests

1. Creating POST HTTP request for adding a **Pessoa**:

```
POST /api/pessoas HTTP/1.1
Host: localhost:8080
Content-Type: application/xml
...

<pessoa>
```

```
  <nif>123123123</nif>
  <nome>Manuel Pedra</nome>
  <data_nascimento>
    <dia>12</dia>
    <mes>12</mes>
    <ano>2000</ano>
  </data_nascimento>
</pessoa>
```



2. Adding more data

```
POST /api/pessoas HTTP/1.1
Host: localhost:8080
Content-Type: application/xml
...

<pessoa>
  <nif>234234234</nif>
  <nome>Maria Forte</nome>
  <data_nascimento>
    <dia>23</dia>
    <mes>2</mes>
    <ano>1999</ano>
  </data_nascimento>
</pessoa>
```

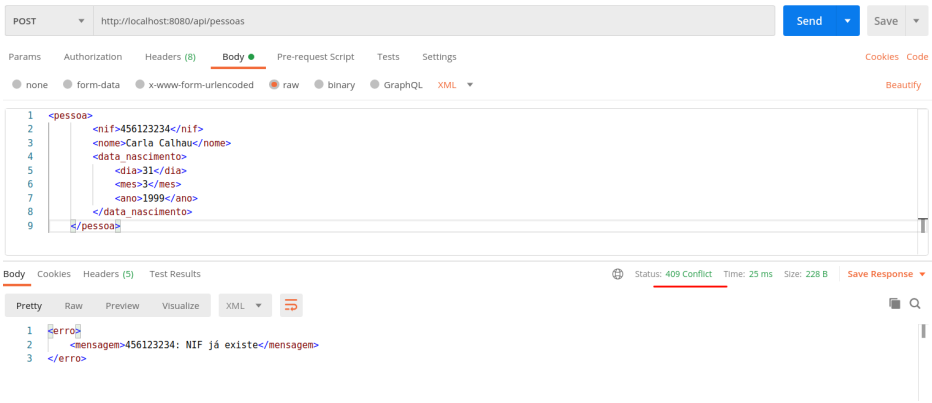
```
POST /api/pessoas HTTP/1.1
Host: localhost:8080
Content-Type: application/xml
...

<pessoa>
  <nif>123234345</nif>
  <nome>Carlos Ferrolho</nome>
  <data_nascimento>
    <dia>21</dia>
    <mes>3</mes>
    <ano>1998</ano>
  </data_nascimento>
</pessoa>
```

```
POST /api/pessoas HTTP/1.1
Host: localhost:8080
Content-Type: application/xml
...
```

```
  <pessoa>
    <nif>456123234</nif>
    <nome>Carla Calhau</nome>
    <data_nascimento>
      <dia>31</dia>
      <mes>3</mes>
      <ano>1999</ano>
    </data_nascimento>
  </pessoa>
```

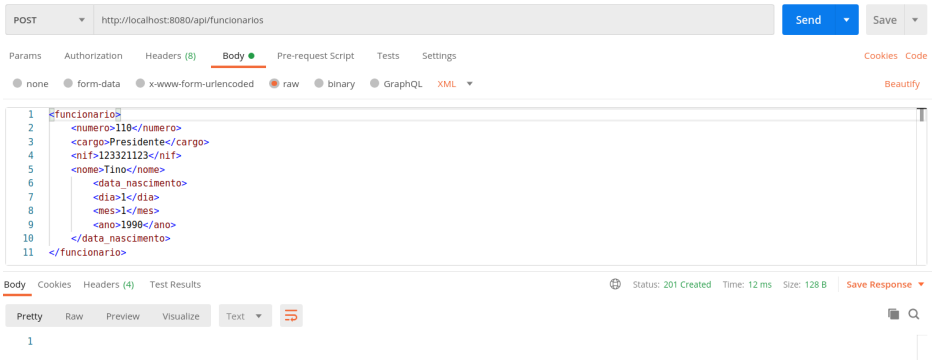
3. Adding an existing `Pessoa`

```
POST /api/pessoas HTTP/1.1
Host: localhost:8080
Content-Type: application/xml
...

<pessoa>
  <nif>456123234</nif>
  <nome>Carla Calhau</nome>
  <data_nascimento>
    <dia>31</dia>
    <mes>3</mes>
    <ano>1999</ano>
  </data_nascimento>
</pessoa>
```



4. Adding `Funcionario`

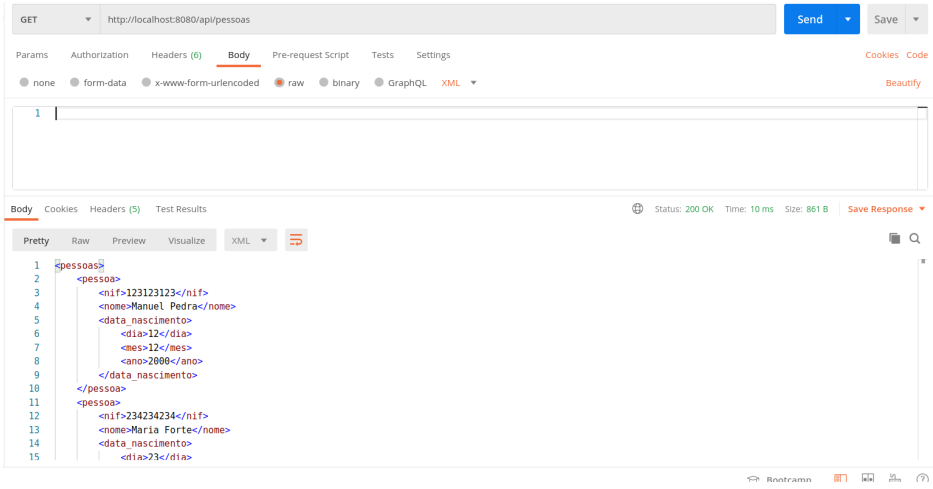
```
POST /api/funcionarios HTTP/1.1
Host: localhost:8080
Content-Type: application/xml
...

<funcionario>
  <numero>110</numero>
  <cargo>Presidente</cargo>
  <nif>123321123</nif>
  <nome>Tino</nome>
    <data_nascimento>
    <dia>1</dia>
    <mes>1</mes>
    <ano>1990</ano>
  </data_nascimento>
</funcionario>
```



6.2 Making GET HTTP requests

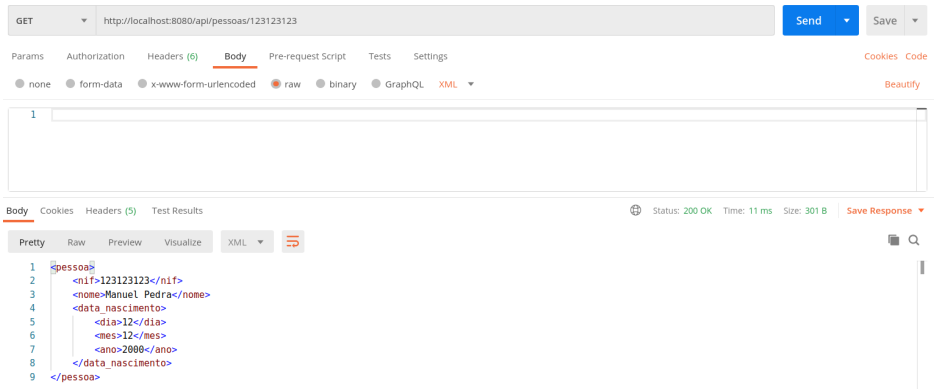
1. Creating GET HTTP request for getting all `Pessoa`:

```
GET /api/pessoas HTTP/1.1
Host: localhost:8080
...
```



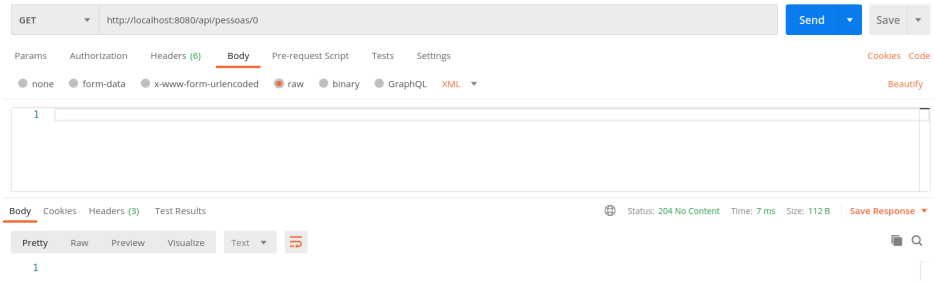
2. Creating GET HTTP request for getting one `Pessoa`:

```
GET /api/pessoas/123123123 HTTP/1.1
Host: localhost:8080
...
```



3. Creating GET HTTP request for getting one non-existing `Pessoa`:

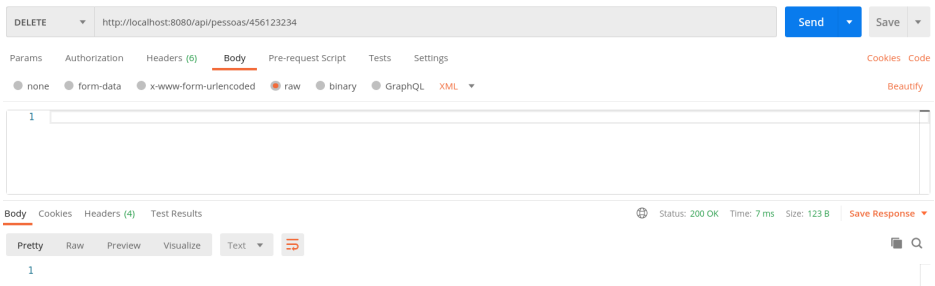
```
GET /api/pessoas/0 HTTP/1.1
Host: localhost:8080
...
```



6.3 Making DELETE HTTP request

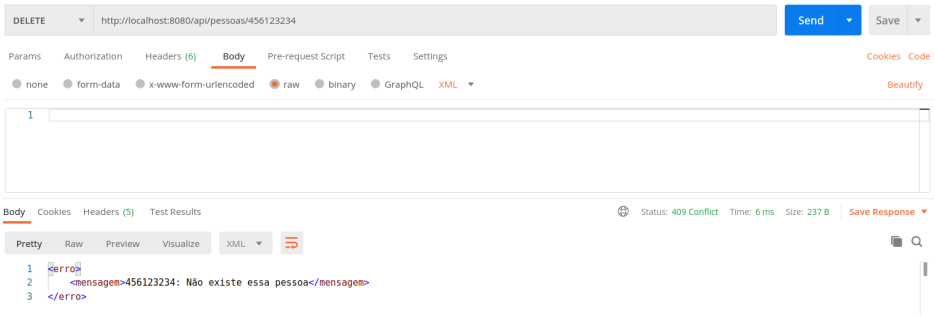
1. Creating DELETE HTTP request for removing an existing `Pessoa`.

```
DELETE /api/pessoas/456123234 HTTP/1.1
Host: localhost:8080
...
```



2. Creating DELETE HTTP request for removing a non-existing `Pessoa`.

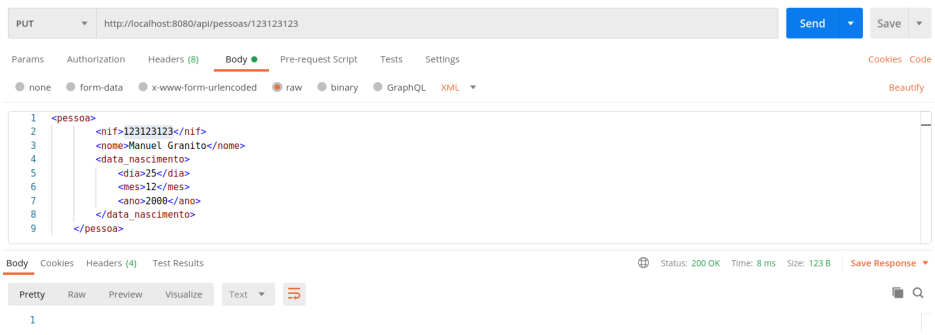
```
DELETE /api/pessoas/456123234 HTTP/1.1
Host: localhost:8080
...
```



6.4 Make a PUT HTTP request

1. Creating PUT HTTP request for updating an existing `Pessoa`.

```
PUT /api/pessoas/123123123 HTTP/1.1
Host: localhost:8080
Content-Type: application/xml
...

<pessoa>
  <nif>123123123</nif>
  <nome>Manuel Granito</nome>
  <data_nascimento>
    <dia>25</dia>
    <mes>12</mes>
    <ano>2000</ano>
  </data_nascimento>
</pessoa>
```



2. Creating PUT HTTP request for updating a non-existing `Pessoa`.

```
PUT /api/pessoas/0 HTTP/1.1
Host: localhost:8080
Content-Type: application/xml
```

```
...

<pessoa>
  <nif>123123123</nif>
  <nome>Manuel Granito</nome>
  <data_nascimento>
    <dia>25</dia>
    <mes>12</mes>
    <ano>2000</ano>
  </data_nascimento>
</pessoa>
```