1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

Web Services

Integração de Sistemas 2020/2021

António Marques Maria — 2017265346
Rui Mário Choupina Reis — 2013134606

ÍNDICE

ANEXO

a. Lists of what the group succeeded to do and what is missing
R: Conseguimos cumprir todos os requisitos do projeto.

b. self-evaluation of the group (0-100%)
R: 95%

c. List of what each student contributed (no repetitions)
Adotámos uma técnica de Pair Programming, logo houve participação de ambos em todo o processo.

d. self-evaluation of each student in the group (0-100%)
R: António - 95%
Rui - 95%

e. hours of effort by each student separately
R: António - 40h
Rui - 40h

INTRODUÇÃO

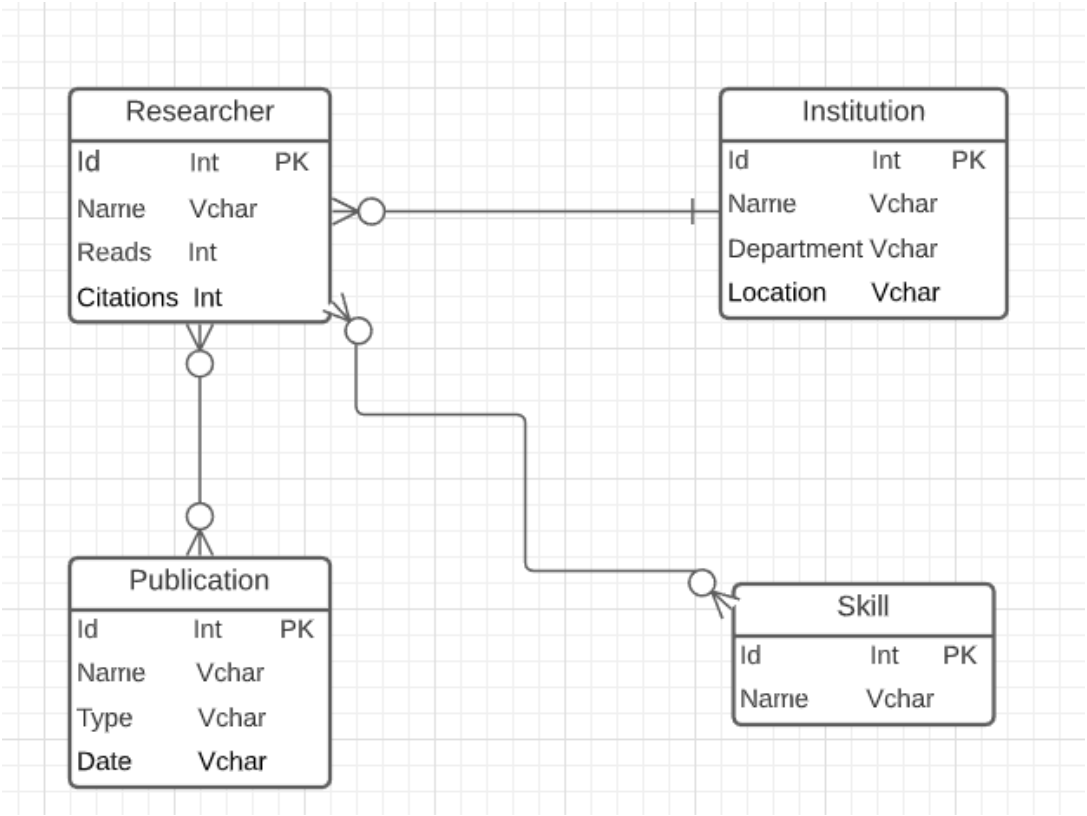
No âmbito da unidade curricular de Integração de Sistemas inserida no Mestrado em Engenharia Informática da Universidade de Coimbra, iremos no decorrer deste relatório descrever o nosso projeto.

Neste trabalho prático é pretendido aprofundar conhecimento acerca do desenvolvimento de aplicativos empresariais em camadas, nomeadamente com o uso de Enterprise JavaBeans (EJB), WebServices (SOAP/REST), Persistence Engine e um cliente java que utiliza todos estes recursos, tal como uma web front-end com uma utilização específica.

De forma a realizar este projeto recorremos à linguagem JAVA tal como a JSP no caso da web front-end de forma a mostrar informação dinâmica. Cada um dos WebServices está numa Enterprise Application Archive (EAR) juntamente com o EJB correspondente de forma a ter acesso às suas funcionalidades.

LOADER

O Loader serviu para carregar informação para a base de dados Postgres e utilizamos este esquema de dados para relacionar as várias entidades.



Para inserir na base de dados usamos JPA/Hibernate de forma a mapear as entidades e persisti-las.

```java
@Entity(name = "Researcher")
@Table(name = "researcher")
public class Researcher implements Serializable {
    private static final long serialVersionUID = 1L;
    // we use this generation type to match that of SQLWriteStudents
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY )
    private long id;
    private String personName;
    private int reads;
    private int citations;


    @ManyToOne
    private Institution institution;
    @ManyToMany(targetEntity = Publication.class,fetch=FetchType.EAGER)
    private Set<Publication> publications = new HashSet<Publication>();
    @ManyToMany(targetEntity = Skill.class,fetch=FetchType.EAGER)
    private Set<Skill> skills = new HashSet<Skill>();
```

```java
@Entity(name = "Institution")
@Table(name = "institution")
public class Institution implements Serializable{

   /**
     *
     */
    private static final long serialVersionUID = 1L;
    @Id
   @GeneratedValue( strategy= GenerationType.IDENTITY )

   private long id;
   private String name;
   private String location;
   private String department;
```

```java
@Entity(name = "Publication")
@Table(name = "publication")
public class Publication implements Serializable{
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue( strategy= GenerationType.IDENTITY )
    private long id;
    private String name;
    private String type;
    private String date;
```

```java
@Entity(name = "Skill")
@Table(name = "skill")
public class Skill implements Serializable {
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY )
    private long id;
    private String nome;
```

Começamos por criar objetos de cada classe com os seus atributos inicializados, de seguida adicionamos ao Researcher o objeto Institution que lhe está associado, tal como o seu Set de Publications e de Skills.

De seguida, persistimos cada um dos objetos e estes são adicionados à base de dados com as suas relações como definimos na classe Researcher.

Também podemos usar comandos SQL para realizar esta operação. Iríamos criar 6 tabelas, Researcher, Institution, Publication, Skill, Researcher_Skill, Researcher_Publication, cada uma delas com os atributos correspondentes que estão descritos no diagrama ER.

1. Para criar a tabela researcher.

```
String sql = “CREATE TABLE IF NOT EXISTS “ + tablenameresearcher + “(id SERIAL
NOT NULL , personname VARCHAR(254), reads INTEGER, citations INTEGER, institu-
tion_id BIGINT , PRIMARY KEY (id), FOREIGN KEY (institution_id))”;
```

2. Para criar a tabela institution.

```
sql = “CREATE TABLE IF NOT EXISTS “ + tablenameinstitution + “(id SERIAL NOT NULL
, Name VARCHAR(254), Location VARCHAR(254), Department VARCHAR(254), PRIMARY KEY
(id))”;
```

3. Para criar a tabela publication.

```
sql = “CREATE TABLE IF NOT EXISTS “ + tablenamepublication + “(id SERIAL NOT NULL
, Name VARCHAR(254), Type VARCHAR(254), Date VARCHAR(254), PRIMARY KEY (id))”;
```

4. Para criar a tabela skill.

```
sql = “CREATE TABLE IF NOT EXISTS “ + tablenameskills + “(id SERIAL NOT NULL ,
Name VARCHAR(254), PRIMARY KEY (id))”;
```

5. Para criar a tabela researcher_publication

```
sql = “CREATE TABLE IF NOT EXISTS “ + tablenameresearcherpublication + “(rese-
archer_id BIGINT NOT NULL , publications_id BIGINT NOT NULL, PRIMARY KEY (rese-
archer_id, publications_id), FOREIGN KEY (publications_id) REFERENCES “ + table-
namepublication + “(id) MATCH SIMPLE, FOREIGN KEY (researcher_id) REFERENCES “ +
tablenameresearcher + “(id) MATCH SIMPLE” + “)”;
```

6. Para criar a tabela researcher_skill

```
sql = “CREATE TABLE IF NOT EXISTS “ + tablenameresearcherskills + “(researcher_id
BIGINT NOT NULL , skills_id BIGINT NOT NULL, PRIMARY KEY (researcher_id, skills_
id), FOREIGN KEY (researcher_id) REFERENCES “ + tablenameresearcher + “(id) MAT-
CH SIMPLE, FOREIGN KEY (skills_id) REFERENCES “ + tablenameskills + “(id) MATCH
SIMPLE” +”)”;
```

De seguida criamos os conjuntos de objetos a inserir e iteramos sobre esse conjunto, fazendo o comando INSERT para cada objeto, inserido-os na base de dados:

1. Na entidade Researcher adicionamos o nome, o reads, as citações e o id da instituição a que pertence. (st é um objeto Researcher num conjunto).

```
sql = “INSERT INTO “ + tablenameresearcher + “(personname, reads, citations, ins-
titution_id)” + “VALUES” + “(‘” + st.getPersonName() + “’,’” + st.getReads() +
“’,’” + st.getCitations() + “’,’” + st.getInstitutionSet().getId() + “’)”;
```

2. Na entidade Institution adicionamos o nome, a localização e o departamento. (st é um objeto Institution num conjunto).

```
sql = “INSERT INTO “ + tablenameinstitution + “(Name, Location, Department)” +
“VALUES” + “(‘” + st.getName() + “’,’” + st.getLocation() + “’,’” + st.getDepart-
ment() + “’)”;
```

3. Na entidade Publications adicionamos o nome, o tipo e a data. (st é um objeto Publication num conjunto).

```
sql = “INSERT INTO “ + tablenamepublication + “(Name, Type, Date)” + “VALUES” +
“(‘” + st.getName() + “’,’” + st.getType() + “’,’” + st.getDate() + “’)”;
```

4. Na entidade Skill adicionamos o nome. (st é um objeto Skill num conjunto).

```
sql = “INSERT INTO “ + tablenameskills + “(Name)” + “VALUES” + “(‘” + st.getNo-
me() +”’)”;
```

5. Na entidade Researcher_Publication adicionamos por cada Publication que pertence ao Set publication_set do Researcher, o id do Researcher associado e o id da publicação que está a ser percorrida. (st é um objeto Researcher presente num conjunto, st2 é um objeto Publication presente no publication_set do Researcher iterado).

```
sql = “INSERT INTO “ + tablenameresearcherpublication + “(researcher_id, publica-
tions_id)” + “VALUES” + “(‘” + st.getId() + “’,’” + st2.getId() + “’)”;
```

6. Na entidade Researcher_Skill adicionamos por cada Skill que pertence ao Set skill_set do Researcher, o id do Researcher associado e o id do Skill que está ser percorrido. (st é um objeto Researcher presente num conjunto, st2 é um objeto Skill presente no skill_set do Researcher iterado).

```
ql = “INSERT INTO “ + tablenameresearcherskills + “(researcher_id, skills_id)” +
“VALUES” + “(‘” + st.getId() + “’,’” + st2.getId() + “’)”;
```

SOAP WEB SERVICE – RESEARCHERS

O SOAP Web Service que vai buscar informação acerca de Researchers tem 3 WebMethods, um que vai buscar informação acerca de todos os presentes na Base de dados, um que filtra pelo nome do investigador, e outro que filtra pelo nome do skill associado aos investigadores, ambos os nomes passados como WebParams.

Cada um destes WebMethods chama um método que se encontra num EJB que vai ser encontrado por eles.

O primeiro método do WebService encontra todos os investigadores e chama o método ejb.Getall() para o fazer. Este por sua vez faz um query à base de dados para selecionar todos os elementos da tabela Researcher e retorna uma lista com os objetos.

O segundo método chama o ejb.GetResearcherByNome(String nome) que recebe um nome como parâmetro que depois é usado para filtrar a query à base de dados de forma a retornar apenas uma lista com os objetos que tenham o atributo personName igual ao que foi passado como parâmetro.

O terceiro método chama o ejb.GetResearcherBySkill(String nome) que recebe também um nome como parâmetro e que é utilizado para ir buscar o id do Skill que tem o nome igual ao que foi passado e que de seguida retorna uma lista com todos os objetos Researcher que tenham esse id do Skill associado.

Este serviço encontra-se disponível através do endpoint: http://localhost:8080/ResearcherBeanWS/ResearcherWebService

SOAP WEB SERVICE – PUBLICATIONS

O SOAP Web Service que vai buscar informação acerca das Publications tem 3 WebMethods, um que vai buscar informação acerca de todos as presentes na Base de dados, um que filtra pelo nome da publicação, e outro que filtra pelo nome do investigador associado à publicação, ambos os nomes passados como WebParams.

Cada um destes WebMethods chama um método que se encontra num EJB que vai ser encontrado por eles.

O primeiro método do WebService encontra todos as publicações e chama o método ejb.Getall() para o fazer. Este por sua vez faz um query à base de dados para selecionar todos os elementos da tabela Publication e retorna uma lista com os objetos.

O segundo método chama o ejb.GetPublicationByNome(String nome) que recebe um nome como parâmetro que depois é usado para filtrar a query à base de dados de forma a retornar apenas uma lista com os objetos que tenham o atributo name igual ao que foi passado como parâmetro.

O terceiro método chama o ejb.GetPublicationByResearcher(String nome) que recebe também um nome como parâmetro e que é utilizado para ir buscar o id de todas as publicações associadas ao Researcher com esse nome, que depois é usado para retornar uma lista com todos os objetos Publication através desses ids.

Este serviço encontra-se disponivel através do endpoint: http://localhost:8080/PublicationBeanWS/PublicationWebService

REST WEB SERVICE – INSTITUTIONS

O REST Web Service vai buscar informação acerca das Institutions e tem 3 métodos, um que vai buscar informação acerca de todos as presentes na Base de dados, um que filtra pelo nome da instituição, e outro que filtra pelo nome do investigador associado à instituição, ambos os nomes passados como QueryParams.

Estes métodos enviam um request GET e retornam o tipo de dados JSON. Cada um destes chama um método associado a um EJB que vai ser encontrado por eles.

O primeiro método do WebService encontra todos as instituições e chama o método ejb.Getall() para o fazer. Este por sua vez faz um query à base de dados para selecionar todos os elementos da tabela Institution e retorna uma lista com os objetos.

O segundo método chama o ejb.GetInstitutionByNome(String nome) que recebe um nome como parâmetro que depois é usado para filtrar a query à base de dados de forma a retornar apenas uma lista com os objetos que tenham o atributo name igual ao que foi passado como parâmetro.

O terceiro método chama o ejb.GetInstitutionByResearcher(String nome) que recebe também um nome como parâmetro e que é utilizado para ir buscar o id da instituição associada ao Researcher com esse nome, que depois é usado para retornar o objeto Institution através desse id.

Os métodos estão disponíveis nos endpoints:

1. Getall
http://localhost:8080/InstitutionBeanRestWS/rest/InstitutionWebService/getall

2. GetInstitutionByNome
http://localhost:8080/InstitutionBeanRestWS/rest/InstitutionWebService/getbyname

3. GetInstitutionByResearcher
http://localhost:8080/InstitutionBeanRestWS/rest/InstitutionWebService/getbyresearcher

CLIENT APPLICATION

A aplicação JAVA que utiliza todas as funcionalidades dos WebServices apresenta um menu gráfico básico para ser possível selecionar a operação que o utilizador quer realizar. Existem várias funções auxiliares para auxiliarem no acesso à informação da base de dados.

Função auxiliar do seletor:
A função lerInt(int init, int fin) lê um inteiro e verifica se é um valor entre init e fin e retorna-o.

Função auxiliar do REST:
A função GetAllInst() chama o Rest WebService Getall e faz parsing do array JSON que recebe como resposta de forma a imprimir a informação pretendida.
A função GetInstByName(String name) chama o Rest WebService GetInstByName com o QueryParam name e faz parsing do array JSON que recebe como resposta de forma a imprimir a informação da instituição retornada.
A função GetInstByResearcher(String name) chama o Rest WebService GetInstByResearcher com o QueryParam name e faz parsing do array JSON que recebe como resposta de forma a imprimir a informação da instituição retornada.

Função auxiliar do SOAP:
A função createSoapEnvelope(SOAPMessage soapMessage, String method, String param) cria o envelope da mensagem SOAP que vai ser enviado ao WebService para chamar o método pretendido. Recebe a mensagem SOAP a ser enviada, o método que está a ser chamado e o parâmetro do método. Como os métodos Getall de ambos os serviços não usam argumentos é apenas adicionada ao envelope o nome do método, caso seja algum dos outros métodos é ainda adicionado o argumento ao envelope.

```
<soapenv:Envelope xmlns
    <soapenv:Header/>
    <soapenv:Body>
        <unk:Getall/>
    </soapenv:Body>
</soapenv:Envelope>
```

Exemplo de um envelope SOAP sem parametros

```
<soapenv:Envelope xmlns:soapenv="
   <soapenv:Header/>
   <soapenv:Body>
      <unk:GetResearcherByNome>
         <!--Optional:-->
         <arg0>Nuno Lopes</arg0>
      </unk:GetResearcherByNome>
   </soapenv:Body>
</soapenv:Envelope>
```

Exemplo de um envelope SOAP com parametros

A função callSoapWebService(String soapEndpointUrl, String soapAction, String method, String param) faz a conexão e recebe a resposta do WebService pretendido. Recebe o Url onde está hospedado o WebService, a soapAction a chamar (no nosso caso é sempre vazio), o método a ser chamado e os seus parâmetros de forma a enviar para a função que cria o SoapRequest.
A função createSOAPRequest(String soapAction,String method,String param) cria a mensagem Soap que vai ser enviada ao WebService. Recebe a soapAction a executar, o método a ser chamado e os seus parâmetros de forma a enviar para a função que cria o envelope Soap.
A função parseSOAPMessageResearcher(SOAPMessage message) extrai a informação do Researcher da resposta Soap que foi enviada pelo WebService. Vai buscar o elemento return da mensagem, que é cada objeto Researcher retornado pelo query, e percorre todos os nós filhos do return e imprime a informação contida neles.

A função parseSOAPMessagePublication(SOAPMessage message) extrai a informação da Publication da resposta Soap que foi enviada pelo WebService. Vai buscar o elemento return da mensagem, que é cada objeto Publication retornado pelo query, e percorre todos os nós filhos do return e imprime a informação contida neles.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/
   <soap:Body>
      <ns2:GetResearcherByNomeResponse xmlns:ns2="http
         <return>
            <citations>2</citations>
            <id>3</id>
            <institutionSet>
               <department>Desporto</department>
               <location>Porto</location>
               <name>Universidade do Porto</name>
            </institutionSet>
            <personName>Nuno Lopes</personName>
            <reads>3</reads>
            <skillSet>
               <nome>Java Programming</nome>
            </skillSet>
            <skillSet>
               <nome>Technology</nome>
            </skillSet>
         </return>
      </ns2:GetResearcherByNomeResponse>
   </soap:Body>
</soap:Envelope>
```

Resposta Soap de um Researcher com query “Nuno Lopes”

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:GetallResponse xmlns:ns2="http://unknown.namespace/">
         <return>
            <date>January 2011</date>
            <name>Hibernate para totós</name>
            <type>Book</type>
         </return>
         <return>
            <date>December 2018</date>
            <name>CDI para totós</name>
            <type>Book</type>
         </return>
         <return>
            <date>November 2011</date>
            <name>Ice Age</name>
            <type>Article</type>
         </return>
         <return>
            <date>May 2009</date>
            <name>O futuro da tecnologia</name>
            <type>Conference Paper</type>
         </return>
         <return>
            <date>May 2013</date>
            <name>JPA para totós</name>
            <type>Book</type>
         </return>
         <return>
            <date>June 2019</date>
            <name>XML para totós</name>
            <type>Book</type>
         </return>
         <return>
            <date>January 2012</date>
            <name>REST para totós</name>
            <type>Book</type>
         </return>
```

Resposta Soap com várias Publications

WEB FRONT-END

O Web front-end consiste num servlet que ao receber um pedido GET chama o WebService Soap das publicações com o método Getall e constrói uma lista de Publications com a resposta Soap recebida e de seguida o insere num atributo do request de forma a puder ser chamado no JSP. O Servlet termina executando o ficheiro “display.jsp”

O front-end apresenta a informação a dizer que está a mostrar as publicações da base de dados e de seguida itera pela lista que foi passada pelo servlet e constrói uma tabela com a informação de cada publicação.

As funções auxiliares de Soap são as mesmas que o cliente JAVA com a mudança que, ao invés de imprimir a informação na consola, este cria um objeto Publication por cada “return” presente na resposta Soap e o adiciona a um Array de Publications.

List of Publications in the Database

#	Name	Date	Type
1	Hibernate para totós	January 2011	Book
2	CDI para totós	December 2018	Book
3	Ice Age	November 2011	Article
4	O futuro da tecnologia	May 2009	Conference Paper
5	JPA para totós	May 2013	Book
6	XML para totós	June 2019	Book
7	REST para totós	January 2012	Book
8	O estudo Volume II	November 2016	Book
9	O passado da tecnologia	June 2019	Conference Paper
10	JavaBeans para totós	January 2015	Book
11	WebServices para totós	February 2013	Book
12	JMS para totós	March 2014	Book

CONCLUSÃO

Através da realização deste trabalho foi-nos possível aprofundar conhecimento no que diz respeito ao desenvolvimento de aplicações empresariais.
De forma a alcançarmos isto utilizamos uma base de dados para guardar a informação acerca de investigadores, que posteriormente utilizamos para ir buscar os seus dados através de SOAP Web Services e REST.

Estes WebServices utilizam Enterprise JavaBeans para fazer um query à base de dados e retornar a lista de dados pretendida para os respectivos serviços.
Para alcançar a visualização da informação criámos um cliente JAVA que utiliza todas as funcionalidades dos Web Services, lendo as respostas SOAP enviadas no caso dos investigadores e publicações, tal como o JSON no caso do REST para a informação das instituições. Também criámos um Web front-end capaz de mostrar a lista de publicações que existem na base de dados, lendo a resposta SOAP e mostrando os dados numa tabela.

Concluímos que este tipo de aplicativos pode ser bastante útil a nível empresarial tal como permite uma melhor flexibilidade e acessibilidade aos dados.