

UNIVERSITÀ DI BOLOGNA

Scuola di Ingegneria
Corso di Laurea in Ingegneria Informatica

Relazione Tirocinio

Studente:

Giorgio Mastrotucci

Anno accademico 2021/2022

Contents

1	Introduzione	5
1.1	Cineca	5
1.2	Stand-up e sviluppo: metodologia Agile	6
2	Tecnologie utilizzate	7
2.1	Jet Database Engine	7
2.2	Maschere	8
2.3	Visual Basic for Application	8
3	Realizzazione	9
3.1	Access	9
3.1.1	VBA e scripting	11
3.2	Ricezione Json ed inserimento nel back-end	14
3.2.1	Da json ad oggetto java	16
3.2.2	insMig : inserimento nel database	18
3.2.3	Testing	19
4	Conclusioni	22

List of Figures

1.1	Organizzazione settoriale U-Gov Cineca	6
2.1	Microsoft Jet Database Engine DDLs	7
3.1	Struttura tabella Atenei	9
3.2	Struttura tabella Anagrafica	10
3.3	Maschera di selezione tabella	11

Abstract

Il tirocinio ha come obiettivo la migrazione da Database Microsoft Access ad Oracle Database, focalizzandosi inizialmente sull'esportazione di tabelle in formato JSON tramite l'utilizzo delle "maschere" fornite da Access e l'ambiente di sviluppo VBA (Visual Basic for Application).

Durante il suo svolgimento un corso di formazione su framework di sviluppo della soluzione Esse3. Durante il corso il tirocinante apprenderà i dettami architetturali della soluzione Esse3 di Cineca e imparerà a realizzare manufatti software con l'ausilio delle librerie Java sviluppate da Cineca. Il corso è erogato in modalità "training on the job" con l'inserimento in un team di sviluppo al fine di partecipare alla "vita" del team con partecipazione agli stand-up secondo la metodologia Agile.

Chapter 1

Introduzione

Questo progetto deriva dalla richiesta di alcune università musicali di dimensione medio/piccola di aderire al consorzio Cineca, le quali internamente utilizzano Access per salvare i propri dati.

Essendo questo il primo approccio al problema, si è deciso di affrontare l'importazione massiva, in modo da poter avere dati su cui basarsi, per le prossime implementazioni nel database aziendale.

1.1 Cineca

Il CINECA (Consorzio Interuniversitario dell'Italia Nord Est per il Calcolo Automatico, in seguito Consorzio INteruniversitario pEr il Calcolo Automatico) è un consorzio interuniversitario italiano senza scopo di lucro, cui aderiscono 69 università italiane, 2 Ministeri, 27 Istituzioni pubbliche Nazionali.

Il sistema per la gestione studenti (Student Management System) ESSE3 costituisce uno dei servizi “core” della suite dei prodotti Cineca a supporto della “Didattica e Studenti” in ambito universitario.

In primis ESSE3, permette di gestire tutto l'attraversamento o ciclo di vita dello studente, nei diversi livelli della formazione universitaria:

- Primo Ciclo: Corso di Laurea
- Secondo Ciclo - Corso di Laurea Magistrale, Master Universitari di I Livello
- Terzo Ciclo: Dottorato di Ricerca, Corso di Diploma di Specializzazione, Master Universitari di II Livello

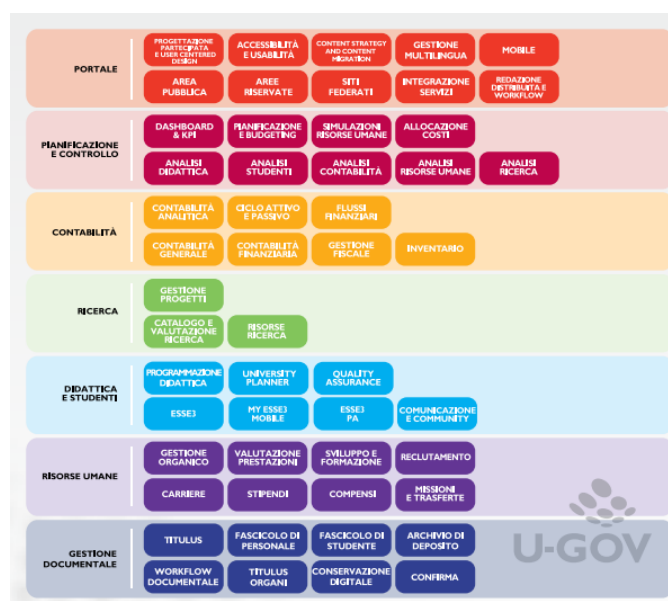


Figure 1.1: Organizzazione settoriale U-Gov Cina

1.2 Stand-up e sviluppo: metodologia Agile

Quando si parla di “Stand-up” si intende una riunione a cui prende parte tutto il team di lavoro, della durata di circa venti minuti ed organizzata con una frequenza molto elevata.

L’incontro ha la finalità di condividere le informazioni più importanti riguardo il progetto di lavoro e la rimozione dei “problemi” che precludono la buona riuscita dei lavori: fornisce quindi le basi per mantenere il controllo e tracciare il progresso dei lavori svolti, da svolgere e degli obiettivi condivisi.

Chapter 2

Tecnologie utilizzate

La principale tecnologia utilizzata è Access, un motore di database pseudo-relazionale di Microsoft. Fa parte della suite di applicazioni Microsoft Office che include anche Word, Outlook ed Excel, tra gli altri. Access utilizza Jet Database Engine per l'archiviazione dei dati, viene utilizzato per distribuzioni di database di piccole e grandi dimensioni, sfruttando l'interoperabilità con altre applicazioni e piattaforme come SQL Server di Microsoft e Visual Basic for Applications (VBA).

2.1 Jet Database Engine

Jet, acronimo di Joint Engine Technology, essendo parte di un RDBMS (Relational Database Management System), consente la manipolazione di database relazionali. Offre un'unica interfaccia che altri software possono utilizzare per accedere ai database Microsoft e fornisce supporto per la sicurezza, l'integrità, l'elaborazione delle transazioni, l'indicizzazione, il blocco di record e pagine e la replica dei dati. Nelle versioni successive, il motore è stato esteso per eseguire query SQL, archiviare i dati dei caratteri in formato Unicode, creare visualizzazioni di database e consentire la replica bidirezionale con Microsoft SQL Server.

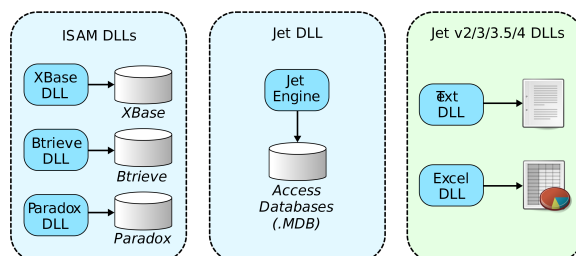


Figure 2.1: Microsoft Jet Database Engine DDLs

Esistono tre moduli per Jet: il primo è il driver ISAM Jet nativo, una DDL (libreria di collegamento dinamico) che può manipolare direttamente i file di database (MDB) di Microsoft Access utilizzando l'API dei file del sistema. Altri moduli contengono driver ISAM, DLL che forniscono l'accesso a vari database ISAM utilizzando metodi di accesso sequenziale, inclusi xBase, Paradox, Btrieve e FoxPro, a seconda della versione di Jet. L'ultimo modulo è la DLL Data Access Objects (DAO). DAO fornisce un'API che consente ai programmatori di accedere ai database JET utilizzando qualsiasi linguaggio di programmazione.

2.2 Maschere

Di grande importanza risultano le maschere, questi oggetti possono essere usate per creare un'interfaccia utente per un'applicazione di database.

Una maschera associata è collegata direttamente a un'origine dati come una tabella o una query che può essere usata per immettere, modificare o visualizzare dati da tale origine dati.

In alternativa è possibile creare una maschera non associata che non è collegata direttamente a un'origine dati ma contiene ugualmente pulsanti di comando, etichette o altri controlli necessari per usare l'applicazione.

2.3 Visual Basic for Application

È un'implementazione di Visual Basic che è integrata in applicazioni Microsoft come la suite Microsoft Office o Visio, altri programmi hanno implementazioni parziali di VBA (ad esempio AutoCAD e WordPerfect). VBA estende i linguaggi di scripting precedentemente inclusi in tali applicazioni (come Word Basic di Microsoft Word) e può essere utilizzato per controllare praticamente ogni aspetto dell'applicazione host, come l'interfaccia utente (manipolazione di menu, barre degli strumenti, ecc.).

I principali oggetti di questo linguaggio sono subroutine e funzioni.

La subroutine, chiamata anche procedura o macro, esegue automaticamente un insieme di operazioni, nella cartella, foglio o cella selezionate al momento del lancio. L'utente può aggiungere delle funzioni personalizzate, da lui definite. La funzione, a differenza della subroutine, richiede come input almeno un valore numerico o testuale per almeno una variabile indipendente.

Chapter 3

Realizzazione

3.1 Access

Il database preso in considerazione per la migrazione è così formato:

Figure 3.1: Struttura tabella Atenei

ATIS_ID	DES	ISTAT_COD
1	Università Politi	70001
2	Università degli	70002
3	Politecnico di B	70048
4	LUM "Jean Moni	70137
5	Università degli	70038
6	Università degli	70051
7	Università degli	70003
8	Libera Universit	70130
9	Università degli	70046
10	Università degli	70004
11	Università degli	70005
12	Università degli	70006
13	Università degli	70007
14	Università "Carli	70050
15	Università degli	70008
16	Università degli	70125
17	Università degli	70053
18	Università degli	70009
19	Università degli	70010
20	Università degli	70129
21	Università degli	70011
22	Università degli	70135
23	Università degli	70055
24	Università del S	70012
25	Università degli	70013
26	Università degli	70014
27	Università degli	70015
28	Università degli	70132
29	Politecnico di M	70066
30	Università Catt	70057
31	Università Com	70056
32	Libera Universit	70058
33	Libera Universit	70119
34	Università degli	70017
35	Università degli	70019
36	Università degli	70018
37	Università degli	70049
38	Università degli	70041
39	Università degli	70043

Figure 3.2: Struttura tabella Anagrafica

Tutti gli oggetti di Access

ANAGRAFICA

EXT_PERS_COD	CODICE_FIG	COGNOME	NOME	SESSO	DATA_NASC	NASC_NAZ_ID	DOM_NAZ_ID	RES_NAZ_ID	CITT_NAZ_ID	NASC_COM_ID_ITA
001	mstgrg0	Mastroiucchi	Giorgio	M	22/03/2001	1	1	1	1	9999
002	mrrs02	Rossi	mario	M	09/03/2022	0	0	0	0	9999
003	dfstfddsa	Duca	Luca	M	08/04/2022	0	0	0	0	9999

Records: 1 - 4 of 4

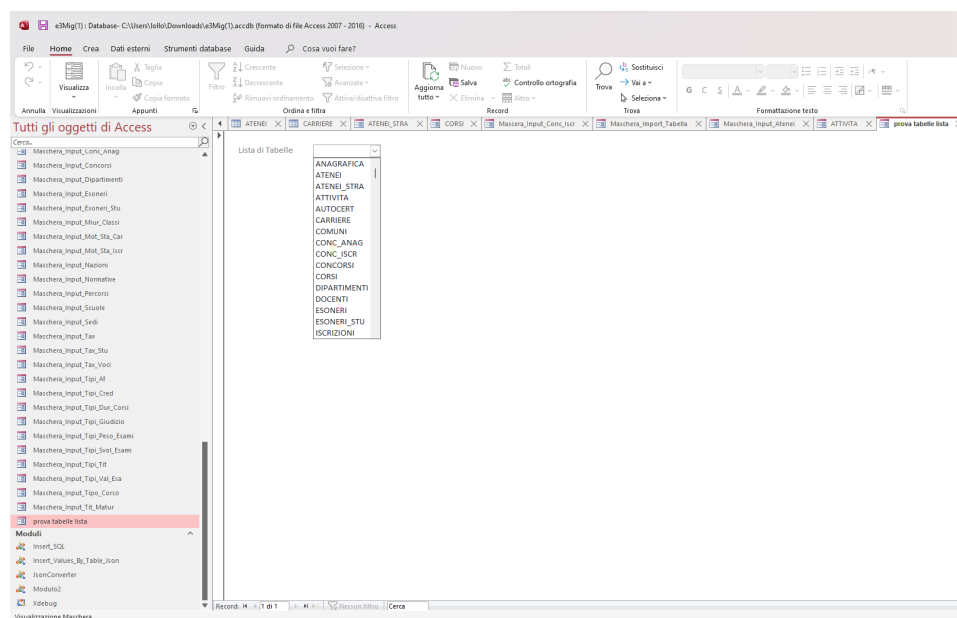
3.1.1 VBA e scripting

Il primo approccio alla migrazione avviene attraverso l'utilizzo delle maschere integrate in Access.

La maschera utilizzata in questo caso permette di selezionare tramite un menù a tendina la tabella interessata, una volta effettuata la selezione verrà generato un file JSON di quest'ultima tramite codice VBA.

VBA non può compilare un proprio progetto in un file eseguibile, ma dovrà sempre appoggiarsi all'applicativo Microsoft Office che lo ospita per poter eseguire il programma.

Figure 3.3: Maschera di selezione tabella



```

Option Compare Database
Function toJSON(PassTblQry)
' EXPORT JSON FILE FROM TABLE OR QUERY
Dim mydb As Database, rs As Recordset
Dim VarField(255), VarFieldType(255)
Dim fld As DAO.Field, VarDat As String
Set db = CurrentDb
fn = "C:\Users\giorgio.mastrotu.ext\Desktop\vba_code" & "\" &
    PassTblQry & ".json" ' define export current folder query
    date/time
Open fn For Output As #1 ' output to text file
Recs = DCount("*", PassTblQry) ' record count
Set rs = db.OpenRecordset("Select * from [" & PassTblQry & "]")
Nonulls = True ' set Nonulls = true to remove all null values
    within output ELSE set to false
fieldcount = 0
' Save field count, fieldnames, and type into array
For Each fld In rs.Fields
    fieldcount = fieldcount + 1
    VarField(fieldcount) = fld.Name
    'Debug.Print VarField(fieldcount)
    VarFieldType(fieldcount) = "TEXT"
    Select Case fld.Type
        Case 4, 5, 6, 7 ' fieldtype 4=long, 5=Currency, 6=
            Single, 7=Double
            VarFieldType(fieldcount) = "NUMBER"
    End Select
Next
Set fld = Nothing
Print #1, "[" ' start JSON dataset
' build JSON dataset from table/query data passed
Do While Not rs.EOF
    Print #1, "{" ' START JSON record
    ' build JSON record from table/query record using fieldname
    and type arrays
    For loop = 1 To fieldcount
        VarFT = VarFieldType(loop)
        If VarFT = "NUMBER" Then QuoteID = "" ' No quote
            for numbers
        QuoteID = Chr(34) ' double quote for text
        If IsNull(rs(VarField(loop)).Value) Then ' deal with
            null values
            VarDat = "Null": QuoteID = "" ' no quote for
                nulls
        If Nonulls = True Then VarDat = "": QuoteID = Chr
            (34) ' null text to empty
                quotes
        If Nonulls = True And VarFT = "NUMBER" Then VarDat
            = "0": QuoteID = "" ' null number to zero
                without quotes
        Else
            VarDat = Trim(rs(VarField(loop)).Value)
    Next loop
    Print #1, QuoteID & VarDat & "}"
    rs.MoveNext
Loop
Print #1, "]"
Close #1
End Function

```

```

End If
VarDat = Replace(VarDat, Chr(34), "'") ' replace double
      quote with single quote
VarDat = Replace(VarDat, Chr(8), "") ' remove
      backspace
VarDat = Replace(VarDat, Chr(10), "") ' remove line
      feed
VarDat = Replace(VarDat, Chr(12), "") ' remove form
      feed
VarDat = Replace(VarDat, Chr(13), "") ' remove
      carriage return
VarDat = Replace(VarDat, Chr(9), " ") ' replace tab
      with spaces
jsonRow = Chr(34) & VarField(looper) & Chr(34) & ":" &
      QuoteID & VarDat & QuoteID
If looper < fieldcount Then jsonRow = jsonRow & "," '
      add comma if not last field
Print #1, Chr(9) & jsonRow
Next looper
Print #1, "}"; ' END JSON record
rs.MoveNext
If Not rs.EOF Then
    Print #1, "," ' add comma if not last record
Else
    Print #1, ""
End If
Loop
Print #1, "]" ' close JSON dataset
Close #1

End Function

```

Il codice mostrato è presente nella maschera e permette di esportare un Json in locale partendo da una tabella del database.

Per prima cosa viene creato il file “*.json” e lo si definisce come output dello script, dopodiché viene letta una intera tabella e salvata in una variabile “Set” in questo modo:

```
Set rs = db.OpenRecordset("Select * from [" & PassTblQry & "]")
```

al termine dell’operazione viene salvato e chiuso il file precedentemente creato.

Ottenuta la variabile si effettuano due cicli:

- nel primo, “For Each fld In rs.Fields” si contano i campi della tabella e si salva il loro numero in “fieldcount” e vi si associa il tipo
- nel secondo, “Do While Not rs.EOF” si scorre fino alla fine il Set ed al suo interno vi è un terzo ciclo “For looper = 1 To fieldcount” che utilizza la variabile “fieldcount” creata precedentemente ed in questo modo si crea il dataset json che verrà scritto su file.

3.2 Ricezione Json ed inserimento nel back-end

Di seguito verrà mostrato il codice delle classi inerenti alla tabella “Lauree” come esempio, essendo l’organizzazione delle classi e il codice molto simile anche per le altre tabelle.

Il metodo generale “caricaTabelleMig” è utilizzato per caricare la tabella ottenuta dal database Access su quello Oracle del Cineca.

Le seguenti righe realizzano lo scopo del metodo, creando l’oggetto java inerente alla tabella presa in considerazione ed incaricando dell’inserimento il metodo “insMig”.

```
“ExportLauree exp = ResourceUtils.toObject (json, ExportLauree.class)”
“insMig(migElabDettId.value, exp)”
```

```
public class n_am_01_import_lauree_s extends NSession implements
↳ n_am_01_import_lauree_sPx{

    private boolean isValid = true;

    public int caricaTabelleMig(String json, RefLong
↳ migElabId, RefLong migElabDettId ) {
        final String METHODNAME = "importLauree";
        fpLog (METHODNAME, "INIZIO");
        int ret = Constants.OK;
        isValid = true;

        ExportLauree exp = ResourceUtils.toObject (json,
↳ ExportLauree.class);
        insMig(migElabDettId.value, exp);

        // Aggiornamento stato elaborazione TAA o TAE
        n_am_04_pkg04_sPx pxLog =
↳ (n_am_04_pkg04_sPx) SessionProxy.create (n_am_04_pkg04_sPx.COMP_NAME);
        if (isValid) {
            ret =
↳ pxLog.updateStatoElabMig (migElabId.value,
↳ "TAA");
            pxLog.fnCambiaStatoDettaglioElencoMig (
↳ migElabDettId.value, "TAA");
```

```
    } else {  
        ret =  
        ↪ pxLog.updateStatoElabMig(migElabId.value,  
        ↪ "TAE");  
        pxLog.fnCambiaStatoDettaglioElencoMig(  
        ↪ migElabDettId.value, "TAE");  
        if(ret == 1) {  
            ret = 0;  
        }  
    }  
  
    fpLog(METHODNAME, "FINE. Valore di ritorno " +  
    ↪ ret);  
  
    return ret;  
}
```

3.2.1 Da json ad oggetto java

All'interno del metodo “caricaTabelleMig” creiamo un oggetto che rappresenti la tabella che vogliamo inserire partendo da una stringa contenente il file json, utilizziamo “ExportLauree” per rappresentare la tabella, la quale si basa su “LaureeImport”, classe che rappresenta i singoli record e contiene i nomi degli attributi della tabella.

Di sotto viene mostrata la classe “ExportLauree” per maggiore chiarezza.

```
public class ExportLauree implements DecoratedModel {

    private static final long serialVersionUID = 1L;
    public static List < LaureeImport > LAUREE = new ArrayList <
        LaureeImport > ();
    @Debug @Key(value = "debugInfo", order = 0)
    private ModelDecorator decorator;

    public ExportLauree() {
        Esse3LayoutModel layoutModel =
            Esse3LayoutModelFactory.getInstance().newEsse3LayoutModel(this.getClass());
        decorator = new ModelDecorator(layoutModel, DebugLevel.NONE);
    }

    public static List < LaureeImport > getLauree() {
        List < LaureeImport > lista = LAUREE;
        return lista;
    }

    public void setLauree(List < LaureeImport > laurea) {
        ExportLauree.LAUREE = laurea;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("class ExportLauree {\n");
        sb.append("    LAUREE:
            ↪ ").append(toIndentedString(LAUREE)).append("\n");
        sb.append("}");
        return sb.toString();
    }

    private String toIndentedString(Object o) {
        if (o == null) {
            return "null";
        }
    }
}
```

```
    }  
    return o.toString().replace("\n", "\n    ");  
}  
public ModelDecorator getDecorator() {  
  
    return decorator;  
}  
  
public DecoratedModel cloneDecoratedModel() {  
  
    ExportLauree exportLauree = new ExportLauree();  
  
    exportLauree.setLauree(ModelUtils.cloneObj(LAUREE));  
    return exportLauree;  
  
}  
}
```

3.2.2 insMig : inserimento nel database

Il principale metodo della classe “caricaTabelleMig”, vista precedentemente, è ”insMig” che riguarda il binding tra i campi delle tabelle Access e quelle del database Oracle. Tramite il metodo “appendRow” che esegue l’ “execute” della classe “Insert” e permette l’inserimento della tabella nel database date le associazioni eseguite.

```
private void insMig(Long migElabDettId, ExportLauree exp) {

    MigLaureeDao migLaureeDao =
        ↳ BaseDao.newInstance(MigLaureeDao.class);
    List < LaureeImport > midLauree = ExportLauree.getLauree();
    Long migLaureeId = 0 L;

    AnnotationFieldValidator validator = new
        ↳ AnnotationFieldValidator(midLauree);
    List < DettaglioErroreAggiuntivo > errorList =
        ↳ validator.validateWithAnnotationsWithErrors(false);

    if (!KBase.isEmpty(errorList)) {
        insertLogErr(migElabDettId, errorList, "LAUREE");
    }

    for (LaureeImport element: midLauree) {

        InsertUpdateMap < FieldsMigLauree > bindings = new
            ↳ InsertUpdateMap < MigLaureeDao.FieldsMigLauree > ();
        migLaureeId = Long.valueOf(
            ↳ ProgressiviFactory.getProgr(NProgressiviRepository.MIG_LAUREE_MIG_LAUREE_ID))

        bindings.addBind(FieldsMigLauree.MIG_DOM_CT_ID,
            ↳ migLaureeId);

        bindings.addBind(FieldsMigLauree.AD_COD_TESI,
            ↳ element.getAD_COD_TESI());
        bindings.addBind(FieldsMigLauree.ANNO_CT,
            ↳ element.getANNO_CT());
        bindings.addBind(FieldsMigLauree.CORREL_DOC_MAT,
            ↳ element.getCORREL_DOC_MAT());

        bindings.addBind(FieldsMigLauree.DATA_CT,
            ↳ KBase.date(element.getData_CT(), DD_MM_YYYY));
```

```

bindings.addBind(FieldsMigLauree.DATA_INS,
    ↪ inTransaction.fnGetserverdatetime());
bindings.addBind(FieldsMigLauree.DES_SESSIONE,
    ↪ element.getDES_SESSIONE());
bindings.addBind(FieldsMigLauree.DOM_CT_ID, "");
bindings.addBind(FieldsMigLauree.EXT_STU_COD,
    ↪ element.getEXT_STU_COD());
bindings.addBind(FieldsMigLauree.GIUDIZIO,
    ↪ element.getGIUDIZIO());
bindings.addBind(FieldsMigLauree.LODE,
    ↪ setDefault(element.getLODE()));
bindings.addBind(FieldsMigLauree.MIG_ELAB_DETT_ID,
    ↪ migElabDettId);
bindings.addBind(FieldsMigLauree.REL_DOC_MAT,
    ↪ element.getREL_DOC_MAT());
bindings.addBind(FieldsMigLauree.TESI_ID, "");
bindings.addBind(FieldsMigLauree.TITOLO_TESI,
    ↪ element.getTITOLO_TESI());
bindings.addBind(FieldsMigLauree.USR_INS_ID,
    ↪ fpGetUserId(getSessionid()));
bindings.addBind(FieldsMigLauree.VOTO, element.getVOTO());
bindings.addBind(FieldsMigLauree.VOTO_MAX_LAUREA,
    ↪ element.getVOTO_MAX_LAUREA());
bindings.addBind(FieldsMigLauree.VOTO_MIN_LAUREA,
    ↪ element.getVOTO_MIN_LAUREA());

migLaureeDao.appendRow(bindings);

    }

}

```

3.2.3 Testing

Una volta create le classi per l'inserimento della tabella, il tutto può essere testato tramite il seguente metodo che instaura una connessione con il database Oracle e richiama i metodi per l'inserimento.

```

public void testInserimento() throws IOException {

```

```

BufferedReader buff = new BufferedReader(new
↳ FileReader(PATH_TO_JSON));
String json = "";
String line;
int i = 0;
while ((line = buff.readLine()) != null) {
    if (i == 1) {
        String nome = (String) line.subSequence(1, line.length() -
↳ 3);
        String ini = (String) line.subSequence(0, 1);
        String fin = (String) line.subSequence(line.length() - 3,
↳ line.length());
        line = ini + nome.toLowerCase() + fin;
        //System.out.println(line);
    }
    json += line;
    i++;
}

ServerConnection conn =
↳ RemoteSessionProxy.connect("http://localhost:9000/esse3be",
↳ "***", "***");

NSrvservicesPx logonSvc = RemoteSessionProxy.create(conn,
↳ NSrvservicesPx.class);
StrLogon logon = new StrLogon("***", "***");
KRef < StrSession > session = new KRef < StrSession > (new
↳ StrSession());
int rv = logonSvc.fnLogin("ESSE3", logon, session);

if (rv == 1) {

    ↳ SessionData.getCurrentInstance().setSessionid(session.value.sSessionid);
    try {

        n_am_import_carriera_mig_sPx px1 =
↳ RemoteSessionProxy.create(conn,
↳ n_am_import_carriera_mig_sPx.class);

        n_am_01_import_lauree_sPx px =
↳ RemoteSessionProxy.create(conn,
↳ n_am_01_import_lauree_sPx.class);
        System.out.println(px);

        RefLong refElabId = new RefLong();

```

```
RefLong refElabDettId = new RefLong();

px1.createElenco(null, null, null, refElabId,
    ↪ refElabDettId);
px.caricaTabelleMig(json, refElabId, refElabDettId);
assertTrue(json.length() > 0);
} finally {
    logonSvc.fnEndconnection(session.value.sSessionid);
}
}

assertTrue(json.length() > 0);
}
```

Chapter 4

Conclusioni

Il lavoro svolto, come visto, ha portato alla realizzazione di un'importazione massiva di tabelle, questa tipologia di importazione verrà utilizzata principalmente durante la migrazione tra database Access ed Oracle.

La maschera realizzata è stata sviluppata per essere utilizzata dalla segreteria dell'università di riferimento per comunicare i dati all'ambiente di sviluppo aziendale.

Lavorando con VBA ho potuto notare alcune caratteristiche ad esempio come sia perfettamente integrato con la suite Microsoft ma al contempo al di fuori dell'automazione e dell'interazione dei programmi di quest'ultima ha poche applicazioni e viene facilmente superato da altri linguaggi più recenti e di facile comprensione, di fatti è stato il primo ostacolo da superare durante il suo studio è stata la sintassi, quasi unica e molto diversa rispetto a quella dei linguaggi tradizionali.

Guardando al futuro sarà necessario programmare degli sviluppi per rispondere a specifiche esigenze come l'aggiunta di singoli record evitando di importare nuovamente l'intera tabella ad ogni modifica effettuata, questo potrà essere effettuato tramite l'utilizzo di un "form" (i "form" in Access sono un oggetto che è possibile utilizzare per creare un'interfaccia utente per un'applicazione di database). Come ulteriore miglioramento sarebbe utile una funzione che inoltri il json appena creato direttamente ad un ambiente dove gli sviluppatori possano utilizzarlo invece di doverlo trasferire in locale, questo velocizzerebbe il processo di migrazione.