

**Semester Project Report**

**TRACK PICKER**

Major : Multimedia

Student : LE Thi Nhu Nguyen

Supervisors: **Raphaël Troncy**

**Ghislain August Atemezing**

Valbonne, 10 June 2013

**CONTENTS**

[1. Introduction 1](#_Toc359354013)

[2. Project requirements 2](#_Toc359354014)

[3. Related work 2](#_Toc359354015)

[4. Curriculum regulations of Eurecom 2](#_Toc359354016)

[5. Vocabulary selection 4](#_Toc359354017)

[6. Data conversion: JSON to RDF 9](#_Toc359354018)

[7. TrackPicker Architecture 10](#_Toc359354019)

[8. TrackPicker implementation 11](#_Toc359354020)

[8.1. Sketching the application 11](#_Toc359354021)

[8.2. Languages and techniques for implementation 13](#_Toc359354022)

[8.3. Connecting to the triplestore and getting data 13](#_Toc359354023)

[8.4. Collecting the user’s choice 13](#_Toc359354024)

[8.5. Displaying the course and teacher information for user 14](#_Toc359354025)

[8.6. Checking the user’s choice 15](#_Toc359354026)

[8.7. Making the recommended track 15](#_Toc359354027)

[8.8. Statistics on Internship Offers at Eurecom 19](#_Toc359354028)

[9. Lesson learned 19](#_Toc359354029)

[10. Conclusion and future work 20](#_Toc359354030)

[APPENDIX 21](#_Toc359354031)

[REFERENCES 25](#_Toc359354032)

**LIST OF FIGURES**

[Figure 1. New properties related with InternshipOffer 5](#_Toc359353149)

[Figure 2. The graph representing REVE2 ontology 8](#_Toc359353150)

[Figure 3. TrackPicker workflow 11](#_Toc359353151)

[Figure 4. TrackPicker sketching 12](#_Toc359353152)

[Figure 5. User’s choice collecting page: 14](#_Toc359353153)

[Figure 6. Course information details in the case of the “ATMM” course 15](#_Toc359353154)

[Figure 7. Warning of user’s choice 15](#_Toc359353155)

[Figure 8. Announcement about the user’s choice 15](#_Toc359353156)

[Figure 9. The result of recommendation strategy with details explaining why it happens 17](#_Toc359353157)

[Figure 10. The editing page to fit the Eurecom rules checker 18](#_Toc359353158)

[Figure 11. Internship page 19](#_Toc359353159)

# Introduction

The idea of this project relies on the demand of engineering student at Eurecom. At the beginning of the first semester of engineering student, they are faced up to many choices in courses and deciding one track to pursue. Besides, they have to register courses such that they are satisfied with curriculum regulations of Eurecom. These tasks cause the confusion for students. In this circumstance, one web application to help students is necessary. This project is realized for this purpose.

The tendency of web development is semantic web. With semantic web, data can be shared and reused easily and effectively. This project inherits the data model and ontology of the project in last year – “Building data.eurecom.fr” [1]. The project “Building data.eurecom.fr” in the last year is done by Anne-Elisabeth Gazet. In that project, she designed an RDF data model to solve problem that is storing all of data of Eurecom such that this data can be interlinked with other data and reusable to develop many applications that provide useful services to students and professors. The data are about students, professors, courses, publications, departments, rooms, schedules, etc. The vocabulary is created to describe data which is REVE ontology. Many external vocabularies are selected to use such as:

FOAF [2]: use to describe people and their relations to other people and objects.

Dublin Core term [3]: use to describe numeric and physical resources.

Participation [4]: use to describe the roles that people play within group.

AIISO [5]: use to describe the internal organizational structure of an academic institution.

BIBO [6]: use to describe the articles in Eurecom’s scientific publications repository.

LODE [7]: use to publish descriptions of historical events, we use to describe course sessions.

OWL-Time [8]: use to describe the temporal aspects of course sessions.

Rooms [9]: use to describe rooms and buildings where the courses take place.

After having the REVE ontology and the triples of data, we can develop many applications. In the last project, Anne-Elisabeth Gazet developed the application which is a map of Eurecom publications, and she gave some more applications in the future work. Among of them, there is application to help students choose track at Eurecom, which is TrackPicker application that we are doing. We reuse the REVE ontology and edit and add some components to be satisfied with the requirements.

# Project requirements

The object of this project is to help students choose the courses that they want to study such that these courses are satisfied of the curriculum regulations of Eurecom [10]. These curriculum regulations here are about the amount of credits students have to obtain and the schedule of course.

From information about courses that students choose, we will recommend the suitable track and help student arrange the studying schedule. The technical courses are strong relationship with the track that student should choose. And in the technical courses, there are three types of courses which are mandatory, optional and free courses. There are constraints about the amount of credits according to each type of courses. So, we will count the amount of each type’s credits according to each track and return the recommended track and the details of our advices through the table of details which has how much credits of mandatory, optional and free courses according to each track.

# Related work

The requirements of this project is fairly similar to MIT Courses Picker project of MIT University (<http://picker.mit.edu/>). Course Picker is a scheduling tool to help MIT students plans their schedules using a semantic data source and a clean user interface. By choosing the fields of courses, the application will search all of courses related with this field and supply for student more criterion for student to filter the courses they want and the schedule for each courses. The application supplies the schedule interface for students arrange their study plan, count the number of credits they choose and some information about the courses they choose. After they finish the courses choice, they can login to register for those classes.

# Curriculum regulations of Eurecom

With engineering students, their curriculum has 4 semesters, two fall semesters and one spring semester, and the last one is the internship. There are many regulations that students have to comply with, they are listed in the Curriculum document of Eurecom. In this report, we summary the regulations which are related with the functions of the application.

At the moment we do this project, Eurecom supplies for engineering program 58 courses and 7 tracks. Each track contains 58 courses. But the difference of the different tracks is: one course is a mandatory course of track A, but this course is an optional course of track B, and is a free course of track C.

Every track have technical courses which include mandatory, optional and free courses; non-technical courses which include *Economy, Law, Human Sciences, Project Management*; and language courses.

These are the constraints of the amount of credits that students have to obtain:

* The amount of total credits for all of courses is equal or more than 90 credits. They have a maximum of 3 years to obtain 90 credits. However, before they go to do internship, they have to obtain at least 60 credits.
* The amount of technical credits for 3 semesters is equal or more than 60 credits, in which the mandatory credits is at least 25, optional credits is at least 18, and no minimum number for free credits.
* The amount of non-technical credits for 3 semesters is at least 12 credits, in which credits must be acquired in three different subject matters (*Economy, Law, Human Sciences, Project Management*).
* They obtain at least 6 credits in foreign languages, in at least two languages with 2 mandatory credits per semester.
* They cannot register to two courses that take place at the same time.

Therefore, we can summarize the constraints as below:

One student is qualified to graduate at Eurecom if he obtains:

In three semesters, at least 90 total credits, in which:

At least 60 technical credits

In technical credits, at least 25 mandatory credits and 18 optional credits

At least 12 non-technical credits and must be acquired in three different subject matters which are in *Economy, Law, Human Sciences and Project Management*.

At least 6 language credits and must be acquired in two foreign languages.

In each semester, at least 2 language credits and there are not two courses that take place at the same time.

On the contrary, he is not qualified to graduate.

The information of these constraints are expressed in natural language and have not had any form of storing yet. But there is no need to edit current ontology of Eurecom to store these information. The solution here is to hard store it into the application.

# Vocabulary selection

In this project, the REVE ontology is reused, and add some necessary components to use for new functions.

In the new ontology, we add some new classes to model our requirements

* **reve:LanguageCourse** class which is subclass of *reve:Course*.

This class is used to describe a language course which is one of three types (Language, Technical and Non-technical) of one course.

* **reve:Curriculum** class which is a title of a training period offered at Eurecom which consists of credits compatible with the LMD European standard.

At Eurecom, there are many curriculums that students pursue, one of which is Engineering, and each of curriculum has different number of semester and special schedule. In order to store these information, we add class *reve:Curriculum* with two properties between *reve:Curriculum* and *reve:Semester* as below.

* **reve:InternshipOffer** class to store a set of proposals of topics to be assigned to students in companies, or universities, as part of their curriculum. It is mandatory for each student to fulfill this condition to be graduated at Eurecom.
* **reve:Company** class to store companies that has internship offer for Eurecom students.

We use the organization ontology to store the name of the company.

We add some new properties of the relationship between Curriculum and Semester:

* **reve:hasDuration** property with domain *reve:Curriculum* and range *reve:Semester*. This property indicates which semester is belong to the curriculum.
* **reve:isDurationOf** property with domain *reve:Semester* and range *reve:Curriculum*. This property indicates one semester is belong to which curriculum. The *reve:isDurationOf* property is inverse of *reve:hasDuration*.

Some new properties of the relationship between internship offer and company as in the below figure:

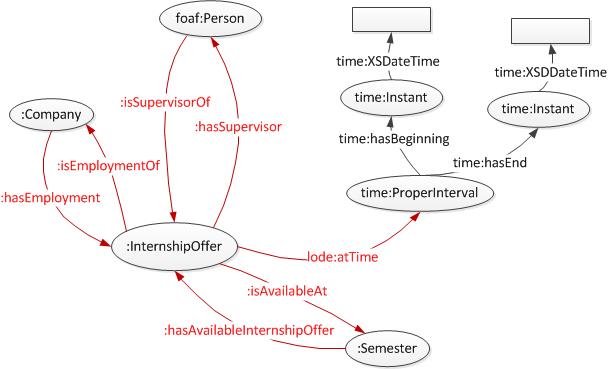


Figure 1. New properties related with InternshipOffer

* **reve:hasSupervisor** property with domain *reve:InternshipOffer* and range *foaf:Person*. This property indicates who the supervisor of this internship offer is.
* **reve:isSupervisorOf** property with domain *foaf:Person* and range *reve:InternshipOffer*. This property indicates one person is the supervisor of which internship offer. The *reve:isSupervisorOf* property is inverse of *reve:hasSupervisor*.
* **reve:isEmploymentOf** property with domain *reve:InternshipOffer* and range *reve:Company*. This property indicates one internship is belongs to which company.
* **reve:hasEmployment** property with domain *reve:Company* and range *reve:InternshipOffer*. This property indicates one company has the internship offer. The *reve:hasSupervisor* property is inverse of *reve:isSupervisorOf*.
* **reve:hasAvailableInternshipOffer** property with domain *reve:Semester* and range *reve:InternshipOffer*. This property indicates one semester has which internship offer.
* **reve:isAvailableAt** property with domain *reve:InternshipOffer* and range *reve:Semester*. This property indicates one internship is in which semester. This property is inverse of *reve:hasAvailableInternshipOffer*.
* **lode:atTime** property with domain *reve:InternshipOffer* and range *reve:ProperInterval*. This property indicates one internship offer is done in which time interval.

Some new properties of the relationship have been added between course and track:

* **reve:isFreeFor** property with domain *reve:Course* and range *reve:Track*. This property indicates one course is free course of one track.
* **reve:hasFreeCourse** property with domain reve:*Track* and range *reve:Course*. This property indicates one track has one course that is free course. This property is inverse of *reve:isFreeFor*.
* Add disjointness in pairs between *reve:GeneralCourse*, *reve:TechnicalCourse* and *reve:LanguageCourse*. This property is created to ensure that each course is only belong to one type of course.

And we model new instances for three current classes which are *reve:Curriculum*, *reve:GeneralCourse* and *reve:Semester*

* For *reve:Curriculum* class, we add the different tracks:

**reve:engineering** for engineering curriculum

**reve:masterOfScience**  for master of science

**reve:postmaster** for post graduate

**reve:doctoralProgram** for doctoral program

There are four program of Eurecom, each of them has the special schedule and the number of semester is different. So, we add these instances to store the information about semesters of each program and we can get this information to use when necessary.

* For *reve:GeneralCourse* class, we add three different subject matters :

**reve:law**

**reve:economy**

**reve:humanSciences**

**reve:projectManagement**

These are four matters in General Course, and each engineering student have to obtain at least three of these four matters in their three semesters. We store it here to define the matter of each general course in order to check this constraint when student choose courses they want to study.

* For “reve:Semester” class, we add four different semesters:

**reve:fall1**

**reve:spring**

**reve:fall2**

**reve:internship**

Each course is set up in the special semester and in the engineering curriculum, each student has two fall semester and one spring semester to study the courses. So, the choice of courses to study need to be defined at the beginning of the curriculum. These instances are created and have the connection with *reve:Course* to show which courses are in which semester.

The graph representing this ontology is in Figure 2 and the ontology is store in RDF/XML format [26].

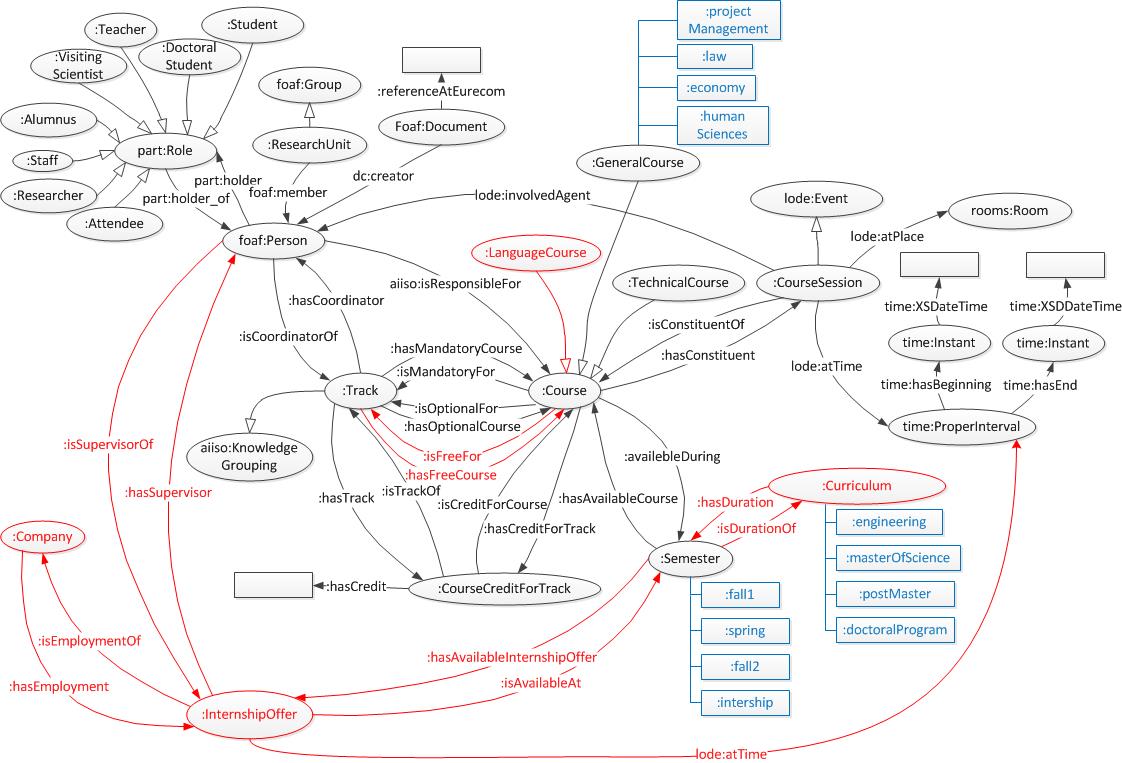


Figure 2. The graph representing REVE2 ontology

# Data conversion: JSON to RDF

The input data comes from an export of the internal database of Eurecom in JSON. Data is divided into many files: courses, courses sessions, tracks, publications, teacher, PHD, researcher, doctor.

We implement in Java a module to transform data to RDF using Jena RDF API [11]. Jena is a Java API which is used to create and manipulate RDF graphs. Jena has Resource interface, Property interface, Literal interface to represent resources, properties, literals and Model interface to represent graph. These classes are in package com.hp.hpl.jena.rdf.model of Jena. So, after referencing the Jena library into project and import these classes, we use them to create models, resources and properties according to data got from JSON file.

Besides using existing vocabulary that comes with Jena (for example, FOAF vocabulary), it is necessary to import the package com.hp.hpl.jena.ontology to create ontology model in our project to use (such as AIISO, LODE, TIME, ROOMS and REVE).

OntModel modelAIISO = null;

String AIISO\_URL= "http://vocab.org/aiiso/schema#";

modelAIISO = ModelFactory.createOntologyModel();

modelTrack.setNsPrefix("AIISO", AIISO\_URL);

After that, when we need to use the definitions of this ontology, we only need to call functions of the model like below:

eachTrack.addProperty(modelAIISO.getProperty(AIISO\_URL+"code"), modelTrack.createLiteral(track.get("code").toString()));

After converting all the necessary data from JSON file to RDF triples, we save these model in files.

About input and output file configuration, we read one configuring file in JSON format in which we list the name of input (JSON data file) and output file (RDF/XML file). Every year, some new files will be added in the dataset, we will add in the configuring file one line for each added file in data (for example, in 2014, coursesession2014 and internship2014 will be added). The construction of configuring file is as following:

{

"input":

{

"track": [

{

"name":"/Data/track.json"

}

],

"course": [

{

"name": "/Data/course.json"

}

],

…

}

"output":

{

"track":

{

"name": "/OutputRDF/track.rdf"

},

"course":

{

"name": "/OutputRDF/course.rdf"

},

…

}

}

We have not done the function to read data online directly from the Eurecom server, so whenever the Eurecom data updates, we have to put new data file on the Input folder of the Convert program and run it.

# TrackPicker Architecture

The TrackPicker application is divided into 4 modules: Conversion module, Checking module, Recommendation module and Statistics module. The function of each module is displayed bebow.

* **Conversion module**: This module is used to transform data of the internal database of Eurecom in JSON format to RDF/XML data file.
* **Checking module**: This module is used to check the user’s choice such that it is satisfied with the curriculum regulations of Eurecom about both the number of credits and the schedule to study.
* **Recommendation module**: This module is used to make the recommended track depend on the user’s choice.
* **Statistics module**: This module statistic the information about internship offer according to year, company and the number of internship offer.

TrackPicker workflow: In order to describe the sequence of work in this project, we depict it in the workflow as in the figure below:

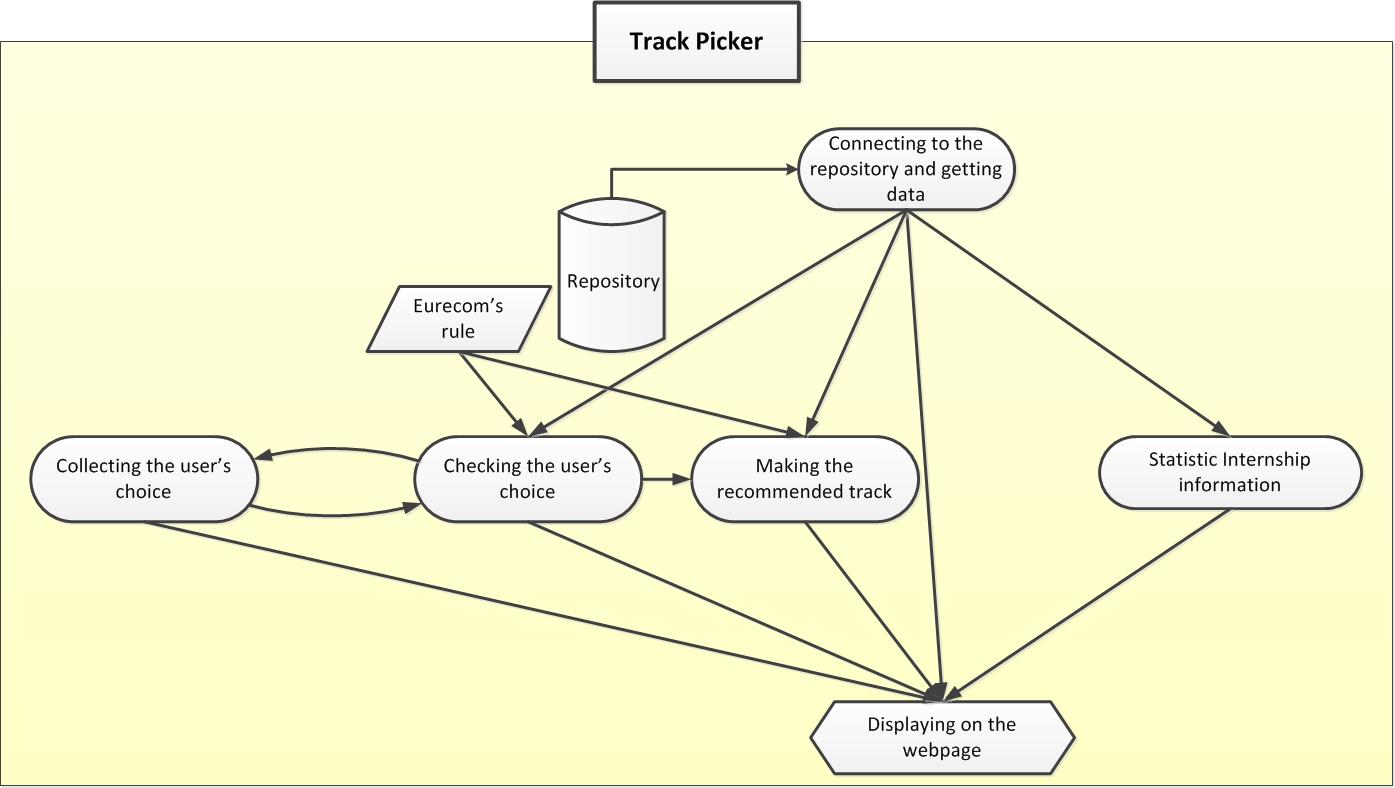


Figure 3. TrackPicker workflow

After converting data to RDF format, we put them on server. In Track Picker project, the first task is connecting to the repository and getting data to use in our project. We display information about courses, teacher and rule of Eurecom on the web interface. The next task is collecting the user’s choice, which is the courses that the users want to study. The parallel task with collecting the user’s choice is checking if the user’s choice is satisfied with Eurecom’s rule or not (both the rule of credit requirement and the rule of schedule). After that, we make the recommended track. We need display each step for the user. The remaining information that the user also need to know is the internship offer. In order to understand easily about the internship offer, we make some chart to display the statistic number of internship offer, and display for the user.

# TrackPicker implementation

## Sketching the application

In order to have a good picture about the web site interface, and display all important things that the users want from our application, we need to do the sketching task. We sketched all web page and all steps are needed to do in my application. But in this report, we only show one total picture of the collecting user’s choice step of the sketching, to show the basic ideas about the interface of the application as in the below figure.

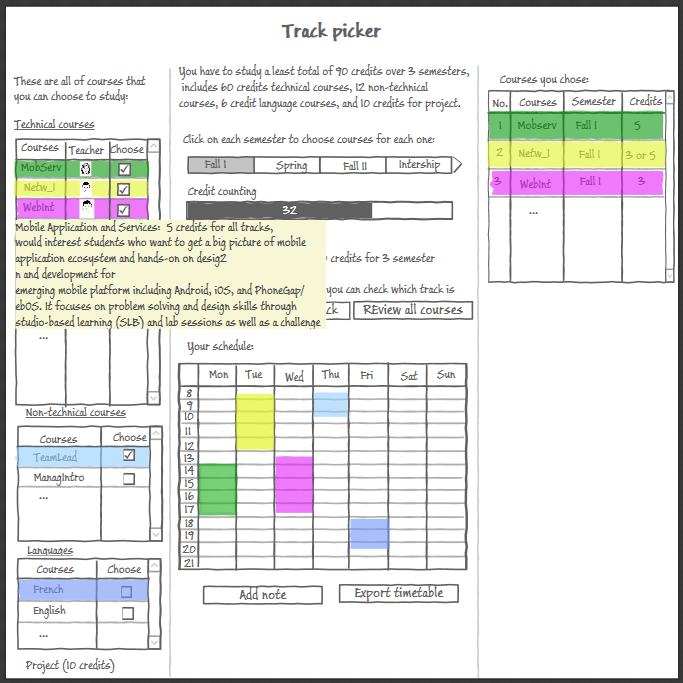


Figure 4. TrackPicker sketching

Design requirements: In the interface of our application, we need to arrange the list of all the courses that Eurecom supplies, the information of each course, the rule about the credits Eurecom, the list of courses that user choose and the schedule of the courses appropriately. Besides, we also need to display some information while the user choose course such as the amount of credits that the user chose and colorizing the course in the list according to teacher’s department for user to easily understand, without mixing up all the amount of information.

## Languages and techniques for implementation

Besides using HTML5, CSS3 as languages to display and define the layout for the web page, Javascript will be used for processing in the project because of many advantages which are listed below:

**Compatibility:** Website users do not need special software or downloads to view JavaScript. Most major Internet browsers support using Javascript.

**Client- Side processing:** JavaScript is quick to load and execute from webpages because the Internet browser does all of the processing work.

**Development:** JavaScript does not require any special compilers or editors with which to work. We can code JavaScript on any text editor.

## Connecting to the triplestore and getting data

We use AJAX to connect to the server and get data for the application. We can take advantage of AJAX in the project:

* The XMLHttpRequest object is used to exchange data with a server behind the scenes.
* AJAX allows web pages to be updated asynchronously, without reloading the whole page. In this project, when the user changes the semester to choose courses, the web page does not need to load again, only the necessary parts are reloaded.
* Waiting time and the traffic to and from the server is reduced.

## Collecting the user’s choice

We collect the list of courses that the user want to study according to each semester. As in the below figure, the list of courses that the user can choose is showed in left hand side, and the list of courses that the user choose is in the right hand side. The courses is classified to show by the semester and the type of course (technical or non-technical courses) and colorized depend on the teacher’s department.

The user can choose courses by checking into checkbox or drag and drop course into the schedule.

In order to help the user know how much credits they choose and which is lacked in their choice, we make the credit counting bar and the comment to pay attention to them.



Figure 5. User’s choice collecting page:

On the left, list of coursesl; On the center, the semester and a calendar and on the right, the list of user’s courses choice.

## Displaying the course and teacher information for user

When the mouse is moving on the name of course, the information is displayed to supply more information for user.

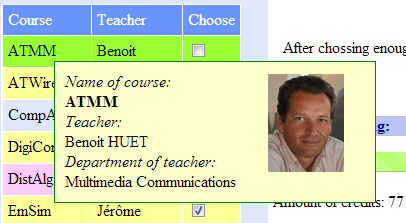
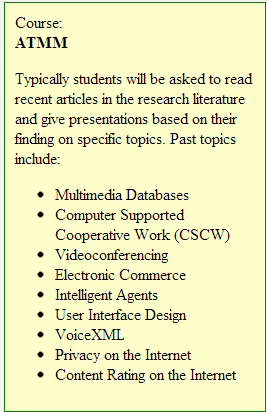
 

Figure 6. Course information details in the case of the “ATMM” course

## Checking the user’s choice

Every time the user choose one courses, besides the work of counting credits and marking on the schedule, we have to check the user’s choice with the rule of Eurecom (both about the rule of credits and guaranteeing that the schedule is not overlapped). And we show the warning if it exists the problem.

As in the image below, the application displays the warning with the choice of the user.

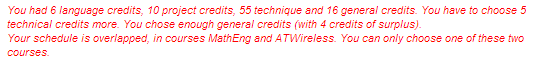


Figure 7. Warning of user’s choice

On the contrary, if there is not any warning, the application tells to the user that they choose enough courses and the “Check track” is set to disable.



Figure 8. Announcement about the user’s choice

## Making the recommended track

The way of processing recommended track is based on the following rule: The track which has the smallest of the sum of the lack of mandatory credits and optional credits amount is chosen track.

SUM (lack of mandatory credits, lack of optional credits) == smallest

On the other hand, we make this rule in the math formulation:

If there are many tracks that has the same of this sum, the application shows all these tracks, the users will decide by themselves.

In the previous step, the user only choose courses such that it is enough technical, non-technical and total credits, they do not know their choice is for which track. So, with each track, the user choice will be lacked some of mandatory or optional credits, they have to edit their choice such that it is satisfied with the rule of Eurecom about the mandatory and optional credits.

As in the picture below, it is the result of recommended track (for example, the result is “Transmission Technologies”), the user can see the details and choose track by himself (in this case, the user want to choose “Real-time and Embedded systems”). When the user click on the “Edit your choices” button, we will load the editing choice page as the next image.

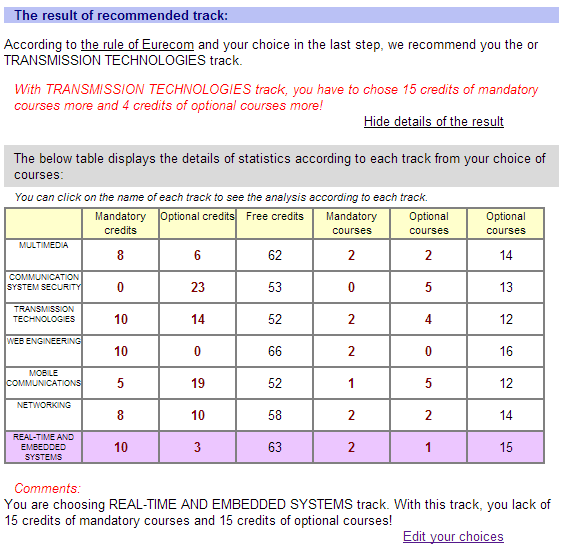


Figure 9. The result of recommendation strategy with details explaining why it happens

We use the web storage (SessionStorage capability) to store necessary values when moving the other pages.

In the editing choice page, the list of courses will be colorized according to the type of courses (mandatory, optional or free courses) and depends on the track that user choose.

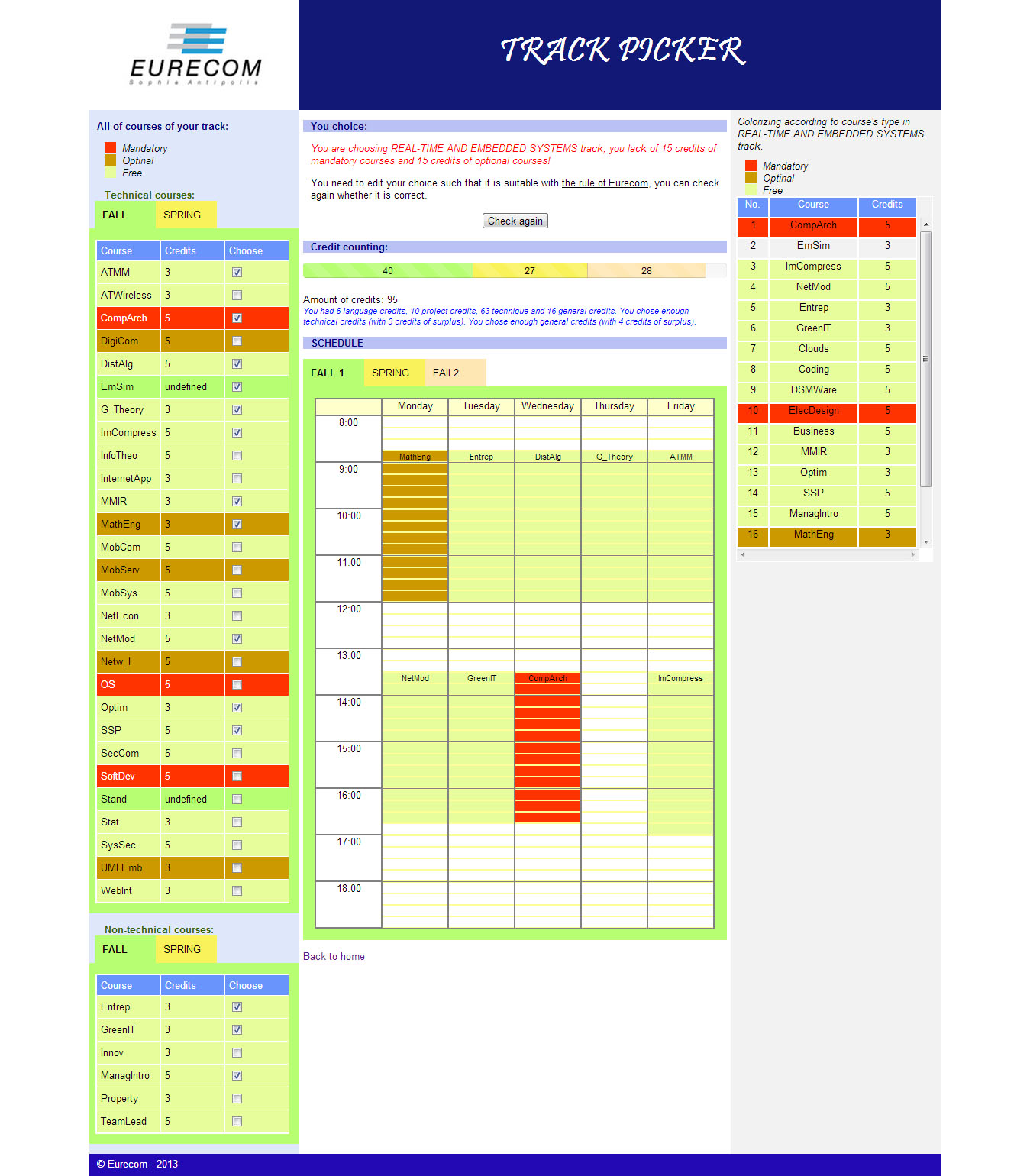


Figure 10. The editing page to fit the Eurecom rules checker

Depending on the color of the courses, the user know the type of course, and they can arrange their choice thanks to the schedule, and the comment line on the top of the page will announce with the user about the problem of their choice.

## Statistics on Internship Offers at Eurecom

In order to supply more information about the internship offer, we create the internship page to display the internship list and some charts that summary the internship situation of previous years.

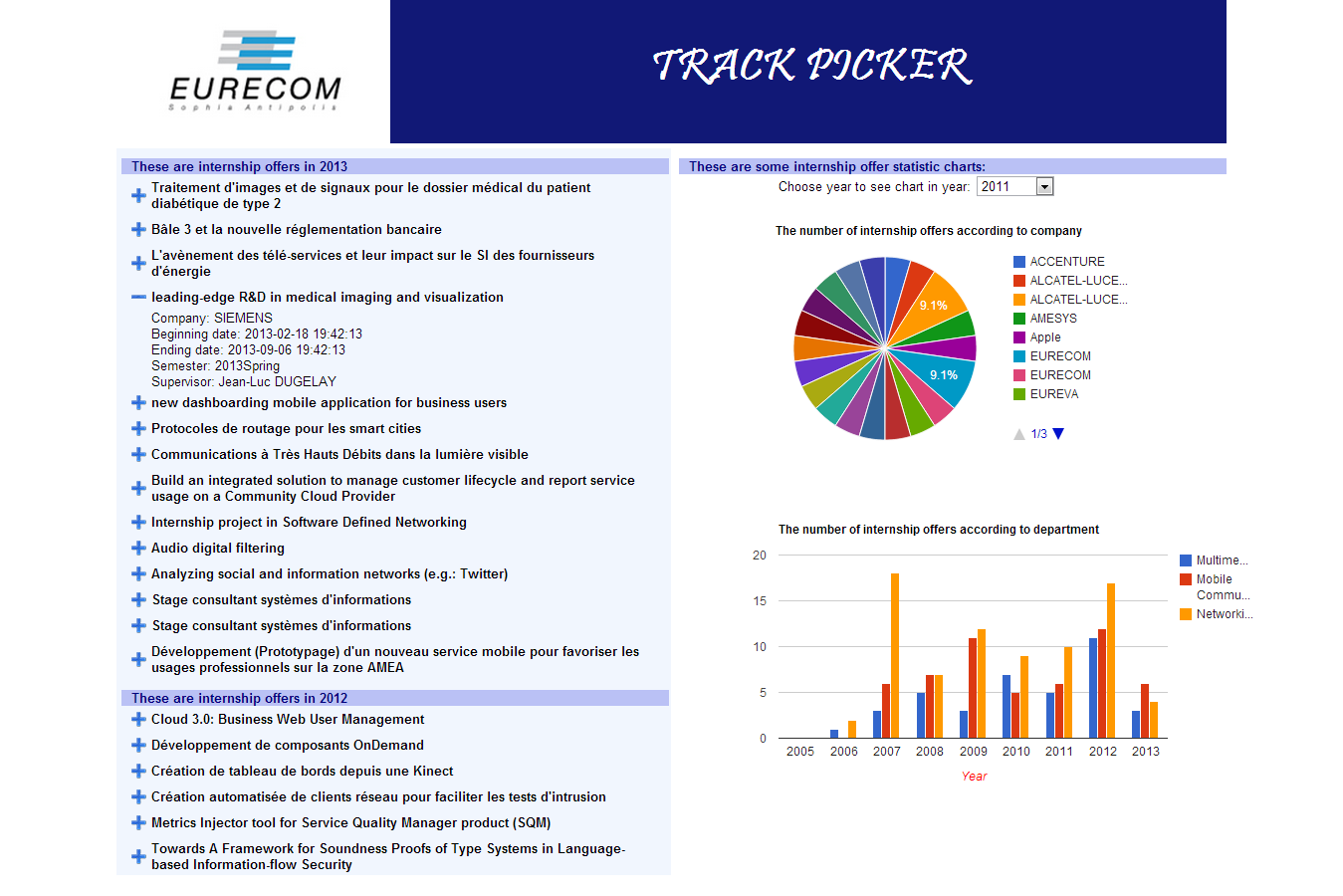


Figure 11. Internship page

In the above image, on the left hand side is the list of internship offer in each year. On the right hand, the first chart displays the percentage of internship offer according to each company and year. The second chart is statistic about the number of internship offers according to the department of the supervisor of all years. It is possible to choose on the top of the page the year in which the statistics are displayed.

# Lesson learned

In the progress of making the TrackPicker application, we found out some lacks in the data of Eurecom. We also help to detect some errors that is not easy to realize in the current database.

* In order to colorize the list of courses according to the teacher’s department of course, we need the information about the department of teacher. But in the database, there does not exist this information for each teacher in *teacher.json* file. However, this information is found in the *researcher.json* file. After this lack is discovered, it is corrected.
* Besides, we also detect the lack of the information about the type of course (mandatory, optional or free) and the number of credits for course according to each track.
* This project creates the demand for supplementing the information about the image of teacher and the information of internship offer in the database of Eurecom.

# Conclusion and future work

The project created the basic web application to help the students in choosing courses to study arrange their schedule such that it does not violate the rule of Eurecom and suggest for the student the track that they can study depend on their studying hobby. The application can help the students reduce the confusing with many choices in Eurecom and they can imagine the first step that they have to decide in their study in Eurecom.

Project improvement: The project have some problems that should be improved if time permits.

* Solve problems of updating data – Reread data directly from Eurecom server according to the defined time frame.
* The function for user can take notes on the calendar and export this timetable and notes for user.
* The integration of social profile of the user in the recommendation process. For example, using Google+, Twitter or Facebook account to log in and use network relationship with other friends to choose a course.

# APPENDIX - SPARQL QUERIES

In this appendix, we list the all of query sentences that we used in the TrackPicker application.

The query sentence to get information of all courses (course's name, semester, description, teacher's name, teacher's image, teacher's department):

SELECT ?course (sql:SAMPLE (?semester) AS ?semester) (sql:SAMPLE (?des) AS ?des) (sql:SAMPLE (?teacherFirstName) AS ?teacherFirstName) (sql:SAMPLE (?teacherLastName) AS ?teacherLastName) (sql:SAMPLE (?teacherImage) AS ?teacherImage) (sql:SAMPLE (?department) AS ?department) (sql:SAMPLE (?courseType) AS ?courseType)

FROM <http://data.eurecom.fr/reve>

WHERE {

{?code AIISO:code ?course. ?code rdf:type REVE:Course} {{?code rdf:type ?courseType}{?courseType rdfs:subClassOf REVE:Course}}

OPTIONAL {{?code REVE:availableDuring ?semesterID} {?semesterID rdfs:label ?semester}} {?code AIISO:responsibilityOf ?teacherID} {?teacherID FOAF:firstName ?teacherFirstName} {?teacherID FOAF:family\_name ?teacherLastName}

OPTIONAL {?teacherID FOAF:img ?teacherImage}

OPTIONAL {?teacherID PART:holder\_of ?role. ?role PART:role\_at ?departmentID. ?departmentID rdfs:label ?department}

OPTIONAL {?code DC:description ?des}

}GROUP BY ?course ORDER BY ?course

The query sentence to get name of all courses:

SELECT ?course

FROM <http://data.eurecom.fr/reve>

WHERE {{?code AIISO:code ?course. ?code rdf:type REVE:Course}} GROUP BY ?course

ORDER BY ?course

The query sentence to get the schedule of one course with name of course is replaced with ‘courseName’ in this query:

SELECT ?course ?timeB ?timeE

FROM <http://data.eurecom.fr/reve>

WHERE {

{?code AIISO:code 'courseName'. ?code AIISO:code ?course. ?code rdf:type REVE:Course}

{?code REVE:hasConstituent ?sess}

{?sess LODE:atTime ?session}

OPTIONAL{{?session TIME:hasBeginning ?nodeB} {?nodeB TIME:inXSDDateTime ?timeB}}

OPTIONAL{{?session TIME:hasEnd ?nodeE} {?nodeE TIME:inXSDDateTime ?timeE}}

}

The query sentence to get type of course and amount of credits of all courses according to one track, with the name of track is replaced with ‘trackName’ in this query:

SELECT DISTINCT ?course ?typeCourse ?credit ?trackID

FROM <http://data.eurecom.fr/reve>

WHERE {

{?code AIISO:code ?course. ?code rdf:type REVE:Course}

OPTIONAL {?code ?typeCourse ?trackID. ?trackID rdf:type REVE:Track. ?trackID rdfs:label \""+ trackName +"\"@en}

OPTIONAL {?cCFT REVE:isCreditForCourse ?code. ?cCFT REVE:hasCredit ?credit. ?cCFT REVE:hasTrack ?trackID. ?trackID rdfs:label "trackName"@en }

}ORDER BY ?course

The query sentence to get course’s name and course’s teacher of all technical courses:

SELECT ?course (sql:SAMPLE (?teacher) AS ?teacher)

FROM <http://data.eurecom.fr/reve>

WHERE {

{?code AIISO:code ?course. ?code rdf:type REVE:Course. ?code rdf:type REVE:TechnicalCourse}{?code AIISO:responsibilityOf ?teacherID}{?teacherID FOAF:firstName ?teacher}

} GROUP BY ?course ORDER BY ?course

The query sentence to get course’s name, course’s teacher of all courses in one semester with semester’s name is replaced with ‘semesterName’ in this query:

SELECT ?course (sql:SAMPLE (?teacher) AS ?teacher) (sql:SAMPLE (?semester) AS ?semester)

FROM <http://data.eurecom.fr/reve>

WHERE {

{?code AIISO:code ?course. ?code rdf:type REVE:Course. ?code rdf:type REVE:TechnicalCourse}

{?code AIISO:responsibilityOf ?teacherID}

{?teacherID FOAF:firstName ?teacher}

{{?code REVE:availableDuring ?semesterID}

{?semesterID rdfs:label ?semester}}.

FILTER regex (?semester, "semesterName", "i")

} GROUP BY ?course ORDER BY ?course

The query sentence to get course’s name and course’s teacher of all technical courses:

SELECT ?course (sql:SAMPLE (?teacher) AS ?teacher)

FROM <http://data.eurecom.fr/reve>

WHERE {

{?code AIISO:code ?course. ?code rdf:type REVE:Course. ?code rdf:type REVE:GeneralCourse}{?code AIISO:responsibilityOf ?teacherID}{?teacherID FOAF:firstName ?teacher}

} GROUP BY ?course ORDER BY ?course

The query sentence to get the teacher’s department of each course:

SELECT ?code ?department

FROM <http://data.eurecom.fr/reve>

WHERE {

{?course AIISO:code ?code. ?course rdf:type REVE:Course}

{?course AIISO:responsibilityOf ?teacherID}

{?teacherID PART:holder\_of ?role. ?role PART:role\_at ?departmentID. ?departmentID rdfs:label ?department}

}

The query sentence to get the name of all companies that have internship offer and count the number of internship offer with Eurecom:

SELECT ?companyName (COUNT(?companyName) as ?count)

FROM <http://data.eurecom.fr/reve>

WHERE {

{?internship rdf:type REVE:InternshipOffer. ?internship rdfs:label ?internshipTitle}

{?internship REVE:isEmploymentOf ?company. ?company SKOS:prefLabel ?companyName}

}GROUP BY ?companyName ORDER BY ?companyName

The query sentence to get the information of each internship offer (name, the supervisor’s name, the department of the supervisor, the semester of this offer, the begin and the finish day of this offer):

SELECT ?internshipTitle ?companyName ?firstName ?lastName ?image ?department ?semester ?begindate ?enddate

FROM <http://data.eurecom.fr/reve>

WHERE {

{?internship rdf:type REVE:InternshipOffer. ?internship rdfs:label ?internshipTitle }

{?internship REVE:isEmploymentOf ?company. ?company SKOS:prefLabel ?companyName}

OPTIONAL{?internship LODE:atTime ?interval. ?interval TIME:hasBeginning ?begindateNode. ?begindateNode TIME:inXSDDateTime ?begindate. ?interval TIME:hasEnd ?enddateNode. ?enddateNode TIME:inXSDDateTime ?enddate}

OPTIONAL {?internship REVE:hasSupervisor ?supervisor. ?supervisor FOAF:firstName ?firstName. ?supervisor FOAF:family\_name ?lastName. ?supervisor FOAF:img ?image. ?supervisor PART:holder\_of ?role. ?role PART:role\_at ?departmentID. ?departmentID rdfs:label ?department}

{?internship REVE:isAvailableAt ?semesterID. ?semesterID rdfs:label ?semester}

}

# REFERENCES

[1] Anne-Elisabeth Gazet. *REVE Ontology,* 2011. <https://github.com/aegazet/REVE-Ontology>

[2] Dan Brickley, Libby Miller. *FOAF Vocabulary*, 2000-2010. <http://xmlns.com/foaf/spec/>

[3] DCMI Usage Board. *Dublin Core terms*, 2012. <http://dublincore.org/documents/dcmi-terms/>

[4] Rob Styles, Chris Wallace and Knud Moeller. *Participation Schema,* 2008. <http://vocab.org/participation/schema>

[5] Rob Styles and Nadeem Shabir. *AIISO Ontology*, 2008. <http://vocab.org/aiiso/schema>

[6] Bruce D'Arcus, Frédérick Giasson. BIBO Ontology, 2008-2013. <http://bibliontology.com/specification>

[7] Ryan Shaw. *LODE Ontology*, 2010. <http://linkedevents.org/ontology/>

[8] Jerry R. Hobbs and Feng Pan. *Time Ontology in OWL*, 2006. <http://www.w3.org/TR/owl-time/>

[9] Richard Cyganiak. *Buildings and Rooms Vocabulary*, 2012. <http://vocab.deri.ie/rooms>

[10] *Curriculum regulations of Eurecom.* <http://www.eurecom.fr/sites/www.eurecom.fr/files/documents/2-Formation/Formation_Ingenieur/CURRICULUM%20REGULATIONS%20inges%2024.pdf>

[11] *Apache Jena*. <http://jena.apache.org/>

[12] Toby Segaran, Colin Evans, and Jamie Taylor, *Programming the Semantic Web book*, 2009

[13] *Jena tutorials*. <http://jena.apache.org/tutorials/index.html>

[14] *Java Web Programming with Eclipse*. <http://cse.csusb.edu/turner/java_web_programming/>

[15] *User Guide for Sesame 2.3*.

<http://www.openrdf.org/doc/sesame2/2.3.2/users/userguide.html#chapter-repository-api>

[16] *Pellet Reasoner Plug-in for Protégé 4*. <http://clarkparsia.com/pellet/protege/>

[17] *Introducing JSON*. <http://json.org/>

[18] *AJAX Tutorial*. <http://www.tutorialspoint.com/ajax/what_is_xmlhttprequest.htm>

[19] *SPARQL Query Language for RDF.* <http://www.w3.org/TR/rdf-sparql-query/>

[20] *An organization ontology*. <http://www.w3.org/TR/vocab-org/#class--organization>

[11] *Virtuoso Universal Server*. <http://virtuoso.openlinksw.com/>

[22] *Google code website – Session storage tutorial.* <https://code.google.com/p/sessionstorage/>

[23] *Google charts.* <https://developers.google.com/chart/>

[24] Dave Reynolds, Epimorphics Ltd. *An organization ontology,* 2012. <http://www.w3.org/TR/vocab-org/#class--organization>

[25] *Eurecom Website*, <http://www.eurecom.fr/en/teaching>

[26] *The resourses at Github account*, <https://github.com/lethinhunguyen/TrackPicker/tree/Version1.0>