POLITECNICO
MILANO 1863

Rest Api Project

https://github.com/marckw94/HS

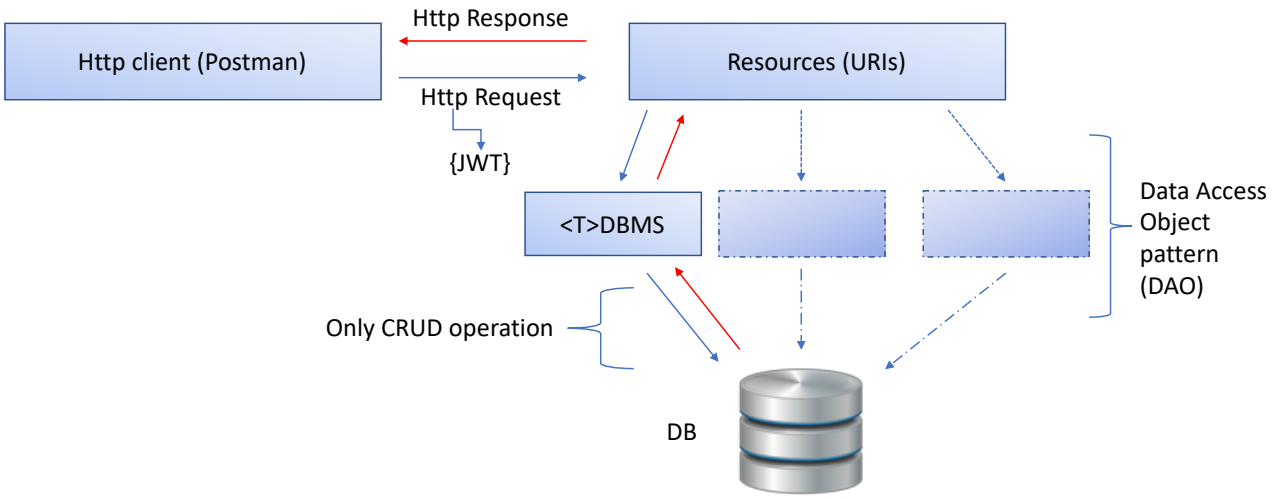
Middleware Technologies for Distributed Systems
Prof. Guinea Sam

Paola Sanfilippo 882892
Francesco Tinarelli 883738
Marco Wenzel 883732

INDEX

General Architecture

We decided to use the JAX-RS framework, because, thanks to the annotations, it allows to create a REST API in a more efficient and easier way.
In addition to this we chose to use Hibernate, because it allows us to map java classes into database tables, keeping the same relations among entities. Together with hibernate we used the DAO paradigm for the database: we created a DBMS so that we could use the correct CRUD operation, according to the resource that calls it.

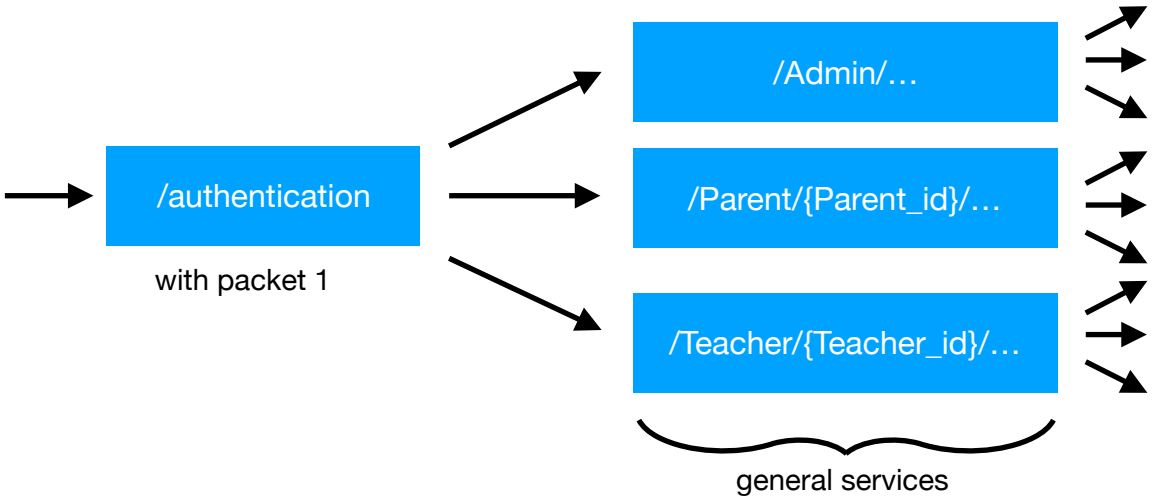


Installation

- The repository of the project can be found at https://github.com/marckw94/HS
- Configure a maven project on eclipse
- Download and configure the latest version of Xamp server (for mac) or Wamp server (for windows)
- Download the latest version of Hibernate from http://hibernate.org/orm/releases/
- Add the Hibernate libraries to the maven project
- Create an empty database on the chosen local server
- Create a hibernate configuration file (hibernate.cfg.xml): it must conform to the hibernate 3 configuration DTD which is available at hibernate site.
- Edit hibernate configuration file inserting the credentials and the url of the local database

Implementation choices

- At first users log in at “/authentication”. Once he’s logged in, the path is built accordingly. The different resources and methods are available in the next chapters.



- There’s an exception for what concerns the Appointment resource: as it is a resource shared between teachers and parents, the path is “Appoint/{user_id}/…”, instead of starting with the kind of user.
- The Course resource has been thought as the combination of a specific subject, taught by a teacher to a class. This means that we could have different courses with the same subject taught by the same teacher but in different classes.
- The authentication part is managed through JWT (JSON Web Token, see TokenManager.java). The hashing algorithm used is HMAC256, the token has 2 custom claims, username and category (Admin, Teacher or Parent), and expires after one hour of usage.
- Statelessness: no information is stored on the client side. The only usage of cookies that we perform is intended to store the JSON web tokens.
- For what concerns the content negotiation, users can perform both XML and JSON requests.
- Exception Handling: every status code, that we used, is handled with a specific exception that explains what went wrong
- In order to make easier to read the http response and to avoid the creation of an ad hoc container, when the response is composed by a list of elements, the common links are shown only for the first element.

Resources

Admin

(In yellow there are the methods that the Admin can reach from the “general services” page)

Method	Packet	Reachable links
• newAdministrator (POST) (create a new admin)	2	• self • general services
• newParent (POST) (create a new parent)	4	• Parent/{parent_id}/newSon • allParents • self • general services
• newClass (POST) (create a new class)	6	• newCourse • newClassCourse/{class_id}/{course_id} • allClasses • self • general services
• newTeacher (POST) (create a new teacher)	3	• newCourse • newTeacherCourse/{teacher_id}/{course_id} • allTeachers • self • general services
• newCourse (POST) (create a new course)	7	• newTeacher • newTeacherCourse/{teacher_id}/{course_id} • allCourses • self • general services
• newPayment (POST) (create a new payment)	8	• self • general services
• newTeacherCourse/{teacher_id}/{course_id} (PUT) (create a new association between a specific teacher and course)		• Teacher/{teacher_id} • Course/{course_id} • self • general services
• newClassCourse/{class_id}/{course_id} (POST) (create a new association between a specific class and course)		• Class/{class_id} • Course/{course_id} • allClassCourse • self • general services
• allParents (GET) (returns all parents)		• Parent/{parent_id} • newParent • self • general services
• allClasses (GET) (returns all classes)		• studentsPerClass/{class_id} • newClass • self • general services
• allTeachers (GET) (returns all teachers)		• Teacher/{teacher_id} • newTeacher • self • general services

Method	Packet	Reachable links
• allCourses (GET) (returns all courses)		• Course/{course_id} • newCourse • self • general services
• allStudents (GET) (returns all students)		• Parent/{parent_id}/newSon • Student/{student_id} • self • general services
• allClassCourse (GET) (returns all associations between classes and courses)		• Course/{course_id} • Class/{class_id} • self • general services
• studentsPerClass/{class_id} (GET) (returns all the students of a specific class)		• Student/{student_id} • enrollment • self • general services
• Parent/{parent_id} (GET) (returns a specific parent)		• Parent/{parent_id}/newSon • self • general services
• Class/{class_id} (GET) (returns a specific class)		• TimeTable/{class_id} • studentsPerClass/{class_id} • newClassCourse/{class_id}/{course_id} • self • general services
• Teacher/{teacher_id} (GET) (returns a specific teacher)		• newTeacherCourse/{teacher_id}/{course_id} • newCourse • self • general services
• Course/{course_id} (GET) (returns a specific course)		• newTeacherCourse/{teacher_id}/{course_id} • newTeacher • newClassCourse/{class_id}/{course_id} • self • general services
• Student/{student_id} (GET) (returns a specific student)		• self • general services
• TimeTable/{class_id} (GET) (returns the timetable of a specific class)		• self • general services
• enrollment (PUT) (enroll a student in a class)	14	• studentsPerClass/{class_id} • Class/{class_id} • Student/{student_id} • self • general services
• TimeTable/{course_id} (POST) (adds a time slot for a course)	13	• self • general services
• Parent/{parent_id}/newSon (POST) (adds a child to a parent)	5	• allClasses • enrollment • self • general services

Method	Packet	Reachable links
• Notif/newParentNotif/{user_id} (POST) (creates a new notification to a parent)	11	• Notif/allParents • self • general services
• Notif/newTeacherNotif/{user_id} (POST) (creates a new notification to a teacher)	11	• Notif/allTeachers • self • general services
• Notif/allParents (POST) (creates a new notification to all the parents)	11	• Notif/newParentNotif/{user_id} • self • general services
• Notif/allTeachers (POST) (creates a new notification to all the teachers)	11	• Notif/newTeacherNotif/{user_id} • self • general services
• Notif/ClassParents/{class_id} (POST) (creates a new notification to all the parents of a class)	11	• Notif/newParentNotif/{user_id} • Notif/classTeachers/{class_id} • self • general services
• Notif/ClassTeachers/{class_id} (POST) (creates a new notification to all the teachers of a class)	11	• Notif/newTeacherNotif/{user_id} • Notif/classParents/{class_id} • self • general services

Teacher

The path for the resources available to the teachers is build in this way:
Teacher/{teacher_id}/ + one of the following methods (left)
Mind that the teacher id is the username.
(In yellow there are the methods that the Teacher can reach from the “general services” page)

Method	Packet	Reachable links
• personal (GET) (returns the personal data of the user)		• personal (PUT) • allClasses • allCourses • self • general services
• personal (PUT) (allows to modify the personal data of the user)	3	• personal (GET) • self • general services
• allCourses (GET) (returns all courses)		• Course/{course_id} • self • general services
• Course/{course_id} (GET) (returns a specific course)		• allCourses • Course/{course_id}/students • self • general services
• Course/{course_id}/students (GET) (returns the students of a specific course)		• Course/{course_id} • Course/{course_id}/students/{student_id}/Mark (GET) • Course/{course_id}/students/{student_id}/Mark (POST) • self • general services
• Course/{course_id}/students/{student_id}/Mark (GET) (returns the grades of a specific student of a specific course)		• Course/{course_id}/students • Course/{course_id}/students/{student_id}/Mark (POST) • self • general services
• Course/{course_id}/students/{student_id}/Mark (POST) (adds a new grade of a specific student of a specific course)	9	• Course/{course_id}/students • Course/{course_id}/students/{student_id}/Mark (GET) • self • general services
• allClasses (GET) (returns all classes)		• Class/{class_id} • self • general services
• Class/{class_id} (GET) (returns a specific class)		• allClasses • TimeTable/{class_id} • Class/{class_id}/students • self • general services
• Class/{class_id}/students (GET) (returns all the students of a specific class)		• Class/{class_id} • self • general services
• TimeTable/{class_id} (GET) (returns the timetable of a specific class)		• self • general services

Method	Packet	Reachable links
• allNotifications (GET) (returns all the notifications for that teacher)		• Notification/{notification_id} • self • general services
• Notification/{notification_id} (GET) (returns a specific notification)		• allNotifications • self • general services

Parent

The path for the resources available to the parents is build in this way:
Parent/{parent_id}/ + one of the following methods (left)
Mind that the parent id is the username.
(In yellow there are the methods that the Parent can reach from the “general services” page)

Method	Packet	Reachable links
• personal (GET) (returns the personal data of the user)		• personal (PUT) • children • self • general services
• personal (PUT) (allows to modify the personal data of the user)	4	• personal (GET) • self • general services
• children (GET) (returns all the children)		• Child/{child_id} • self • general services
• Child/{child_id} (GET) (returns the data of a son/daughter)		• children • Child/{child_id} (PUT) • Child/{child_id}/marks • self • general services
• Child/{child_id} (PUT) (modify the data of a son/daughter)	5	• Child/{child_id} (GET) • self • general services
• Child/{child_id}/marks (GET) (returns the marks of a son/daughter)		• children • self • general services
• Payment/allPending (GET) (returns all the pending payments)		• Payment/allPaid • Payment/{payment_id} • self • general services
• Payment/allPaid (GET) (returns all the paid payments)		• Payment/allPending • self • general services
• Payment/{payment_id} (PUT) (pay one of the pending payments)	10	• Payment/allPaid • Payment/allPending • self • general services
• pubNotif (GET) (returns the public notifications)		• Notification/{notification_id} • privNotif • self • general services
• privNotif (GET) (returns the private notifications, those intended only for that parent)		• Notification/{notification_id} • pubNotif • self • general services
• Notification/{notification_id} (GET) (returns a specific notification)		• self • general services

Appointment

Appointments are resources shared between teachers and parents. The methods are the same for both, but in this case the path doesn't start with the kind of user (such as Teacher/{teacher_id} or Parent/{parent_id}).
Instead it is composed in this way:

Appoint/{user_id}/ + one of the following methods (left)

(In yellow there are the methods that the Parent and the Teacher can reach from their "general services" page)

Method	Packet	Reachable links
• allAppointments (GET) (returns all the appointments of the user)		• Appointment/{appointment_id} (GET) • self • general services
• newAppointment (POST) (add a new appointment)	12	• Appointment/{appointment_id} (GET) • Appointment/{appointment_id} (PUT) • self • general services
• Appointment/{appointment_id} (GET) (returns a specific appointment)		• Child/{child_id} • self • general services
• Appointment/{appointment_id} (PUT) (allows to modify an appointment of the user)	12	• personal (GET) • self • general services
• Appointment/{appointment_id} (DELETE) (allows to delete an appointment)		• children • Child/{child_id} (PUT) • Child/{child_id}/marks • self • general services

Packets
XML

---

1. Log in

```
<logIn>
     <category>Admin</category>
     <password>Fenice</password>
     <username>Silente</username>
</logIn>
```

Category: Admin/Teacher/Parent

---

2. Create an administrator: newAdministrator

```
<administrator>
     <password>MrsPurr</password>
     <username>Gazza</username>
</administrator>
```

---

3. Create a teacher: newTeacher & personal (PUT)

```
<teacherWrapper>
     <name>Rubeus</name>
     <surname>Hagrid</surname>
     <teacherId>RubeusH</teacherId>
     <password>Fierobecco</password>
</teacherWrapper>
```

---

4. Create a parent: newParent & personal (PUT)

```
<parentWrapper>
     <name>Sirius</name>
     <surname>Black</surname>
     <username>SiriusB</username>
     <password>Felpato</password>
</parentWrapper>
```

---

5. Create/Modify a student: newStudent & Child/{child_id} (PUT)

```
<wrapper>
     <lastName>Potter</lastName>
     <name>Harry</name>
</wrapper>
```

---

6. Create a new class: newClass

```
<classWrapper>
     <className>3A</className>
</classWrapper>
```

7. Create a course: newCourse

```
<courseWrapper>
     <classRoom>Aula5</classRoom>
     <courseDescription>Cura creature magiche 1 anno</courseDescription>
     <courseName>Cura creature magiche</courseName>
</courseWrapper>
```

8. Create a new pending payment: newPayment

```
<paymentWrapper>
     <cost>820.0</cost>
     <parentUsername>LuciusM</parentUsername>
     <payed>false</payed>
     <payementDate>2018-12-21T08:10:25</payementDate>
     <paymentDescription>First Rate</paymentDescription>
</paymentWrapper>
```

9. Evaluate a student: Course/{course_id}/students/{student_id}/Mark

```
<EvaluationWrapper>
     <courseId>0</courseId>
     <mark>8</mark>
     <sonId>0</sonId>
</EvaluationWrapper>
```

10. Pay: Payment/{payment_id}

```
<payRequest>
     <payID>1</payID>
     <parentUsername>LuciusM</parentUsername>
</payRequest>
```

11. Send a new notification : various

```
<notificationWrapper>
     <content>Cambio orario colloquio</content>
     <contentType>PRIVATE</contentType>
     <receiver>MollyW</receiver>
</notificationWrapper>
```

ContentType: PRIVATE/PUBLIC (it can be chosen by the admin regardless of the receiver)
In order to keep the same structure in the notifications:

- all parents:
  receiver = parents
- all teachers:
  receiver = teachers
- classParents or classTeachers:
  receiver = class

12. Create/Modify an appointment: Appointment/{appointment_id}

```
<appointmentWrapper>
     <appointmentDate>2018-12-21T16:00:00</appointmentDate>
     <parentUsername>LuciusM</parentUsername>
     <teacherId>SeverusP</teacherId>
</appointmentWrapper>
```

13. Create a timetable: TimeTable/{course_id}

```
<timeTableWrapper>
     <courseId>0</courseId>
     <day>Monday</day>
     <finishTime>2019-06-21T16:00:00</finishTime>
     <startingTime>2018-09-22T14:00:00</startingTime>
</timeTableWrapper>
```

14. Enroll a student to a class: enrollment

```
<enrollRequest>
     <idStud>4</idStud>
     <idClass>0</idClass>
</enrollRequest>
```

JSON

---

1. Log in

```
{
  "category": "Admin",
  "password": "Fenice",
  "username": “Silente"
}
```

Category: Admin/Teacher/Parent

---

2. Create an administrator: newAdministrator

```
{
   "password": "MrsPurr",
   "username": "Gazza"
}
```

---

3. Create a teacher: newTeacher & personal (PUT)

```
{
  "name": "Rubeus",
  "surname": "Hagrid",
  "teacherId": "RubeusH",
  "password": "Fierobecco"
}
```

---

4. Create a parent: newParent & personal (PUT)

```
{
  "name": "Sirius",
  "surname": "Black",
  "username": "SiriusB",
  "password": "Felpato"
}
```

---

5. Create/Modify a student: newStudent & Child/{child_id} (PUT)

```
{
  "lastName": "Potter",
  "name": "Harry"
}
```

---

6. Create a new class: newClass

```
{
  "className": "3A"
}
```

7. Create a course: newCourse


```
{
  "classRoom": "Aula5",
  "courseDescription": "Cura creature magiche 1 anno",
  "courseName": "Cura creature magiche"
}
```


8. Create a new pending payment: newPayment


```
{
  "cost": "820.0",
  "parentUsername": "LuciusM",
  "payed": "false",
  "payementDate": "2018-12-21T08:10:25",
  "paymentDescription": "First Rate"
}
```


9. Evaluate a student: Course/{course_id}/students/{student_id}/Mark


```
{
  "courseId": "0",
  "mark": "8",
  "sonId": "0"
}
```


10. Pay: Payment/{payment_id}


```
{
  "payID": "1",
  "parentUsername": "LuciusM"
}
```


11. Send a new notification : various


```
{
  "content": “Cambio orario colloquio”,
  "contentType": "PRIVATE",
  "receiver": "MollyW"
}
```


ContentType: PRIVATE/PUBLIC (it can be chosen by the admin regardless of the receiver)
In order to keep the same structure in the notifications:

- all parents:
  receiver = parents
- all teachers:
  receiver = teachers
- classParents or classTeachers:
  receiver = class

12. Create/Modify an appointment: Appointment/{appointment_id}


```
{
  "appointmentDate": "2018-12-21T16:00:00",
  "parentUsername": "LuciusM",
  "teacherId": "SeverusP"
}
```


13. Create a timetable: TimeTable/{course_id}


```
{
"courseId": "0",
"day": "Monday",
"finishTime": "2019-06-21T16:00:00",
"startingTime": "2018-09-22T09:30:00"
}
```


14. Enroll a student to a class: enrollment


```
{
  "idStud": "4",
  "idClass": "0"
}
```