

# **An Open Source Solution for Classroom Management EduXes**

## **V Master on Free Software Projects Development and Management 2011-2012**

— Author —

José Antonio Salgueiro Aquino  
<info@joseantonio.org>

—Tutor—

Manuel Rego Casasnovas

December 29, 2012



# Copyright

Copyright ©2012 José Antonio Salgueiro Aquino. Some rights reserved. This work is free and is licensed under the conditions of Creative Commons Attribution - Share Alike 3.0 Unsupported license. You can use, distribute and reuse this work if the same license is applied and the author is quoted Full text of the license can be read on <http://creativecommons.org/licenses/by-sa/3.0/deed.en>





# Acknowledgments

Acknowledgments to my mentor, Manuel Rego. To Ramón Castro Pérez who helped me to run Siestta work. To José Manuel Ciges Regueiro, which memory I used as a template for my own work, and every 5th Master Software Libre classmate.



# Description of the practicum

The main objective of this Master Thesis consist in the development of an mobile application to be used int highschoools by teachers. It could allows teachers to carry on control students attendance, their behavior. Also it permits quick assessment by activity. Teachers would read students reports: weekly and daily assessment, by activity assessments and total marks.

- Name : José Antonio Salgueiro Aquino.
- Birth date: August 5th 1970
- Education: B.Sc. in Fundamental Physics, University of Santiago de Compostela University 1988-1993.
- Address: Marín (Pontevedra). Spain.
- Current position: Secondary School teacher in Technology.

Working times (planned) : 300 hours. From 6th August, to 30 September, on an eight hours day basis.

Technologies involved:

- Java <sup>TM</sup>language.
- Android <sup>TM</sup>. The operating system.
- PhoneGap <sup>TM</sup>(alias Cordova ) framework to develop multi-platform applications.
- JQuery and JQueryMobile to development of mobile oriented applications.
- JavaScript with Web Database.
- Git for version control system.
- L<sup>A</sup>T<sub>E</sub>X for documentation.

Meetings:

- Technologies to be used were stated, work methodologies, first application windows (pages), Android version to be used (2.3.3) because is the most popular.
- Several emails and gtalk conversations about organization, general problems were written.

Teleworking is carried on

Materials and special equipment used:

- Hardware: Intel Quad, 6GiB RAM, 500GiB HD.
- Software: Debian GNU/Linux Wheezy (testing), Eclipse Juno, JQuery 1.8.1, jQueryMobile 1.1.1, and PhoneGap-Cordova 1.8.1, Android Virtual Manager 2.3.3, Git 1.7.10.4-1.
- Real testing: Sony-Ericsson Xperia V mobile phone, with USB cable and wifi.



# Contents

<b>Copyright</b>	<b>i</b>
<b>Acknowledgments</b>	<b>iii</b>
<b>Description of the practicum</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Working plan</b>	<b>3</b>
2.1 Description and objectives . . . . .	3
2.2 Motivation . . . . .	7
2.3 Methodology . . . . .	7
2.4 Tools . . . . .	9
2.5 Work plan . . . . .	10
2.5.1 HTML . . . . .	10
2.5.2 Interface code . . . . .	12
2.5.3 Database code . . . . .	14
2.5.4 Database . . . . .	16
2.5.5 Interface Diagram . . . . .	17
2.5.6 Problems and drawbacks . . . . .	17
<b>3 State-of-art solutions</b>	<b>23</b>
3.1 Siestta . . . . .	23
3.2 Sixa . . . . .	26
3.3 Android applications . . . . .	26
3.3.1 Grade Book . . . . .	26
3.3.2 Attendance . . . . .	28
3.3.3 Teacher Organizer . . . . .	29
3.3.4 Teacher Aide . . . . .	29

---

<b>4</b>	<b>Description</b>	<b>31</b>
4.1	Preparation of development . . . . .	31
4.2	Application Skeleton . . . . .	32
4.3	Upload into Git repository . . . . .	32
4.4	Development . . . . .	34
4.4.1	Groups . . . . .	35
4.4.2	Students . . . . .	40
4.4.3	Main Page . . . . .	43
4.4.4	Attendance . . . . .	45
4.4.5	Activities . . . . .	49
4.4.6	Assessment . . . . .	52
4.5	Database . . . . .	56
<b>5</b>	<b>Results</b>	<b>59</b>
5.1	Objectives completed . . . . .	59
5.2	Further objectives . . . . .	59
5.3	Problems faced . . . . .	60
5.4	Statistics . . . . .	61
<b>6</b>	<b>Personal evaluation of the practicum</b>	<b>63</b>
	<b>Bibliography</b>	<b>67</b>

# List of Figures

2.1	Diagram . . . . .	18
3.1	Siestta Attendance Page . . . . .	23
3.2	Siestta Main Page . . . . .	24
3.3	Siestta SQL . . . . .	25
3.4	Sixa Main Page . . . . .	26
3.5	Sixa Schedule Page . . . . .	27
3.6	Sixa Reports Page . . . . .	27
3.7	Grade Book . . . . .	28
3.8	Attendance . . . . .	29
3.9	Teacher Organizer . . . . .	30
3.10	Teacher Aide . . . . .	30
4.1	List of Groups . . . . .	35
4.2	Students . . . . .	40
4.3	Main Page . . . . .	43
4.4	Schedule . . . . .	45
4.5	Attendance . . . . .	46
4.6	Attendance Report . . . . .	49
4.7	Assessment . . . . .	55
4.8	Assessment Report . . . . .	56
4.9	Database . . . . .	57



# Chapter 1

## Introduction

In a high school, there are classes which attendance, assessment and group dynamics is difficult to control. This is especially relevant for the technology workshop, this could be a noisy, annoying, even dangerous place, which requires teacher's standing monitoring. This workshop needs a lightweight, reliable and also complete software tool. A simple web solution is not enough, because it could be too complicated and not fast enough; even use a tablet can be heavy. Therefore, this problem requires a new approach.

The development of a multiplatform tool, open source, for smartphones, including tablets, for attendance monitoring and evaluation of students is suggested. To achieve this goal a client-side application will be developed using a multiplatform framework: Phonegap (Cordova) [Pho12c]. Phonegap allows you to develop quickly, utilizing well-known languages as Javascript and HTML. It permit us to deploy an application for both Android, WebOS, iOS and others.

For data management, a built-in database, SQLite, will be employed which has all capabilities needed (autoincremental indexes, relation among tables and several built-in functions ).

The tool that enables rapid development of the application and integration of performance tests is Eclipse. Also Eclipse is well integrated with Android SDK, and its Android virtual machine (AVM).



# Chapter 2

## Working plan

### 2.1 Description and objectives

An open source multi-platform management application for high school teachers is developed. It can be run on a smartphone or tablet. The actual objectives of the application are students management:

- Attendance and punctuality control.
- Misbehaviour control.
- Activities assessment. Each student will have an activity mark.

Application would also include these features:

- Data visualization. As table-like format. Attendance and misbehaviour
- Server synchronization with a custom application or Xade [dEXdG12].

The final goal is to develop an application to make teacher's work easier and comfortable. This application is focused on user (teacher) experience. Also an objective is to write extensible, easy to read code, which allows external developers to take part into application development.

Below, a list of tasks to be done to fulfil application development:

- Study state-of-art solutions:

Find out other solutions: PDAs and smart-phone or tablet related and web-based applications.

Download to study and reuse graphical user interfaces, code or/and database structure.

- Develop database structure: tables (field names and type of data), and relationships among tables.
- Preparation for development:

Build development and staging environment:

Install Eclipse [Fou01], Android Virtual Machine [Inc01], Aptana Plugin for Eclipse, JQuery, JQueryMobile [Mob12a] and Phonegap [Pho12c] from their respective websites [Pho12b].

Choose application name and folder's policy.

Make a simple application: only a blank page.

Configure a git repository and upload the application: [Aqu12].

Development will follow these stages:

- Populate database with sample data. Firstly, data will be hard-coded into source code to tagging. Several tables will be created: groups, students, sessions, teacher schedule, students attendance, activities, student activities, and activities per group. Secondly the appropriate windows to manage these tables will be built.
- Groups: Several groups (four) of students will be hard-coded into javascript source code, with three or four students each other.

Make list of groups window. This will list the four groups.

Groups management window. Another group could be added, or removed.

- Students information:

Make list of students window per group and complete list of students.

Students management window: to insert and update data students: name, surname, birthday, address, e-mail, tutor name, landline and cell phone numbers and nationality.

- Sessions. Each lecture has a description (as 'first hour', or 'recreation'), starting and ending time. These sessions will be hard-coded on first version.
- Teacher schedule. For the current teacher, it contains weekly and daily schedule: name of group, session and day of the week. This information will be also hard-coded.



- Main window (or *page* in PhoneGap notation) will be created. Teacher can set current data and go to *daily page*, manage activities, students and groups or manage reports: assessment and attendance. Next page will be *daily page*.
- Timetable for current date: daily schedule, list of groups for each day ordered by session.
- Attendance page will be next to be build. It contains a list of students by group. Teacher can assign an attendance code to each student (attendance, misbehaviour, unpunctuality or excused).
- Assessment page is next to previous one. It contains one upper list of activities and a list of students similar to *attendance page* . Teacher could set activity and assign adequate mark to each student.
- Activities. This page manage activities (add and disable activities <sup>1</sup>): name and description of assessable exercises.

Activities group will be set in activities page, and activities student will be related with assessment.

- Reports pages.

List activities.

List groups.

List students by group.

List students attendance by week. User can select previous or next week, or set another date.

List students marks and final mark.

- Error handling. While developing each SQL query, any error will be caught and an error window will appears.
- Eventually, test into real hardware: an Android 2.3.3 mobile phone, tablets, and so on.

Next steps will include :

- Load data from an external file. This is well-documented [Pho12d].
- Load images from disk (SD-Card).

---

<sup>1</sup>still not coded

- Save or download data from database to disk.
- Xade web interface.
  - a) Study Xade web interface.
  - b) Develop an ad-hoc application for retrieve Xade's data. With javascript or a native one.
- 8. Develop an ad-hoc application for store data.
- 9. Synchronization with a custom server or with Xade.
- Test units. To ensure previous step were implemented.
- User documentation. Manual with images.
- Developer's documentation.
- Find out a website to host a forum, a bug report system, documentation and application download.

Task above could be achieved in a 300 hour basis. In the following table a broad estimation of time spent in each task are shown.

Task	Time (hours)
State-of-art solutions	10
Develop database	8
Preparation for development	40
Development.	
Populate database with sample data.	20
Groups. List and management	50
Students. List and management.	30
Timetable for current date.	80
Add attendance, behaviour.	50
Add error handling.	2
Retrieve and insert data from and to database.	30
List of attendance, misbehaviour incidents.	12
Add activities grades for each student.	12
List students marks and final mark.	14
Activities management window.	14
Management of student notes.	20
List of student notes.	2
Test into real hardware	20
Total	384

## 2.2 Motivation

As a Technologies teacher, in my daily work I have to evaluate students work such as working with tools, cooperative work with other classmates etc., besides usual activities as written exercises. A long data sheet could be used, or an awkward long spreadsheet, but a portable device with a custom application should be desirable.

This application tries to increase teacher's productivity because teacher only has to write attendance, or unpunctuality two times (on official report and on application's window), and classroom notes and activity grades on very easy way.

The most important feature is to be as easy, fast and intuitive as possible. It could be desirable to be platform independent (Android, iOS, Windows RT), but Android is preferred because it is open source and has a higher market share.

On the other hand, development of this application improves my computer science skills in mobile-phone applications development: JQuery, jQueryMobile, PhoneGap/Cordova, SQLite, Android, git repository management.

## 2.3 Methodology

This work was carried on building small blocks, also called pages, and make it up into final application. A page is a visible window, only first page is visible, and other pages are called from this one.

Database structure was separated from interface, and interface was also separated into dynamic and static contents. Each new functionality was written, tested, and polished.

Foremost a new window/page is designed from previous HTML code or from sample code [Cas12b] and written in **index.html**, *id* are set with ad-hoc names (e.g. *id="edit\_student"*, *id="in\_name\_student"*), these *id's* are used in javascript code.

### Sample Page. index.html

```
<div data-role="page" id="daily\_work">
  <div data-role="header" data-add-back-btn="true" data-theme="a">
    <h1>Day</h1>
  </div>
  <div data-role="content" data-theme="a">
    <div class="ui-grid-b">
      <div class="ui-block-a">
        <div data-role="fieldcontain">
          <label for="date" >Date:</label>
        </div>
      </div>
      (...)
    </div>
  </div>
  <div data-role="navbar">
    <ul>
      <li>
        <a href="#" onClick="onGeneralFile()"
          data-role="button" data-icon="star"
          data-theme="a" > File</a>
      </li>
      <li>
        (...)
      </li>
    </ul>
  </div><!-- /navbar -->
  <div data-role="footer" class="footer-docs" data-theme="a">
    <p style="text-align:center;" id="teachers\_name"></p>
  </div>
</div>
```

Secondly **interface.js** code are written: show progress icon (a *PhoneGap* function), a customized function which loads data into new HTML code (e.g. *loadGroupAssessment*), and, finally a *PhoneGap* function to show new page/window.

### Sample interface.js code

```
$.mobile.showPageLoadingMsg();
id_group=global_id_group;
loadGroupAssessment(global_db,id_group);
$.mobile.changePage("#list_students_assessment");
```

Thirdly **database.js** code are written: Declaration of previous function, which usually implements a SQL-query to SQLite database. These functions also fill html code, so HTML *id*'s names are so significant.



### Sample database.js code

```
var list_asset=$( '#list_assessment_select' );
sql="SELECT ";
sql += " ACTIVITIES_GROUP.a_date (...) ";

db.transaction(function(tx) {
    tx.executeSql(sql, [],
        dbSuccessFunc = function(tx, results) {
            var html = "";
            for (var i=0;i<results.rows.length;i++) {
                a_id = results.rows.item(i).a_id;
                a_name = results.rows.item(i).a_name;
                (...)
                html += ' <option value="'+a_id+'">'+a_name+'</option> ';
            }
            list_asset.empty().append(html);
            list_asset[0].selectedIndex = 0;

            if(results.rows.length>0) {
                $('#current_group_assessment').text(
                    results.rows.item(0).g_data);
            }
            global_id_group = id_group;
            listStudentsByGroupAssessment( db, id_group,
                $('#students_assessment_ul'));
        },
        dbErrorFunc (...));
});
```

After successful tests using Android Emulator provided by Android SDK [Inc01], **TODO** file could be updated, and code will be upload to remote git [Aqu12] repository.



### Git init shell

```
$ git add -A *
$ git commit -m BRIEF_DESCRIPTION_OF_CHANGES
$ git push origin master
```

From time to time application is downloaded from git repository into real hardware and tested. It is usually faster on a mobile-phone than on emulator.

## 2.4 Tools

Tools involved in *EduXes* development are :

- IDE. Eclipse [Fou01]. The most popular Integrated Development Environment for Java and other languages. It integrates browser, contextual help, even Android Virtual Machine is integrated.
- Aptana [App05]. Aptana Plugin is used HTML, CSS and JavaScript editing.
- Android Development Kit [Inc01]. Android SDK provides the API libraries and developer tools necessary to build, test, and debug apps for Android.
- Git. Also included as an Eclipse plugin.

To build this documentation several applications were used:

- Sqlfairy. Tranforms SQL language into a png image.
- Gimp 2.8.2 to get screen-shots.
- LibreOffice 3.5.4.2 to write previous document
- LateXila 2.7.0 pre to write this document.
- Other tools from GNU/Debian Wheezy October 2012

## 2.5 Work plan

Following previous methodology 2.3, a detailed plan is shown below:

- HTML. User interface.
- interface.js. Functional code and database agnostic.
- database.js. Database related code.
- DATABASE.sql. Database declarations as a standalone file.

### 2.5.1 HTML

There are only two html files: **index.html** and **remove.html**. The most important: **index.html** file is build by blocks called pages, gathered together [Mob12b]. Each page is its own **div** with custom properties (*propieraty* properties in Eclipse jargon) : **data-role="page"**. Below as show an example: list of several reports, user could choose one and a function (*onReportListAttendance()* or *onReportListAssessment()*) is executed .

Inside each page there are several identificatives *id="name"* which are used by application to fill with data (e.g. within *daily\_schedule* page *current\_day* id is used to set date to current date).



## Reports Page

```
<div data-role="page" id="list_reports" data-add-back-btn="false">
  <div data-role="header" data-back-btn-text="previous"
    data-add-back-btn="false" >
    <a href="#daily_work" data-icon="arrow-l" data-theme="a"
      data-role="button">Back</a>
    <h1>Reports</h1>
  </div>
  <div data-role="content">

    <ul data-role="listview" id="list_reports_ul"
      data-split-icon="gear" data-split-theme="a"
      data-filter="false"
      data-inset="true" data-theme="a" >
      <li><a href="#" onClick="onReportListAttendance();"
        >Attendance</a></li>
      <li><a href="#" onClick="onReportListAssessment();"
        >Assessment</a></li>
      <li><a href="#" onClick="" >File</a></li>
    </ul>

  </div>
  <div data-role="footer" class="footer-docs"
    data-rel="back"
    data-theme="a">
  </div>
</div>
```

A complete list of pages is shown below:



## List of Pages

intro	Introductiony page
daily_work	daily work
list_reports	List reports
list_settings	List settings
file	Exit application
daily_schedule	List of groups per day
list_students_attendance	
list_students_assessment	
list_students_attendance_by_group	
general_list_attendance	
list_groups_attendance	
list_groups_assessment	
list_students_assessment_reports	
list_groupsto_edit	
edit_new_group	
edit_groups	
list_groups	
list_students_by_group	
list_students	
edit_student	
edit_new_student	
list_all_activities	
update_activity	

### 2.5.2 Interface code

There is only one file which deals with interface interactions (events from **html** files): *interface.js*. If the function is called directly from *html* code, function name contains **on** prefix. There are several groups of functions:

- *Students* and *Groups* functions, which manage and list general students and groups information.
- *Activities* functions, which update, list and add new activities.
- *Assessment* functions, list and update students marks.
- *Attendance* functions, list when student attend classes and update their values.

Whether these functions need access to data (every function but *onGeneral\** ) they call their counterpart function in **database.js** file.





### List of Functions: interface.js

Name	Task	Group
<i>onDeviceReady()</i>	Firstly loaded, create and populate database, load initial page	Initial
<i>init()</i>	Initial	
<i>initialize_data ()</i>	Load default date	Initial
<i>open_daily_page()</i>	Daily work page: list of groups	Schedule
<i>listStudentsAttendance()</i>	Students Attendance page	
<i>onAddNewGroup()</i>	Edit Group	Groups
<i>onAddNewGroup()</i>	Update Group	Groups
<i>onDeleteGroup()</i>	Delete Group	Groups
<i>onSaveNewGroup()</i>	Save Group	Groups
<i>onListAllGroups()</i>	List All Groups	Groups
<i>EditStudent()</i>	Edit Student	Students
<i>onDeleteStudent()</i>	Delete Student	Students
<i>onAddNewStudent()</i>	Add new Student	Students
<i>onSaveStudent()</i>	Save Student	Students
<i>onSaveNewStudent()</i>	Save New Student	Students
<i>listStudents()</i>	Full list of students	Students
<i>listStudentsByGroup( )</i>	List Students by group	Students
<i>onListAllStudents()</i>	Full list of students	Students
<i>onListAllActivities()</i>	List All Activities	Activities
<i>onUpdateActivity()</i>	Activity update	Activities
<i>onAddNewActivity()</i>	Add new Activity	Activities
<i>onSaveNewActivity()</i>	Save New Activity	Activities
<i>onOpenStudentsAssessment()</i>	Open Students Assessment	Assessment
<i>onRefreshGroupAssessment()</i>	Reload Group Assessment	Assessment
<i>onOpenStudentsAttendance()</i>	Open Student Attendance	Attendance
<i>onReportListAttendance()</i>	List Attendance	Attendance Reports
<i>listStudentsByGroupAttendance()</i>	List Students by Group Attendance	Attendance Reports
<i>studentsAttendanceListPrevious()</i>	List Students by Group Previous week	Attendance Reports
<i>studentsAttendanceListNext()</i>	List Students by Group Next week	Attendance Reports
<i>onReportListAssessment()</i>	List Groups for Assessment	Assessment Reports
<i>onListStudentsAssessment()</i>	List Students for Assessment	Assessment Reports
<i>studentState()</i>	Student State	Student
<i>Attendance()</i>	Student Attendance	Student Attendance
<i>onGeneralFile()</i>	Exit	Exit
<i>onGeneralListReports()</i>	List Reports	Reports
<i>onGeneralListSettings()</i>	List Settings	Settings

### 2.5.3 Database code

The file which contains functions which carry on queries and data manipulation (on data base (SQLite)) is : *database.js*

There are global variables on top of this file. These variables are used by functions because there are not obvious ways to pass values from **html** code through functions ( e.g. as day week, database, current date, ... ):

Variable	Meaning
global_id	General and unique identification of any table as students or groups <sup>a</sup>
table_global	students, groups table
global_id_group	Identification of a group
global_id_student	Identification of a student
global_id_activity	Identification of an activity
global_max_activities	Number of activities
global_no_groups	Number of groups
global_week_day	Number of week day (0-6)
global_db	Database pointer
global_session	Selected session
global_actual_date	Current date
global_reports_date	Date for reports
global_is_new	Whether activity is new
global_exist	if exist current record
STATE_NONE	Default Student state
STATE_ABSENCE	Truancy
STATE_UNPUNCTUAL	Unpunctuality
STATE_EXCUSED	Excused unattendance
STATE_BEHAVIOR	Bad behaviour

<sup>a</sup>will be deprecated

Several approaches have been considered, a lot of tiny, one function equals one simple task, but code were growing in complexity, became very difficult to read, maintain and pass variables to them. Other perspective was to write only several complete functions which carry all (or almost all) the work. From this point of view functions become longer, (up to a hundred of lines of code) but easier to test and follow.

Next tables list every function used in **database.js**, gathered together by task, or menu options.



## List of Functions: database.js

Name	Task
<i>errorCB()</i>	Error handler
<i>successCB()</i>	Success handler
┌ <i>loadSchedule()</i>	Main Window → Load Daily Schedule
<i>querySchedulePerDayDB()</i>	
└ <i>queryScheduleSuccess()</i>	
┌ <i>loadAllStudents()</i>	Settings → List all students
<i>queryAllStudentsDB()</i>	
└ <i>queryAllStudentsSuccess()</i>	
┌ <i>loadStudent()</i>	Settings → List all students → Fill Students page
<i>queryStudentSuccess()</i>	
<i>queryStudentDB()</i>	
┌ <i>loadNewStudent()</i>	Settings → List all students → New Student
<i>queryNewStudentDB()</i>	
┌ <i>loadStudents()</i>	Delete Student → List all students
<i>queryStudentsDB()</i>	
└ <i>queryStudentsSuccess()</i>	
<i>loadStudentsByGroup()</i>	Settings → Groups → List Students
┌ <i>loadStudentAttendance()</i>	Main Window → Load Daily Schedule → Groups (Default: Attendance)
<i>queryStudentsAttendanceDB()</i>	
<i>queryStudentsAttendanceSuccess()</i>	
└ <i>fillSelectStudent()</i>	
┌ <i>loadAllActivities()</i>	Settings → Activities
└ <i>queryLoadAllActivitiesDB()</i>	
<i>loadActivity()</i>	Settings → Activities List → Load activity information
<i>updateActivity()</i>	Settings → Activities List → Load activity → Update Activity.
<i>updateActivitiesGroup()</i>	idem. for each group
<i>insertNewActivity()</i>	Settings -> Activities -> New
<i>insertActivitiesGroup()</i>	idem. for each group
<i>loadGroupsActivitiesEdit( )</i>	Settings → Activities → New (List groups)
┌ <i>listMaxActivities( )</i>	List Number of activities
└ <i>queryListMaxActivities()</i>	
<i>Peers ( )</i>	Activities auxiliar object



### List of Functions (2): database.js

Name	Task
<i>insertNewGroup()</i>	Settings → Groups → Insert Group
<i>updateGroup()</i>	Settings → Groups → Update Group
<i>deleteGroup()</i>	Settings → Groups → Delete Group
<i>loadGroup()</i>	Settings → Groups → Select
┌ <i>loadAllGroups()</i>	Settings → Groups
└ <i>queryAllGroupsDB()</i>	
<i>insertNewStudent()</i>	Settings → Insert new Student
┌ <i>querySaveStudentDB()</i>	Settings → Update Student
└ <i>saveStudent()</i>	
<i>deleteStudent()</i>	Settings → Delete Student
<i>insertStudentStateL( )</i>	Settings → Insert New Student
<i>updateStudentStateL( )</i>	Settings → Update Student
┌ <i>loadGroupAssessment( )</i>	Assessment <sup>a</sup> → Group → List options
<i>listStudentsByGroupAssessment()</i>	
<i>refreshGroupAssessment()</i>	
┌ <i>fillSelectStudentAssessment( )</i>	
<i>onChangeStudentAssessment()</i>	Assessment → Change Student Mark
<i>updateStudentAssessmentL()</i>	Id. Update Student Assessment
<i>insertStudentAssessmentL()</i>	Id. Insert Student Assessment
<i>loadGroupsAssessment()</i>	Report → Assessment → Group
<i>loadStudentsAssessment()</i>	Report → Assessment → Students <sup>b</sup>
┌ <i>loadGroupsAttendance()</i>	Reports → Attendance → List groups
└ <i>queryGroupsAttendanceDB()</i>	
┌ <i>reportAttendanceDB()</i>	Reports → Attendance → Group
<i>queryReportAttendanceDB()</i>	
└ <i>queryReportAttendanceSuccess()</i>	
<i>deleteRawRecord( )</i>	Delete any row from any table
<i>stateCheck( )</i>	Check whether student state changes
<i>updateStudentState( )</i>	Update student state

<sup>a</sup>Main Window → Attendance → go to Assessment

<sup>b</sup>Assessment List

#### 2.5.4 Database

Database is the application most important data structure, it requires a special study. Actual database is inherit of Siestta application [P06]. This database was developed from *groups* table, to *activities\_student* following less related path ( *groups* → *students* → *session* → *teacher\_schedule* → *attendance...* ).

### Snippet database

```
(...)
-- Students Attendance
  DROP TABLE IF EXISTS ATTENDANCE;
  CREATE TABLE IF NOT EXISTS ATTENDANCE (id integer primary key ,
    id_group integer, id_student integer, id_session integer,
    a_type integer, a_date text,
    FOREIGN KEY (id_student) REFERENCES STUDENTS (id),
    FOREIGN KEY (id_group) REFERENCES GROUPS(id),
    FOREIGN KEY (id_session) REFERENCES SESSIONS(id) );

-- Activities
  DROP TABLE IF EXISTS ACTIVITIES ;
  CREATE TABLE IF NOT EXISTS ACTIVITIES
    (id integer primary key, name text, date_init text,
    date_end text, weight integer, final integer );
  DROP TABLE IF EXISTS activities_student;
  (...)
```

## 2.5.5 Interface Diagram

This figure 2.1 illustrates application workflow. Dotted lines are instructions and continuous lines data flow. Readings from database are not displayed.

## 2.5.6 Problems and drawbacks

From initial plan, there was several changes, application is evolving along time.

One relevant change was *Edit Update* ↔ *Edit New* unification, only in students, not in groups and activities yet.

The most difficult page: *List of assessment* by student: *loadStudentsAssessment* function. Pointedly a comment in code is set with a **XXX** tag in `database.js` file. The main problem was to deal with several students with different activities each other, therefore I realized a need a matrix, but there are not matrices in javascript. I asked for a hint in IRC (freenode.net) in `#javascript` channel, they pointed out an object: *Peers*. Eventually *Peers* object is an object which contains activity id, mark and its weight.

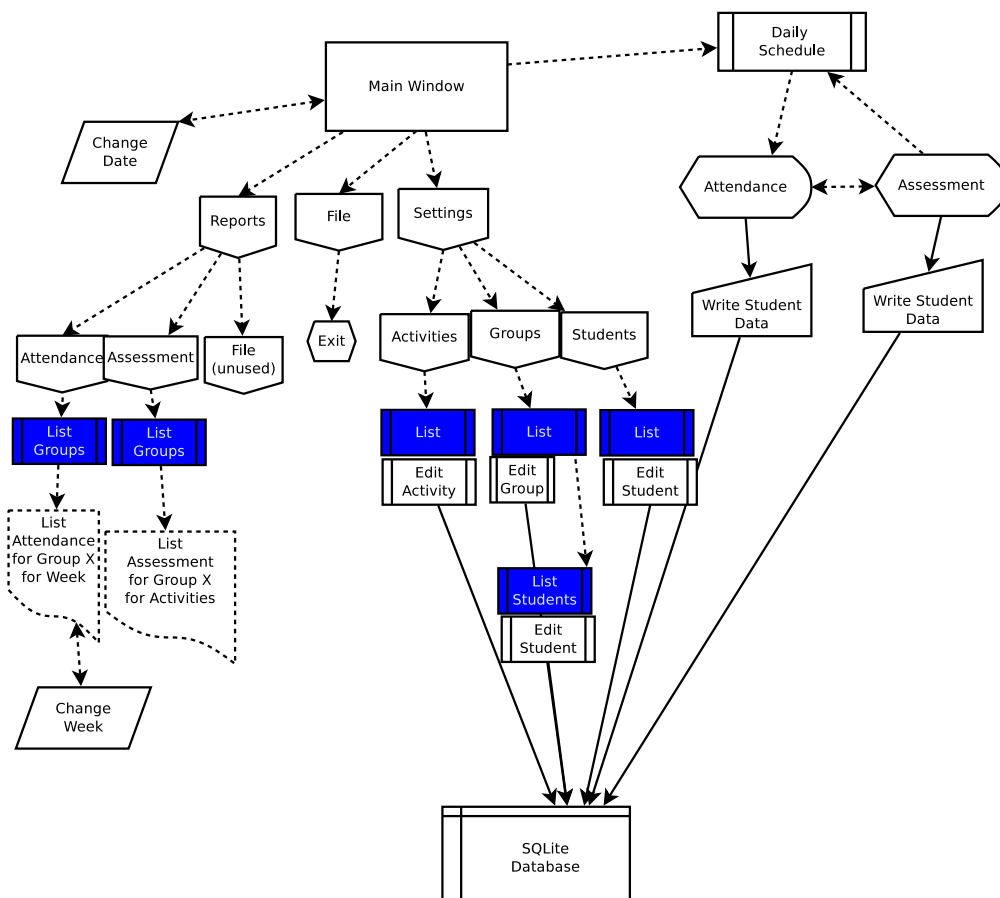


Figure 2.1: Diagram

```
// XXX: Warning with 1 begining indexes Adjust
// Assessment - List Students
// http://www.w3schools.com/js/js_objects.asp
//      activity-1  activity-3  activity-5
// A              8          7
// B              7          5
// C              3
// Max actividades => 3
// Max students => 3 !
// act_id_1. act_id_3, act_id_5
```

*Report attendance* was difficult to implement. Main difficulty was week change, this implied a new global variable was set *report\_date* which allow week change. Initially an ad-hoc function was developed to face date format problems, but an external javascript library *moment.js* was found, and it is used to calculate week days.

Another problem was a design decision: *Assessment* page. it is tightly related with *Attendance*. After a long standby, it became evident that *Assessment* had to contain two main elements: list of activities (by default it was set to 0), and list of students with a combo box each one. On bottom there is a button to change between *Assessment* and *Atendance*.

On the first page, initially current date was set by hand (or with a native widget which depends on Android version), this was a problem because user has to dealt with dates in *month/day/year* format. My mentor, Manuel Rego, told me about a widget *LibrePlan* app [Cas12a] was using: *Mobiscroll*[Med12].

Other main change was a **philosophical** concept. Firstly main idea was to develop shortest functions, each one doing a simple task, but I have problems passing parameters to that functions without using global variables. Eventually, I gathered code and function is longer but clearer (e.g. *loadGroupsAssessment*).

### Snippet loadGroupsAssessment

```
function loadGroupsAssessment(db){
var sql="SELECT id, data, other_data FROM groups;";
db.transaction(function(tx) {
    tx.executeSql(sql, [],
        dbSuccessFunc = function(tx,results){
            var len = results.rows.length;
            var ul_list=$('#groups_assessment_ul');
            ul_list.empty();
            for (var i = 0; i < len; i++) {
                var id = results.rows.item(i).id;
                html = "<li>";
                html += " <a onClick='onListStudentsAssessment(" + id + ");' ";
                html += " href='#' data-transition='slideup'>";
                html += results.rows.item(i).data + "</a>";
                html += "<a onClick='global_id=" + results.rows.item(i).id ;
                html += "; table_global=\"groups\";' href='#' data-rel='dialog' ";
                html += " data-transition='slideup'>";
                html += "</a></li>";
                ul_list.append(html);
            }
            ul_list.listview('refresh');
            return true;
        },
        dbErrorFunc = function(tx, e) {
            if (tx.message) e = tx;
            log("fillSelectStudent. SQL "+sql);
            alert("fillSelectStudent. There has been an error SELECT stateCheck: "
                + e.message);
            return false;
        }
    );
}
);
}
```

Another change was to how manage success and errors events in a SQL call. First solution was to control it in *transaction* call 2.5.6. But it was difficult to manage when success event happened.

### Transaction

```
db.transaction(queryCallbackDB,
    errorCallback,
    successCallback);
```

Next idea was to manage success and error events in *cursor*. This is fine grained and easier to control application flux 2.5.6.



**Cursor**

```
tx.executeSql(sql, [],  
    dbSuccessFunc = function(tx, results){ (... ) },  
    dbErrorFunc = function(tx, e) { (... ) } );  
}
```

Finally, another change was to not implement show and load student image in student *Edit* page. Main reason for this is that there are a previous decision about how to import external data, via a webpage, a SD card loaded file, webcam or other ideas. Of course it is in next milestone.



# Chapter 3

## State-of-art solutions

Only one open source application was found suitable for study, Siestta , nevertheless there are a lot of educational software (Sixa [S.L09], Unisoft [Uni05] ) but they are privative, Microsoft Windows freeware or both (SAS académico).

### 3.1 Siestta

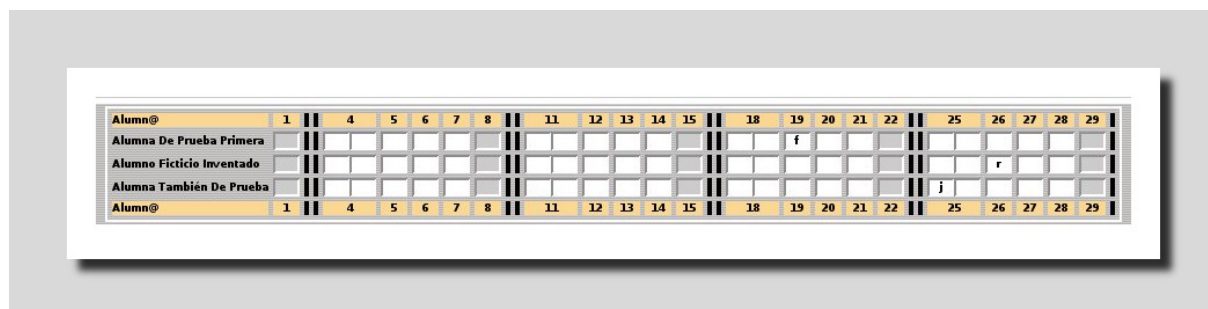
Technically Siestta[P06] is an GPL'ed 1990's style PHP-based web application with Ajax, an interactive editor, fckeditor and fpdf to generate reports.

From user point-of-view there are online documentation. This application includes management of students, attendance, marks, tasks, incidents, general queries, letters to parents, interviews with parents, messages, appointments, exams and more.

Several screenshots were taken and their structure will be reused in current application, specially attendance page 3.1, and daily schedule 3.2.

This application are also available for PDAs, it could be a valid solution but it is server-side with outdated technologies. Data structure from Siestta is standard and fully functional, and it will be partially reused by EduXes.3.3.

Source code are also shown: calendario.php. It shows us a PHP application which uses sessions variables and is not Model-View-Controller oriented.



**Figure 3.1:** Siestta Attendance Page



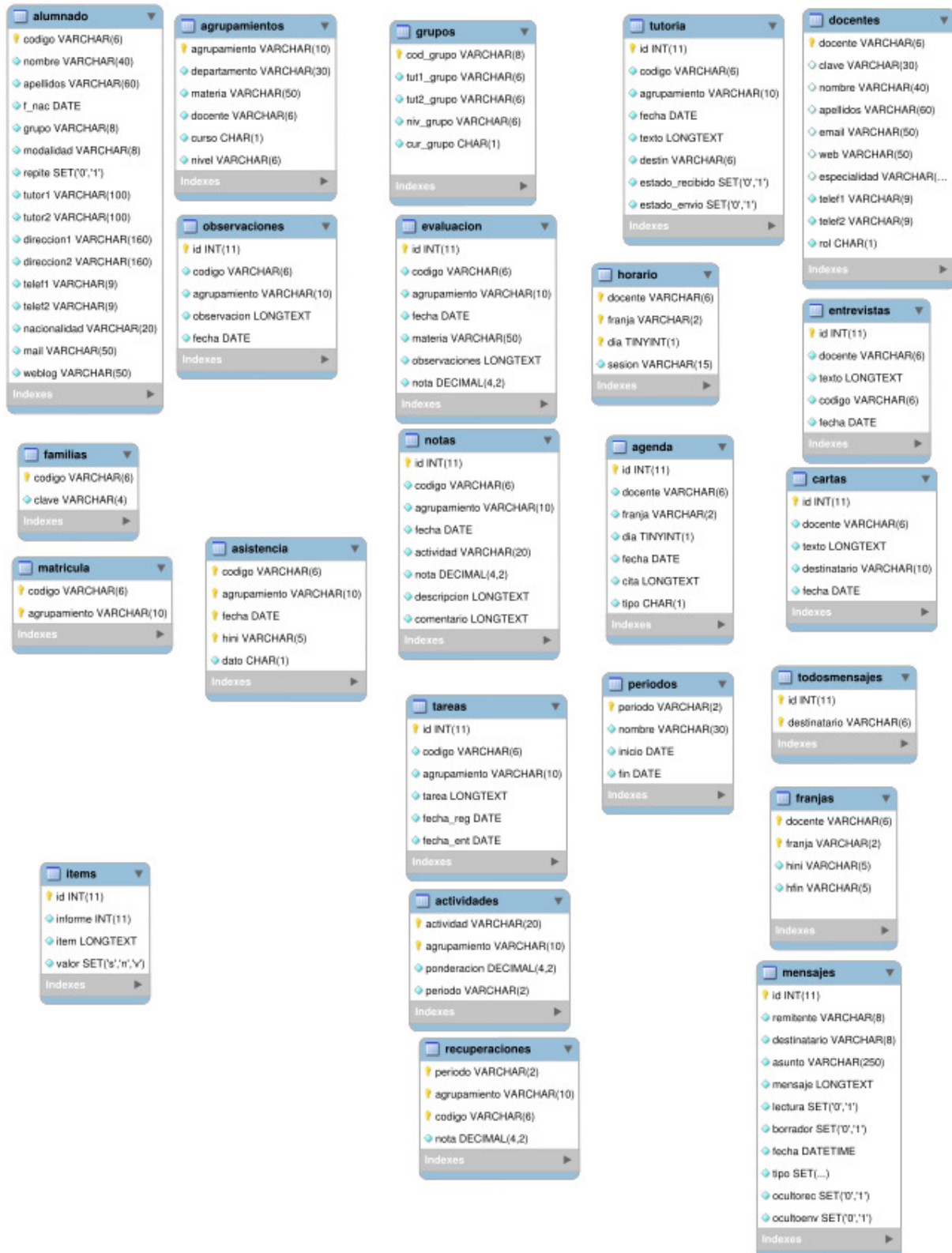


Figure 3.3: Siestta SQL



Figure 3.4: Sixa Main Page

## 3.2 Sixa

Although not GPL'ed, and there are no source code available, several design ideas could be considered: for main page 3.4, schedule page 3.5 and reports page 3.6.

It has a PDA version (not shown), and it Xade data import capable. Also has a long list of features [S.L09].

## 3.3 Android applications

Several Android applications are listed below, in a short list:

### 3.3.1 Grade Book

- Name: Grade Book [Aca11]

**SIXA - Aplicación do Profesorado**

Aplicación: **profesorado** [Axuda](#)

**Horario do docente**

Listaxes **Caderno clase** Sección conduta

« Semana do 24/12/2012 até o 28/12/2012 »

	Luns	Martes	Mércores	Xoves	Venres
08:40	Garda de clase	-	-	-	-
11:30	-	-	Garda de clase	Titoría pais/alum	-
12:40	Garda de clase	Garda de clase	-	-	-

**Estudiantes**

[Personalizar](#) [Mensaxes](#) [Imprimir caderno](#)

Versión: 3.22.3

© SIXA É UNHA PLATAFORMA DESENVOLTA POR E~XENIO  
Deseño baseado nun modelo de Just Web Templates (<http://www.justwebtemplates.com>)

Figure 3.5: Sixa Schedule Page

**SIXA - Aplicación do Profesorado**

Aplicación: **profesorado** [Axuda](#)

**Listaxes do alumnado**

Listaxes **Caderno clase** Sección conduta

[Volver](#)

☐ Listaxe por grupos --

☐ Listaxe por materia --

☐ Listaxe por pendentes -- --

[Ver listaxe](#)

Versión: 3.22.3

© SIXA É UNHA PLATAFORMA DESENVOLTA POR E~XENIO  
Deseño baseado nun modelo de Just Web Templates (<http://www.justwebtemplates.com>)

Figure 3.6: Sixa Reports Page



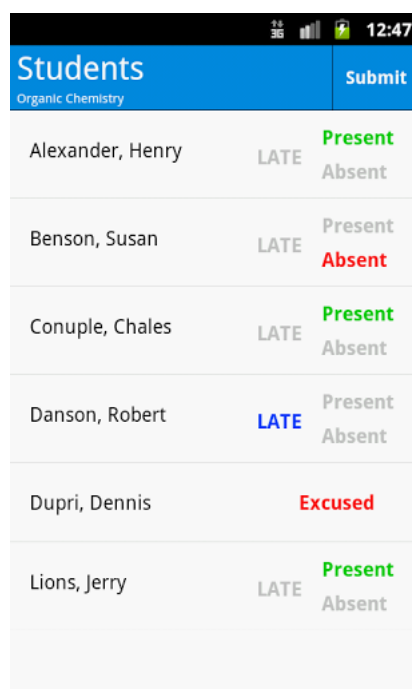
**Figure 3.7:** Grade Book

- Description: Now teachers can manage their students grades directly on their Android device!
- Key Features:
  - Sync with Google Spreadsheets
- Updated: July 7, 2011.
- Price : 4 €
- Screenshot: 3.7

### 3.3.2 Attendance

- Name: Attendance [fA10]
- Description: Attendance control sync with Google Spreadsheet.
- Key Features: attendance.
- Updated: January 13, 2012.
- Price : Free
- Screenshot: ??





Students		Submit
Organic Chemistry		
Alexander, Henry	LATE	Present Absent
Benson, Susan	LATE	Present Absent
Conuple, Chales	LATE	Present Absent
Danson, Robert	LATE	Present Absent
Dupri, Dennis		Excused
Lions, Jerry	LATE	Present Absent

Figure 3.8: Attendance


### 3.3.3 Teacher Organizer

- Name: Teacher Organizer [Lle12]
- Description: Gradebook and attendance, notes, schedule a teacher (high school teacher).
- Key Features: Unified information resource teacher developed within diploma projects.
- Updated: December 26, 2012.
- Price : Free.
- Screenshot: 3.9

It has a drawback: its web page is in Russian, and it is not translated. This application seems to be very professional. There is no code available.

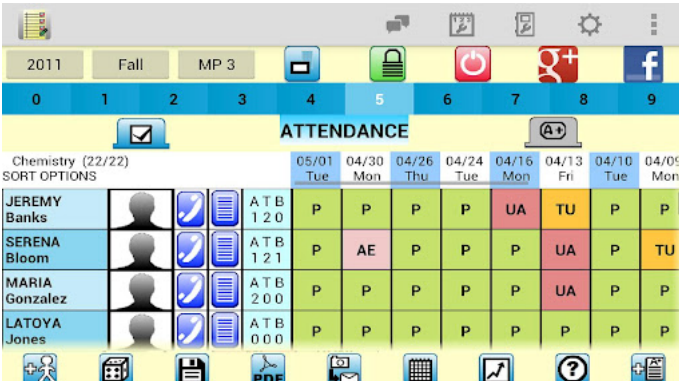
### 3.3.4 Teacher Aide

- Name: Teacher Aide [iagdc12].
- Description: The app allows teachers to take attendance and record grades on their phone or tablet.
- Key Features: Import - export data via CSV. Print via a Google service.



дисциплина 2	4 2 2012	28.5 2012	5.6 2012	12.7 2012	19.8 2012	26.9 2012	3.10 2012	10.11 2012	17.12 2012	
Вот это фамилия 01 Имя 01	5	5-	4+	4	4-	3+	3	3-		
Вот это фамилия 02 Имя 02	5-	4+	4	4-	3+	3	3-	2		
Вот это фамилия 03 Имя 03	4+	4	4-	3+	3	3-	2	н		
Вот это фамилия 04 Имя 04	4	4-	3+	3	3-	2	н	н		
Вот это фамилия 05 Имя 05										

Figure 3.9: Teacher Organizer



		0	1	2	3	4	5	6	7	8	9
<b>ATTENDANCE</b>											
Chemistry (22/22)											
SORT OPTIONS											
		05/01 Tue	04/30 Mon	04/26 Thu	04/24 Tue	04/16 Mon	04/13 Fri	04/10 Tue	04/09 Mon		
JEREMY Banks	ATB 120	P	P	P	P	UA	TU	P	P		
SERENA Bloom	ATB 121	P	AE	P	P	P	UA	P	TU		
MARIA Gonzalez	ATB 200	P	P	P	P	P	UA	P	P		
LATOYA Jones	ATB 000	P	P	P	P	P	P	P	P		

Figure 3.10: Teacher Aide

- Updated: 26 December 2012.
- Price : Free (Limited number of students).
- Screenshot: 3.10

Excellent but not open source or even free.

# Chapter 4

## Description

A detailed list of procedures to build and prepare environment to develop *EduXes* application, will be described. Also a description of how development was carried out, step by step are show.

### 4.1 Preparation of development

Firstly Java Development Kit (JDK) version 1.6 is needed, to build application itself and the Eclipse IDE, is downloaded from *Oracle*: [Ora95]

```
http://www.oracle.com/technetwork/java/javase/downloads/index.html
```

As root user downloaded file is unpacked into `/usr/lib/jvm` and configured to be the Java default:



#### Update Java

```
| # update-java-alternatives -s JDK_1.6_NAME
```

*Eclipse Juno (4.6)* is downloaded from its site [Fou01]: *Download* → *Linux 64bits*

```
http://www.eclipse.org
```

Android Development Toolkit (ADT)[Inc01] is downloaded following instructions on this page:

```
http://developer.android.com/sdk/installing/installing-adt.html
```

In Eclipse a new line is included into repository software (*Help* → *Install New Software* → *Add*):

```
http://dl-ssl.google.com/android/eclipse/
```

Next step is to select all the related software listed.

For Aptana Plugin the line to be added into Eclipse is:

`http://download.apptana.com/studio3/plugin/install`

Furthermore JQuery, JQueryMobile and Phonegap are needed, and were downloaded from their web sites:

- JQuery 1.8.1 (no newer versions), from `http://jquery.com/` :

JQuery will be copied into **assets/www/js** folder.

- JQueryMobile 1.1.1 from `http://jquerymobile.com/`:

JQueryMobile is a zip file which will be uncompressed and copied into **assets/www/js** folder.

- PhoneGap - Cordova 1.8.1 is downloaded from `https://github.com/phonegap/phonegap/zipball/1.8.1` and installed following reference [Pho12b]

To create a PhoneGap application there are very important instructions (Getting Started with Android) should be followed step by step: [Pho12a]

## 4.2 Application Skeleton

Next step is to choose application name and folders policy. Name *EduXes* stands for "Educaci3n" and "Xesti3n", is a educational management software. About folders policy, a folder is created (**assets/www/js**) which contains javascript (**\*.js**) files except *JQuery* and *JQueryMobile* which is included into another folder ( **assets/www/js/jquery** ), do not forget style sheets files (**\*.css** )

To finish building application skeleton a simple application is created which contains only a blank page.

Getting started with Android [Pho12a] is followed step by step.

## 4.3 Upload into Git repository

Following stage is to upload that simple application into a git repository. An account is created in Github[Git08], which is the largest code host in the world, and a new application is initialized. This are the source code project page `https://github.com/joseantoniosa/EduXes`.

Then source code are upload to Github:



### Git init shell

```
$ git init
$ git add -A *
$ git remote add EduXes git@github.com:joseantoniosa/EduXes.git
$ git push origin master
```

Each time an update in code is done, code is uploaded :



### Git update

```
$ git add -A *
$ git commit -m 'CHANGES_DESCRIPTION'
$ git push origin master
```

## 4.4 Development

Before beginning with development itself is compulsory to explain some concepts already shown above in Methodology (2.3), but with more detail: JQueryMobile [Mob12a] and by extension, PhoneGap [Pho12c] applications, are structured in pages:

```
<div data-role="page">
```

which are very similar to desktop applications windows. Therefore, from Javascript code, to change to a new page

```
$.mobile.changePage("#daily_work");
```

opens "*daily\_work*" page.

As shown before (2.3) *EduXes* contains several files:

- Four files contain application code:

**interface.js** : It contains information and decisions related to interface and application workflow, completely independent from database. It is located in **assets/www/js** folder.

**database.js** : It contains database related code: SELECT, INSERT, etc. It is located in **assets/www/js** folder.

**index.html**, **remove.html** only contain HTML framework, page properties, and static content. They are located in **assets/www/** folder.

- There are three important files which contain documentation:

**TODO.txt**. List of goals to be achieved and milestone reached. It is located in root application folder.

**DATABASE.sql**. Data-base structure in SQL format. This file are shown below. It is located in root application folder.

**README.txt**. Only contains general information about this application. It is located in root application folder.

Also, *Eclipse* generates several files, the most important is the application file, which is an **apk** package: **bin/EduXes.apk**.

Next step is to build database, this will be described below 4.5. Following database, sample data is needed to begin application development.

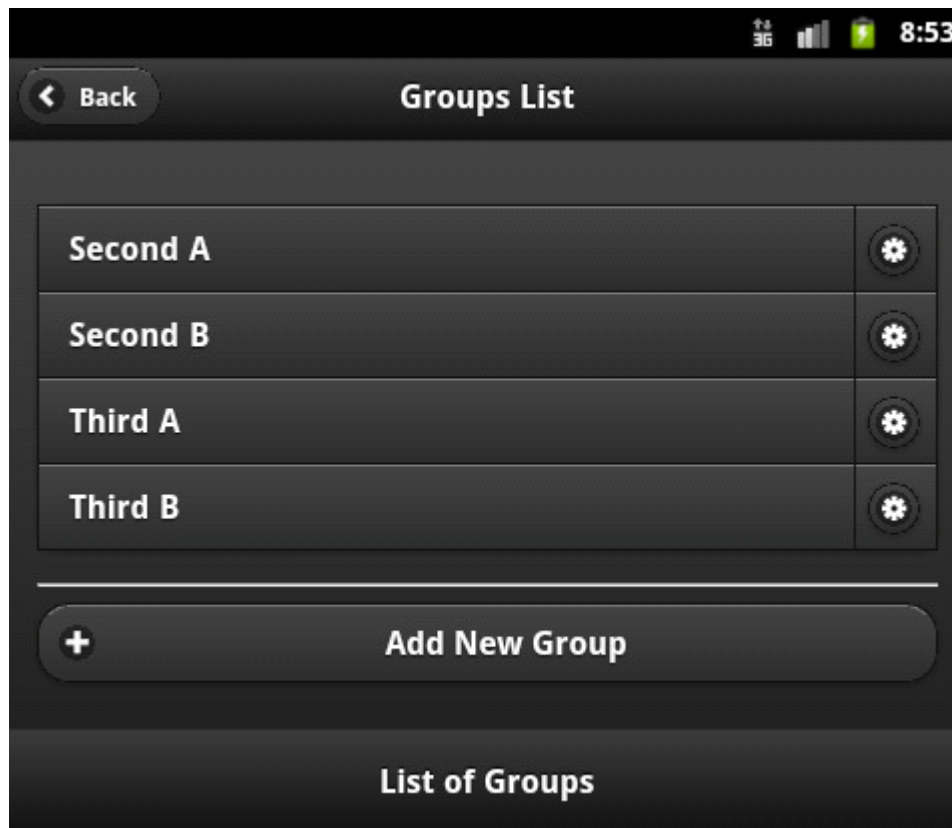


Figure 4.1: List of Groups

#### 4.4.1 Groups

The first window to be developed is *List of Groups*: as shown above 2.3, firstly a *page* is created inside **index.html**. This page, as every page, has three important elements:

- *header*. It could contain a backward button, which return application flow to previous visited page. This is disabled, because it could cause problems when previous page is an edit page, and another solution is preferred.
- *content*. Data itself, it has an important *id* to be filled with content by javascript code.
- *footer*. Usually page name.

### List Groups page. index.html

```
<div data-role="page" id="list_groups" data-add-back-btn="true">
  <div data-role="header" data-add-back-btn="false" >
    <a href="#list_settings" data-icon="arrow-l"
      data-theme="a" data-role="button">Back</a>
    <h1>Groups List</h1>
  </div>

  <div data-role="content">
    <ul id="groups_ul" data-role="listview" data-inset="true"
      data-split-icon="gear" data-split-theme="a" >
    </ul>
    <hr />
    <a style="text-align: center" onClick="onAddNewGroup();"
      data-role="button" data-icon="add">Add New Group</a>
  </div>
  <div data-role="footer" data-rel="back" data-theme="a">
    <p style="text-align: center">List of Groups</p>
  </div>
</div>
```

To fill this list of elements `<ul id=groups_ul (...)>` a function `onListAllGroups()` is called.

Firstly a loading ... logo appears, then a function from **database.js** is called (`loadAllGroups()` ).

### List Group page. interface.js

```
function onListAllGroups(){
  $.mobile.showPageLoadingMsg();
  table_global='GROUPS';
  loadAllGroups(global_db); // #groups_ul
  $.mobile.changePage("#list_groups", { transition: "slideup" } );
}
```

Function (`loadAllGroups()` ) calls (`queryAllGroupsDB()`) from **database.js**. This function fills `$('#groups_ul')` combo box with group data: `name`. Moreover user can access a list of students of that group (`listStudentsByGroup()`) or change group information (`EditGroup()`).



### List Group page. database.js

```
function queryAllGroupsDB(tx) {
    log("Query All Groups \n");
    var ul_list = $('#groups_ul');
    tx.executeSql('SELECT * FROM GROUPS', [],
        dbSuccessFunc = function(tx, rs) {
            ul_list.empty();
            var html = "";
            for (var i = 0; i < rs.rows.length; i++) {
                id = rs.rows.item(i).id ;
                html = "<li>";
                html += "<a onClick='global_id=\"" + id +
                    "\"; global_id_group=\"" + id
                    + "\"; table_global=\"groups\"; \"";
                html += " listStudentsByGroup(" + id + " );'  ";
                html += " href='#' >";
                html += rs.rows.item(i).data;
                html += "</a>";
                html += " <a data-role='button' data-position-to='window'";
                html += " data-iconpos='notext' ";
                html += " style='float:right;' href='#' ";
                html += " data-rel='dialog' data-theme='a' ";
                html += " data-transition='slideup' ";
                html += " onClick=\"EditGroup(" + id + "); \"";
                html += " global_id_group=\"" + id
                    + "\"; global_id=\"" + id + "\"; \">Edit</a>";
                html += "</li>";
                ul_list.append(html);
            }
            ul_list.listview('refresh'); },
        dbErrorFunc = function(ttx, e) {
            if (ttx.message)
                e = ttx;
            log(" There has been an error Select * from groups : "
                + e.message);
            return false;
        });
}
```

Second window to be developed is edit group information. The contents of *Edit Group* page from **index.html** is filled by **loadGroup()** function **database.js** as shown below 4.4.1.

### Edit Group page. index.html

```
<div data-role="page" id="edit_groups" data-add-back-btn="true">
  <div data-role="header" data-add-back-btn="true" >
    <h1>Group Edition</h1>
  </div>
  <div data-role="content">
    <label for="nombre_grupo">Name</label>
    <input id="in_nombre_grupo" enabled="true" />
    <label for="nivel_grupo">Level</label>
    <input id="in_nivel_grupo" enabled="true" />
    <a onClick="onUpdateGroup();" data-role="button"
      data-icon="check">Save</a>
  </div>
</div>
```

### Edit Group page. interface.js

```
function EditGroup(id_group){
  $.mobile.showPageLoadingMsg();
  global_id = id_group;      //local variable goes global
  table_global = 'GROUPS';
  loadGroup(global_db,id_group);
  $.mobile.changePage("#edit_groups", { transition: "slideup"});
}
```

Inside *loadGroup* function a sql query is built (`SELECT ... FROM ...`), then it is executed and returned values are loaded in result set variable (*rs*). With this data html form is populated (`$('#in_nombre_grupo')`). Whether an error is triggered then *dbErrorFunc* is called and a message appears on user interface.

These procedures will be very similar to all "*load*" functions (students, activities).

**Load Group page. database.js**

```
function loadGroup(db, id_group){
    db.transaction(function (tx) {
        log("Query Group \n");
        var sql = 'SELECT id, data, other_data ':
        sql += ' FROM GROUPS WHERE id =' + id_group;
        tx.executeSql(sql, [],
            dbSuccessFunc = function(ttx,rs){
                $('#in_nombre_grupo').val(rs.rows.item(0).data);
                $('#in_nivel_grupo').val(rs.rows.item(0).other_data);
            },
            dbErrorFunc = function(tx, e) {
                if (tx.message) e = tx;
                log(" There has been an error queryGroupDB: "
                    + e.message);
                alert(" There has been an error queryGroupDB: "
                    + e.message);
                return false;
            });
    });
}
```

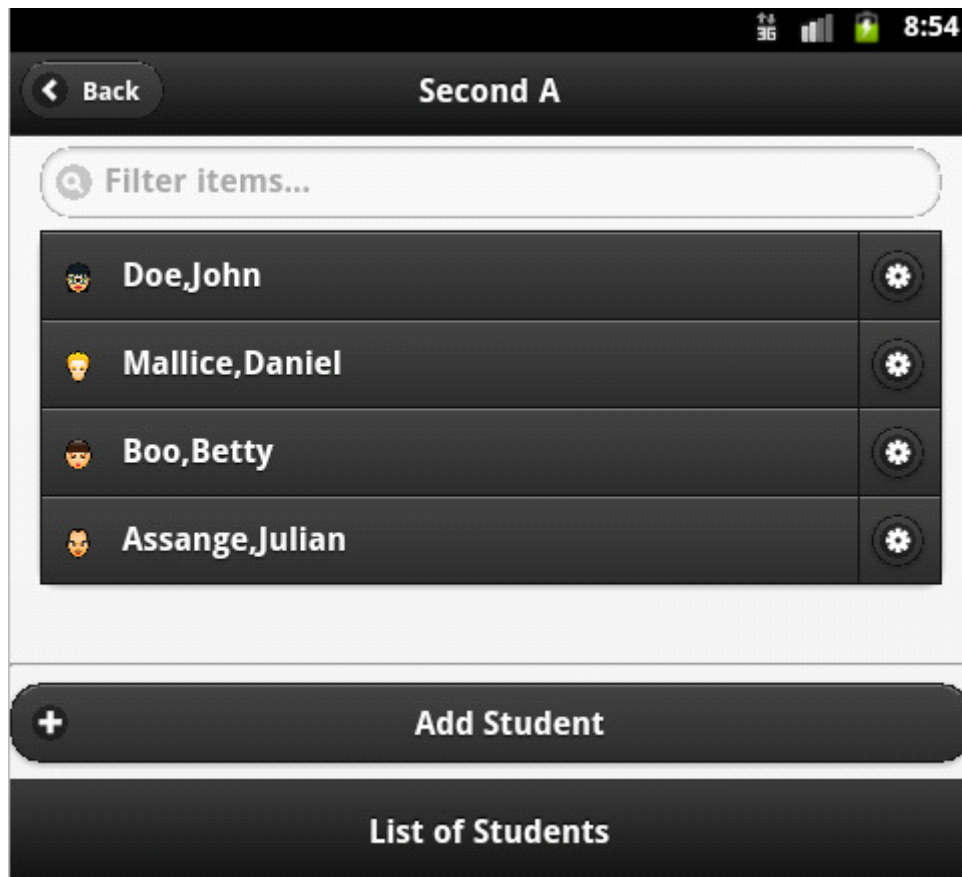


Figure 4.2: Students

#### 4.4.2 Students

Students pages are very similar to groups but, obviously, contain different fields (a complete list of fields is shown in database definition 4.5). Only list of students by group page and *LoadStudentsByGroup* code are shown. They are very similar to their *groups* counterpart. The main concern is about escape characters and colons, code is straightforward to read. Several lines were removed towards readability (...).



### List Students page. index.html

```
<div data-role="page" id="list_students_by_group"
  data-add-back-btn="true">
  <div data-role="header" data-back-btn-text="previous"
    data-add-back-btn="true" >
    <h1 id="id_list_students_by_group"> Students List</h1>
  </div>
  <div data-role="content">
    <ul id="list_students_by_group_ul" data-role="listview"
      data-theme="a" data-split-theme="a" data-split-icon="gear"
      data-filter="true" data-inset="true" >
    </ul>
  </div>
  <hr />
  <a onClick="onAddNewStudent();" data-role="button"
    data-icon="add" data-theme="a" >Add Student</a>
  <div data-role="footer" class="footer-docs" data-rel="back"
    data-theme="a">
    <p style="text-align: center">
      List of Students
    </p>
  </div>
</div>
```

### Load Student snippet. database.js

```
function loadStudentsByGroup(db,id_group) {
  db.transaction(
    function (tx) {
      var sql = 'SELECT STUDENTS.id as id, ';
      sql += ' STUDENTS.id_group as id_group,';
      sql += ' STUDENTS.name as name, ';
      (...)
      sql += ' FROM STUDENTS, GROUPS WHERE ';
      sql += ' STUDENTS.id_group = g_id AND id_group=';
      sql += ' + id_group;
      tx.executeSql(sql, [],
        dbSuccessFunc = function(tx, results) {
          var len = results.rows.length;
          var ul_list = $('#list_students_by_group_ul');
          $('#id_list_students_by_group').text(
            results.rows.item(0).data );
          var html;
          ul_list.empty();
          var id = 0;
          for (var i = 0; i < len; i++) {
            id = results.rows.item(i).id;
            html="<li >";
            html+="<a onClick='global_id="+id;
            html+=";table_global=\"students\";'";
            html+=" href='#' data-rel='dialog' ";
            html+=" data-transition='slideup'>";
            (...)
            html+=results.rows.item(i).surname + "," ;
            html+=results.rows.item(i).name+"</a>";
            html+="<a data-role='button' ";
            html+=" data-position-to='window' ";
            html+=" data-iconpos='notext' ";
            html+=" style='float:right;' href='#' ";
            html+=" data-rel='dialog' ";
            html+=" data-transition='slideup' ";
            html+=" onClick=\"EditStudent(" ;
            html+=id + ");\">Edit</a> </li>";
            ul_list.append(html);
          }
          ul_list.listview('refresh'); },
        dbErrorFunc = function(tx, e) {
          if (tx.message) e = tx;
          log(sql);
          log(" Error loadStudentsByGroup: "+e.message);
          alert("Error loadStudentsByGroup: " + e.message);
          return false;
        });
    }
  );
}
```

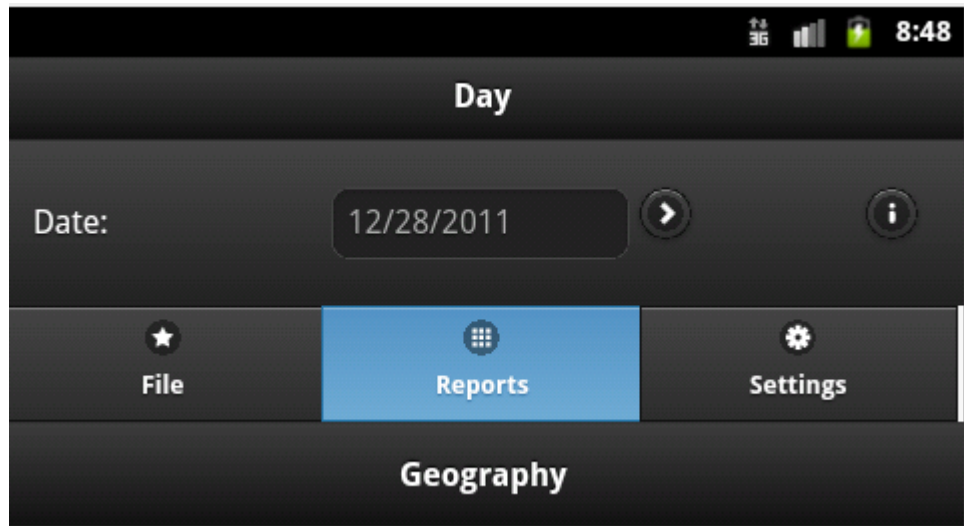


Figure 4.3: Main Page

#### 4.4.3 Main Page

Main page was not the very first page to be built, because it is related with attendance, sessions, groups and students, and also reports. Its design it has to follow several basic principles: simple but with no more than four clicks (even less) away to access to every application window-page. The most important item is current data, and the ">" symbol on the right to access *Attendance* page.

There are a table with three columns (*class="ui-grid-b"*), which contains a label, the current date, and an info button. And a list of navigation buttons to list data (**report**) or manage students, groups and activities (**settings**)

Of course there are a lot of elements that could be changed to improve user experience in this page.

## Main Page. index.html

```

<!-- Main Window    <!-->
<div data-role="page" id="daily_work">
  <div data-role="header" data-add-back-btn="true" data-theme="a">
    <h1>Day</h1>
  </div>
  <div data-role="content" data-theme="a">
    <div class="ui-grid-b">
      <div class="ui-block-a">
        <div data-role="fieldcontain">
          <label for="date" >Date:</label>
        </div>
      </div>
      <div class="ui-block-b">
        <input id="daily_date_scroller"
          name="daily_date_scroller" />
      </div>
      <div class="ui-block-c">
        <a href="#" data-role='button'
          data-icon='info' data-iconpos='notext'
          style='float:right;' onClick="help('date');">Help</a>
        <a href="#" data-role="button"
          data-icon="arrow-r" data-iconpos="notext"
          onClick="open_daily_page() " >Go</a>
      </div>
    </div>
  </div>
  <div data-role="navbar">
    <ul>
      <li>
        <a href="#" onClick="onGeneralFile() "
          data-role="button" data-icon="star"
          data-theme="a" > File</a>
      </li>
      <li>
        <a href="#" onClick="onGeneralListReports();"
          data-role="button" data-icon="grid"
          data-theme="a" >Reports</a>
      </li>
      <li>
        <a href="#" onClick="onGeneralListSettings();"
          data-role="button" data-icon="gear"
          data-theme="a" >Settings</a>
      </li>
    </ul>
  </div><!-- /navbar -->
  <div data-role="footer" class="footer-docs" data-theme="a">
    <p style="text-align:center;" id="teachers_name"></p>
  </div>
</div>

```



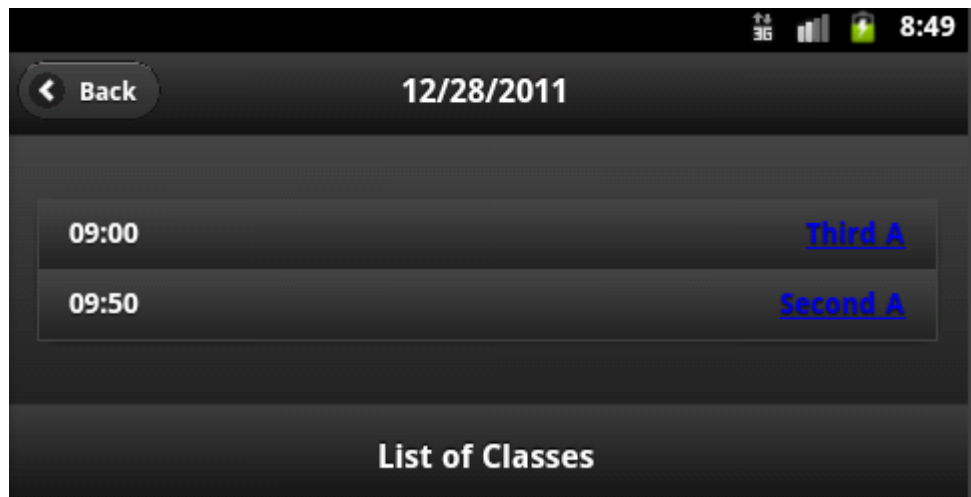


Figure 4.4: Schedule

#### 4.4.4 Attendance

Attendance per group is one of most important pages, and it list attendance, behaviour of students of a group. There is a previous page: list of groups for current day (*Schedule page*) (Figure: 4.4) 4.4.4.

```

Schedule Page. index.html
<div data-role="page" id="daily_schedule"
  data-add-back-btn="false" data-theme="a" >
  <div data-role="header"
    data-back-btn-text="previous"
    data-add-back-btn="false" data-theme="a">
    <a href="#daily_work"
      data-icon="arrow-l" data-theme="a"
      data-role="button">Back</a>
    <h1 id="current_day"> Current Day</h1>
  </div>
  <div data-role="content" data-theme="a">
    <ul id="groups_day_ul" data-role="listview"
      data-inset="true" data-split-icon="gear"
      data-split-theme="a" >
    </ul>
  </div>
  <div data-role="footer" class="footer-docs"
    data-add-back-btn="true" data-theme="a">
    <p style="text-align:center;" > List of Classes</p>
  </div>
</div>

```

The *queryStudentsAttendanceSuccess* 4.4.4 function fills *Attendance* page 4.4.4 with photos, names and surnames of student of the adequate group. User can choose among

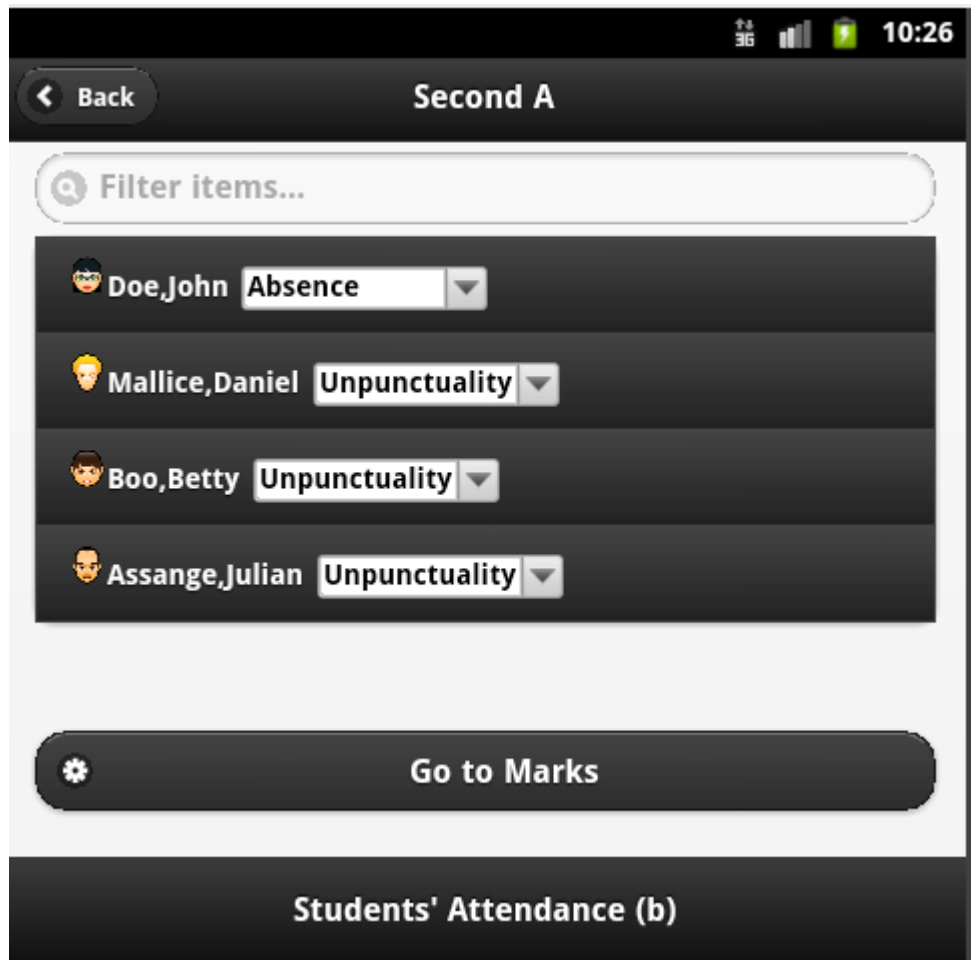


Figure 4.5: Attendance

---

different student "states", these "states" are filled in *fillSelectStudent* function. When user changes combo values function *studentState* is called which update or insert student state into *attendance* table in database.

### Attendance Page. index.html

```

<div data-role="page" id="list_students_attendance"
  data-add-back-btn="false">
  <div data-role="header" data-back-btn-text="previous"
    data-add-back-btn="false" data-theme="a" >
    <a href="#" onClick="open_daily_page();" data-icon="arrow-l"
      data-theme="a" data-role="button">Back</a>
    <h1 id="current_group_attendance">Student List for Group</h1>
  </div>
  <div data-role="content"> <!--Data goes here //-->
    <ul data-role="listview" id="students_attendance_ul"
      data-autodividers="true" data-split-icon="gear"
      data-split-theme="a" data-filter="true" data-inset="true"
      data-theme="a" ></ul>
  </div>
  <div data-role="content">
    <a href="#" onClick="onOpenStudentsAssessment();"
      data-icon="gear" data-theme="a"
      data-role="button">Go to Marks</a>
  </div>
  <div data-role="footer" class="footer-docs"
    data-rel="back" data-theme="a">
    <p style="text-align:center;" >Students' Attendance (b)</p>
  </div>
</div>

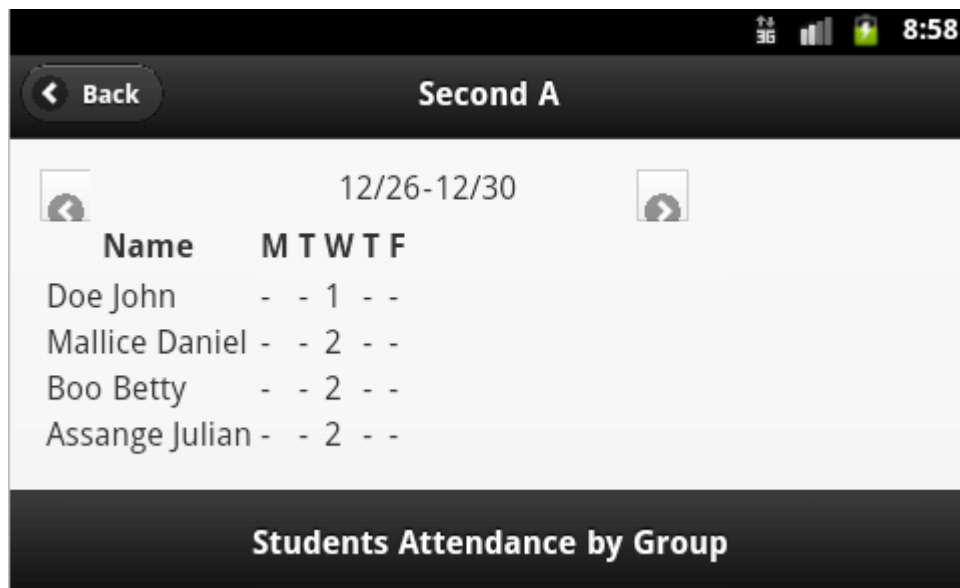
```

### Fill Attendance database.js

```

function queryStudentsAttendanceSuccess(tx, results) {
  (...)
  for (var i=0;i<results.rows.length;i++) {
    id = results.rows.item(i).id_student;
    photo = results.rows.item(i).photo;
    name = results.rows.item(i).name;
    surname = results.rows.item(i).surname;
    id_group = results.rows.item(i).g_id;
    id_session = global_session; //
    html = "<li data-role='fieldcontain'> ";
    html+= "<label for='select_student_"+id+"' class='select'>";
    (...)
    html+= surname + "," + name + "</label> " ;
    html+="<select name='select_student_"+id
      +"' id='select_student_"+id+"' ";
    html+= " onChange='studentState(\"+id + \",\"
      +id_group + \",\"+id_session+ \");'>";
    html+="</select>";
    html+= "</li>";
    $('#students_attendance_ul').append(html);
    fillSelectStudent(global_db,"select_student_"+id,id_session,id);
  }
  $('#students_attendance_ul').listview('refresh');
}

```



Name	M	T	W	T	F
Doe John	-	-	1	-	-
Mallice Daniel	-	-	2	-	-
Boo Betty	-	-	2	-	-
Assange Julian	-	-	2	-	-

Students Attendance by Group

Figure 4.6: Attendance Report


#### 4.4.5 Activities

Activities list is very similar to groups and students.

##### List Activities: index.html

```
<div data-role="page" id="list_all_activities" name="activities"
  data-add-back-btn="false" data-direction="reverse"
  data-theme="a" >
  <div data-role="header" data-add-back-btn="false" >
    <a href="#list_settings" data-icon="arrow-l" data-theme="a"
      data-role="button">Back</a>
    <h1>List of Activities</h1>
  </div>
  <div data-role="content">
    <h2 data-role="listview"
      id="header_activities" ></h2>
    <ul data-role="listview"
      id="activities_ul" ></ul>
  </div>
  <div data-role="footer" class="footer-docs" data-rel="back"
    data-theme="a">
    <a onClick="onAddNewActivity();" data-role="button"
      data-icon="add">Add New Activity</a>
    <p style="text-align: center"> List of Activities</p>
  </div>
</div>
```

Javascript code for list of activities 4.4.5 is similar to their students and groups counterparts.

 **Load all activities: database.js**

```
function queryLoadAllActivitiesDB(tx) {
  tx.executeSql('SELECT * FROM ACTIVITIES', [],
    dbSuccessFunc = function(tx, results) {
      var len = results.rows.length;
      if(len>0) {
        var ul_list = $('#activities_ul');
        var html;
        ul_list.empty();
        var id = 0;
        for (var i = 0; i < len; i++) {
          id = results.rows.item(i).id;
          html = "<li >";
          html += "<a onClick='global_id=" + id + " ";
          table_global=\"activities\";
            onUpdateActivity(\"+id+\") ' ";
          html += " href='#' data-rel='dialog'
            data-transition='slideup'>";
          html += results.rows.item(i).name ;
          html += "</a>";
          html += "<a data-role='button'
            data-position-to='window' ";
          html += " data-iconpos='notext'
            style='float:right;' href='#' ";
          html += " data-rel='dialog'
            data-transition='slideup' ";
          html += " onClick=\"onUpdateActivity(" +
            id + ");\">Edit</a>";
          html += "</li>";
          ul_list.append(html);
        }
        ul_list.listview('refresh');
        global_max_activities = results.rows.length;
      }
    },
    (...))
};
```

Function *loadActivity* is more complex than student or groups because when an activity is updated, information about groups (name) is compulsory to determine whether a group will carry out that activity.

### Load Activity: database.js

```
function loadActivity(db, id_activity ) {
var sql = " SELECT  id, name , date_init , date_end ,
weight , final FROM ACTIVITIES WHERE id="+id_activity;
db.transaction(function(tx) {
    tx.executeSql(sql, [],
        dbSuccessFunc = function(tx, results) {
            if(results.rows.length>0 ){
                $('#in_name_activity').val(results.rows.item(0).name);
            (...)
            var sql= "SELECT id, data, other_data FROM groups;";
            tx.executeSql(sql, [],
                dbSuccessFunc = function(ttx, rs) {
                    if(rs.rows.length>0) {
            (...)
                for (var i = 0; i < rs.rows.length; i++) {
                    id = rs.rows.item(i).id;
                    html = "<li >";
            (...)
                    html += rs.rows.item(i).data + "</label>";
                    html += " </li></br>";
                    ul_list.append(html);
                }
                var sql = "SELECT  activities_group.id_group,
                    activities_group.id_activity, ";
            (...)
                tx.executeSql(sql, [],
                    dbSuccessFunc = function(txx, rrs) {
                        for (var i = 0; i < rrs.rows.length; i++) {
                            var id_group = rrs.rows.item(i).id_group;
                            var in_act = $("#in_group_activity_" + id_group );
                            if(rrs.rows.item(i).enabled !=0) {
                                in_act.attr("checked",true);
                            (...)
                        }
                    }
                }
            }
        }
    }
}
```

### 4.4.6 Assessment

*Assessment* page is very similar to *Attendance*, and only Assessment reports is markedly different from *Attendance*. *loadStudentsAssessment* function fills table:

```
$('#students_assessment_reports_table');
```

with data from assessment: each column is a different activity and each row a student. This procedure does these actions:

- Seek database for activities and students of a particular group.
- Fill a matrix (*SActivity*) with marks and weight of that mark.
  - Each row contains data of a student (first index).
  - Each column contains data of an activity (second index).
- Fill the *students\_assessment\_reports\_table* with data from that matrix.
- Calculate average mark.





### Load Assessment Report (1/2): database.js

```
function loadStudentsAssessment(db, id_group){
    $('#current_group_assessment_reports').text("Name of the group");
    var sql=" SELECT DISTINCT students.id as s_id, students.name as
    (...)
    sql += " AND GROUPS.id=students.id_group AND GROUPS.id="+id_group;
    sql += " ORDER BY students.id ; " ;
    db.transaction(function(tx) {
        tx.executeSql(sql, [],
            dbSuccessFunc = function(tx,results){
                var ul_list=$('#students_assessment_reports_ul');
                var table = $('#students_assessment_reports_table');
                (...)
                var len = results.rows.length; // max number of students
                (...) // Initialize to zero Students-Activities Matrix: SActivity
                var SActivity = new Array (len+1);
                (...)
                for (var i = 0; i < len; i++) {
                    s_id = results.rows.item(i).s_id;
                    (...)
                    activity_id = results.rows.item(i).a_id;
                    if(s_id!=old_s_id) {
                        if (is_new==1){
                            is_new=0;
                        } else {
                            no_activities=
                                Math.max(no_activities,a_no_activities);
                        }
                        no_students ++;
                        student_a.push(s_surname+", "+s_name);
                        a_no_activities=1;
                        k=0;
                    } else {
                        a_no_activities++;
                        k++;
                    }
                }
                SActivity[no_students][activity_id].mark = mark;
                SActivity[no_students][activity_id].weight = weight;
                SActivity[no_students][activity_id].activity = activity_id;
                activity_name_a[activity_id] = activity_name ;
                old_s_id=s_id;
            }
        );
    });
}
```

## Load Assessment Report (2/2): database.js

```

        max_students = no_students;
        html = "";
        for(var i=0;i<=max_students; i++) {
            html += "<tr><td>" + student_a[i] + "</td>";
            measure=0.0;
// XXX: Check DB ID's, if it begins in 0 or 1 (assumed 1)
            for(j=1;j<global_max_activities+1;j++) {
                html += "<td>"
                    + SActivity[i][j].mark + "</td>";
// XXX: Weight sum should return 100
                measure += SActivity[i][j].mark*
                    SActivity[i][j].weight/100.0;
            }
            html += "<td>(" + measure + ")</td>";
            html += "</tr>";
        }
        html_pre = "<tr>";
        html_pre += "<th> Name </th>";
// XXX: Check DB ID's, if it begins in 0 or 1 (assumed 1)
        for(var j=1;j<global_max_activities+1 ; j++) {
            html_pre += "<th>" + activity_name_a[j] + "</th>";
        }
        html_pre += "<th>Mean</th></tr>";
        table.empty().append("<thead>" + html_pre
            + "</thead><tbody>" + html + "<tbody>");
        return true;
    },
    (...)
} );
    } );
}

```

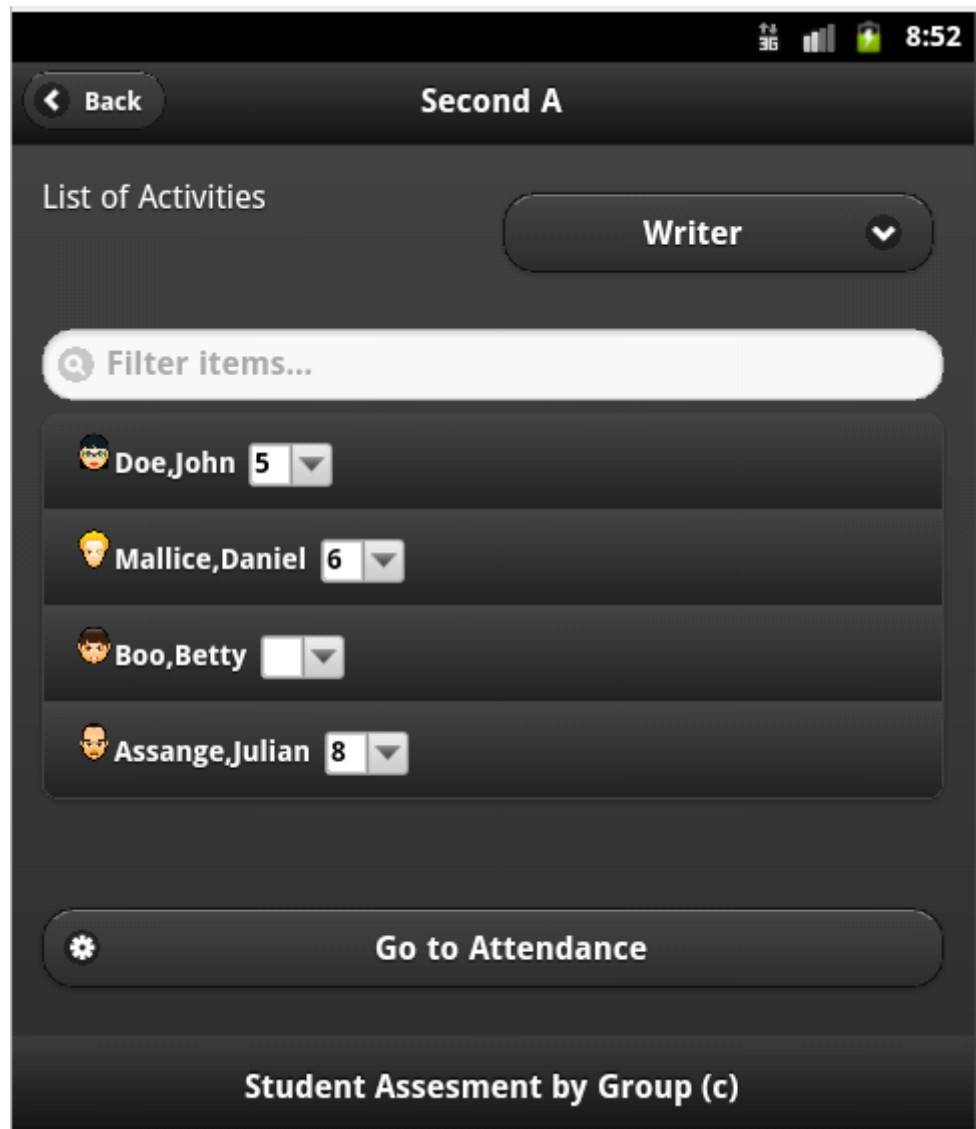
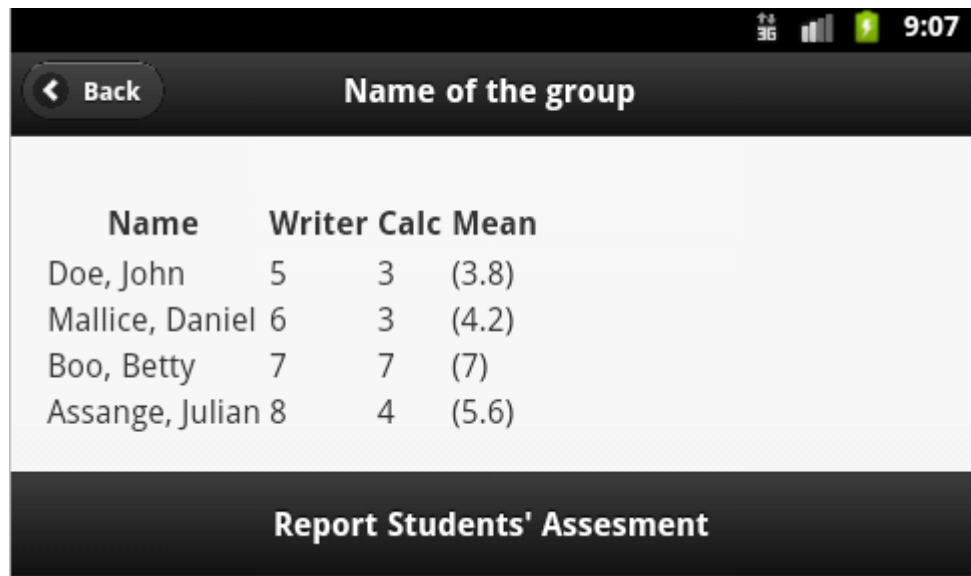


Figure 4.7: Assessment



Name	Writer	Calc	Mean
Doe, John	5	3	(3.8)
Mallice, Daniel	6	3	(4.2)
Boo, Betty	7	7	(7)
Assange, Julian	8	4	(5.6)

Figure 4.8: Assessment Report

## 4.5 Database

Next figure 4.9 is a graphical representation of EduXes database 4.5 obtained using *sqlfairy* program [SQL12].

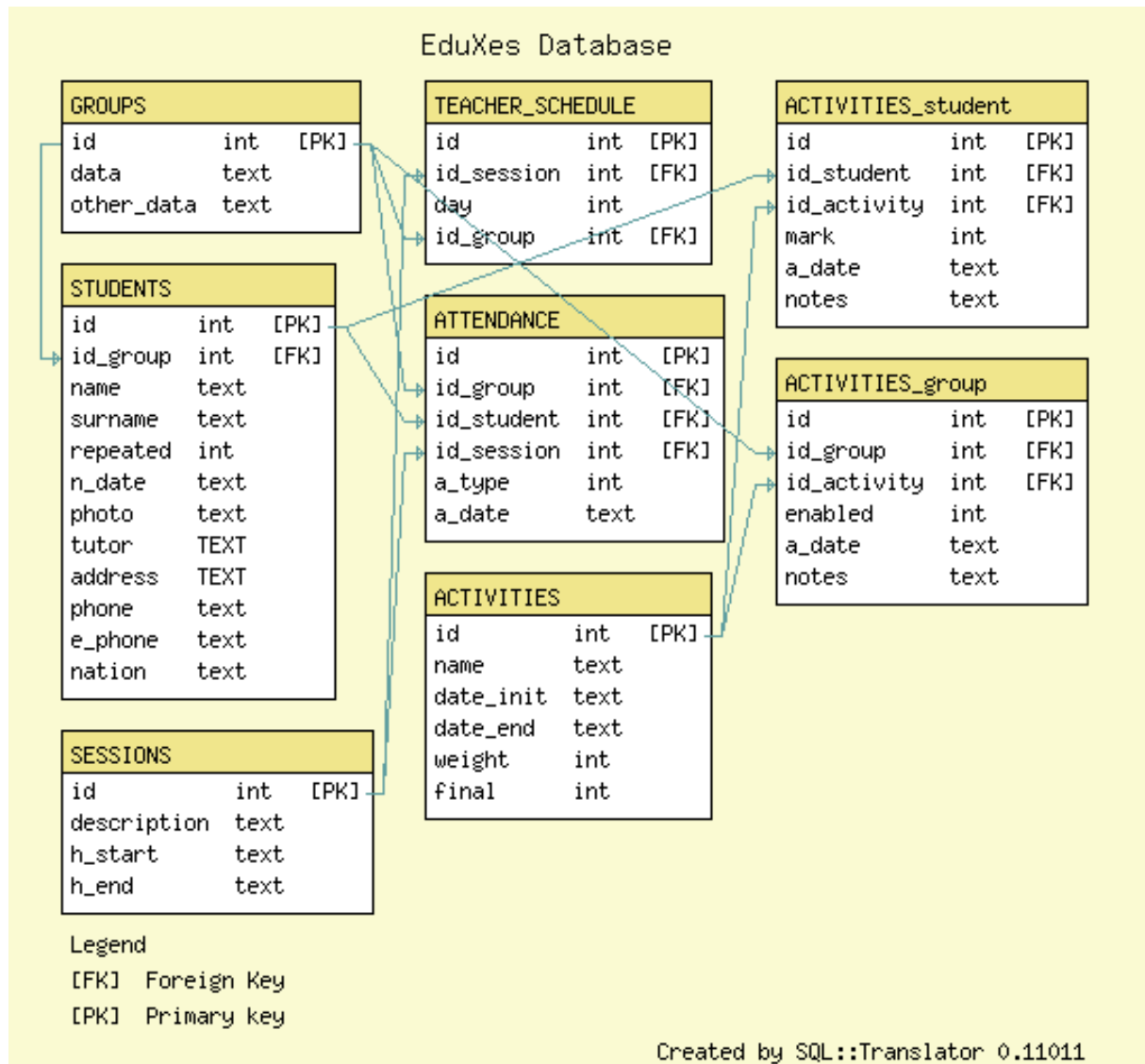


Figure 4.9: Database



## EduXes Database structure

```
-- Groups
CREATE TABLE IF NOT EXISTS GROUPS (id integer primary key ,
    data text , other_data text);

--- Students
CREATE TABLE IF NOT EXISTS STUDENTS (
    id integer primary key, id_group integer not null,
    name text, surname text,
    repeated integer, n_date text , photo text,
    tutor TEXT, address TEXT, phone text, e_phone text,
    nation text,
    FOREIGN KEY(id_group) REFERENCES GROUPS(id));

-- Sessions ( franja horaria)
CREATE TABLE IF NOT EXISTS SESSIONS (id integer primary key,
    description text, h_start text, h_end text);

-- Teacher's schedule
CREATE TABLE IF NOT EXISTS TEACHER_SCHEDULE (id integer primary key,
    id_session integer, day integer, id_group integer,
    FOREIGN KEY(id_group) REFERENCES GROUPS(id),
    FOREIGN KEY(id_session) REFERENCES SESSIONS(id));

-- Students Attendance
CREATE TABLE IF NOT EXISTS ATTENDANCE (id integer primary key ,
    id_group integer, id_student integer, id_session integer,
    a_type integer, a_date text,
    FOREIGN KEY (id_student) REFERENCES STUDENTS (id),
    FOREIGN KEY (id_group) REFERENCES GROUPS(id),
    FOREIGN KEY (id_session) REFERENCES SESSIONS(id) );

-- Activities
CREATE TABLE IF NOT EXISTS ACTIVITIES
    (id integer primary key, name text, date_init text,
    date_end text, weight integer, final integer );
CREATE TABLE IF NOT EXISTS activities_student
    (id integer primary key , id_student integer,
    id_activity integer,
    mark integer, a_date text, notes text,
    FOREIGN KEY (id_student) REFERENCES students (id),
    FOREIGN KEY (id_activity) REFERENCES activities(id) );
CREATE TABLE IF NOT EXISTS activities_group
    (id integer primary key , id_group integer,
    id_activity integer,
    enabled integer, a_date text, notes text,
    FOREIGN KEY (id_group) REFERENCES groups (id),
    FOREIGN KEY (id_activity) REFERENCES activities(id) );
```

# Chapter 5

## Results

Application is evolving from list, edit students and groups, to its final goals. These objectives were fulfilled:

### 5.1 Objectives completed

- Manage students and group of students.

List students and groups.

- Management of attendance and misbehaviour for each student.

List weekly attendance for each group.

- Management of assessment for each group and student.

List group assessment for each activity

- Access to daily schedule to set assessments and attendance information.
- Filter for only day of week days.
- Test in real hardware: Android 4.0 and Android 2.3.3. Does not work with fluency in 4.0. It could be caused because application is designed to Android 2.3.3

### 5.2 Further objectives

There are several objectives not fulfilled yet, those are, in priority order:

- Fix several bugs (blinking pages, assessments, etc.).
- Test units.
- User Documentation.

- Developer documentation. API documentation (JSDoc).
- Add an image or photo to student.
- Import data (students, groups, sessions, schedule, activities) from file or URL.
- Timetable management. A window to manage groups timetable. When a group has class with this teacher.
- Server synchronization with a custom application or XadeWeb [dEXdG12].
- Export data to a file.
- Host application.
- Upload to Google Play.
- Xade web interface. It could be done through another an ad-hoc application.

And the most important objective is to build a community around this application. Firstly testers who help to polish, add robustness and more functionality to EduXes.

These objectives were not fulfilled because lack of time, author's skills, too complicated to be achieved without a robust code, and a community behind it.

### 5.3 Problems faced

Several problems were faced during this application development:

Eclipse environment: A stable, reliable and up-to-date IDE, with several plug-ins is needed. Download vanilla Eclipse Juno from its web-site was chosen because it is more stable, reliable, compatible with newer versions. Aptana Javascript plugin was chosen because Aptana allows source code auto-completion in JQuery.

PhoneGap and Android incompatibilities. Android 2.3.3 requires JQuery-1.8.1 and does not work on higher versions.

Error handlers. There were several problems with *tx.executeSql(...)* function, it was confused with *db.transaction(...)*:

```
tx.executeSql(sql, [parameters], successHandler, errorHandler)
```

and

```
db.transaction(queryFunction, errorHandler, successHandler)
```



have up to four and three parameters respectively, only first one is mandatory. First one was used because success and error handlers for `tx.executeSql` allows an atomic error control.

Passing variables to functions: Only whether another solution is not known or feasible, global variables are used: named after *global\_*, and there are several global variables in block capitals (for enumerators).

## 5.4 Statistics

With *Ohloh* source code line counter, results are:



### Ohloh counter

```
$ ohcount -i assets/www/js/database.js assets/www/js/interface.js \
assets/www/js/create_populate_db.js assets/www/index.html \
assets/www/remove.html
```

Examining 5 file(s)

Ohloh Line Count						
Language	Code	Comment	Comment%	Blank	Total	File
javascript	1306	148	10.2%	201	1655	database.js
javascript	388	84	17.8%	57	529	interface.js
javascript	402	57	12.4%	46	505	create_populate_db.js
html	535	42	7.3%	97	674	index.html
javascript	1	0	0.0%	0	1	index.html
html	28	1	3.4%	8	37	remove.html

With David A. Wheeler's 'SLOCCount' statistics tools [Whe05], results are (with Javascript patch[Whe07]):



## David A. Wheeler's 'SLOCCount'

```
$ /usr/local/bin/sloccount .
```

SLOC Directory SLOC-by-Language (Sorted)

```
1694    source_code    javascript=1694
```

Totals grouped by language (dominant language first):

```
javascript:    1694 (100.00%)
```

Total Physical Source Lines of Code (SLOC) = 1,694

Development Effort Estimate, Person-Years (Person-Months)= 0.35 (4.17)

(Basic COCOMO model, Person-Months = 2.4 \* (KSLOC\*\*1.05))

Schedule Estimate, Years (Months) = 0.36 (4.30)

(Basic COCOMO model, Months = 2.5 \* (person-months\*\*0.38))

Estimated Average Number of Developers (Effort/Schedule) = 0.97

Total Estimated Cost to Develop = \$ 46,989

(average salary = \$56,286/year, overhead = 2.40).

Generated using David A. Wheeler's 'SLOCCount'

# Chapter 6

## Personal evaluation of the practicum

My mentor, Manuel Rego guided me through the application design and coding, he taught me how JQueryMobile, PhoneGap works, and how to go step-by-step. Also I have reused several SergasApp functions from Manuel's application.

Firstly, it was really difficult to prepare environment, because there a lot of incompatibilities among plug-ins, Eclipse versions and so on. Eventually I find out an uptodate version (Juno) which works with Aptana Plugins, Android SDK, even Git control versions. As well I found the correct version of each library.

On the other hand, in the beginning write code was not so complicated, despite of application work-flow confusion and increasing complexity. These complexity were decreasing as far as I used one function to do only one task and I did not need to add more global variables. To avoid that complexity I have to rewrite several functions.

Honestly, I believed that I was not a good coder, nor a good graphical user interface designer neither a good database designer; but at this point I realized how important is to simplify objects (windows, functions, ...) as much as possible, and reuse ideas from one point to another. And, is very important to study and learn from others applications (interface and code). Also I realized that Siestta developer did a good job.

Eventually, I have learnt several technologies (Android, WebSQL, git, etc) and I have done my first Android App, I am confident that soon I will use this application in my daily work.



# Bibliography

- [Aca11]     Academics. Classroom management system. <https://play.google.com/store/apps/details?id=com.gradebook.academics>, 2011.
- [App05]     Inc. Appcelerator. Aptana is a complete environment to build java, php, python applications, along with complete html, css and javascript editing. <http://www.aptana.org/>, 2005.
- [Aqu12]     José Antonio Salgueiro Aquino. Classroom management for high school teachers. <https://github.com/joseantoniosa/EduXes>, 2012.
- [Cas12a]     Manuel Rego Casanovas. Application to access libreplan. <https://github.com/Igalia/libreplanapp/>, 2012.
- [Cas12b]     Manuel Rego Casanovas. Application to access sergas appointment system. <http://mrego.github.com/sergasapp/>, 2012.
- [dEXdG12]   Consellería de Educación. Xunta de Galicia. Xade web is a web interface for educational data management. <http://edu.xunta.es/xade/>, 2012.
- [fA10]     Android for Academics. Attendance. <http://androidforacademics.com/2010/09/attendance-instruction-manual/>, 2010.
- [Fou01]     Eclipse Foundation. Eclipse interface development environment. <http://www.eclipse.org/>, 2001.
- [Git08]     GitHub. Powerful collaboration, review, and code management for open source and private development projects. <https://github.com/>, 2008.
- [iagdc12]     inpocketsolution at gmail dot com. Teacher aide. <https://sites.google.com/site/teacheraidepro/>, 2012.
- [Inc01]     Android Inc. Android simple development kit. <http://developer.android.com/sdk/index.html>, 2001.

- [Lle12] Llerik. Teacher organizer. <http://4pda.ru/forum/index.php?showtopic=345457&st=80>, 2012.
- [Med12] Acid Media. Image only scrollers. <http://mobiscroll.com/>, 2012.
- [Mob12a] JQuery Mobile. JQuery mobile. <http://jquerymobile.com/demos/1.1.1/>, 2012.
- [Mob12b] JQuery Mobile. The jquery mobile "page" structure. <http://jquerymobile.com/demos/1.1.1/docs/pages/page-anatomy.html>, 2012.
- [Ora95] Oracle. Java standard edition. <http://www.oracle.com/technetwork/java/javase/downloads/index.html>, 1995.
- [P06] Ramón Castro Pérez. Classroom management web application. <http://siesta.sourceforge.net/doc/index.html>, 2006.
- [Pho12a] PhoneGap. Phonegap. android getting started guide. [http://docs.phonegap.com/en/1.8.1/guide\\_getting-started\\_android\\_index.md.html](http://docs.phonegap.com/en/1.8.1/guide_getting-started_android_index.md.html), 2012.
- [Pho12b] PhoneGap. Phonegap getting started with android. [http://docs.phonegap.com/en/1.8.1/guide\\_getting-started\\_android\\_index.md.html#Getting%20Started%20with%20Android](http://docs.phonegap.com/en/1.8.1/guide_getting-started_android_index.md.html#Getting%20Started%20with%20Android), 2012.
- [Pho12c] PhoneGap. Phonegap is a free and open source framework that allows you to create mobile apps using standardized web apis for the platforms you care about. <http://www.phonegap.com>, 2012.
- [Pho12d] PhoneGap. Phonegap local storage. [http://docs.phonegap.com/en/1.8.1/cordova\\_storage\\_storage.md.html#Storage](http://docs.phonegap.com/en/1.8.1/cordova_storage_storage.md.html#Storage), 2012.
- [S.L09] Enxenio S.L. Complete classroom management system. <http://www.sixa.es>, 2009.
- [SQL12] SQLFairy. Sqlfairy - the sql translator. <http://sqlfairy.sourceforge.net>, 2012.
- [Uni05] Unisoft. Académico <sup>TM</sup> is a software to school management. <http://www.unisoftcolombia.com/unisoftcolombia/index.php?link=academico>, 2005.

- [Whe05] David A. Wheeler. David a. wheeler's 'sloccount'. <http://sourceforge.net/projects/sloccount/>, 2005.
- [Whe07] David A. Wheeler. David a. wheeler's 'sloccount' patch for javascript. [https://sourceforge.net/tracker/index.php?func=detail&aid=1730985&group\\_id=171819&atid=859390](https://sourceforge.net/tracker/index.php?func=detail&aid=1730985&group_id=171819&atid=859390), 2007.