A
Setup Guide

This chapter aims to guide through the integration of all components used to develop the proposed IoMT pervasive game, therefore it guides for the preparation of the development environment.

A.1
Android Studio Setup

Android Studio IDE was chosen for the development of the IoMT pervasive game, so all integration that is described in this chapter will consider that. Thus, it is recommended that the same IDE is used for any project that wishes to follow the integration described ahead.

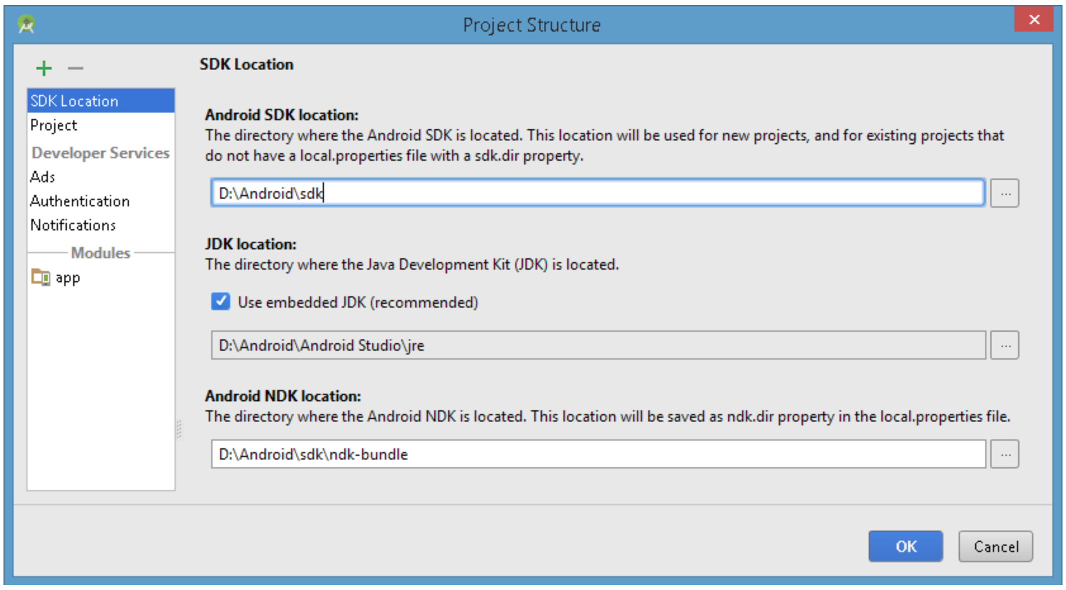
When it comes to setting Android Studio up, the only concern is to make sure to include the Android SDK on the installation, as it is necessary to the development of the application. In case you already have the SDK, you do not need to install it again, but you will have to link the SDK location in your computer on Android Studio configuration, as shown in figure A.1. You may use any Java Development Kit version 8 or newer, but you can use the embedded JDK version 8.

We will not be creating our Android Studio project through the IDE, instead we will prepare the project beforehand in order to integrate the framework LibGDX. This is due to the existence of a project generator for the framework that reduces the effort into creating a LibGDX project.

A.2
LibGDX Project

To aid the development of projects that use LibGDX framework, LibGDX project generator (figure A.2) creates a template project with all the necessary components to enable the use of the framework. In the generator, it is possible to pick the name of the project, package, main class, as well as choose the project destination path, but you must ensure that the Android SDK location is correct.

Once the project is generated, the next step is to import it into Android Studio.



SDK Location

Android SDK location:
The directory where the Android SDK is located. This location will be used for new projects, and for existing projects that do not have a local.properties file with a sdk.dir property.

D:\Android\sdk

JDK location:
The directory where the Java Development Kit (JDK) is located.

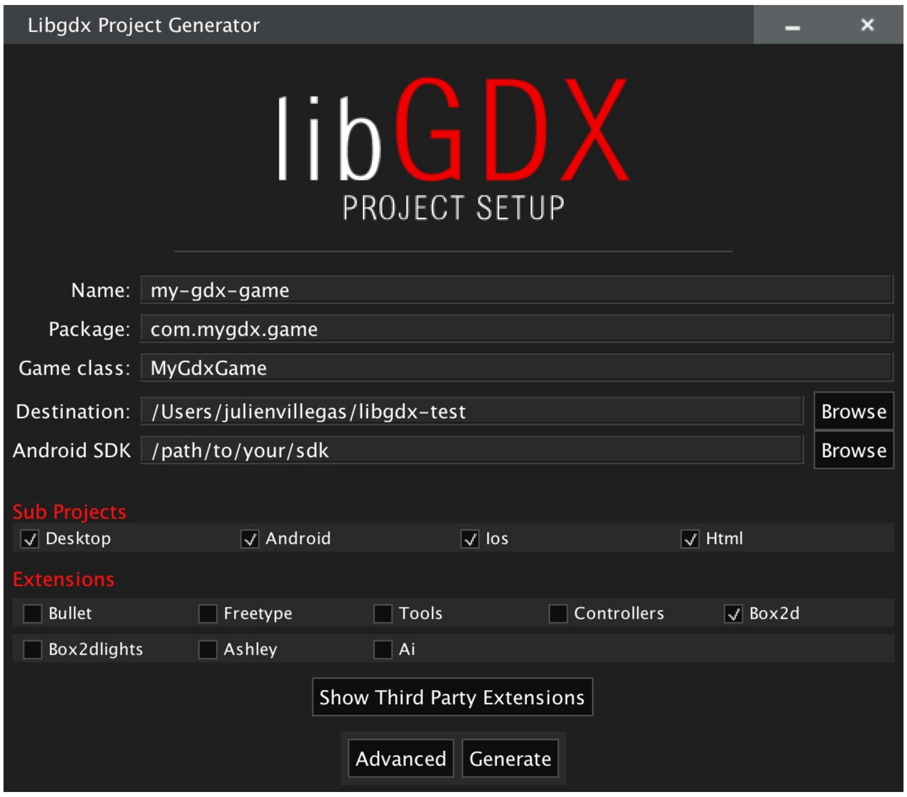
☑ Use embedded JDK (recommended)

D:\Android\Android Studio\jre

Android NDK location:
The directory where the Android NDK is located. This location will be saved as ndk.dir property in the local.properties file.

D:\Android\sdk\ndk-bundle

Figure A.1: Android Studio SDK Location configuration



Name: my-gdx-game
Package: com.mygdx.game
Game class: MyGdxGame
Destination: /Users/julienvillegas/libgdx-test
Android SDK /path/to/your/sdk

Sub Projects: ✓ Desktop ✓ Android ✓ Ios ✓ Html

Extensions: ☐ Bullet ☐ Freetype ☐ Tools ☐ Controllers ✓ Box2d ☐ Box2dlights ☐ Ashley ☐ Ai

Figure A.2: LibGDX Project Generator

A.3 ContextNet Setup

To better structure ContextNet's Setup guide, we divided into two different sub-guides: the first one details the setup for the SDDL Core (that runs the server side), and the second one details the setup for the Mobile Hub extension of the ContextNet middleware.

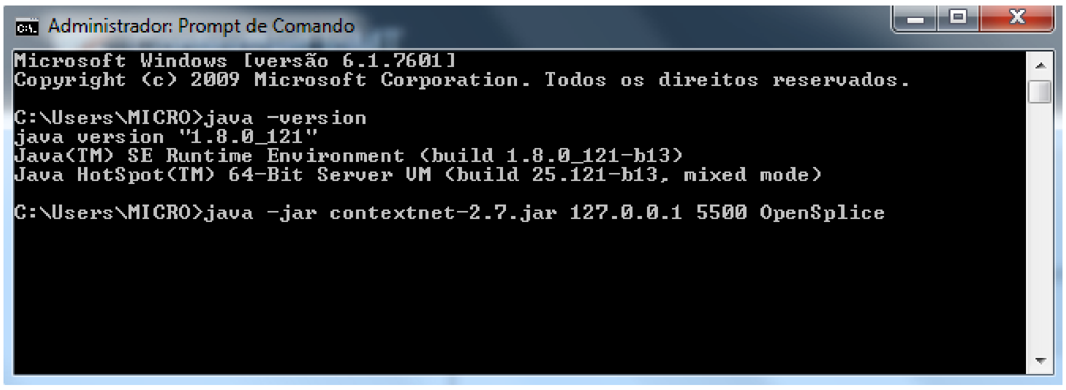
A.3.1
SDDL Core Setup

The ContextNet middleware is distributed as a set of software modules in the form of JAR files. In order to develop applications based on ContextNet services, you will need to include these libraries in your application build path [45]. In the case of our application, which is a LibGDX project, we must use Gradle to include these libraries, which will be explained further in the Gradle Setup section.

To provide real-time communication services, the ContextNet's middleware communication layer, Scalable Data Distribution Layer (SDDL), uses the Data Distribution Service (DDS) protocol in its core network and the MR-UDP protocol in the edges [45]. It is necessary to utilize some DDS product. For the latest ContextNet, version 2.7, OpenSplice 6.7 is the recommended option for an open source implementation of DDS. A detailed tutorial on how to install OpenSplice can be found at the ContextNet's tutorial page [45]. Additionally, it also requires Sun/Oracle Java 1.6 runtime or newer.

To execute a ContextNet application, one needs to first create the gateway, which will instantiate the core infrastructure if none exists. You can create a gateway by running the ContextNet middleware jar file. The gateway receives three parameters: a public IP address, a given port number, and a DDS vendor implementation to be used [45]. Figure A.3 serves an example of a command line that locally deploys the gateway through loopback IP "127.0.0.1". A helpful tip is to create a batch file with the command line to save time when deploying the gateway.

Once the gateway is running, ContextNet's SDDL will be fully usable, and will provide the means to distribute messages between the game's client and the game's server in a very reliable manner.



```
Microsoft Windows [versão 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.

C:\Users\MICRO>java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)

C:\Users\MICRO>java -jar contextnet-2.7.jar 127.0.0.1 5500 OpenSplice
```

Figure A.3: Command line that deploys gateway locally

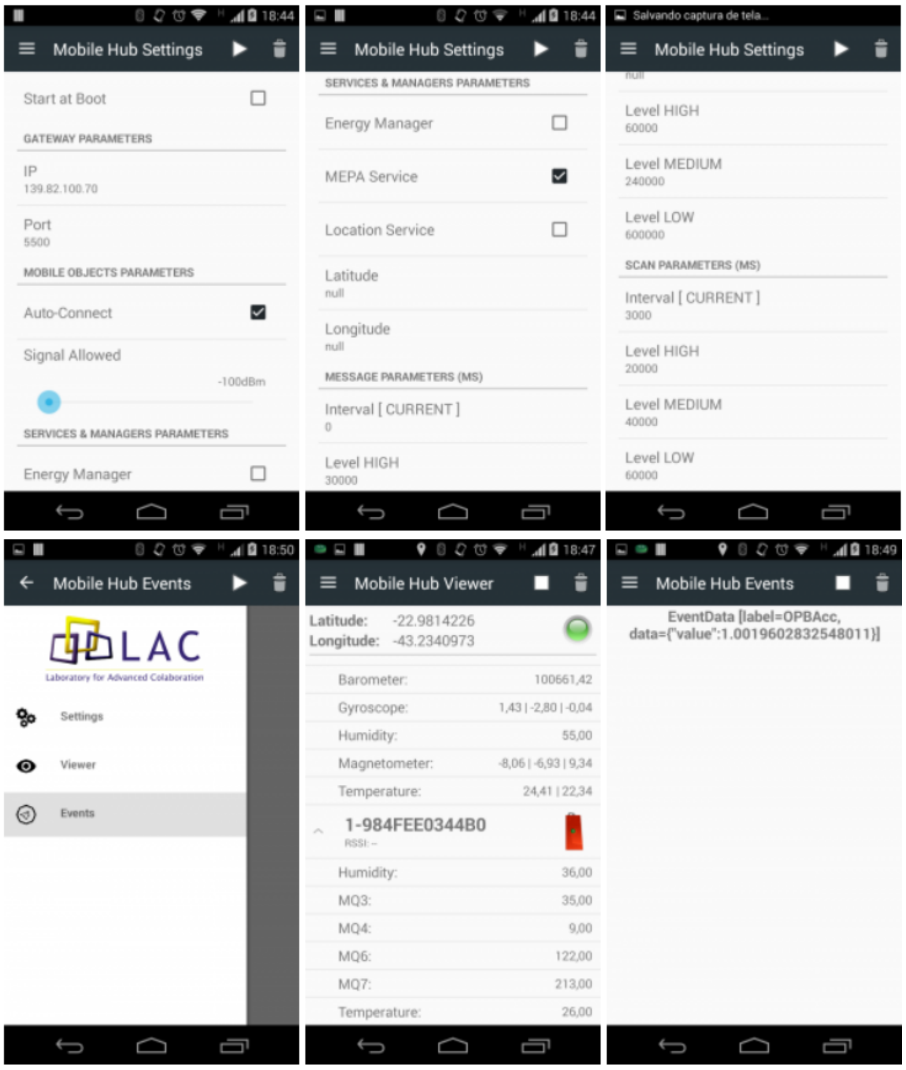
A.3.2 Mobile Hub Setup

The Mobile Hub (M-Hub) is a general-purpose middleware that enables mobile personal devices (Android smartphones and tablets) to become the propagator nodes (i.e. gateways to the Internet) for the simpler IoT objects or Mobile Objects (M-OBJ) (sensors/actuators) with only short-range WPAN interfaces. It provides context information such as current local time and/or the (approximate) location to the data obtained from the M-OBJs to which it is connected. The M-Hub extends the ContextNet with IoT access to smart devices (with WPAN interfaces) [26].

It is possible to use M-Hub Android application package (figure A.4) that is provided by the developers of the middleware in the Mobile Hub project page [26], which allows the user to easily start using it and enable your device to collect M-OBJ data. With minimal and simple parameter configuration, such as the IP address and the port of a Gateway that should be connected (as seen in the first image of figure A.4), any data from all nearby sensors of IoT devices can be sent to the server through ContextNet's SDDL. The user can also configure other parameters, for instance: a Bluetooth LE strength threshold, time intervals between BLE scans, and enabling or disabling of specific services (as shown in the top images of figure A.4). Data collected can be visualized in the Mobile-Hub app's (as displayed in the bottom images of figure A.4). It is important to note that the M-Hub application requires Android 4.3 version or newer to work [26].

In this project, where the functionality of scanning surroundings of the mobile device in order to find sensors and actuators is a part of the gameplay, we wanted to let the player decide when to start and stop searching for M-OBJs, as opposed to enable the scanning throughout the whole execution of the game. This user-driven device search we took not only fits within the context of games, but also saves battery of the user device, as it not only prevents the device to be scanning for M-OBJs all the time, but it also removes the necessity of having the BLE interface turned on while playing the game, even when is not needed. The result is that when the player wants to scan, the Bluetooth will automatically be enabled, and when the user wants to stop, the Bluetooth will be disabled.

Thus, we deviated from using the original Mobile Hub application, as it it controls the scanning with its own on and off button. Instead, we wanted to control the scanning from inside the game, facilitating the scanning for the player and not having to switch between mobile applications. To achieve that we used The Mobile Hub API, that provides all the existing services that



Start at Boot ☐
GATEWAY PARAMETERS
IP
139.82.100.70
Port
5500
MOBILE OBJECTS PARAMETERS
Auto-Connect ✓
Signal Allowed
-100dBm
SERVICES & MANAGERS PARAMETERS
Energy Manager ☐

SERVICES & MANAGERS PARAMETERS
Energy Manager ☐
MEPA Service ✓
Location Service ☐
Latitude
null
Longitude
null
MESSAGE PARAMETERS (MS)
Interval [CURRENT]
0
Level HIGH
30000

null
Level HIGH
60000
Level MEDIUM
240000
Level LOW
600000
SCAN PARAMETERS (MS)
Interval [CURRENT]
3000
Level HIGH
20000
Level MEDIUM
40000
Level LOW
60000

Latitude: -22.9814226
Longitude: -43.2340973
Barometer: 100661,42
Gyroscope: 1,43 | -2,80 | -0,04
Humidity: 55,00
Magnetometer: -8,06 | -6,93 | 9,34
Temperature: 24,41 | 22,34
1-984FEE0344B0
RSSI: --
Humidity: 36,00
MQ3: 35,00
MQ4: 9,00
MQ6: 122,00
MQ7: 213,00
Temperature: 26,00

EventData [label=OPBAcc, data={"value":1.0019602832548011}]

Figure A.4: Mobile Hub application

compose M-Hub. We used the service called S2PA, which is responsible for monitoring and discovering M-OBJs.

Once you have the Mobile Hub API [26] in your project, you will need to be able to use the S2PA service of M-Hub. Thus, you must declare it in the Android manifest file as shown in figure A.5. For guidance, the complete Android manifest file used in this project is available in the appendix section.

```
<!-- Begin Services -->

<service android:name="br.pucrio.inf.lac.mhub.services.S2PAService" />
<service android:name="br.pucrio.inf.lac.mhub.services.ConnectionService" />
<service android:name="br.pucrio.inf.lac.mhub.services.LocationService" />
<service android:name="br.pucrio.inf.lac.mhub.services.AdaptationService" />
<service android:name="br.pucrio.inf.lac.mhub.services.MEPAService" />

<!-- End Services -->
```

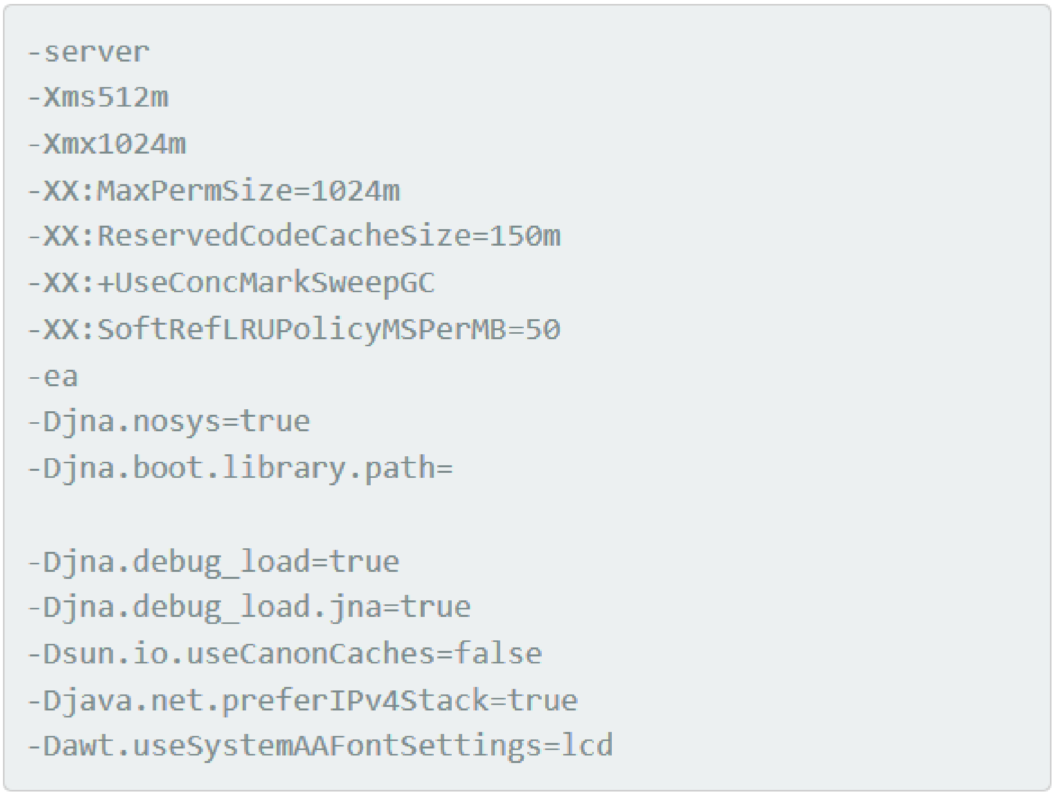
Figure A.5: Android manifest file services declaration

Each declaration in figure A.5 refers to a service present in the Mobile

Hub API. The name that references each service is nothing more than the package path concatenated with the service class name.

The S2PA service will be in charge for discovering and for monitoring data from the M-OBJs, and will start to do so at the moment it is instantiated, but the data retrieved will not be automatically sent to the server, since this is not part of the S2PA service job. In this project, we used our already implemented client-server communication that was created through SDDL to send any data retrieved via the S2PA service data. For that to work, an interface between the Android native code and the LibGDX framework had to be developed. The details of this interface as well as the implementation of this process is discussed in chapter 5.

Finally, the Mobile Hub will, in some cases, require some configurations for the IDE used. In our case, the Android Studio studio.vmptions file should mirror the parameters shown in figure A.6.



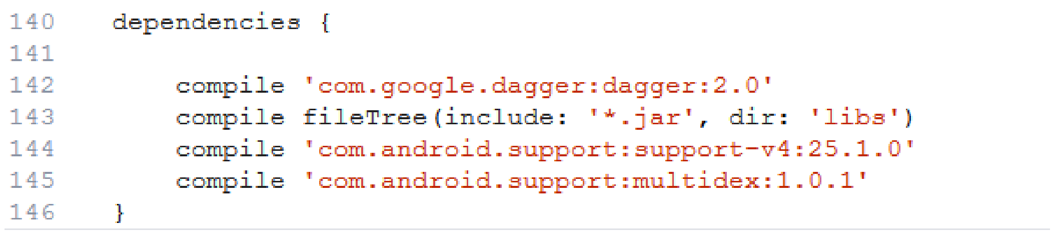
```
-server
-Xms512m
-Xmx1024m
-XX:MaxPermSize=1024m
-XX:ReservedCodeCacheSize=150m
-XX:+UseConcMarkSweepGC
-XX:SoftRefLRUPolicyMSPerMB=50
-ea
-Djna.nosys=true
-Djna.boot.library.path=

-Djna.debug_load=true
-Djna.debug_load.jna=true
-Dsun.io.useCanonCaches=false
-Djava.net.preferIPv4Stack=true
-Dawt.useSystemAAFontSettings=lcd
```

Figure A.6: Android Studio studio.vmptions file

A.4 Gradle Setup

LibGDX framework generated project creates a Gradle-ready project, so gradle can be used to build the project in different IDEs. Following the same protocol, we included any external libraries needed to run the project in the Gradle build, including ContextNet's JAR file. It is also possible to use dependencies that are stored in remote repositories, as illustrated in figure A.7.



```
dependencies {

    compile 'com.google.dagger:dagger:2.0'
    compile fileTree(include: '*.jar', dir: 'libs')
    compile 'com.android.support:support-v4:25.1.0'
    compile 'com.android.support:multidex:1.0.1'
}
```

Figure A.7: Gradle local and remote dependencies

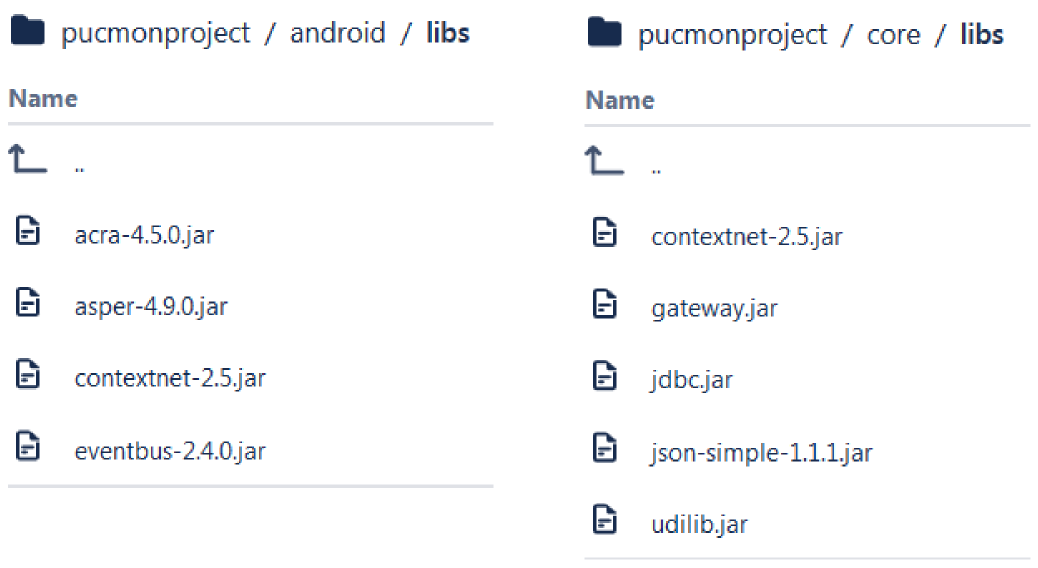
Figure A.7 shows the including of three different remote dependencies, but also shows how to compile local dependencies through the parameter "fileTree" which in this case includes the compilation of any JAR file stored in the local libs directory. The problem is that the generated LibGDX project consists of two separated sub-projects, one that works as a normal Android project that is named "android", and another one where the framework will work, named "core". To build these sub-projects as one bigger project, the Gradle setup ends up getting a little bit more complicated, resulting in a project with multiple Gradle build files and libs folders.

To guide on where each local dependency should be stored, as well as in which Gradle build file should the dependencies be declared, LibGDX provides an extensive document detailing dependency management [47]. Even with the provided guide, it is not trivial to make the LibGDX framework and ContextNet middleware to work all together, so in order to assist future projects we provide in the appendix section the fully working three gradle files, as well as displaying in the figure (A.8) the disposition of the libs folder and the necessary libraries.

One last remark that should be made about Gradle is that the whole project was based on an older version of Gradle, in consequence of the LibGDX generated project Gradle version. If the newer version is used, some keywords and structures may vary, so a compatibility work may be needed.

A.5 Database Setup

To manage the server database, we chose to use phpMyAdmin. It is a free software tool written in PHP, intended to handle the management of MySQL databases over the Web. phpMyAdmin supports a wide range of operations on MySQL. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc) can be performed via the user



pucmonproject / android / **libs**

Name
..
acra-4.5.0.jar
asper-4.9.0.jar
contextnet-2.5.jar
eventbus-2.4.0.jar

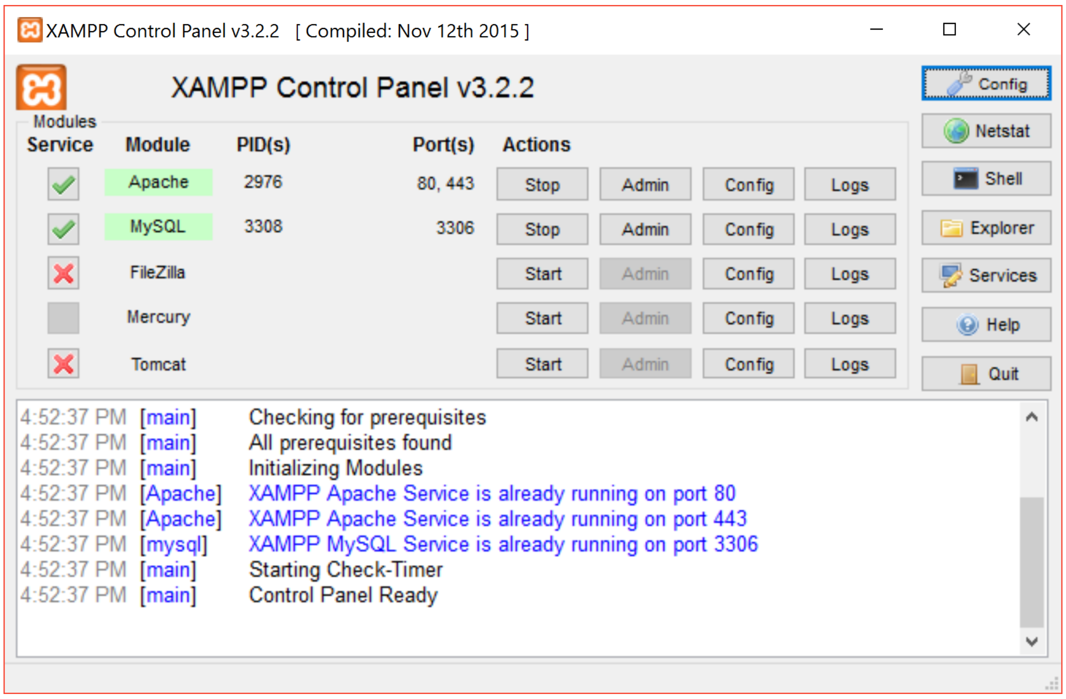
pucmonproject / core / **libs**

Name
..
contextnet-2.5.jar
gateway.jar
jdbc.jar
json-simple-1.1.1.jar
udilib.jar

Figure A.8: Android libs folder (left) and core libs folder (right)

interface, while one still has the ability to directly execute any SQL statement [48].

It is usually ranked as one of the best among MySQL administration tool [49], and is our go-to option for database administration. We used XAMPP as web server (figure A.9), which comes with phpMyAdmin bundled in the installation for database administration.


XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]
XAMPP Control Panel v3.2.2
Config
Netstat
Shell
Explorer
Services
Help
Quit
Modules

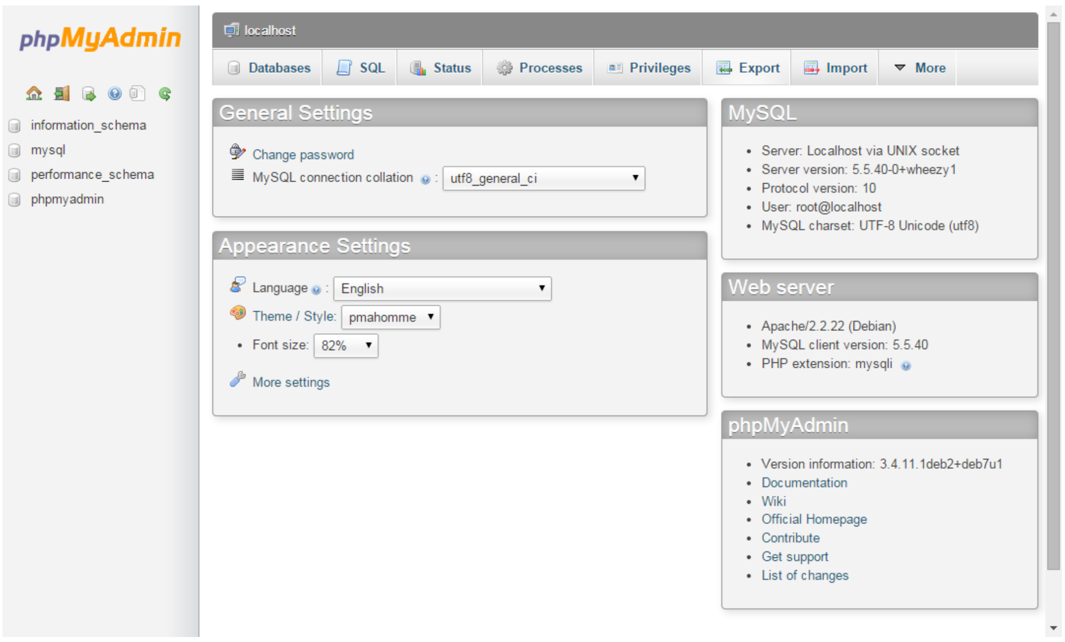
Service	Module	PID(s)	Port(s)	Actions			
✓	Apache	2976	80, 443	Stop	Admin	Config	Logs
✓	MySQL	3308	3306	Stop	Admin	Config	Logs
✗	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
✗	Tomcat			Start	Admin	Config	Logs

```
4:52:37 PM  [main]     Checking for prerequisites
4:52:37 PM  [main]     All prerequisites found
4:52:37 PM  [main]     Initializing Modules
4:52:37 PM  [Apache]   XAMPP Apache Service is already running on port 80
4:52:37 PM  [Apache]   XAMPP Apache Service is already running on port 443
4:52:37 PM  [mysql]    XAMPP MySQL Service is already running on port 3306
4:52:37 PM  [main]     Starting Check-Timer
4:52:37 PM  [main]     Control Panel Ready
```

Figure A.9: XAMPP web server with Apache and MySQL services on

Newer versions of XAMPP comes with MariaDB instead of MySQL, but is possible to replace it with MySQL [50], although we used an older version that comes with MySQL. Once XAMPP is installed, you will only need to start Apach and MySQL services to be able to use and administrate your server database, as shown in figure A.9. You may need to make sure your ports are open for WAN access, but for local and LAN uses you are set up.

Depending if the access is within the server machine, a LAN or a WAN, the phpMyAdmin administration page (figure A.10) must be accessed distinctly, since the IP to access the machine varies. The page URL should be the IP to access the machine concatenated with "/phpmyadmin", so for local access within the machine, the URL would be "http://localhost/phpmyadmin" or "http://127.0.0.1/phpmyadmin". It is also strongly advisable to create a password for phpMyAdmin, ensuring the security of the data.



General Settings

- Change password
- MySQL connection collation: utf8_general_ci

Appearance Settings

- Language: English
- Theme / Style: pmahomme
- Font size: 82%
- More settings

MySQL

- Server: Localhost via UNIX socket
- Server version: 5.5.40-0+wheezy1
- Protocol version: 10
- User: root@localhost
- MySQL charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.2.22 (Debian)
- MySQL client version: 5.5.40
- PHP extension: mysqli

phpMyAdmin

- Version information: 3.4.11.1deb2+deb7u1
- Documentation
- Wiki
- Official Homepage
- Contribute
- Get support
- List of changes

Figure A.10: The phpMyAdmin page accessed locally

Naturally, phpMyAdmin allows the creation of multiple databases. Once a database is created, you may create multiple users with different access privileges for it, but we recommend that users with all privileges should be avoided for safety reasons.

To access and use the database created in the project, the following data will be necessary:

- The name of the database created for the project;
- The IP to access the database machine from the game server machine. As it probably is the same machine, 127.0.0.1 or LAN IP should be used;
- The port to access MySQL. Default is 3306, there is no need to change unless the port is being used for other applications;

- The username and the password of a user that has access to the database created for the project.

B
Android Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mygdx.game" >

    <uses-sdk android:minSdkVersion="16"
        android:targetSdkVersion="25" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission
        android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission
        android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission
        android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
        android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
        android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/GdxTheme"
        >

        <activity
            android:name="com.mygdx.game.AndroidLauncher"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            android:configChanges="keyboard|keyboardHidden|orientation|screenSize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- Begin Services -->

        <service
            android:name="br.pucrio.inf.lac.mhub.services.S2PAService"
            />
        <service
            android:name="br.pucrio.inf.lac.mhub.services.ConnectionService"
            />
        <service
            android:name="br.pucrio.inf.lac.mhub.services.LocationService"
            />
        <service
            android:name="br.pucrio.inf.lac.mhub.services.AdaptationService"
            />
        <service
            android:name="br.pucrio.inf.lac.mhub.services.MEPAService"
            />

        <!-- End Services -->

        <!-- Begin Broadcast Receivers -->

        <receiver
            android:name="br.pucrio.inf.lac.mhub.broadcastreceivers.OnBootReceiver"
            >
            <intent-filter>
                <action
                    android:name="android.intent.action.BOOT_COMPLETED"
                    />
            </intent-filter>
        </receiver>

        <receiver
            android:name="br.pucrio.inf.lac.mhub.broadcastreceivers.PowerReceiver"
            >
            <intent-filter>
                <action
                    android:name="android.intent.action.ACTION_POWER_CONNECTED"
                    />
                <action
                    android:name="android.intent.action.ACTION_POWER_DISCONNECTED"
                    />
            </intent-filter>
        </receiver>

        <receiver
            android:name="br.pucrio.inf.lac.mhub.broadcastreceivers.ConnectivityReceiver"
            >
            <intent-filter>
                <action
                    android:name="android.net.conn.CONNECTIVITY_CHANGE"
                    />
            </intent-filter>
        </receiver>

        <receiver
            android:name="br.pucrio.inf.lac.mhub.broadcastreceivers.BatteryReceiver"
            />

        <!-- End Broadcast Receivers -->

    </application>

</manifest>
```

C
Gradle Build Files

C.1
Root Folder Gradle File

```
/************************
*** root/build.gradle ***
************************/

buildscript {


    repositories {
        mavenLocal()
        mavenCentral()
        maven { url
            "https://oss.sonatype.org/content/repositories/snapshots/"
            }
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.3'

    }
}

allprojects {
    apply plugin: "eclipse"
    apply plugin: "idea"

    version = '1.0'
    ext {
        appName = "pucmon"
        gdxVersion = '1.9.6'
        roboVMVersion = '2.3.1'
        box2DLightsVersion = '1.4'
        ashleyVersion = '1.7.0'
        aiVersion = '1.8.0'
    }

    repositories {
        mavenLocal()
        mavenCentral()
        maven { url
            "https://oss.sonatype.org/content/repositories/snapshots/"
            }
        maven { url
            "https://oss.sonatype.org/content/repositories/releases/"
            }
    }
}

project(":android") {
    apply plugin: "android"

    configurations { natives }

    dependencies {
        compile project(":core")
        compile
            "com.badlogicgames.gdx:gdx-backend-android:$gdxVersion"
        natives
            "com.badlogicgames.gdx:gdx-platform:$gdxVersion:natives-armeabi"
        natives
            "com.badlogicgames.gdx:gdx-platform:$gdxVersion:natives-armeabi-v7a"
        natives
            "com.badlogicgames.gdx:gdx-platform:$gdxVersion:natives-arm64-v8a"
        natives
            "com.badlogicgames.gdx:gdx-platform:$gdxVersion:natives-x86"
        natives
            "com.badlogicgames.gdx:gdx-platform:$gdxVersion:natives-x86_64"
        compile "com.badlogicgames.gdx:gdx-box2d:$gdxVersion"
        natives
            "com.badlogicgames.gdx:gdx-box2d-platform:$gdxVersion:natives-armeabi"
        natives
            "com.badlogicgames.gdx:gdx-box2d-platform:$gdxVersion:natives-armeabi-v7a"
        natives
            "com.badlogicgames.gdx:gdx-box2d-platform:$gdxVersion:natives-arm64-v8a"
        natives
            "com.badlogicgames.gdx:gdx-box2d-platform:$gdxVersion:natives-x86"
        natives
            "com.badlogicgames.gdx:gdx-box2d-platform:$gdxVersion:natives-x86_64"

    }
}

project(":core") {
    apply plugin: "java"


    dependencies {
        compile "com.badlogicgames.gdx:gdx:$gdxVersion"
        compile "com.badlogicgames.gdx:gdx-box2d:$gdxVersion"
        compile fileTree(dir: 'libs', include: '*.jar')

    }
}

tasks.eclipse.doLast {
    delete ".project"
}
```

C.2 Android Folder Gradle File

```
/********************************
*** root/android/build.gradle ***
*********************************/

android {
    buildToolsVersion "26.0.1"
    compileSdkVersion 25
    sourceSets {
        main {
            manifest.srcFile 'AndroidManifest.xml'
            java.srcDirs = ['src']
            aidl.srcDirs = ['src']
            renderscript.srcDirs = ['src']
            res.srcDirs = ['res']
            assets.srcDirs = ['assets']
            jniLibs.srcDirs = ['libs']
        }

        instrumentTest.setRoot('tests')
    }
    packagingOptions {
        exclude 'META-INF/robovm/ios/robovm.xml'
    }
    defaultConfig {
        applicationId "com.mygdx.game"
        minSdkVersion 16
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        // Enabling multidex support.
        multiDexEnabled true
    }
    dexOptions {
        jumboMode true
        javaMaxHeapSize "4g"
    }
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles
                getDefaultProguardFile('proguard-android.txt'),
                'proguard-rules.pro'
        }
    }
}

// called every time gradle gets executed, takes the native
    dependencies of
// the natives configuration, and extracts them to the proper libs/
    folders
// so they get packed with the APK.
task copyAndroidNatives() {
    file("libs/armeabi/").mkdirs();
    file("libs/armeabi-v7a/").mkdirs();
    file("libs/arm64-v8a/").mkdirs();
    file("libs/x86_64/").mkdirs();
    file("libs/x86/").mkdirs();

    configurations.natives.files.each { jar ->
        def outputDir = null
        if (jar.name.endsWith("natives-arm64-v8a.jar")) outputDir =
            file("libs/arm64-v8a")
        if (jar.name.endsWith("natives-armeabi-v7a.jar")) outputDir =
            file("libs/armeabi-v7a")
        if (jar.name.endsWith("natives-armeabi.jar")) outputDir =
            file("libs/armeabi")
        if (jar.name.endsWith("natives-x86_64.jar")) outputDir =
            file("libs/x86_64")
        if (jar.name.endsWith("natives-x86.jar")) outputDir =
            file("libs/x86")
        if (outputDir != null) {
            copy {
                from zipTree(jar)
                into outputDir
                include "*.so"
            }
        }
    }
}
task run(type: Exec) {
    def path
    def localProperties = project.file("../local.properties")
    if (localProperties.exists()) {
        Properties properties = new Properties()
        localProperties.withInputStream { instr ->
            properties.load(instr)
        }
        def sdkDir = properties.getProperty('sdk.dir')
        if (sdkDir) {
            path = sdkDir
        } else {
            path = "$System.env.ANDROID_HOME"
        }
    } else {
        path = "$System.env.ANDROID_HOME"
    }

    def adb = path + "/platform-tools/adb"
    commandLine "$adb", 'shell', 'am', 'start', '-n',
        'com.mygdx.game/com.mygdx.game.AndroidLauncher'
}
// sets up the Android Eclipse project, using the old Ant based
    build.
eclipse {
    // need to specify Java source sets explicitly, SpringSource
        Gradle Eclipse plugin
    // ignores any nodes added in classpath.file.withXml
    sourceSets {
        main {
            java.srcDirs "src", 'gen'
        }
    }

    jdt {
        sourceCompatibility = 1.6
        targetCompatibility = 1.6
    }

    classpath {
        plusConfigurations += [project.configurations.compile]
        containers 'com.android.ide.eclipse.adt.ANDROID_FRAMEWORK',
            'com.android.ide.eclipse.adt.LIBRARIES'
    }

    project {
        name = appName + "-android"
        natures 'com.android.ide.eclipse.adt.AndroidNature'
        buildCommands.clear();
        buildCommand
            "com.android.ide.eclipse.adt.ResourceManagerBuilder"
        buildCommand "com.android.ide.eclipse.adt.PreCompilerBuilder"
        buildCommand "org.eclipse.jdt.core.javabuilder"
        buildCommand "com.android.ide.eclipse.adt.ApkBuilder"
    }
```

```
}
// sets up the Android Idea project, using the old Ant based build.
idea {
    module {
        sourceDirs += file("src");
        scopes = [COMPILE: [plus: [project.configurations.compile]]]

        iml {
            withXml {
                def node = it.asNode()
                def builder = NodeBuilder.newInstance();
                builder.current = node;
                builder.component(name: "FacetManager") {
                    facet(type: "android", name: "Android") {
                        configuration {
                            option(name: "UPDATE_PROPERTY_FILES",
                                value: "true")
                        }
                    }
                }
            }
        }
    }
}
dependencies {

    compile 'com.google.dagger:dagger:2.0'
    compile fileTree(include: '*.jar', dir: 'libs')
    compile 'com.android.support:support-v4:25.1.0'
    compile 'com.android.support:multidex:1.0.1'
}
```

C.3 Core Folder Gradle File

```
/*****************************
*** root/core/build.gradle ***
******************************/

apply plugin: "java"
```

```
sourceCompatibility = 1.6
[compileJava, compileTestJava]*.options*.encoding = 'UTF-8'
sourceSets.main.java.srcDirs = ["src/"]
eclipse.project {
    name = appName + "-core"
}
dependencies {

}
```

D
SQL Schema File

```
DROP TABLE IF EXISTS `player_equipments`;
DROP TABLE IF EXISTS `player_items`;
DROP TABLE IF EXISTS `players`;
DROP TABLE IF EXISTS `accounts`;
DROP TABLE IF EXISTS `market`;
DROP TABLE IF EXISTS `mission_storage`;
DROP TABLE IF EXISTS `sensors`;
DROP TABLE IF EXISTS `sensors_input`;

CREATE TABLE `accounts`
(
   `id` INT NOT NULL AUTO_INCREMENT,
   `name` VARCHAR(32) NOT NULL DEFAULT '',
   `password` VARCHAR(255) NOT NULL/* VARCHAR(32) NOT NULL COMMENT
       'MD5'*//* VARCHAR(40) NOT NULL COMMENT 'SHA1'*/,
   PRIMARY KEY (`id`), UNIQUE (`name`)
) ENGINE = InnoDB;

INSERT INTO `accounts` VALUES (1, '1', '1');

CREATE TABLE `players`
(
   `id` INT NOT NULL AUTO_INCREMENT,
   `name` VARCHAR(255) NOT NULL,
   `world_map` INT NOT NULL DEFAULT 0,
   `account_id` INT NOT NULL DEFAULT 0,
   `level` INT NOT NULL DEFAULT 1,
   `experience` BIGINT NOT NULL DEFAULT 0,
   `posx` INT NOT NULL DEFAULT 0,
   `posy` INT NOT NULL DEFAULT 0,
   `gold` BIGINT NOT NULL DEFAULT 0,
   `first_login` BOOLEAN NOT NULL DEFAULT TRUE,
   `online` BOOLEAN NOT NULL DEFAULT FALSE,
   PRIMARY KEY (‘id‘), UNIQUE (‘name‘),
   KEY (‘account_id‘),
   FOREIGN KEY (‘account_id‘) REFERENCES ‘accounts‘(‘id‘) ON DELETE
      CASCADE
) ENGINE = InnoDB;

INSERT INTO ‘players‘ VALUES (1, 'Jotun', 0, 1, 1, 0, 0, 0, 0, true,
   false);

CREATE TABLE ‘player_items‘
(
   ‘id‘ INT NOT NULL AUTO_INCREMENT,
   ‘uid‘ INT NOT NULL,
   ‘player_id‘ INT NOT NULL DEFAULT 0,
   ‘level‘ INT NOT NULL DEFAULT 0,
   ‘page‘ INT NOT NULL DEFAULT 0,
   ‘idxi‘ INT NOT NULL DEFAULT 0,
   ‘idxj‘ INT NOT NULL DEFAULT 0,
   PRIMARY KEY (‘id‘),
   KEY (‘player_id‘),
   FOREIGN KEY (‘player_id‘) REFERENCES ‘players‘(‘id‘) ON DELETE
      CASCADE
) ENGINE = InnoDB;

INSERT INTO ‘player_items‘ VALUES (1, 3000, 1, 1, 1, 0, 0);

CREATE TABLE ‘player_equipments‘
(
   ‘id‘ INT NOT NULL AUTO_INCREMENT,
   ‘uid‘ INT NOT NULL,
   ‘player_id‘ INT NOT NULL DEFAULT 0,
   ‘level‘ INT NOT NULL DEFAULT 0,
   ‘slot‘ INT NOT NULL DEFAULT 0,
   PRIMARY KEY (‘id‘),
   KEY (‘player_id‘),
   FOREIGN KEY (‘player_id‘) REFERENCES ‘players‘(‘id‘) ON DELETE
      CASCADE
) ENGINE = InnoDB;

INSERT INTO ‘player_equipments‘ VALUES (1, 3000, 1, 1, 1);
```

```
CREATE TABLE ‘market‘
(
   ‘id‘ INT NOT NULL AUTO_INCREMENT,
   ‘uid‘ INT NOT NULL,
   ‘player_id‘ INT NOT NULL DEFAULT 0,
   ‘level‘ INT NOT NULL DEFAULT 0,
   ‘price‘ BIGINT NOT NULL DEFAULT 0,
   ‘quality‘ INT NOT NULL DEFAULT 0,
   ‘sold‘ BOOLEAN NOT NULL DEFAULT FALSE,
   PRIMARY KEY (‘id‘),
   KEY (‘player_id‘),
   FOREIGN KEY (‘player_id‘) REFERENCES ‘players‘(‘id‘) ON DELETE
       CASCADE
) ENGINE = InnoDB;

INSERT INTO ‘market‘ VALUES (1, 7000, 1, 10, 520, 3, false);

CREATE TABLE ‘mission_storage‘
(
   ‘mission_id‘ INT NOT NULL DEFAULT 0,
   ‘player_id‘ INT NOT NULL DEFAULT 0,
   ‘timestamp‘ timestamp NOT NULL DEFAULT now(),
   PRIMARY KEY (‘mission_id‘, ‘player_id‘),
   KEY (‘player_id‘),
   FOREIGN KEY (‘player_id‘) REFERENCES ‘players‘(‘id‘) ON DELETE
       CASCADE
) ENGINE = InnoDB;

INSERT INTO ‘mission_storage‘ VALUES (0, 1, NULL);

CREATE TABLE ‘sensors‘
(
   ‘sensor_id‘ VARCHAR(255) NOT NULL,
   ‘sensor_type‘ VARCHAR(255) NOT NULL DEFAULT ’mission’,
   ‘mission_id‘ INT,
   ‘thumbnail_id‘ INT,
   ‘code‘ VARCHAR(255),
   ‘n_inputs‘ INT NOT NULL DEFAULT 0,
   ‘input_type‘ VARCHAR(255),
   PRIMARY KEY (‘sensor_id‘)
) ENGINE = InnoDB;
```

```
INSERT INTO `sensors` VALUES ('1-F45EAB2755CC', 'mission', 0, 0,
   'RGBR', 0, NULL);
INSERT INTO `sensors` VALUES ('1-34B1F7D508B2', 'mission', 1, 1,
   NULL, 2, 'integer') ;

CREATE TABLE `sensors_input`
(
  `id` INT NOT NULL AUTO_INCREMENT,
  `sensor_id` VARCHAR(255) NOT NULL,
  `input_type` VARCHAR(255) NOT NULL,
  `input_name` VARCHAR(255) NOT NULL,
  `input_data` VARCHAR(255) NOT NULL,
  `timestamp` timestamp NOT NULL DEFAULT now(),
  PRIMARY KEY (`id`),
  KEY (`sensor_id`),
  FOREIGN KEY (`sensor_id`) REFERENCES `sensors`(`sensor_id`) ON
     DELETE CASCADE
) ENGINE = InnoDB;
```

E
Data Source Factory

```java
/**
 * Factory of MySqlDataSource object
 * that communicates with server database
 *
 * @author Pedro Sampaio
 * @since 1.4
 */
public class DataSourceFactory {

    public static DataSource getMySQLDataSource() {
        Properties props = new Properties();
        FileInputStream fis = null;
        MysqlDataSource mysqlDS = null;
        String path = "sql/db.properties";
        try {
            fis = new FileInputStream(path);
            props.load(fis);
            mysqlDS = new MysqlDataSource();
            mysqlDS.setURL(props.getProperty("MYSQL_DB_URL"));
            mysqlDS.setUser(props.getProperty("MYSQL_DB_USERNAME"));
            mysqlDS.setPassword(props.getProperty("MYSQL_DB_PASSWORD"));
        } catch (IOException e) {
            System.err.println("Could not find db properties file:
                "+path);
            e.printStackTrace();
        }
        return mysqlDS;
    }
}
```