NeoMAD at a glance for developers

Edition date: November 2014

Neomades SAS
Technopole IZARBEL - Hôtel d'Entreprises
45 allée Théodore Monod
64210 - BIDART - FRANCE

Phone : 00 33 (0)5 59 43 85 21
Fax : 00 33 (0)5 59 43 84 05
Web : http://www.neomades.com
Email : contact@neomades.com

Table of contents

1 What is NeoMAD? 4
1.1 Who is NeoMAD for? 4
1.2 What result can you expect from NeoMAD? 4
1.3 Benefits of using NeoMAD 4
1.4 Cross-platform mobile application development 5
2 Using NeoMAD 7
2.1 Structure of a NeoMAD project 7
2.1.1 The project description file (URS) 7
2.1.2 The source code 8
2.1.3 Resources 8
2.2 NeoMAD Generic API 8
2.3 Implementing conditional code 9
2.3.1 Declaring constants 9
2.3.1.1 TargetInfo 9
2.3.1.2 Constants.java 9
2.3.2 Using constants 9
2.3.2.1 In the source code 9
2.3.2.2 For resource declarations 10
2.3.3 Compilation process 10
2.4 Implementing platform specific code 10
2.5 Debugging an application 11
3 Application development sample 12
3.1 The entry point 12
3.2 Switching between screens 12
3.3 Building the UI 12
3.3.1 Defining the UI with Java code 12
3.3.2 Defining the UI in a XML file 13
3.4 Results 14
4 Existing application screenshots 16
4.1 PropertyCross 16
4.2 GCT© Mobility by Gfi Chrono Time 17
4.3 Smartscales by L'Oréal 18
4.4 TNS On The Go by TNS Sofres 19
4.5 CROUS by CNOUS 20
4.6 Nouvelle Chance pour l'Orientation 21

1 What is NeoMAD?

Since 2006 NeoMAD has been bringing solutions to developers wishing to create portable mobile and tablet applications from a single project. NeoMAD Version 3, released in mid 2012, has achieved a higher level of simplicity and portability in a market which is constantly evolving and becoming increasingly complex.

1.1 Who is NeoMAD for?

NeoMAD targets mobile application developers who want to address the whole equipments of the mobility market without learning every mobile platform SDK and language.

The only skill required to develop cross-platform mobile applications with NeoMAD is Java programming. With this programming language and a single Generic API provided by NeoMAD, developers can quickly develop native applications for all mobile platforms (Mobile Phones, Tablets, Setup Boxes, etc.).

NeoMAD is installed on the developer's computer (Windows or Mac OS) and can be used in command line or in Eclipse thanks to a plug-in provided by Neomades.

1.2 What result can you expect from NeoMAD?

NeoMAD helps you to extend your market coverage, to reduce the TTM of your mobile projects and to improve your ROI:

A single source code, multiple targets:

NeoMAD enables you to target all mobile technologies and formats with a single development – you centralise development and maintenance, no longer having to decide on which targets you are aiming at, although you can do so according to how your customer base evolves.

An industrial approach:

NeoMAD enables you to industrialise your mobile developments, to organise your team more efficiently and to reduce your costs: maintenance is centralised and shared for all targets; production tools provide greater traceability of versions and the reproducibility of incidents; finally, creating a clone of applications is simplified thanks to conditioning tools.

Simplified and unified skills:

Thanks to NeoMAD, you control all the skills and uses for each of the devices targetted. This way, you make it easier to share experience and re-use knowledge between projects. You also simplify the transfer of skills in long term projects.

Future proof your mobile assets:

NeoMAD's abstraction API future proofs your application investments, extending your applications' life even when new devices and technologies appear on the market. Adapt to the market by clearly separating development from distribution.

1.3 Benefits of using NeoMAD

NeoMAD allows you to do the following:

- reduce code complexity while increasing developer productivity and optimizing development phases
- write source code that is portable with all supported devices, including devices added with future versions
- easily manage the resources used in the project (images, sounds, text, etc.)
- merge manufacturer development kits for simulation and testing purposes
- quickly and easily support new phones arriving on the market

In addition, NeoMAD improves the quality of the developed software and process:

- by industrialising work processes and production
- by simplifying development: device characteristics are no longer the concern of developers
- by combining portability and re-usability: development becomes independent of existing and future devices (future-proofing)
- by making the maintenance process easier: a single source and a complete integration with all SCM tools
- by automating the generation process and thus its reproducibility.

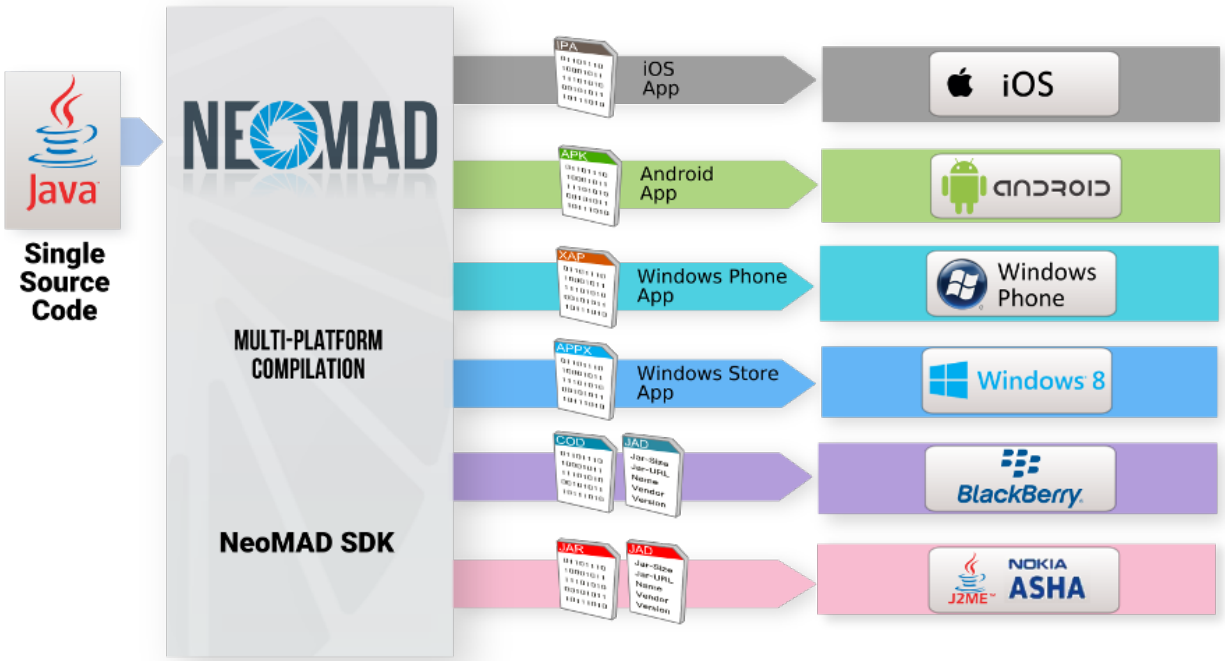
1.4 Cross-platform mobile application development

NeoMAD is a revolution for the development of applications for mobile platforms:

- Based on a single Java source code, NeoMAD can address multiple technologies natively
- Using a conditional process filled dynamically by a knowledge database, NeoMAD allows various functional branches to be implemented in order to optimally address each technology and mobile platform
- With its trans-compilation mechanism, NeoMAD ensures the scalability of an application on current and future technologies

With the NeoMAD Generic API you can cover market needs in terms of programming languages and technologies. With a single source code you can:

- Create User Interfaces according to devices guidelines and good practices. The Generic API relies on natives API of each target; thus, the applications produced with NeoMAD will have a native look and feel.
- Easily access to standardised web services (like REST, SOAP, etc.).
- Access to many devices sensors (compas, accelerometer, GPS, etc.) and handling their data in a unified process.
- Handle push and locale notifications in a generic way; NeoMAD can use any push SDK and is already embeding You N'Push and Parse SDKs.
- Speak with newest connected devices; NeoMAD already support iBeacon technology.
- Access to native map APIs of each platform.
- Access to file system and store data in SQLite databases.
- Use third-party libraries like payment modules, electronic signatures, pdf annotations, etc.

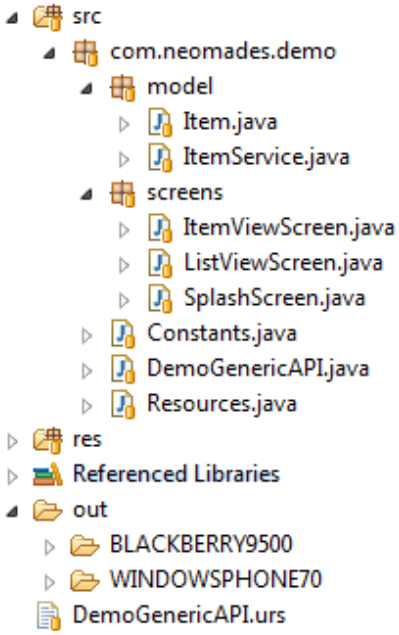


2 Using NeoMAD

NeoMAD provides plugins for the Eclipse IDE and thus enables mobile projects to be developed in a very assisted and traditional way for Java developers.

2.1 Structure of a NeoMAD project

A NeoMAD project is similar to any Java project, with Java source files organized in packages. The project parameters and resources are defined in an XML project description file (.urs).



2.1.1 The project description file (URS)

The URS file contains project information and parameters, resource declarations, external libraries to be included, output folders where binaries and native source code will be generated, etc.

A NeoMAD project contains at least three types of folders:

- **src**: source folder containing Java files
- **res**: resource folder
- **out**: output folder for binaries

```
<?xml version="1.0"?>
<urs xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
  "http://www.neomades.com/XSD/3.0/urs.xsd">
    <parameters>
        <mainclassname>HelloWorld</mainclassname>
        <applicationname>HelloWorld</applicationname>
        <packagename>com.neomades.helloworld</packagename>
        <vendor>Neomades</vendor>
        <description>HelloWord application example</description>
        <icon path="res/icon.png" />
        <version>1.0.0</version>
        <srcpath>src</srcpath>
        <outputpath>out</outputpath>
    </parameters>
```

```
    <strings path="res/strings.csv" />
</urs>
```

2.1.2 The source code

The source code of a NeoMAD project is fully written in Java and uses API provided by NeoMAD for portable application development. Some parts of the source code can be conditioned using Java constants in order to produce versions of the applications with different behaviors. This conditioning mechanism can also be used to add native source code in the NeoMAD application code in order to add features specific to a platform.

2.1.3 Resources

A Resources.java file is generated by the NeoMAD tool in order to provide resource identifiers in the Java code.

E.g.

a text string is declared in the CSVfile “TXT_HELLOWORLD;Hello World;Bonjour le monde”.

NeoMAD will generate a constant named TXT_HELLOWORLD in Resources.java. The developer can use this source code to access the text value for “Hello world” in the selected language.

2.2 NeoMAD Generic API

The NeoMAD Generic API is a library provided with NeoMAD that allows developers to produce native looking mobile applications. It is the main entry point for cross-platform mobile application development with NeoMAD. Whatever the programming language, OS or technical characteristics of the targeted devices, developers can quickly develop applications that will run on them with a single Java source code and the NeoMAD generation tool. The three main components of the NeoMAD Generic API are:

- **An application structure**
    - Life cycle (start, interruptions, stop)
    - Screen sequence
- **A UI kit**
    - Native looking graphical components (button, textfield, etc.)
    - Theme manager
- **A technical kit**
    - HTTP/HTTPS communications
    - Browser launch, text message, voice call, vibration
    - Local storage
    - XML, JSON, etc.

Each component of the Generic API corresponds to a native component for each platform.

E.g.

Buttons correspond to ButtonField (BlackBerry), Button (Android, Windows Phone)

2.3 Implementing conditional code

The conditioning process consists of declaring constants and using them to vary parts of the source code or the declaration of resources. These constants will be dynamically valued during the generation process.

2.3.1 Declaring constants

A condition is based on a constant defined and provided by NeoMAD or by the developer of the application. Thanks to this mechanism you can generate different binaries from a single source code in order to cover several functional cases (using a feature that is not available on all devices, phones / tablets variations, re-branding, etc).

2.3.1.1 TargetInfo

TargetInfo is a Java Interface provided by NeoMAD that defines constants relating to device characteristics (OS, name, screen size, keyboard type, etc). During the NeoMAD build process this interface automatically declares different values for each target.

E.g.

TargetInfo.TARGET_NAME gives the name of the device during compilation

2.3.1.2 Constants.java

Constants is a Java class written by the application developer. The developer can declare constants to enable or disable certain features of the mobile application according to external conditions, adapt part of the application to specific requests for particular platforms, etc. The Constants class must implement TargetInfo. Here is an example of a Constants class defining a FEATURE_ADS constant used to activate advertising on specific versions.

```
/**
* TargetInfo is an interface with phone constants values
*
* (each phone has specific values but same list of constants)
*/
public class Constants implements TargetInfo {
    /** The ANDROID, IOS, BLACKBERRY and TOUCH_SCREEN constants
    are provided by NeoMAD in the TargetInfo interface */
    public static boolean FEATURE_ADS
        = ANDROID | IOS | (BLACKBERRY & TOUCH_SCREEN);
}
```

All constants from Constants and TargetInfo can be used as a condition for source code or resource declarations.

2.3.2 Using constants

2.3.2.1 In the source code

The developer can use constants to condition parts of the application code. In this case, the corresponding code will only be kept if the condition is true.

```
if (Constants.FEATURE_ADS) {
    screen.add(new Ads());
}
```

2.3.2.2 For resource declarations

The developer can condition the declaration of resources in the URS file. In this case, resources will only be added in the application if the condition is true. Constants can also be used to define paths or values. Here NeoMAD substitutes the value of the constants when analysing them.

```
<resource name="IMG_LOGO" path="RES_PATH/img_logo.png" />
    <resource condition="ANDROID"
        name="IMG_ANDROID_ADS"
        path="RES_PATH/img_android_ads.png" />
```

ANDROID and RES_PATH are provided by TargetInfo and Constants respectively.

- TargetInfo.ANDROID is provided by NeoMAD and can be used to condition the declaration of certain resources for the Android platform.
- Constants.RES_PATH is declared by the developer and defines the resource access path.

2.3.3 Compilation process

When compiling applications NeoMAD uses Constants and TargetInfo constant values in order to optimize source code. During this process a block of code with a false condition will be removed and the generated binary will not include this part of the code.

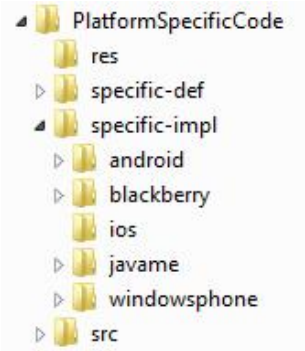
2.4 Implementing platform specific code

NeoMAD comes with an option for platform specific code to be added in a NeoMAD project. In practical terms this option enables source code to be included written in the target platform language (C#, Java, Objective C) in a NeoMAD project to access features that are specific to those platforms (Windows Phone, iOS or Android, for instance). This makes it possible to use a platform specific feature (compass, etc.), to integrate code that has already been written for a specific platform or to access features available on most platforms that are not yet available in the NeoMAD Generic API (camera, file system).

The conditions must observe the target capabilities (NeoMAD will not control this).

To use this option the NeoMAD project must contain two additional folders:

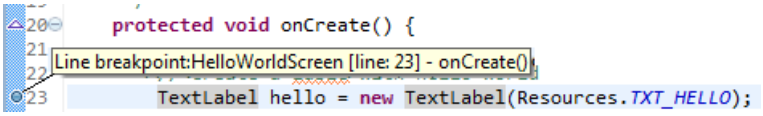
- The specific-def folder contains the Java definition of the specific code. This folder and its content are created by the developer. In order for the non-specific code to interact with the specific code, the developer has to provide a definition of what the specific code will look like. This definition is a classic Java class.
- The specific-impl folder contains the real implementation of the specific code for each platform that will be embedded into the final binary. It contains sub-folders that represent platforms for which specific code will be implemented (such as Android, BlackBerry or Windows Phone). The specific-impl folder and its content are automatically generated by NeoMAD, based on the definition provided by the developer. The generated implementation is the exact image of this definition written in the language of the target platform language. Then the developer can complete the implementation files in order to provide the specific implementation of the feature for each platform.



2.5 Debugging an application

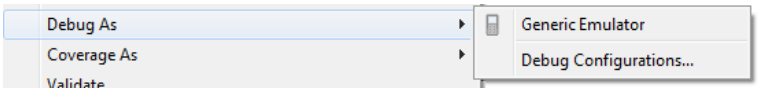
Used in combination with an IDE and an emulator, NeoMAD makes it possible to debug an application. Using the IDE plugins provided with NeoMAD, the developer is able to debug step by step an application launched in an emulator.

With the IDE the developer can add break-points in order to pause the application and follow execution step by step:

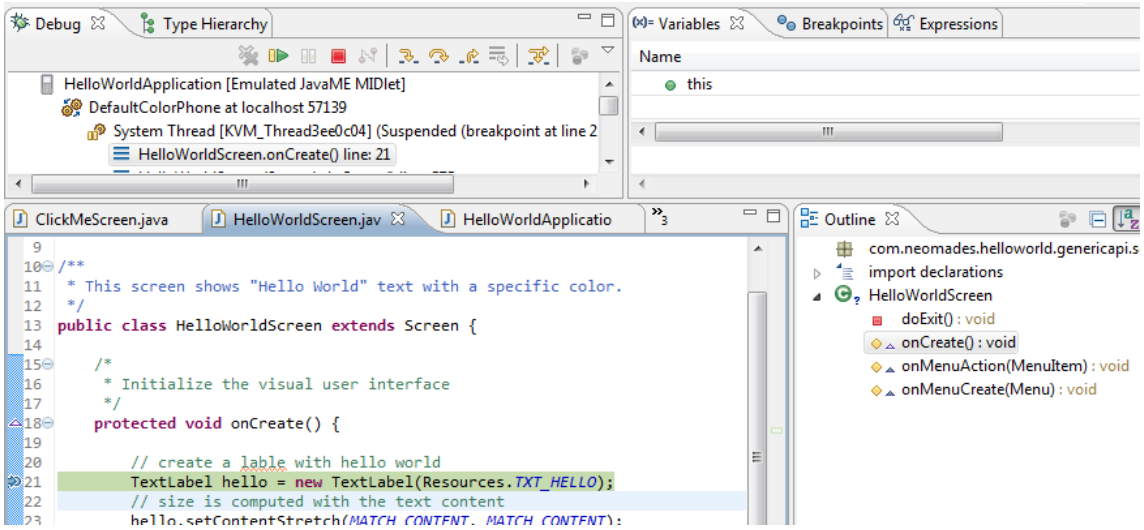


The developer can start debugging the application with the IDE.

Eg. in Eclipse the developer starts debugging with Debug As action



The emulator launches the application and the execution will be paused at the break-point position



3 Application development sample

This section will guide you through the creation of a basic “Hello World” application with the NeoMAD Generic API. The source code of this example can be found in the Examples directory of the NeoMAD installation.

3.1 The entry point

The main class of the application must extend Application. It is the application’s entry point.

```
package com.neomades.helloworld;

import com.neomades.app.Application;
import com.neomades.app.Controller;

public class HelloWorld extends Application {

    public void onStart(Controller controller) {
        // first screen
        controller.pushScreen(HelloWorldScreen.class);
    }
}
```

The onStart() method is overridden to handle what should be done when the application is started. For this application all we have to do is declare the first screen to be displayed. A single Controller instance is used to push and pop screens. Here the pushScreen() method is called to display the first screen (HelloWorldScreen).

3.2 Switching between screens

A single instance of Controller is provided by the Generic API through the Application and Screen classes. Controller offers methods to change the current screen and keeps the screen stack history in order to handle the BACK feature in a native way.

3.3 Building the UI

The NeoMAD Generic API structure imposes one class per screen. Each screen is declared as a class that extends the Screen class. After this the developer must construct the Screen’s UI.

The NeoMAD Generic API provides a complete set of UI elements that can be combined to design rich application screens. However, for this HelloWorld example, we will start with a very simple screen using only a vertical layout and a text label.

There are two ways of defining the UI with NeoMAD: in Java code and in a XML file. Both have the same result, and chosing one or the other is up to developer's own preferences.

3.3.1 Defining the UI with Java code

In order to define the UI with Java code, the only thing to do is to use graphical components of the Generic API directly in the Screen class:

```
package com.neomades.helloworld;

import com.neomades.app.Screen;
import com.neomades.ui.TextLabel;
import com.neomades.ui.VerticalLayout;

/** Application page screen */
```

```
public final class HelloWorldScreen extends Screen {

    protected void onCreate() {
        // Initialize the UI at screen creation

        // *** Main Layout *** //
        // a VerticalLayout aligns its content vertically
        VerticalLayout layout = new VerticalLayout();
        // we want the VerticalLayout to fill all the screen size
        layout.setStretchMode(MATCH_PARENT, MATCH_PARENT);
        // we want the content to be centered
        // into the VerticalLayout
        layout.setContentAlignment(HCENTER | VCENTER);
        // set the VerticalLayout as content for this screen
        setContent(layout);

        // *** Text Label *** //
        // a TextLabel allows to display a simple text
        // on the screen
        TextLabel helloWorldLabel = new TextLabel("Hello World");
        // we want the TextLabel size to match the text
        helloWorldLabel
        .setStretchMode(MATCH_CONTENT, MATCH_CONTENT);
        // add the TextLabel to the VerticalLayout
        layout.addView(helloWorldLabel);
    }
}
```

3.3.2 Defining the UI in a XML file

Several steps are mandatory to define the UI in XML:

- First, a XML file must be created in the ressources directory of the project. This file contains the UI description and must be compliant with the layout.xsd definition delivered by NeoMAD. In our example, this file is called helloworld_screen.xml.

```
<?xml version="1.0"?>
<!-- A VerticalLayout represents a group of views arranged
vertically.
We want the VerticalLayout to fill all the screen size-->
<VerticalLayout
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
        "http://www.neomades.com/XSD/3.2.0/layout.xsd"
    stretchHorizontalMode="MATCH_PARENT"
    stretchVerticalMode="MATCH_PARENT"
    contentAlignment="HCENTER|VCENTER">

    <!-- A TextLabel allows to display a simple text on the
    screen.
    We want the TextLabel size to match the text. -->
    <TextLabel
        stretchHorizontalMode="MATCH_CONTENT"
        stretchVerticalMode="MATCH_CONTENT"
        text="@string/HELLO_WORLD" />

</VerticalLayout>
```

- Then the file must be declared as a ressource in the URS file.

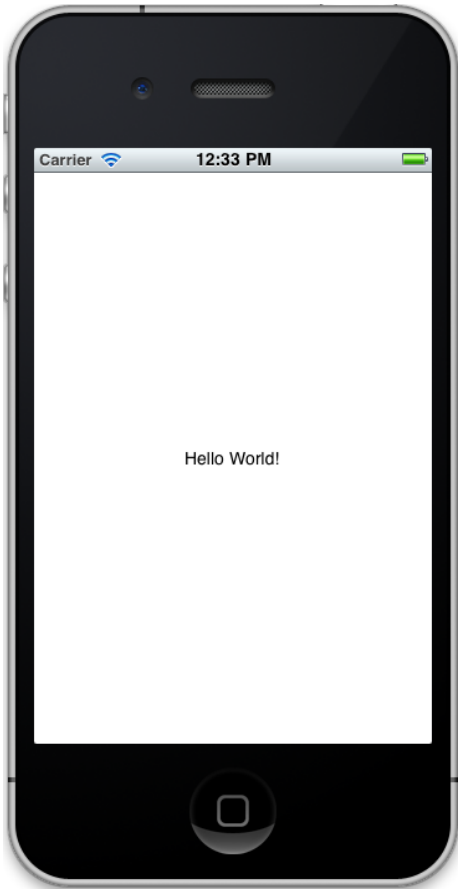
```
<resourcelot>
    <layout
        name="HELLOWORLD_SCREEN"
        path="res/layout/helloworld_screen.xml" />
</resourcelot>
```

- Finaly, the UI must be integrated in the screen content using the ressource id declared in the URS file.

```
package com.neomades.helloworld;

import com.neomades.app.Screen;

/** * Application page screen */
public final class HelloWorldScreen extends Screen {

    protected void onCreate() {
        // Fill the screen with the HELLOWORLD_SCREEN layout
        setContent(Res.layout.HELLOWORLD_SCREEN);
    }
}
```

3.4 Results

Here is what our simple application will look like on Android, BlackBerry, iOS and Windows Phone:



4 Existing application screenshots

This section shows some screenshots of existing applications with NeoMAD.

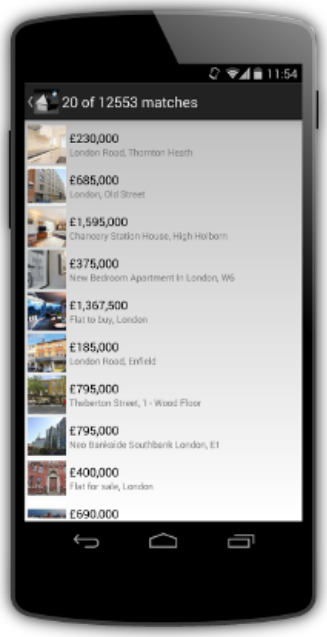
4.1 PropertyCross

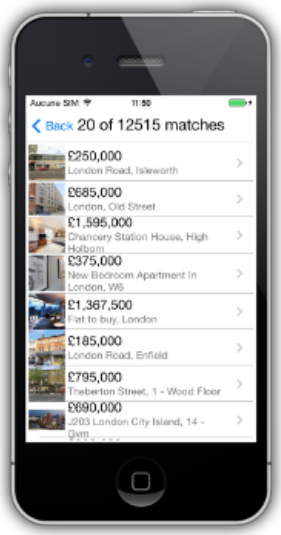
PropertyCross is an initiative by a British group of experts in mobile programming which offers the community a website comparing the different solutions available on the market to develop mobile applications with a cross-platform approach.

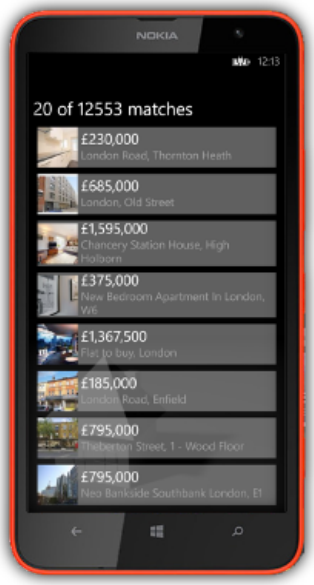
With this aim the website provides a reference application for finding accomodation for rent or buy and a measurement comparer. In order for a solution to be referenced on the site, a developer must create the standard application and make available to the community, on the one hand, the creation's sources, and on the other hand, the resulting executables for iOS, Android and Windows Phone mobiles.

All4Tests has carried out the operation to calibrate NeoMAD and submits elements for comparison which show a 100% identical code and 100% use in compliance with demand.

To access the site http://propertycross.com.







4.2 GCT© Mobility by Gfi Chrono Time

To consolidate its position as a European leader, Gfi Chrono Time has enriched its range of Time Management software packages, and modules available for mobiles and tablets.

Thanks to NeoMAD technology, these new modules can be accessed on all the market's equipment, thus providing users access to databases generated by the software packages used by their companies when in mobile environments.

ChonoGestor's new modules were developed by GCI using NeoMAD V3.

They are available on iOS, Android, Windows Phone and Blackberry mobiles and tablets.

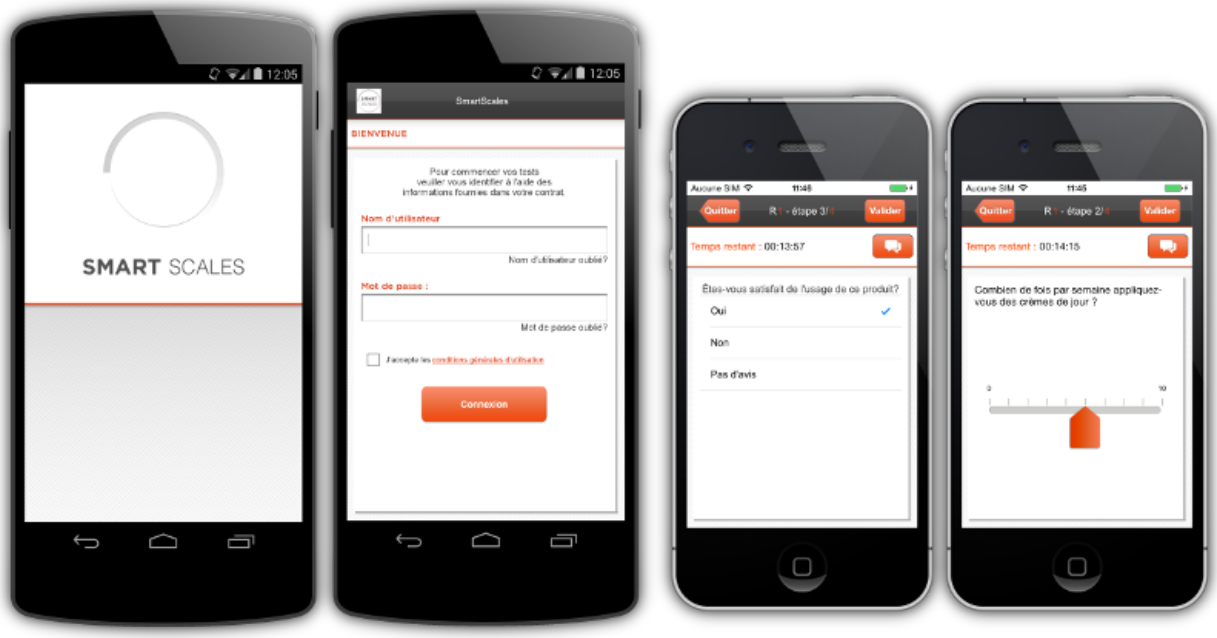
4.3 Smartscales by L'Oréal

The SMARTSCALES application was developed on the request of Oreal's R&D department to manage cosmetic test campaigns.

Thanks to its Back Office, the product manager is able to define all the questions, groups of individuals targetted for testing, the timetable and response schedule.

The application tells testers when it's the right time for them to react to demand, answer questions and provide information, according to the plan determined by the product manager. Once the test has been carried out, the controlled information is sent to the Back Office for aggregation and the application goes into standby mode until the next test. Obviously, the product manager can, based on a partial analysis of the initial results collected, modify the procedure or content of the test as and when required.

This powerful collection system is available for the moment on iOS and Android mobiles and tablets.



4.4 TNS On The Go by TNS Sofres

By publishing TNS' On The GO application, its editor, the marketing department at TNS SOFRES, aims to provide its clients with access to analyses on their mobile equipment as soon as they become available in order to help them react and interact to enrich the debate.

This new tool in TNS SOFRES' PR toolkit enables in-house teams to lead the network of users when appropriate, and to better share and disseminate information produced in different fields of study. Each application used can thus determine the subjects users are passionate about and alert them as soon as new analyses are available, enabling them to react instantly and share their point of view with the community. The implementation of a Back Office developed for this application also enables broadcasts to be pooled by adapting content to different community channels.

This application was created after a series of scaling operations making mobile strategy part of TNS SOFRES' overall PR strategy. It is currently available on both Android and iOS mobiles and tablets.

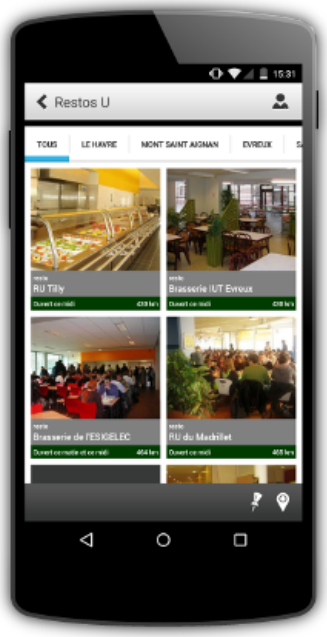
4.5 CROUS by CNOUS

At the head of a national network, the CROUS aims to facilitate student life in many areas of life: eating, accommodation, grants, social and cultural action, international relations, etc.

It was therefore natural for a mobile application to be envisaged to provide access to local information which is esssential to student life: opening times of preferred restaurants, menu of the day, facilities available in different accommodation, performances and entertainment, grants and how to apply, etc.

In addition to this information, the use of mobiles also provides functions for sharing, guiding and searching locally, enabling the student community to use the application as a dedicated tool for everyday life. This application is based on local flows so as to guarantee information is as relevant as possible.

The CNOUS application is available on mobile and tablets using iOS, Android and Windows Phone 8.



4.6 Nouvelle Chance pour l'Orientation

As part of help for initiatives to support economic transformation by the Aquitaine Regional Council, AGEFOS – PME is taking part in a local initiative to support the needs of businesses in the Oloron area in terms of skills.

The ‘‘Local Skills Passport’’ and ‘‘Enhancing Industrial Professions’’ working groups set up to envisage the means to be implemented, decided to carry out an experiment called “New Chance for Careers” with companies and local employment / training partners. The aim of this experiement is to help young people deciding on a career, in training or looking for a first job, discover the 50 most promising jobs in terms of employment prospects in the Aquitaine region.

The experiment is naturally based on a mobile solution, which consists of, on the one hand, an application for mobiles and tablets, and on the other hand an administration back-office enabling employment players to pilot the experiment, its use and users, and finally, resource packs according to the different business sectors, covering on average 10 professions, which can be used dynamically by the application.

This application is currently available on mobiles and tablets using iOS and Android.