NeoMAD en un coup d'oeil pour les développeurs

Date d'édition : mai 2015

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 des matières

1 Qu'est ce que NeoMAD ? 4

1.1 A qui s'adresse NeoMAD ? 4

1.2 Quels résultats attendre de l'utilisation de NeoMAD ? 4

1.3 Pourquoi utiliser NeoMAD ? 4

1.4 Développement cross plateformes pour la mobilité 5

2 Utiliser NeoMAD 7

2.1 Structure d'un projet NeoMAD 7

2.1.1 Le fichier de description du projet (URS) 7

2.1.2 Le code source 8

2.1.3 Les ressources 8

2.2 L'API Générique NeoMAD 8

2.3 Mettre en œuvre du code conditionnel 9

2.3.1 Déclaration de constantes 9

2.3.1.1 TargetInfo 9

2.3.1.2 Constants.java 9

2.3.2 Utiliser les constantes 10

2.3.2.1 Dans le code source 10

2.3.2.2 Pour la déclaration de ressources 10

2.3.3 Le processus de compilation 10

2.4 Utiliser du code spécifique natif pour une cible 10

2.5 Déboguer son application 11

3 Exemple de développement d'application 12

3.1 Le point d'entrée 12

3.2 Passage d'un écran à l'autre 12

3.3 Construction de l'Interface Utilisateur 12

3.3.1 Définition de l'interface utilisateur en code Java 12

3.3.2 Définition de l'interface utilisateur en XML 13

3.4 Résultats 14

4 Screenshots d'applications existantes 16

4.1 PropertyCross 16

4.2 GCT© Mobility par Gfi Chrono Time 17

4.3 Smartscales par L'Oréal 18

4.4 TNS On The Go par TNS Sofres 19

4.5 CROUS par le CNOUS 20

4.6 Nouvelle Chance pour l'Orientation 21

1 Qu'est ce que NeoMAD ?

Depuis 2006, NeoMAD apporte des solutions aux développeurs souhaitant créer des applications mobiles portables sur plusieurs plateformes à partir d'un seul projet. NeoMAD Version 3, publié au milieu de l'année 2012, atteint un niveau supérieur de simplicité et portabilité et permet à partir d'un code unique Java de couvrir l'ensemble des technologies et équipements mobiles d'un marché en permanente évolution et devenant de plus en plus riche et complexe.

1.1 A qui s'adresse NeoMAD ?

NeoMAD est destiné aux éditeurs d'applications mobiles qui veulent couvrir tout ou partie des équipements mobiles du marché, sans avoir à maîtriser les kits de développement correspondants, les langages de programmation induits et les ergonomies sous-jacentes.

La seule compétence requise pour développer des applications mobiles multi plateformes avec NeoMAD, est la programmation en langage Java. En utilisant ce langage de programmation et l'API Générique fournie avec NeoMAD, les développeurs peuvent créer et produire rapidement des applications natives pour toutes les plateformes mobiles (téléphones, tablettes, …).

NeoMAD est installé sur la machine du développeur (Windows ou Mac OS) et peut être utilisé en ligne de commande ou dans un IDE comme Eclipse grâce à un plugin fourni par NeoMAD.

1.2 Quels résultats attendre de l'utilisation de NeoMAD ?

NeoMAD vous permet d'élargir votre couverture et d'accélérer votre présence sur le marché tout en contrôlant votre ROI.

Un code unique, des cibles multiples :

NeoMAD vous permet d'adresser tous les formats et toutes les technologies mobiles à partir d'un développement unique. Vous centralisez vos développements et leur maintenance et vous choisissez les cibles en fonction de l'évolution de votre marché.

Une approche industrielle :

NeoMAD vous permet d'industrialiser vos développements mobiles, d'augmenter l'efficacité de votre organisation et de diminuer les coûts : la maintenance est centralisée et commune pour toutes les cibles ; les outils de production permettent une meilleure traçabilité des versions et le contrôle de la qualité du code ; enfin, créer un clone d'une application est simplifié grâce aux outils de conditionnement.

Des compétences simplifiées et unifiées :

NeoMAD simplifie les compétences nécessaires à votre équipe de développement et permet de couvrir l'ensemble des usages et technologies inhérents aux équipement ciblés. De cette manière, vous optimisez votre équipe, vous facilitez le partage d'expérience, la réutilisation des acquis et le transfert de compétences.

Vos actifs mobiles protégés face à l'évolution du marché :

L'API d'abstraction NeoMAD protège vos investissements applicatifs des évolutions rapides du marché et augmente ainsi la durée de vie de vos applications même lors de l'arrivée de nouveaux équipements et technologies. Vous traitez vos besoins d'adaptation au marché en séparant clairement le développement de la distribution.

1.3 Pourquoi utiliser NeoMAD ?

NeoMAD vous permet :

- de réduire la complexité du code tout en augmentant la productivité des développeurs et en optimisant les phases de développement,

- d'écrire un code source qui est portable sur tous les terminaux mobiles pris en charge, y compris les dispositifs ajoutés dans les versions futures,
- de gérer les ressources utilisées dans le projet (images, sons, textes, …),
- de générer automatiquement des fichiers binaires ou le code source natif pour les terminaux ciblés,
- de fusionner les kits de développement des fabricants à des fins de simulation et de tests,
- d'adresser de nouveaux téléphones et tablettes lorsqu'ils apparaissent sur le marché, sans retoucher le code source.

De plus, NeoMAD améliore la qualité des logiciels développés

- en industrialisant les processus de travail et de production,
- en simplifiant le développement : les caractéristiques de l'appareil ciblé ne sont plus une préoccupation des développeurs,
- en combinant portabilité et réutilisation : l'application développée devient indépendante des dispositifs existants et futurs,
- en automatisant les processus liés à l'amélioration permanente de la qualité : tests unitaires, intégration continue, GCL, outils d'analyse de qualité du code,
- en facilitant le processus de maintenance : un source unique et une intégration complète avec les outils de GCL,
- en automatisant le processus de génération et donc sa reproductibilité.

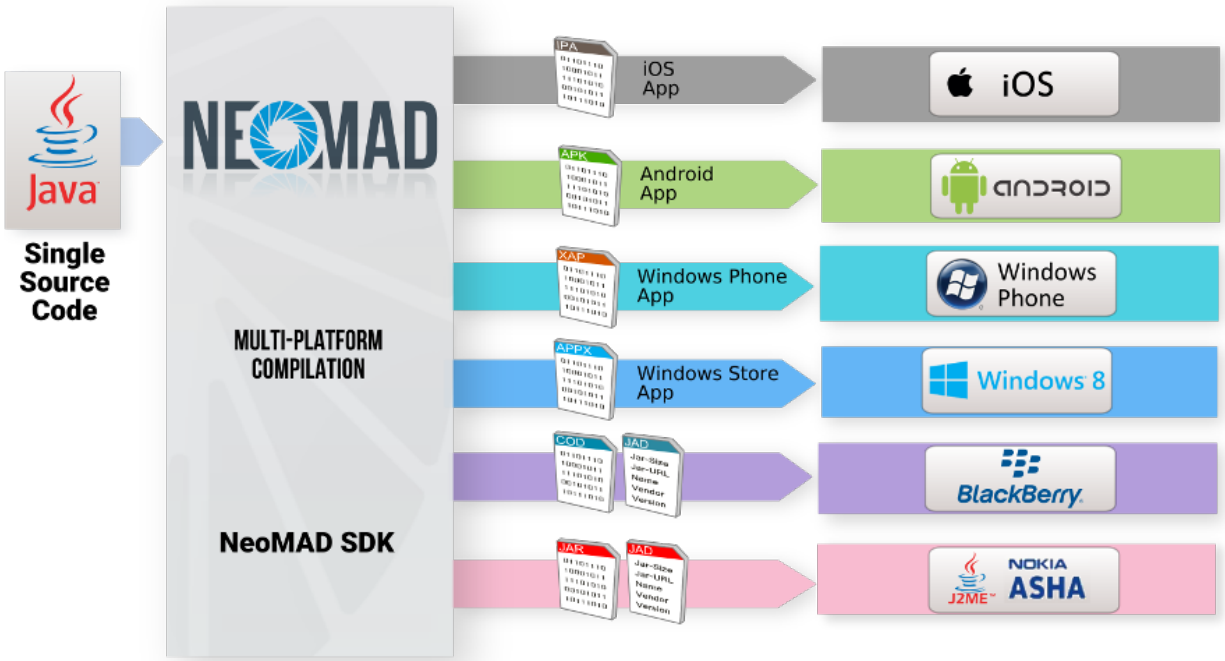
1.4 Développement cross plateformes pour la mobilité

NeoMAD Version 3 apporte une réelle révolution pour le développement d'applications mobiles devant être déployées sur plusieurs plateformes :

- Basé sur un code Java unique, NeoMAD produit des applications natives pour toutes les différentes technologies du marché.
- En utilisant un processus conditionnel valorisé dynamiquement lors de la compilation grâce à une base de connaissances, NeoMAD permet de mettre en œuvre différentes branches fonctionnelles dans l'application, afin d'adresser de façon optimale chaque technologie et plateforme mobile.
- Grâce à son mécanisme de trans-compilation, NeoMAD couvre l'ensemble des technologies du marché actuel de la mobilité et garantit la pérennité de vos applications sur les technologies de demain.

En utilisant l'API Générique fournie avec NeoMAD, vous couvrez tous les besoins du marché en termes de langages de programmation et de SDK ciblés. Avec un seul code, vous pouvez entre-autres :

- Définir l'interface utilisateur de vos applications tout en respectant l'ergonomie et les bonnes pratiques de chaque appareil. L'API Générique repose sur les API natives de chaque cible, apportant ainsi l'assurance que les applications produites avec NeoMAD auront un « look and feel » natif sur ces terminaux
- Accéder à des services réseau standardisés (tels que REST, SOAP, ...) de manière simplifiée
- Accéder aux différents capteurs (compas, boussole, GPS, accéléromètre, ...) et en exploiter les données de façon unifiée
- Gérer les notifications push et locales de façon générique; NeoMAD s'interface avec n'importe quel système push et intègre également les back-offices You N'Push et Parse
- Communiquer avec les objets connectés de dernière génération; NeoMAD intègre entre autres la technologie iBeacon
- Accéder aux systèmes de cartographie natifs présents sur chaque plateforme
- Interagir avec le système de fichier ou enregistrer vos données dans des bases de données SQLite.
- Intégrer des bibliothèques externes telles que des modules de paiement, de signature électronique, d'annotations de pdf...

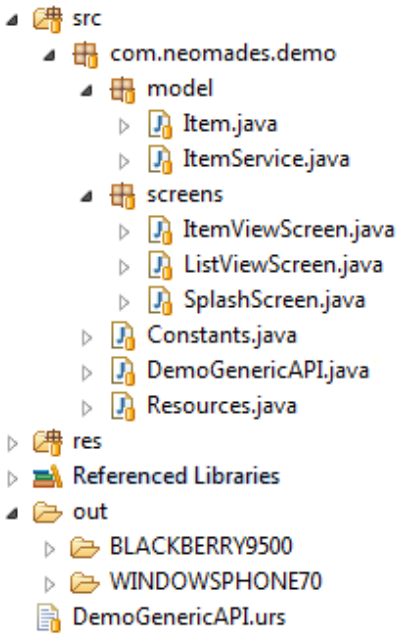


2 Utiliser NeoMAD

NeoMAD fournit un plugin pour l'IDE Eclipse et permet ainsi de développer des projets mobiles d'une façon très assistée et traditionnelle pour les développeurs Java.

2.1 Structure d'un projet NeoMAD

Un projet NeoMAD est similaire à tout autre projet Java, avec des fichiers sources Java organisés en packages. Les paramètres et ressources du projet sont définis dans un fichier de description de projet au format XML nommé .urs :



2.1.1 Le fichier de description du projet (URS)

Le fichier URS décrit les paramètres du projet, ses déclarations de ressources, les bibliothèques externes à intégrer, les dossiers de sortie pour produire les binaires et les codes générés.

Un projet NeoMAD se compose au moins de 3 types de dossiers :

- **src**: les dossiers source qui contiennent le code Java
- **res**: les dossiers de ressources qui contiennent les images, polices, sons …
- **out**: les dossiers pour la production de fichiers résultants

```
<?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 Le code source

Le code source d'un projet NeoMAD est entièrement écrit en Java et repose sur les API fournies par NeoMAD pour le développement d'applications mobiles portables. Certaines parties du code peuvent être conditionnées à l'aide de constantes afin de produire des versions fonctionnellement différentes de l'application. Ce mécanisme de conditionnement peut également être utilisé pour intégrer des parties de code natif afin d'intégrer des fonctionnalités spécifiques à certaines plateformes.

2.1.3 Les ressources

Un fichier Res.java est généré par NeoMAD en fonction des ressources déclarées dans le fichier URS. Il fournit des identifiants de ressources utilisables dans le code Java, répartis en sous classes.

Exemple

Une chaîne de caractères est fournie dans le fichier CSV file “TXT_HELLOWORLD;Hello World;Bonjour le monde”.

NeoMAD générera une constante nommée TXT_HELLOWORLD dans la classe string du fichier Res.java. Le développeur pourra utiliser cette constante dans son code Java pour accéder à la valeur du texte “Hello world” dans la langue choisie.

2.2 L'API Générique NeoMAD

L'API Générique NeoMAD est une bibliothèque fournie avec NeoMAD qui permet aux développeurs de produire des applications natives multi plateformes. C'est le point d'entrée pour le développement d'applications multi plateformes mobiles avec NeoMAD. Quels que soient les langages de programmation, les systèmes d’exploitation ou les caractéristiques techniques des terminaux cibles, l’API Générique permet de développer rapidement des applications qui s’exécuteront sur ces terminaux à partir d’un seul code source Java et de l’outil de génération NeoMAD. Les 3 composants de l'API Générique NeoMAD sont les suivants :

- **Une structure d'application**
    - gestion du cycle de vie de l'application (start, interruptions, stop ...)
    - Enchaînement des écrans
- **Un Kit de fonctions natives d'interface graphique**
    - Composants graphiques au look natif (Button, TextField, …)
    - Composants de haut niveau (TabScreen, SplitScreen, Layout, ...)
- **Un Kit de fonctions techniques**
    - communications HTTP/HTTPS …
    - gestion du navigateur, SMS, VoiceCall, Vibration …
    - stockage local, Base de données …
    - XML, JSON, etc.

A chaque composant de l'API Générique correspond un composant natif pour chacune des plateformes adressées.

Exemple

« Button » correspond à « ButtonField » (BlackBerry), « Button » (Android, Windows Phone), « UIButton » (iOS).

2.3 Mettre en œuvre du code conditionnel

Le processus de conditionnement consiste à déclarer des constantes et à les utiliser pour faire varier des parties du code source ou la déclaration de certaines ressources. Ces constantes sont évaluées dynamiquement lors du processus de génération.

2.3.1 Déclaration de constantes

Une condition est basée sur une constante définie et renseignée par NeoMAD ou par le développeur de l'application. En utilisant ce mécanisme, il est possible de générer différents binaires à partir d'un code source unique, ce afin de couvrir différents cas fonctionnels (utilisation d'une fonctionnalité qui n'est pas présente sur tous les terminaux, variations téléphones / tablettes, re-branding de l'application, etc.).

2.3.1.1 TargetInfo

TargetInfo est une interface Java fournie par NeoMAD qui définit les constantes correspondant aux caractéristiques des équipements (système d'exploitation, nom, taille d'écran, type de clavier, …). Lors du processus de production, cette interface déclare automatiquement les différentes valeurs correspondant à chaque cible.

Exemple

TargetInfo.TARGET_NAME donne le nom du téléphone lors de la compilation

2.3.1.2 Constants.java

Constants est une classe Java écrite par le développeur de l'application. Le développeur peut ainsi déclarer des constantes afin d'activer ou de désactiver certaines fonctionnalités de l'application mobile en fonction de conditions externes ou d'un fonctionnement spécifique de son application pour un client ... La classe Constants doit implémenter l'interface TargetInfo. Voici un exemple de classe Constants définissant une constante FEATURE_ADS utilisée pour activer la publicité sur certaines versions de l'application :

```
/**
* 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);
}
```

Toutes les constantes de Constants et TargetInfo peuvent être utilisées pour conditionner le code ou les déclarations de ressources.

2.3.2 Utiliser les constantes

2.3.2.1 Dans le code source

Le développeur peut utiliser les constantes pour conditionner des blocs de code. Dans ce cas, les traitements correspondants ne seront conservés que si la condition est remplie.

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

2.3.2.2 Pour la déclaration de ressources

Le développeur peut conditionner la déclaration de ses ressources dans le fichier URS. Dans ce cas, les ressources ne seront intégrées à l'application que si la condition est remplie. Il peut également utiliser les constantes pour définir des chemins ou valeurs. Dans ce cas, NeoMAD effectue la substitution lors de l'analyse des ressources.

```
<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 et RES_PATH sont fournis respectivement par TargetInfo et Constants.

- TargetInfo.ANDROID est proposé par NeoMAD et permet de conditionner la déclaration de certaines ressources à la plateforme Android,
- Constants.RES_PATH est déclaré par le développeur et définit le chemin d'accès aux ressources.

2.3.3 Le processus de compilation

Lors de la compilation d'application, NeoMAD utilise les valeurs de Constants et TargetInfo dans le but d'optimiser le code source généré. Durant ce processus, un bloc de code avec une condition fausse sera supprimé et le code généré ne comprendra pas cette partie du code.

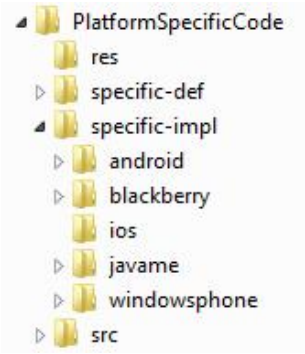
2.4 Utiliser du code spécifique natif pour une cible

NeoMAD propose une option qui permet d'ajouter du code spécifique à une plateforme dans un projet NeoMAD. En pratique, cette option permet d'inclure du code source écrit dans le langage de programmation de la plateforme cible (C#, Java, Objective C) dans un projet NeoMAD afin d'utiliser des fonctionnalités spécifiques à ces plateformes (Windows Phone, iOS ou Android par exemple). Cela permet d'utiliser des fonctionnalités spécifiques (boussole...), d'intégrer du code source ou des bibliothèques pré-existants pour une plateforme, ou d'accéder à des fonctionnalités disponibles sur la plupart des plateformes qui ne sont pas encore couvertes par NeoMAD. Cette option doit être utilisée en connaissance de cause, NeoMAD ne fait aucun contrôle sur la disponibilité sur le terminal ciblé des fonctions utilisées dans le code spécifique.

Pour utiliser cette option, le projet NeoMAD doit contenir deux répertoires supplémentaires :

- le répertoire specific-def contient la définition Java du code spécifique. Ce dossier et son contenu sont créés par le développeur. Afin de permettre l'interaction entre le code spécifique et le reste du code du projet, le développeur doit fournir une définition des parties spécifiques. Cette définition est une classe Java classique.
- le répertoire specific-impl contient l'implémentation du code spécifique pour chaque plateforme, qui sera embarquée dans le binaire produit. Il contient des sous-répertoires représentant les différentes plateformes pour lesquelles le code spécifique peut être implémenté (telles que Android, BlackBerry, Windows Phone, ...) Le répertoire specific-impl et son contenu sont générés automatiquement par NeoMAD, à partir de la définition fournie par le développeur. L'implémentation générée est l'exacte image de la définition dans le langage de programmation de la plateforme cible. Le développeur peut ensuite compléter les fichiers générés afin de fournir l'implémentation native pour chaque plateforme.

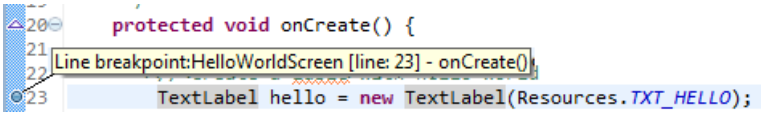
La structure de projet NeoMAD est la suivante :



2.5 Déboguer son application

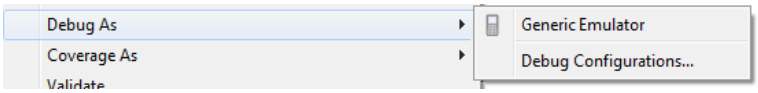
Utilisé en combinaison avec un IDE et un émulateur, NeoMAD permet de déboguer une application. En utilisant les plugins fournis avec NeoMAD, le développeur est en mesure de déboguer pas-à-pas une application s'exécutant dans l'émulateur NeoMAD.

Dans l'IDE, le développeur peut placer des points d'arrêt de manière à suivre pas-à-pas l'exécution :

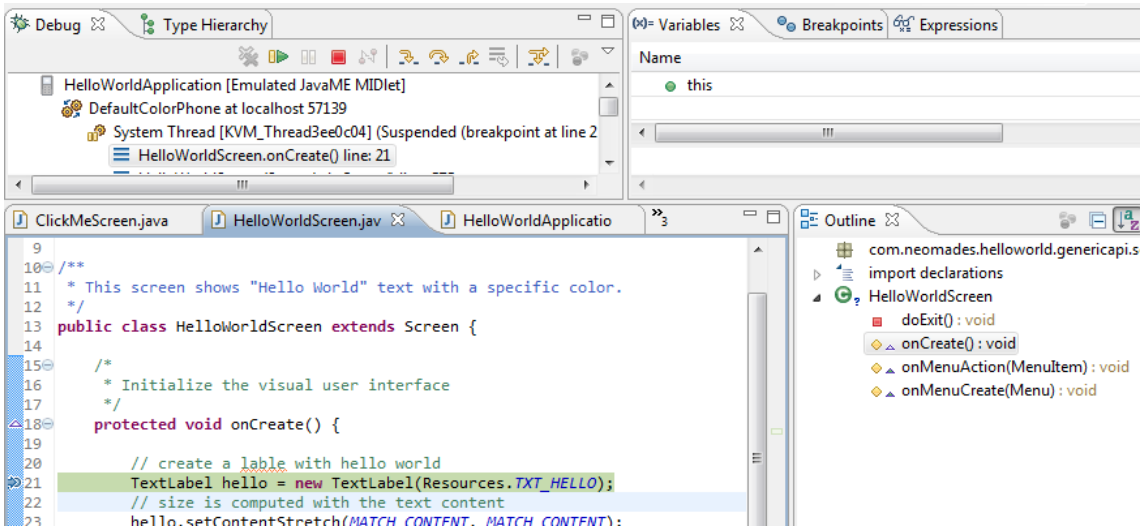


L'accès au débogage se fait via l'IDE.

Dans Eclipse par exemple, l'option utilisée sera « Debug As action ».



L'émulateur lance l'application et l'exécute jusqu'au premier point d'arrêt.



3 Exemple de développement d'application

Ce chapitre vous guide pas-à-pas dans la création d'une application basique avec l'API Générique de NeoMAD. Le code source de cet exemple se trouve dans l'extension « Examples » après l'installation de NeoMAD.

3.1 Le point d'entrée

La classe principale de l'application doit étendre « Application ». C'est le point d'entrée de l'application.

```
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);
    }
}
```

La méthode onStart() est surchargée pour gérer le démarrage de l'application. La seule chose à faire pour notre exemple consiste à déclarer le premier écran à afficher. Une instance unique de Controller est utilisée pour gérer la pile d'écrans (push et pop). Pour notre exemple, la méthode pushScreen() est appelée pour afficher le premier écran (HelloWorldScreen).

3.2 Passage d'un écran à l'autre

Une instance unique de Controller est fournie par l'API Générique NeoMAD en utilisant les classes Application et Screen. Controller propose plusieurs méthodes pour changer l'écran courant et garde l'historique de la pile pour la gérer le retour en arrière de manière native.

3.3 Construction de l'Interface Utilisateur

La structure de l'API Générique NeoMAD impose une classe par écran. Chaque écran est déclaré comme une classe qui étend la classe Screen. Ensuite, le développeur doit construire l'interface utilisateur de son application. L'API Générique de NeoMAD propose un ensemble complet d'éléments d'interface utilisateur pouvant être combinés pour définir des écrans d'application très riches. Cependant, pour cet exemple HelloWorld, nous allons commencer avec un écran très simple composé uniquement d'un layout vertical et d'un champ texte. Il existe deux manières de définir l'interface utilisateur avec NeoMAD : en code Java et dans un fichier XML. Les deux sont équivalentes en terme de rendu, et le choix de l'une ou l'autre dépend des préférences du développeur.

3.3.1 Définition de l'interface utilisateur en code Java

Pour définir l'interface utilisateur en code Java, il suffit d'utiliser les composants graphiques fournis dans l'API Générique directement dans la classe Screen :

```
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 Définition de l'interface utilisateur en XML

Plusieurs étapes sont nécessaires pour définir l'interface en XML.

- Tout d'abord, il faut créer le fichier XML dans le répertoire de ressources du projet et y définir l'interface utilisateur en respectant le schéma layout.xsd fourni par NeoMAD. Dans notre exemple, ce fichier s'appelle 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>
```

- Ce fichier doit ensuite être déclaré comme une ressource dans le fichier URS.

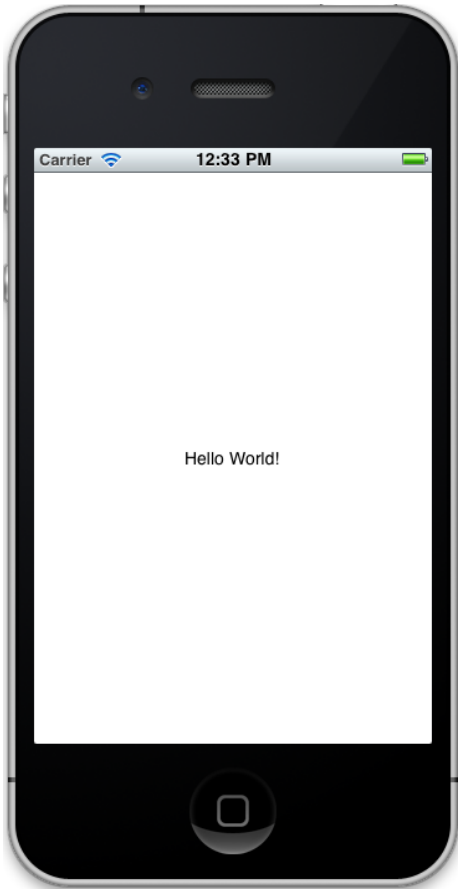
```
<resourcelot>
    <layout
        name="HELLOWORLD_SCREEN"
        path="res/layout/helloworld_screen.xml" />
</resourcelot>
```

- Enfin, il faut intégrer cette interface dans le contenu de l'écran en utilisant l'identifiant de ressource déclaré dans le fichier URS.

```
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 Résultats

Voici ce à quoi ressemble notre petite application sur des terminaux Android, BlackBerry, iOS et Windows Phone :



4 Screenshots d'applications existantes

Voici à titre d'exemple, les sceenshots d'applications réalisées avec NeoMAD.

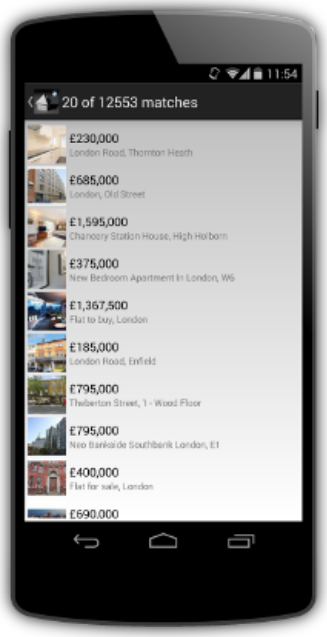
4.1 PropertyCross

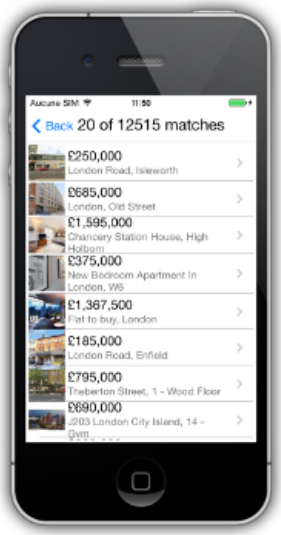
PropertyCross est une initiative d'un groupe d'experts anglais en programmation mobile qui propose à la communauté un site comparant les différentes solutions disponibles sur le marché pour développer des applications mobiles avec une approche cross plateforme.

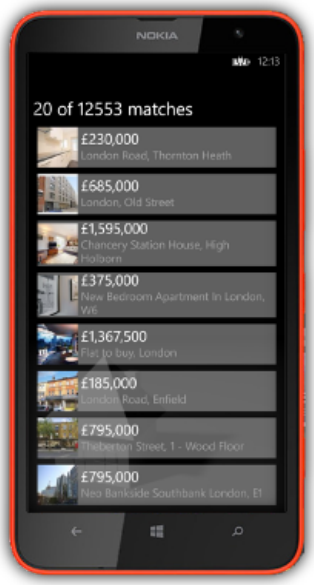
Le site propose dans ce but, une application étalon de recherche de logements à louer ou à acheter et une métrique de comparaison. Pour qu'une solution soit référencée sur le site, un développeur doit réaliser l'application étalon avec et mettre à disposition de la communauté, d'une part les sources de sa réalisation, d'autre part les exécutables résultants pour mobiles iOS, Android et Windows Phone.

La société All4Tests a réalisé l'opération pour étalonner NeoMAD et soumit les éléments de comparaison qui font apparaitre un code 100% identique et un usage 100% conforme à la demande.

Le site est accessible sur : http://propertycross.com.







4.2 GCT© Mobility par Gfi Chrono Time

Pour conforter sa place de leader européen, Gfi Chrono Time enrichit sa gamme de Progiciels de Gestion des Temps, de modules disponibles sur mobiles et tablettes.

Grâce à la technologie NeoMAD, ces nouveaux modules peuvent être accessibles sur l'ensemble des équipements et technologies du marché permettant ainsi aux utilisateurs d'accéder lorsqu'ils sont en situation de mobilité, aux bases de données gérées par les progiciels dans leur entreprise

Les nouveaux modules de ChonoGestor ont été développés par Gfi en utilisant NeoMAD V3.

Ils sont disponibles sur mobiles et tablette iOS, Android, Windows Phone et Blackberry.

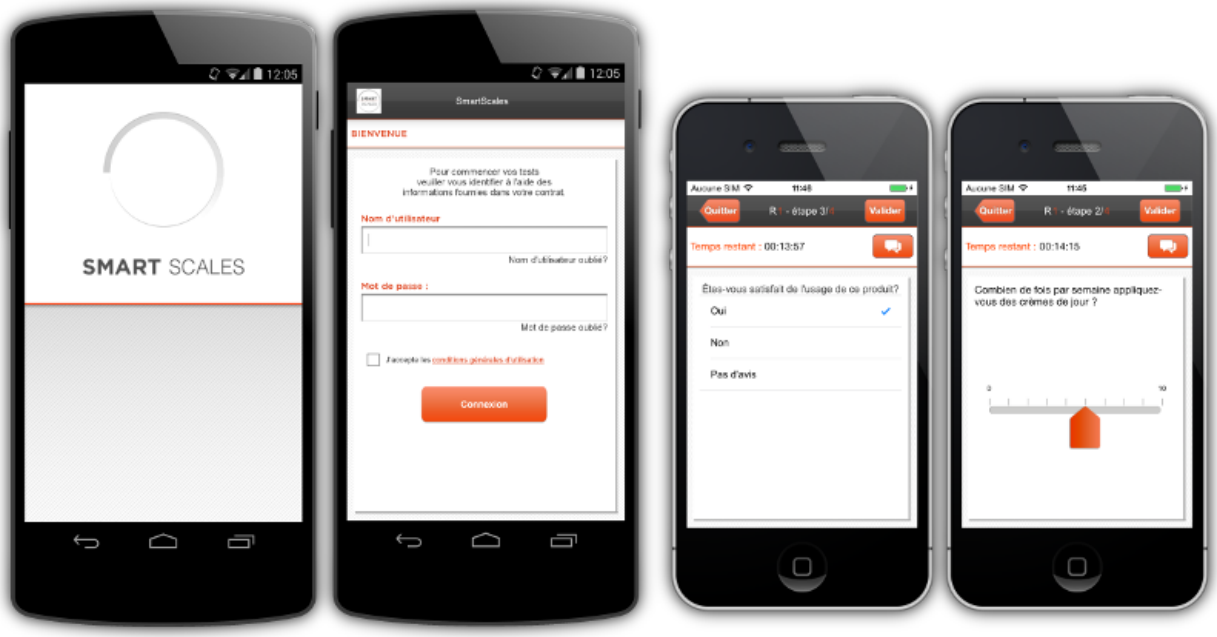
4.3 Smartscales par L'Oréal

L'application Smartscales a été développée à la demande du département Recherche et Innovation de L'Oréal pour gérer les campagnes de tests de cosmétiques.

Grâce à son Back Office, le responsable du produit définit des ensembles de questions, les groupes d'individus ciblés par les tests, le rythme et le planning de réponse.

L'application pour sa part, indique au bon moment à son testeur, qu'il va lui falloir réagir à la demande, répondre aux questions et fournir les informations, selon le plan déterminé par le responsable du produit. Une fois le test réalisé, les informations contrôlées sont envoyées au Back Office pour agrégation et l'application se met en veille jusqu'au moment du test suivant. Évidement, le responsable de produit peut sur la base d'une analyse partielle des premiers résultats collectés, modifier le déroulement et le contenu de son test de manière à infléchir en cours de route ce qui lui semble nécessaire.

Ce système de collecte puissant est disponible pour le moment, sur mobiles et tablettes de technologies iOS et Android.

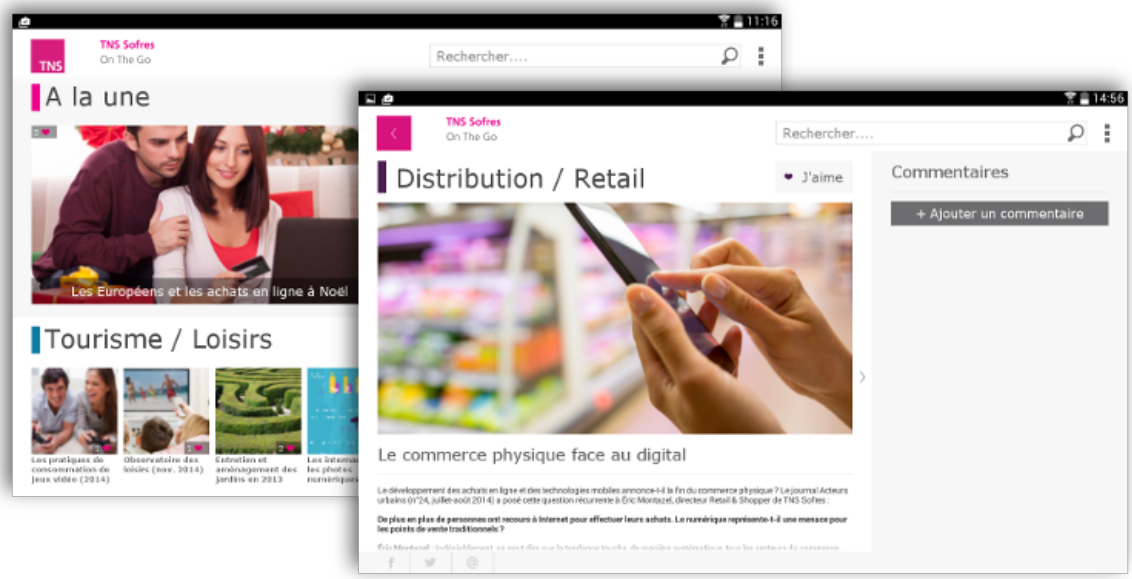


4.4 TNS On The Go par TNS Sofres

En publiant l'application TNS On The GO, son éditeur, la direction marketing de TNS SOFRES, a pour objectif de donner accès à ses clients depuis leurs équipements mobiles, aux analyses dès qu'elles sont disponibles et à les faire réagir et interagir pour enrichir le débat.

Ce nouvel outil dans la palette de communication de TNS SOFRES, permet aux équipes internes d'animer le réseau de leurs utilisateurs lorsqu'ils le souhaitent et de mieux partager et faire connaitre les informations qui sont produites par les différents secteurs d'études. Chaque utilisateur de l'application peut ainsi déterminer les sujets qui le passionnent et demander à être informé dès la mise à disposition d'une nouvelle analyse, réagir instantanément et partager son point de vue avec la communauté. La mise en place d'un Back Office développé en relation avec la réalisation de cette application, permet aussi de mutualiser les diffusions en adaptant les contenus sur les différents canaux communautaires.

Cette application a été réalisée après une série de prestations de cadrage qui ont permis d'inscrire la stratégie mobile dans la stratégie de communication de TNS SOFRES; elle est actuellement disponible sur mobiles et tablettes de technologies Android et iOS.



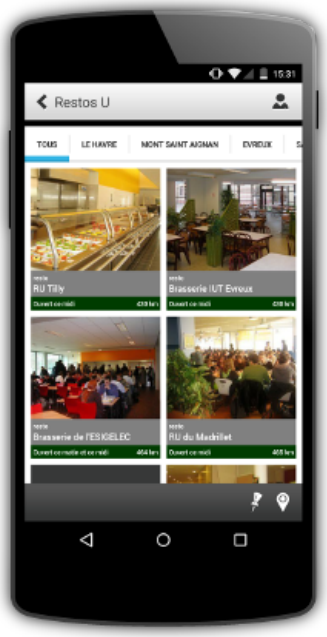
4.5 CROUS par le CNOUS

À la tête du réseau national des Crous, le Cnous a pour vocation de faciliter la vie des étudiants dans de nombreux domaines : restauration, logement, bourses, action sociale et culturelle, ouverture sur l'international…

C'est donc naturellement qu'une application mobile a été pensée puis développée pour permettre d'accéder aux informations locales essentielles de la vie étudiante : les horaires de son restaurant préféré, le menu du jour ou des jours à venir, les équipements disponibles dans les différents logements, les spectacles et animations, les bourses et le moyen d'y accéder...

Au delà de ces informations, l'usage de la mobilité permet d'avoir aussi des fonctions riches de partage, de guidage, de recherche par proximité et zone géographique qui permettent à la communauté des étudiants d'utiliser l'application comme un outil dédié à leur vie de tous les jours. Cette application s'appuie sur des flux locaux de manière à garantir des informations aussi pertinentes que possible.

L'application CNOUS est disponibles sur mobiles et tablettes de technologies iOS, Android et Windows Phone 8.



4.6 Nouvelle Chance pour l'Orientation

Dans le cadre du soutien aux initiatives d'accompagnement des mutations économiques soutenu par le Conseil régional d'Aquitaine, AGEFOS – PME participe à une initiative du territoire pour accompagner les besoins en compétences des entreprises du Piémont Oloronais.

Les groupes de travail « Passeport local de Compétences » et « Valorisation des Métiers de l'industrie » créés pour réfléchir sur les moyens à mettre en œuvre, ont décidé de faire une expérimentation appelée NCO (Nouvelle Chance pour l'Orientation) avec les entreprises et les partenaires emploi / formation du territoire : cette expérimentation a pour objectif de faire découvrir aux jeunes en recherche de leur orientation, en cours d'orientation, en cours de formation ou à la recherche d'un premier emploi, les 50 métiers les plus porteurs en terme de débouchés en Région Aquitaine.

L'expérimentation s'appuie de manière naturelle sur une solution mobile; celle-ci se compose d'une part, d'une application sur mobiles et tablettes, d'autre part, d'un back-office d'administration qui permet aux acteurs de l'emploi de piloter l'expérimentation, son usage et ses usagers, et enfin, de packs de ressources par secteur d'activité, couvrant en moyenne 10 métiers, utilisables dynamiquement par l'application.

Cette application est disponible actuellement sur mobiles et tablettes iOS et Android.