Livrets B4x

B4x Premiers pas

 Edition 1.1

Dernière mise à jour : 2017.10.12

Contributeurs principaux : Klaus Christl (klaus), Erel Uziel (Erel)
Traduit par : Klaus Christl (klaus)

Pour chercher un mot ou une phrase particulière veuillez utiliser la fonction Rechercher dans le menu Edition.

Les codes source avec tous les fichiers nécessaires (layouts, images etc.) de tous les projets exemple dans ce guide figurant dans le dossier CodesSource.
Pour chaque programme il y a trois dossiers.

```
CodesSource
    MonPremierProgramme
        B4A
            MonPremierProgramme.b4a
        B4i
            MonPremierProgramme.b4i
        B4J
            MonPremierProgramme.b4j
```

Les deux programmes MonPremierProgramme et SecondProgramme sont pratiquement les mêmes pour les trois produits B4A, B4i et B4J.

Mis à jour pour les versions ci-dessous :
B4A version 7.30
B4i version 4.30
B4J version 5.90
B4R version 2.20

Autres livrets B4x en français :
B4x Premiers pas
B4x Langage Basic
B4x EDI Environnement de Développement Intégré

Livrets B4x en anglais :
B4x Getting started
B4x Baisc Language
B4x IDE Integrated Development Environment
B4x CustomViews

1 B4x

B4x est une suite de langages de programmation BASIC pour différentes plateformes.

B4x supporte plus de plateformes que n'importe quel autre outil
ANDROID | IOS | WINDOWS | MAC | LINUX | ARDUINO | RASPBERRY PI | ESP8266 | ET PLUS...

B4R, B4A, B4J et B4i ensemble, constituent la meilleure solution de développement pour l'Internet des Objets (en anglais Internet of Things IoT).

- **B4A**  **Android**

  B4A inclut toutes les fonctionnalités pour développer rapidement n'importe quel type d'application pour Android.

- **B4i**  **iOS**

  B4i est un outil de développement pour des applications natives pour iOS.
  B4i suit les mêmes concepts que B4A, et vous permet de réutiliser la plupart du code et produire des applications pour les deux systèmes d'exploitation Android et iOS.

- **B4J**  **Java / Windows / Mac / Linux / Raspberry PI**

  B4J est un outil de développement, **100% gratuit,** pour des applications desktop, serveurs et IoT.
  Avec B4J vous pouvez facilement créer des applications desktop avec interface utilisateur, des applications de console (sans interface utilisateur) et des solutions serveur.
  Les applications compilées peuvent fonctionner sur Windows, Mac, Linux et des cartes ARM (comme Raspberry Pi).

- **B4R** 

Arduino / ESP8266

  B4R est un outil de développement, **100% gratuit,** pour des applications natives pour des microcontrôleurs Arduino et ESP8266.
  B4R suit les mêmes concepts que les autres produits B4x, fournissant un outil de développement simple et puissant.

2 B4A Premiers pas

B4A (Basic for Android) est un environnement de développement simple mais puissant qui cible les dispositifs Android.
Le langage B4A est similaire à Visual Basic avec un support pour des objets.
Les applications compilées sont des applications natives, donc pas besoin d'autres programmes ou dépendances.
Contrairement aux autres EDI (Environnement de Développement Intégré), B4A est 100% axé sur le développement d'applications Android.
B4A comprend un 'Constructeur visuel' puissant permettant de définir des interfaces utilisateur supportant des écrans et orientations multiples.

Pas besoin d'écrire du XML.

Vous pouvez développer et déboguer avec :
- un dispositif réel via B4Abridge
- un dispositif réel via un câble USB
- ou une émulateur Android.

B4A comprend un vaste ensemble de bibliothèques qui rendent facile le développement d'applications avancées.
Incluant : SQL databases, GPS, Serial ports (Bluetooth), Camera, XML parsing, Web services (HTTP), Services (background tasks), JSON, Animations, Network (TCP and UDP), Text To Speech (TTS), Voice Recognition, WebView, AdMob (ads), Charts, OpenGL, Graphics and more.

Android 1.6 et plus sont supportés (incluant les tablettes).

2.1 B4A Version d’essai

Consultez dette page pour des instruction comment utiliser la version d’essai :
https://www.b4x.com/b4a.html

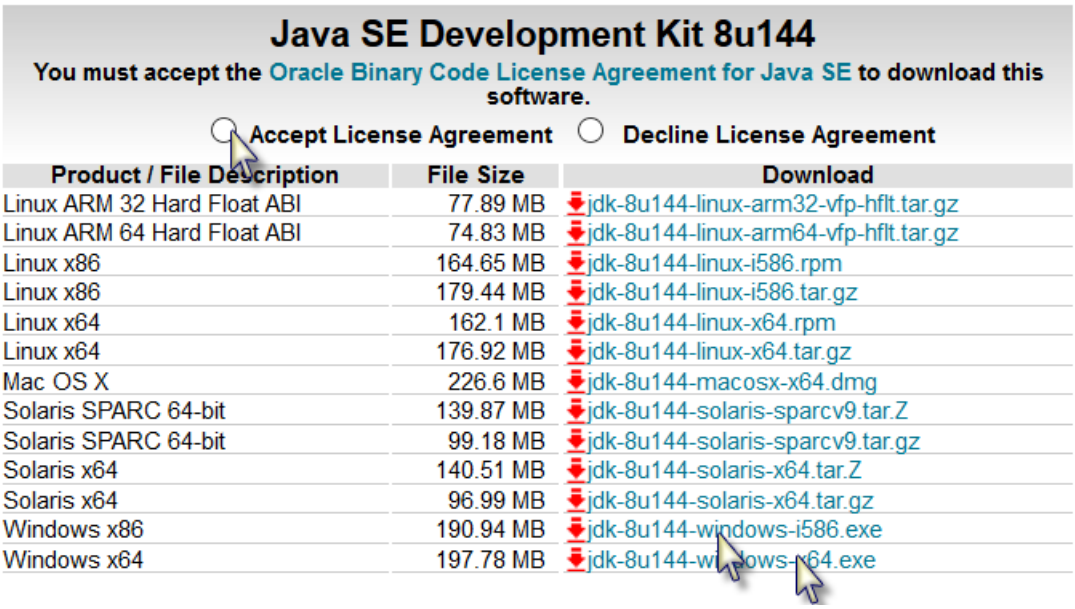
2.2 Installation de B4A et Android SDK

B4A dépend de deux composants supplémentaires (gratuits) :
- Java JDK
- Android SDK

2.2.1 Installation de Java JDK

Instructions d'installation :
La première étape est d'installer le **Java JDK**, car Android SDK l'exige aussi.
Notez que vous pouvez, sans problème, installer plusieurs versions de Java sur un même ordinateur.

- Ouvrez ce lien Java 8 JDK download link.



Java SE Development Kit 8u144

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ **Accept License Agreement** ○ **Decline License Agreement**

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.89 MB	jdk-8u144-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.83 MB	jdk-8u144-linux-arm64-vfp-hflt.tar.gz
Linux x86	164.65 MB	jdk-8u144-linux-i586.rpm
Linux x86	179.44 MB	jdk-8u144-linux-i586.tar.gz
Linux x64	162.1 MB	jdk-8u144-linux-x64.rpm
Linux x64	176.92 MB	jdk-8u144-linux-x64.tar.gz
Mac OS X	226.6 MB	jdk-8u144-macosx-x64.dmg
Solaris SPARC 64-bit	139.87 MB	jdk-8u144-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.18 MB	jdk-8u144-solaris-sparcv9.tar.gz
Solaris x64	140.51 MB	jdk-8u144-solaris-x64.tar.Z
Solaris x64	96.99 MB	jdk-8u144-solaris-x64.tar.gz
Windows x86	190.94 MB	jdk-8u144-windows-i586.exe
Windows x64	197.78 MB	jdk-8u144-windows-x64.exe

- Cochez l'agrément de licence Accept License Agreement.
- Sélectionnez **"Windows x86"** ou **"Windows x64"** (pour des ordinateurs 64 bits) dans la liste des plateformes.
- Téléchargez le fichier et installez-le.

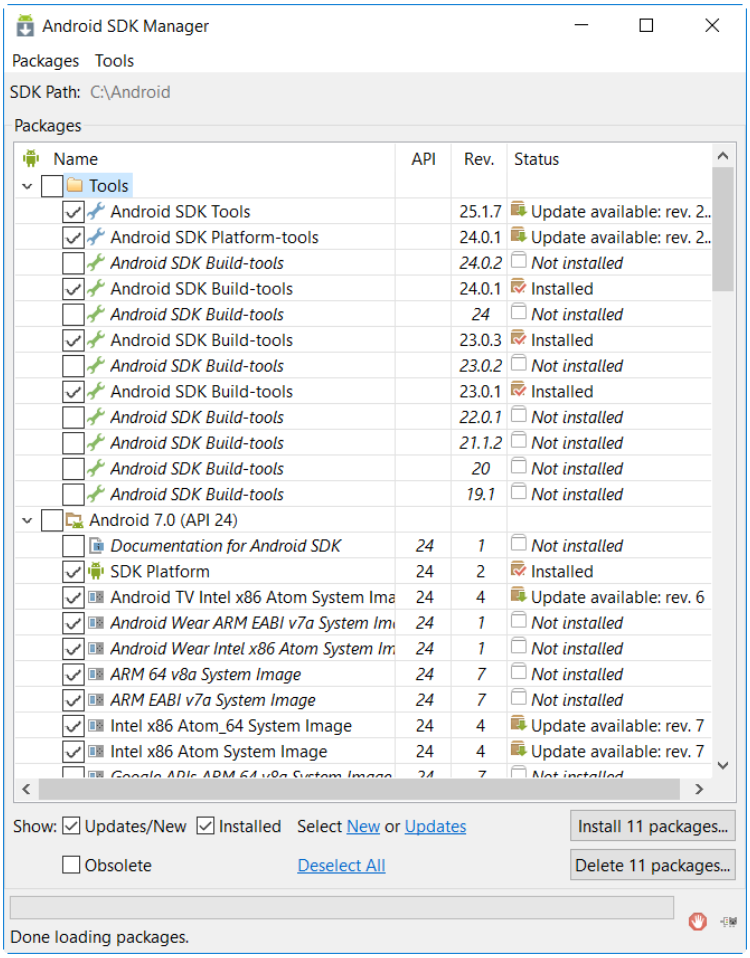
2.2.2 Installation de Android SDK

L'étape suivante consiste à installer Android SDK et une plateforme :

- Installation du SDK . **Le SDK ne fonctionner correctement que s'il est installé dans un dossier dont le nom ne comprend pas des espaces comme (Program Files).**
Il est recommandé de l'installer dans un dossier particulier similaire à C:\Android.

- Vous devez maintenant installer les outils de la plateforme et au minimum une image de plateforme. Utilisez la dernière version mais au moins API 8.

Il est conseillé d'installer aussi Google USB Driver pour pouvoir connecter physiquement un dispositif avec USB. Une liste d'autres pilotes se trouve ici.
Notez que vous pouvez aussi connecter un dispositif Android par un réseau local avec l'outil B4A-Bridge.

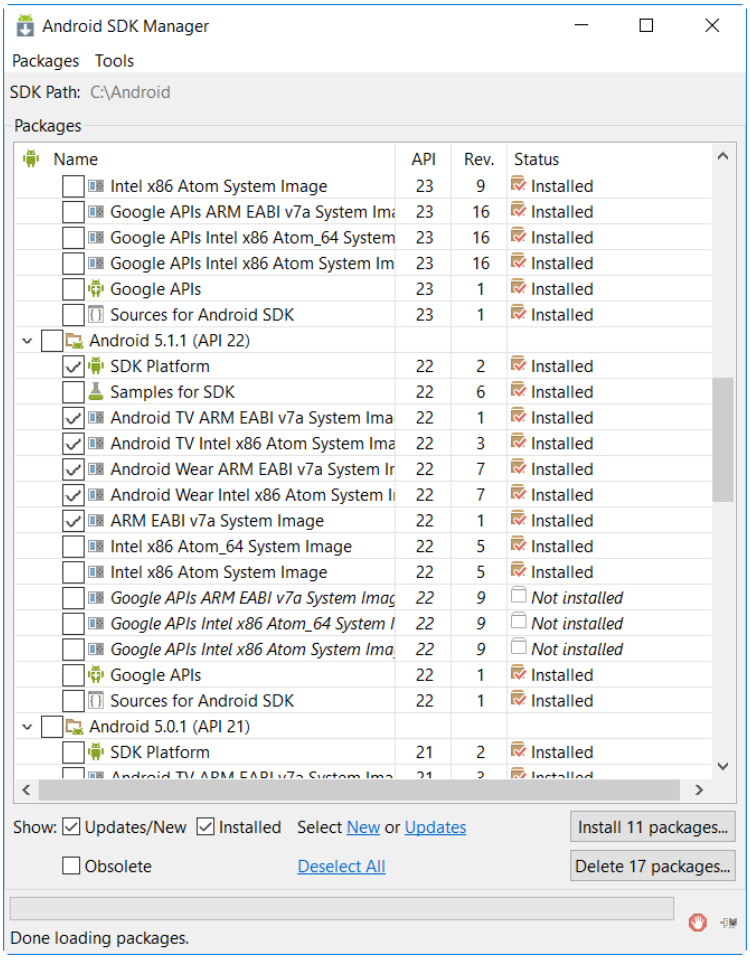


Une fenêtre similaire à celle-ci sera affichée.

Sélectionnez la version de l'API que vous voulez télécharger.
Dans l'exemple j'ai choisi API 24.

Vous pouvez sélectionner plusieurs APIs et les installer en parallèle.

Dans l'exemple, API 22 a aussi été sélectionné.



Notez que vous pouvez en installer d'autres plus tard.

- Pressez sur Install xx packages.

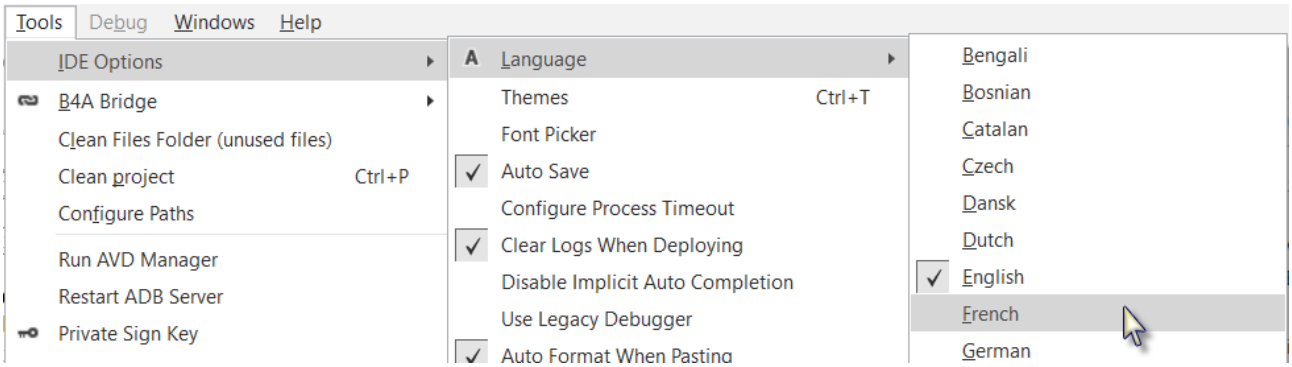
Si vous voulez connecter votre dispositif avec USB vous devez aussi télécharger Google USB driver.

2.2.3 Installation de B4A

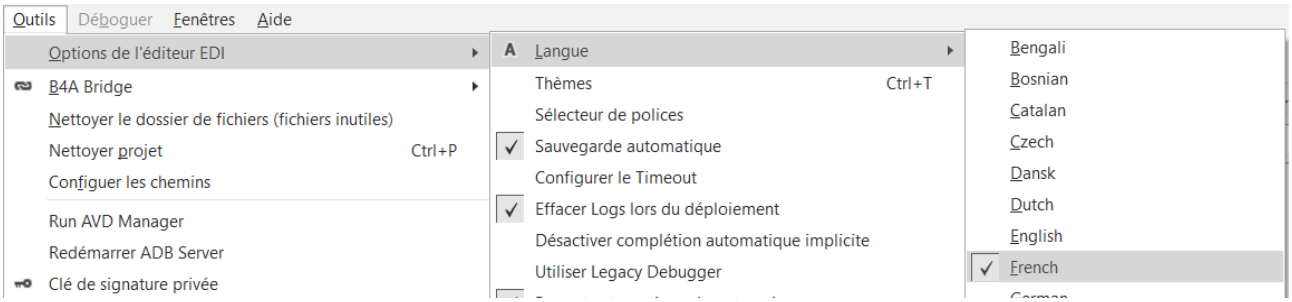
- Téléchargez B4A et installez le.
- Lancez B4A.

2.3 B4A Choix de la langue

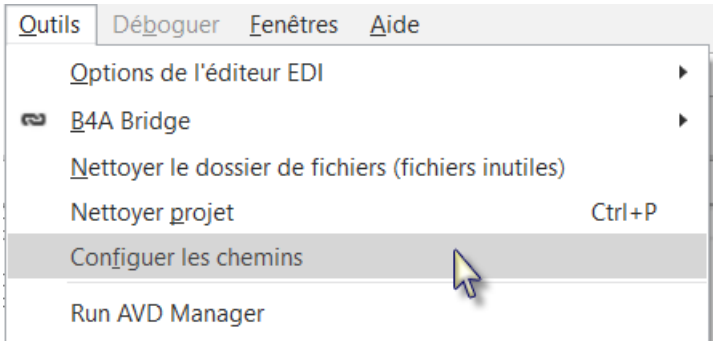
Dans le menu *Tools / IDE Options / Language* sélectionnez la langue de votre choix, French (français) dans l'exemple.



Ou, si l'éditeur est déjà en français vous pouvez choisir une autre langue.

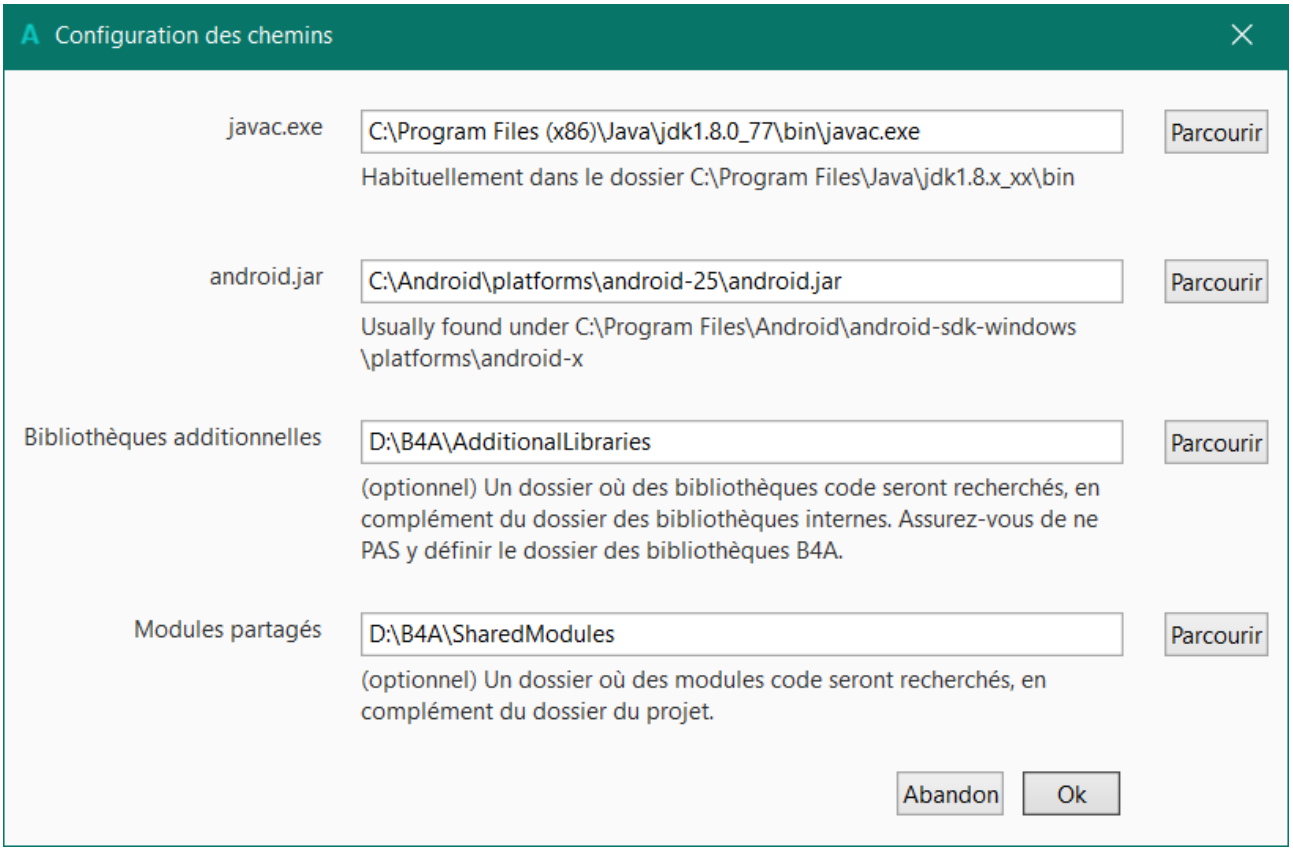


2.4 B4A Configuration des dossiers dans l'EDI



- Lancez B4A.
- Cliquez sur Configuer les chemins dans le menu Outils.

La fenêtre ci-dessous sera affichée.



- Utilisez les boutons Parcourir pour chercher les fichiers «javac.exe" et "android.jar".
javac.exe est situé dans le dossier <dossier java>\bin.
android.jar est situé dans le dossier <dossier Android>\platforms\android-24**.**
Le dossier dépend où vous avez installé Android SDK,
En principe : C:\Android\platforms\android-xx\android.jar
ou C:\Android\platforms\android-24\android.jar.
Les numéros dépendent de la version d'Android que vous avez installée.

Sur des versions plus anciennes, android.jar peut se trouver sous :
C:\Android\android-sdk-windows\platforms\android-8\android.jar.
Sous Windows 64 bit, Java se trouve probablement sous C:\Program Files (x86).

Il est recommandé de créer un dossier spécifique pour les bibliothèques additionnelles.
B4A utilise deux types de bibliothèques :

- Bibliothèques standard, qui sont fournies avec B4A et sont situées dans le dossier Libraries de B4A.
  Ces bibliothèques sont automatiquement mises à jour lorsque vous installez une nouvelle version de B4A.
- Bibliothèques additionnelles, qui ne sont pas fournies avec B4A, et en majorité écrites par des membres du forum. Ces bibliothèques doivent être enregistrées dans un dossier spécifique différent du dossier standard.

Modules partagés : Des fichiers Module peuvent être partagés entre différents projets et ils doivent être enregistrés dans un dossier spécifique.

Les bibliothèques et modules sont expliqués dans le livret B4x Langage Basic.

2.5 B4A Connecter un dispositif réel

Il existe deux moyens de connecter un dispositif réel :

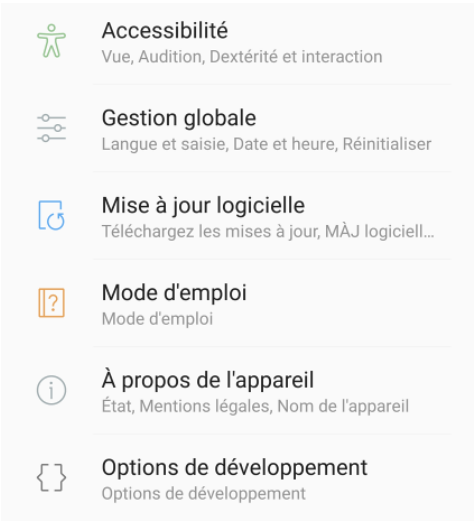
- USB
  Exige que le dispositif supporte le débogage ADB.
  Exige l'activation de Débogage USB sur le dispositif.
- B4A Bridge, via WiFi.

2.5.1 Connexion via USB

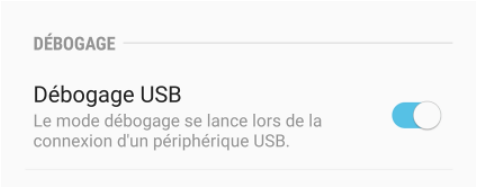
Vous devez télécharger le pilote Google USB Driver dans Android SDK Manager.
Si ce pilote ne fonctionne pas, vous devez chercher un pilote spécifique pour votre dispositif.

Pour pouvoir connecter un dispositif via USB vous devez activer Débogage USB.
C'est aussi nécessaire si vous utilisez un émulateur.

Sur le dispositif, exécutez Paramètres



Déroulez l'écran jusqu'à ce que vous voyiez *Options de développement*.



Déroulez l'écran jusqu'à ce que vous voyiez *Débogage USB.*

Activez-le.

Le dispositif sera automatiquement reconnu par l'EDI.

Dans ce cas, sur certains anciens dispositifs, il n'était pas possible d'accéder à la carte SD depuis le PC. Si vous voulez accéder à carte SD vous devez désactiver Débogage USB.

2.5.2 Connexion via B4A-Bridge

Il est toujours recommandé d'utiliser un dispositif réel au lieu d'émulateurs Android qui sont plus lents compares à des dispositifs réels (spécialement lors de l'installation d'applications).

Néanmoins pas tous les dispositifs ne supportent le débogage ADB.
C'est la raison de l'outil B4A-Bridge.
B4A-Bridge comporte deux composantes. L'une s'exécute sur le dispositif et permet à l'autre composante, qui fait partie de l'EDI, de se connecter et de communiquer avec le dispositif.
La connexion se fait au travers d'un réseau (B4A-Bridge ne fonctionne pas sans réseau).

Une fois connecté, B4A-Bridge supporte toutes les fonctionnalités de l'EDI incluant :
L'installation d'applications, affichage des Logs et le Concepteur Visuel.

Android n'autorise pas une application d'installer d'autres applications sans l'autorisation de l'utilisateur, c'est la raison pour laquelle lorsque vous compilez votre application, B4A-Bridge affiche un écran demandant votre autorisation.

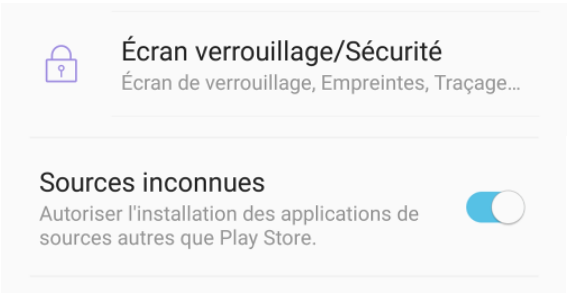
2.5.2.1 Premiers pas avec B4A-Bridge

Vous devez installer B4A-Bridge sur votre dispositif.

B4A-Bridge peut être téléchargé depuis ici :
http://www.basic4ppc.com/android/files/b4a_bridge.apk.

B4A-Bridge est aussi disponible sur Play Store. Chercher pour : B4A Bridge.
Notez que vous devez autoriser l'installation d'applications de "Sources inconnues".

Sur le dispositif, exécutez Paramètres

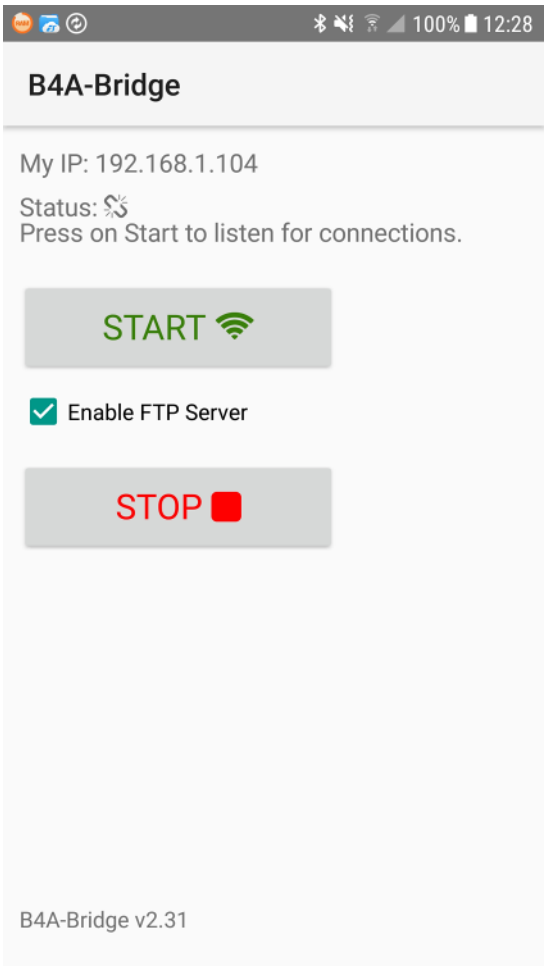


Déroulez l'écran jusqu'à ce que vous voyiez *Options de développement*.
Activez-le.

Déroulez l'écran jusqu'à ce que vous voyiez *Options de développement*.
Activez-le.

B4A-Bridge requiert une carte de stockage en écriture. Sinon, il est impossible d'installer des applications.

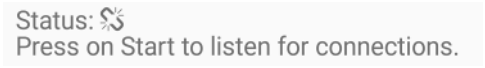
2.5.2.2 Exécuter B4A-Bridge sur votre dispositif



Lancez B4A-Bridge sur votre dispositif, un écran similaire à l'image ci-contre sera affiché.

Sur le haut vous trouvez l'adresse IP du dispositif : 192.168.1.104 dans l'exemple.

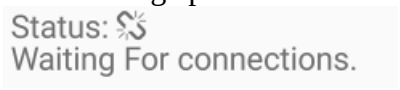
Status sera :



Pressez sur 

pour activer une connexion.

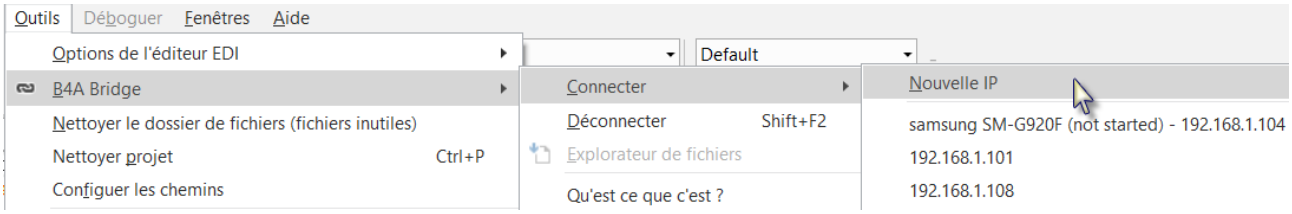
Status change pour :



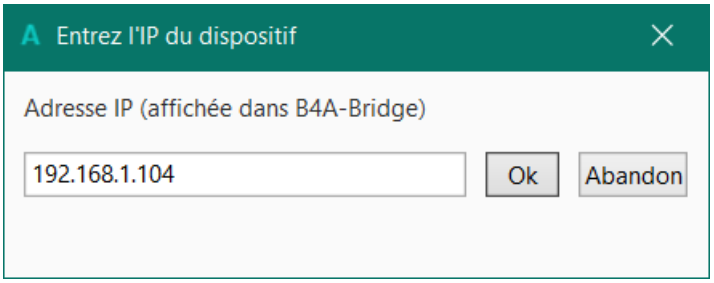
Notez que B4A-Bridge a été écrit avec B4A.

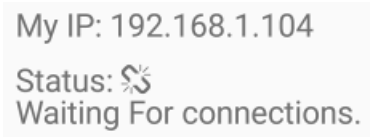
2.5.2.3 Connexion sans fil

Dans l'EDI, dans le menu Outils sélectionnez Nouvelle IP.
Si l'adresse existe déjà, cliquez directement sur cette adresse.
Si le dispositif avait déjà été connecté, pressez simplement la touche F2 pour vous connecter.



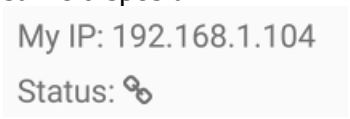
Entrez l'adresse IP de votre dispositif qui se trouve sur le haut de l'écran de B4A-Bridge.



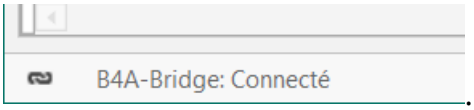


Cliquez sur Ok, le dispositif est maintenant connecté à l'EDI.

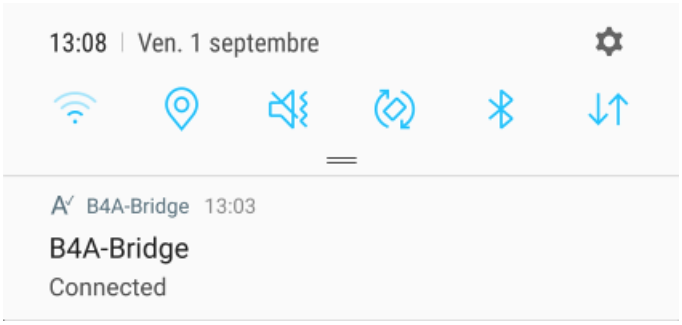
Vous verrez que l'état (status) a changé,
sur le dispositif



et dans l'EDI dans le coin inférieur gauche.



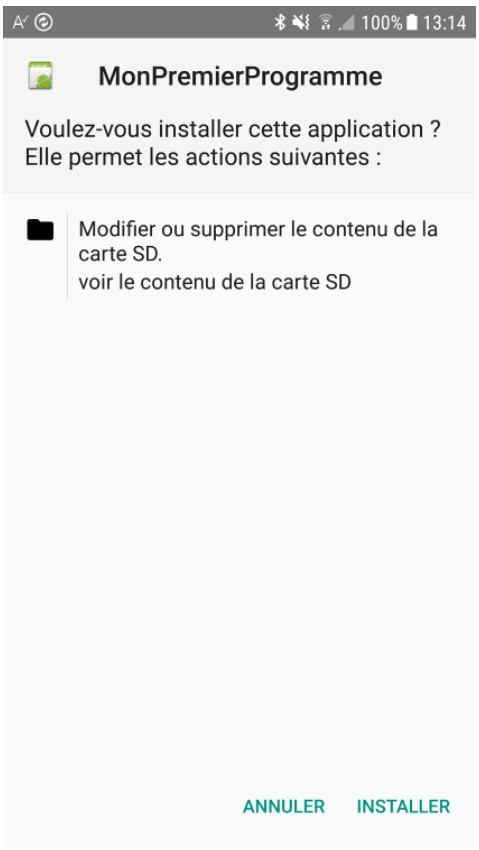
B4A-Bridge s'exécute en tant que service jusqu'à ce que vous pressiez le bouton Stop.



Vous pouvez toujours y avoir accès dans l'écran des notifications.

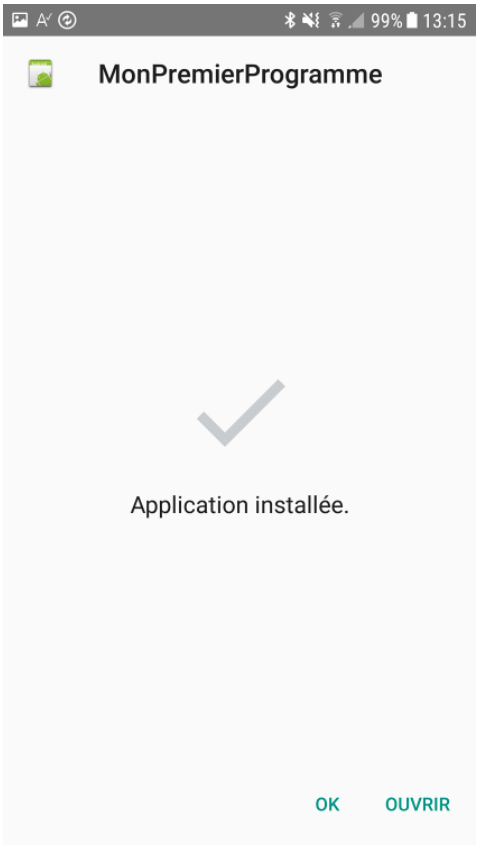
Vous voyez B4A-Bridge avec son état actuel.

Notez que la permission Internet sera automatiquement ajoutée dans le mode débogage



Lorsque vous exécutez une application, vous devez approuver son installation. Vous verrez généralement un écran similaire à celui-ci-contre.

Pressez INSTALLER pour installer le programme.



Si vous avez pressé INSTALLER vous verrez un écran similaire à celui à gauche.

Sur cet écran, pressez OUVRIR pour exécuter l'application.

Si vous essayez d'installer une application existante signée avec une clé différente, l'installation échouera (sans aucun message significatif). Vous devez d'abord désinstaller l'application existante. Allez dans Paramètres – Applications, sélectionnez-la et pressez DÉSINSTALLER.

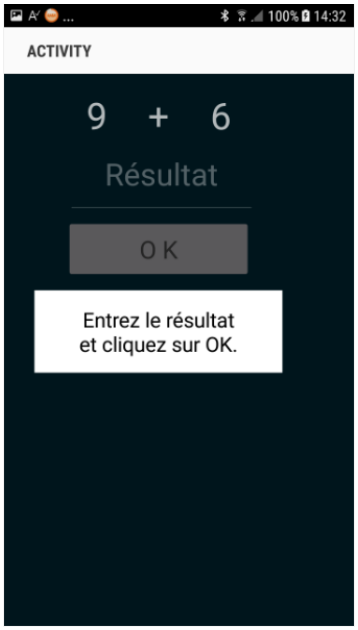
Lorsque vous arrêtez le développement, pressez le bouton Stop dans B4A-Bridge pour économiser la batterie.

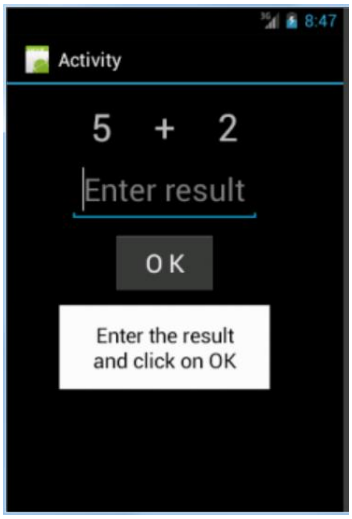
2.6 Mon premier programme B4A (MonPremierProgramme.b4a)

Nous allons écrire notre premier programme B4A. C'est un programme d'entraînement de calcul pour enfants.

Le projet est disponible dans le dossier des codes sources fourni avec le livret :
CodesSource\MonPremierProgramme\ B4A\ MonPremierProgramme.b4a

L'aspect de l'écran est différent selon la version Android des dispositifs, également avec les émulateurs.





Sony xperia z1 Emulateur Android version 4.2 Emulateur Android version 2.2

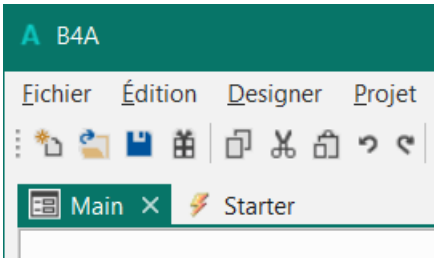
Nous aurons sur l'écran :
- 2 Labels affichant des nombres générés aléatoirement (entre 1 et 9).
- 1 Label avec le signe mathématique (+).
- 1 EditText dans lequel l'utilisateur devra entrer le résultat.
- 1 Button, utilisé soit pour confirmer le résultat entré ou pour générer un nouveau calcul.
- 1 Label avec un commentaire concernant le résultat.

Dans Android :
- Label est un objet pour afficher du texte.
- EditText est un objet permettant à l'utilisateur d'éditer du texte.
- Button est un objet permettant à l'utilisateur des actions, un click.

Nous allons définir le layout (mise en page) de l'interface utilisateur avec le Concepteur visuel du Designer et passerons pas à pas au travers de tout le processus.
Le Designer gère les différents objets de l'interface.
Le Concepteur visuel montre les positions et dimensions des différents objets et permet de les déplacer ou de les redimensionner sur l'écran.
Sur un dispositif nous voyons l'aspect réel

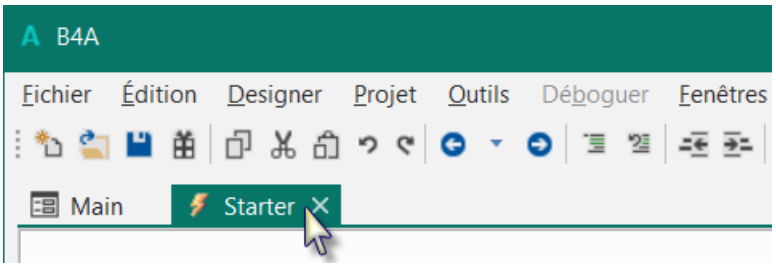
Exécutez B4A 

Lorsque vous exécutez B4A vous verrez sur le haut à gauche deux onglets Main et Starter.

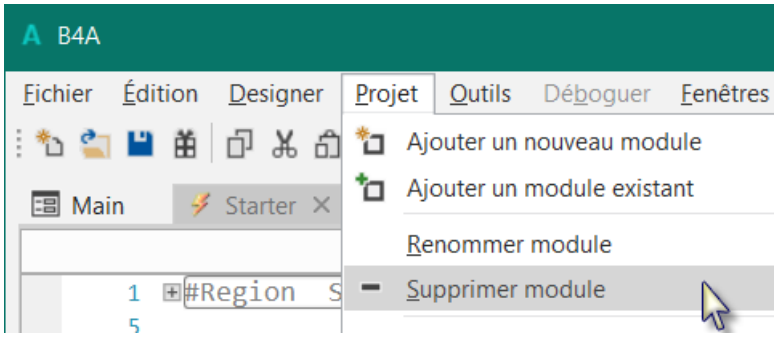


- Main
Est le module principal pour B4A qui est normalement le module de départ. Son nom ne peut pas être changé.
- Starter
Est un service, qui est exécuté au lancement du programme.

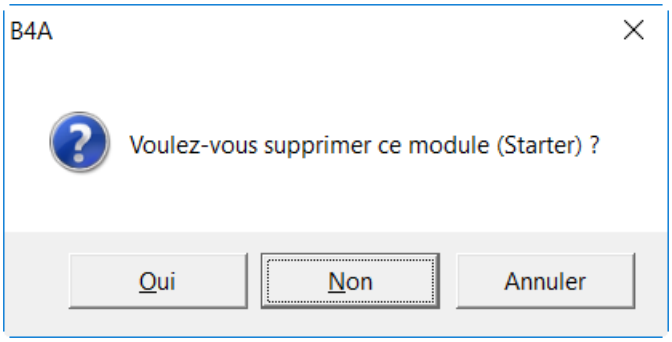
Pour notre premier programme nous n'avons pas besoin du module Starter, nous le supprimons.



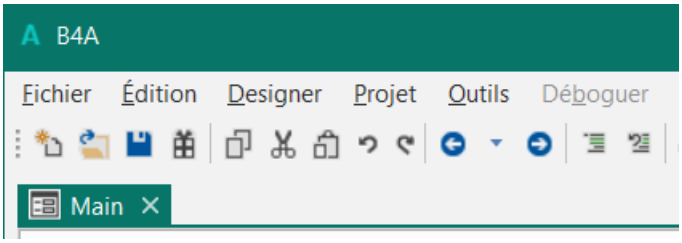
Cliquez sur l'onglet Starter pour sélectionner le module Starter.



Dans le menu Projet cliquez sur Supprimer module.



Vous devez confirmer si vous voulez supprimer le module.
Cliquez sur OUI.

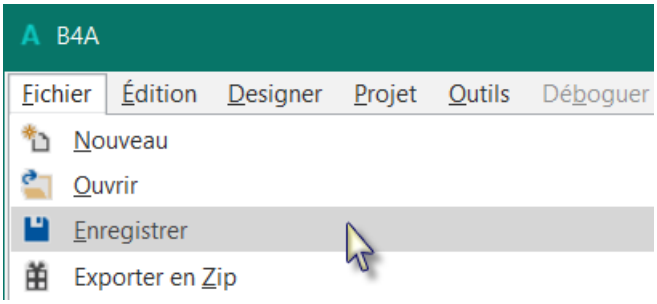


Le module Starter est supprimé.

Vous pouvez aussi laisser le module, sa suppression n'est pas obligatoire.

Enregistrez le projet.

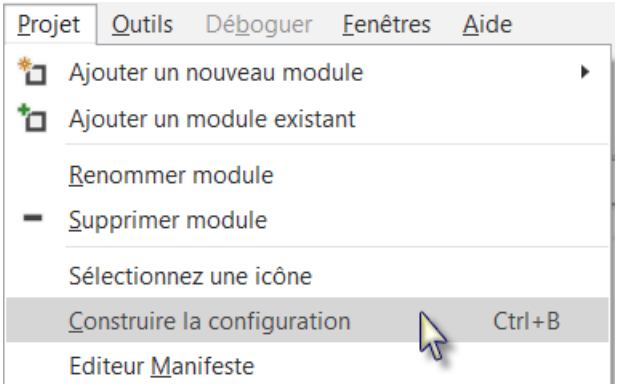
Vous devez d'abord enregistrer le projet avant de pouvoir utiliser le Designer.



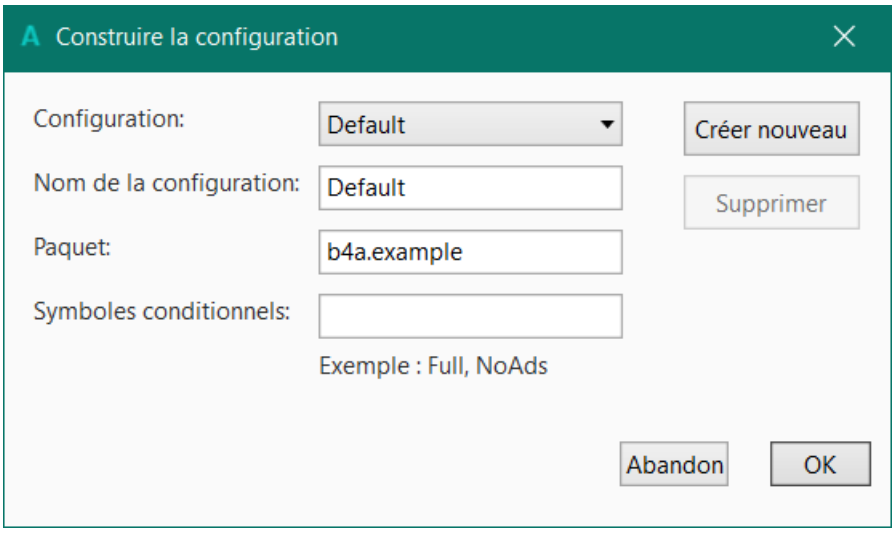
Créez un nouveau dossier MonPremierProgramme et enregistrez le projet avec le nom MonPremierProgramme.

Définissez un nom de paquet (Package Name).

Chaque projet doit avoir un nom de paquet.

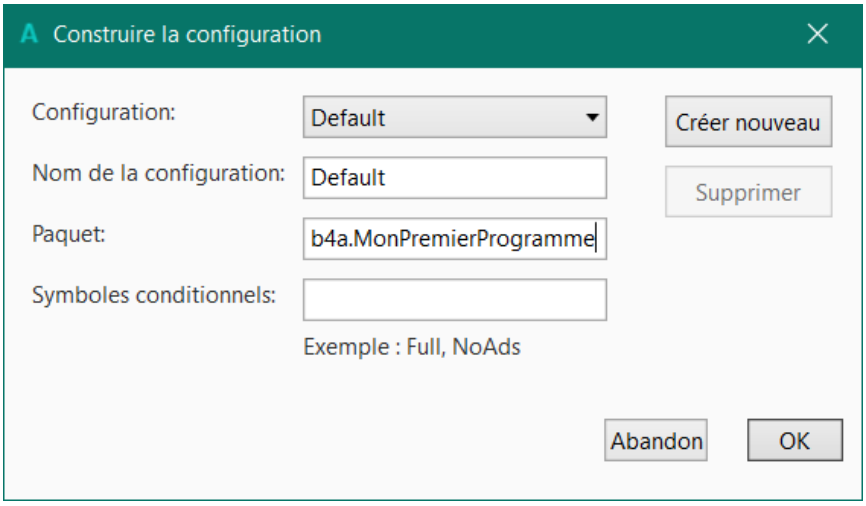


Dans le menu Projet cliquez sur Construire la configuration.



Cette fenêtre sera affichée :

Le nom de Paquet par défaut est b4a.example.



Nous changeons le nom en b4a.MonPremierProgramme.

Puis cliquez sur OK.

Définissez l'ApplicationLabel.

L'ApplicationLabel est le nom qui sera affiché sur le dispositif sous l'icône du projet.
Sur le haut de la zone code, vous voyez les deux lignes ci-dessous montrant deux 'Régions'.

```
1  #Region  Project Attributes
9
10 #Region  Activity Attributes
```

Les Régions sont des parties de code qui peuvent être réduites ou étendues.
Un clic sur ⊞ étend une Région.
Un clic sur ⊟ réduit une Région.
Les Régions sont expliquées dans le chapitre Réduire une Région dans le livret B4x EDI.

```
1  #Region  Project Attributes
9
10 #Region  Activity Attributes
11   #FullScreen: False
12   #IncludeTitle: True
13 #End Region
```

```
#Region  Project Attributes
   #ApplicationLabel: B4A Example
   #VersionCode: 1
   #VersionName:
   'SupportedOrientations possible values: unspecified, landscape or portrait.
   #SupportedOrientations: unspecified
   #CanInstallToExternalStorage: False
#End Region

#Region  Activity Attributes
   #FullScreen: False
   #IncludeTitle: True
#End Region
```

Le nom par défaut est `B4A Example`, mais nous le changeons en `MonPremierProgramme`.

Changez cette ligne :

```
   #ApplicationLabel: B4A Example
```

en

```
   #ApplicationLabel: MonPremierProgramme
```

Les autres lignes sont expliquées dans le chapitre *Project Attributes attributs projet* dans le livret B4x EDI.

Connectez un dispositif

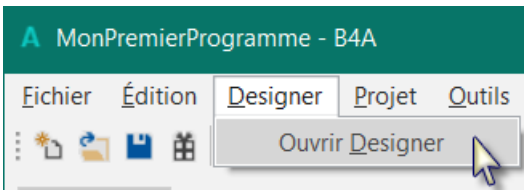
Pour tester le programme vous devez connecter un dispositif à l'EDI.

Vous pouvez connecter l'EDI à un dispositif via (voir chapitre précédent) :

- B4A-Bridge
- Câble USB

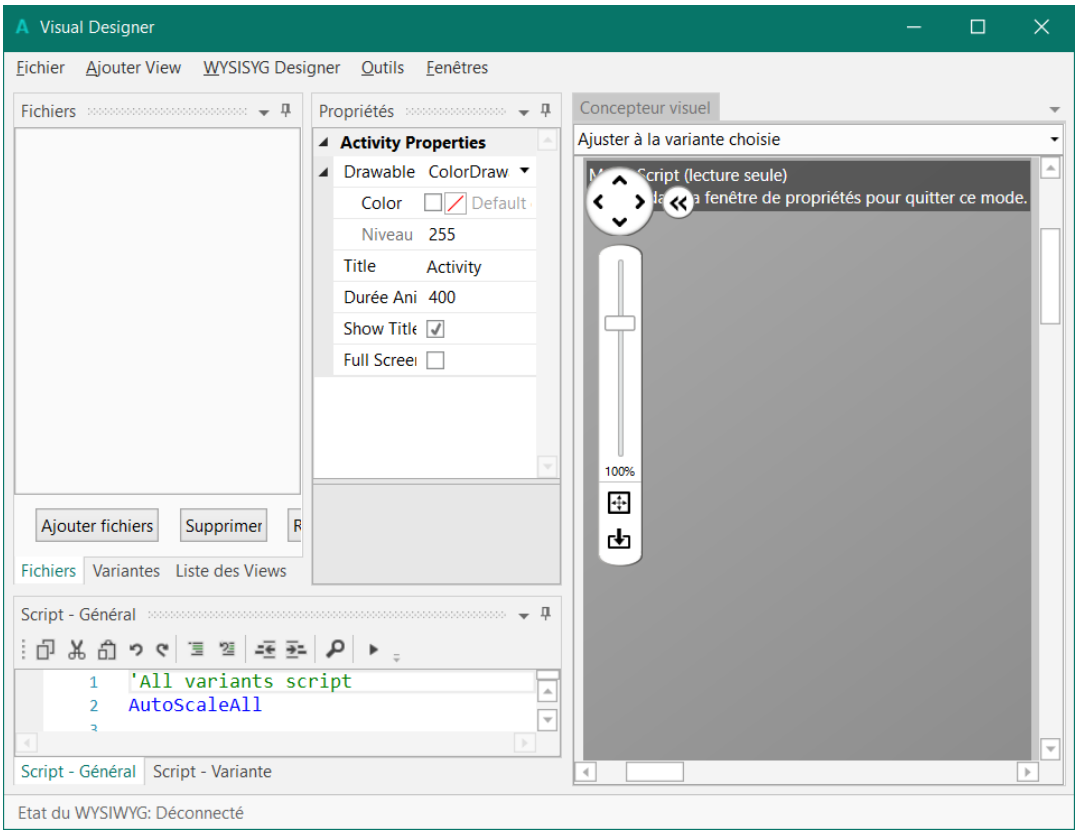
Il est aussi possible de connecter un émulateur, mais c'est déconseillé à cause de la lenteur.

Dans l'EDI lancez le Designer.



Cliquez sur Ouvrir Designer.

Le Visual Designer ressemble à l'image ci-dessous.



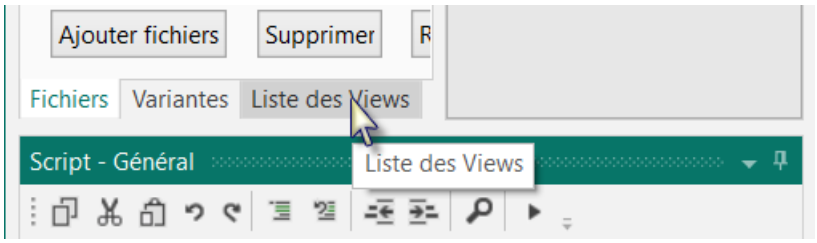
Il y a différentes fenêtres :

- Fichiers affiche tous les fichiers ajoutés au projet.
- Propriétés affiche toutes les propriétés de la view sélectionnée.
- Concepteur visuel affiche les views sur l'écran.
- Script - General permet de 'fignoler' des layouts (mises en page).

Le Designer est expliqué dans le livret B4xVisualDesigner.

Dans le premier projet nous nous intéressons seulement aux fenêtres ci-dessous :

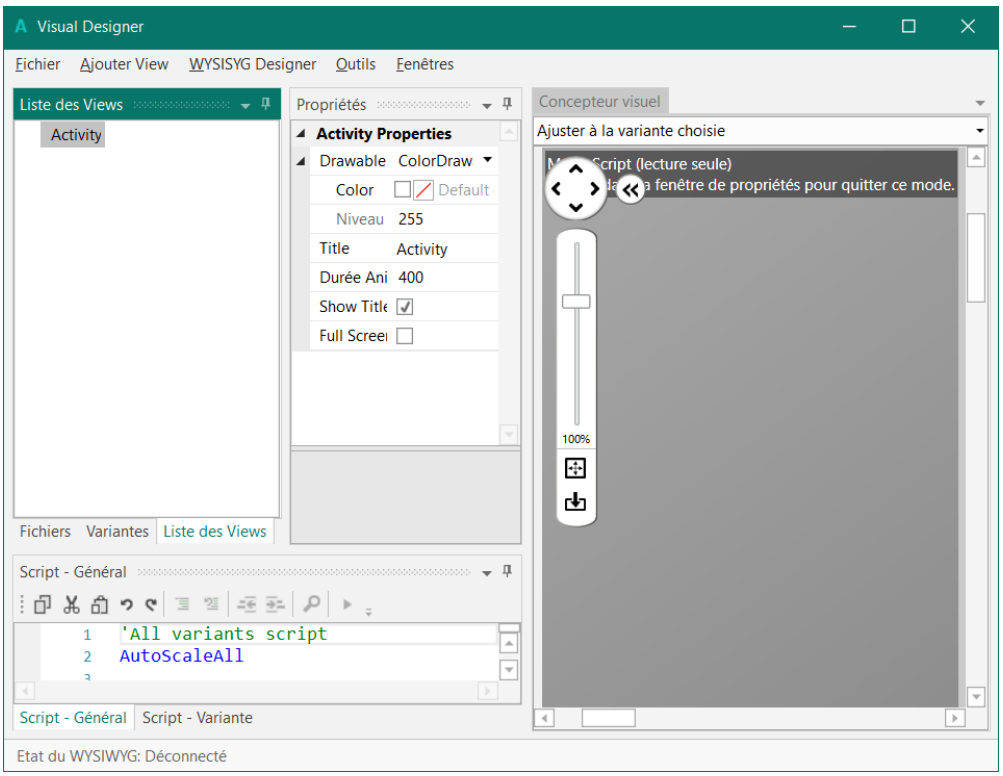
- Liste des views
- Propriétés
- Concepteur visuel

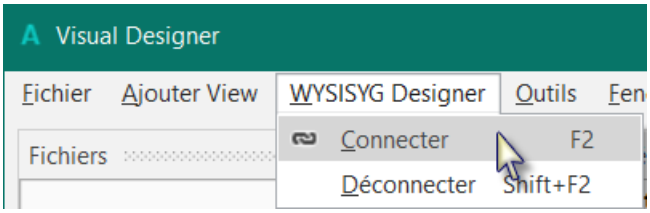


Affichons la Liste des Views.

Cliquez sur Liste des Views.

Le Designer ressemblera à l'image ci-dessous.

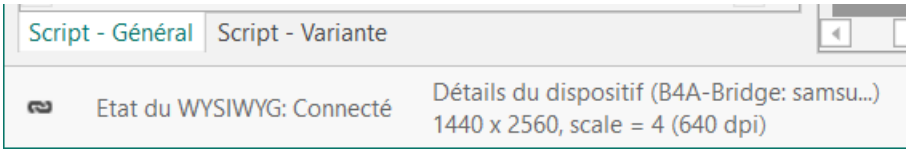




Pour afficher les views sur le dispositif, vous devez le connecter au Designer.

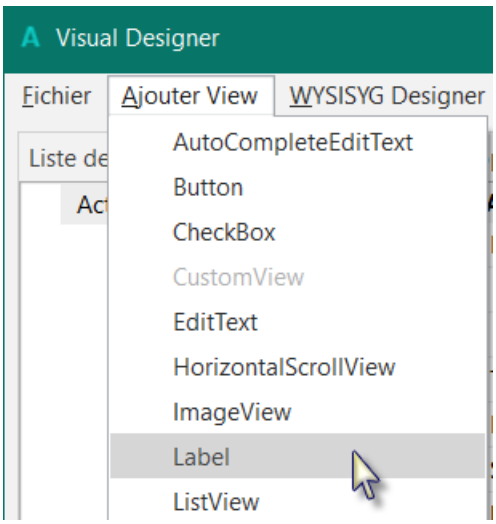
Dans le menu WYSISYG Designer cliquez sur Connecter.

Attendez jusqu'à ce que le dispositif et le Designer soient connectés. Ceci peut prendre du temps, soyez patient.
Vous verrez, dans le coin inférieur gauche, l'état de la connexion ainsi que les paramètres du dispositif connecté :



Maintenant nous ajoutons 2 Labels pour les nombres.

Dans le Designer, ajoutez un Label.



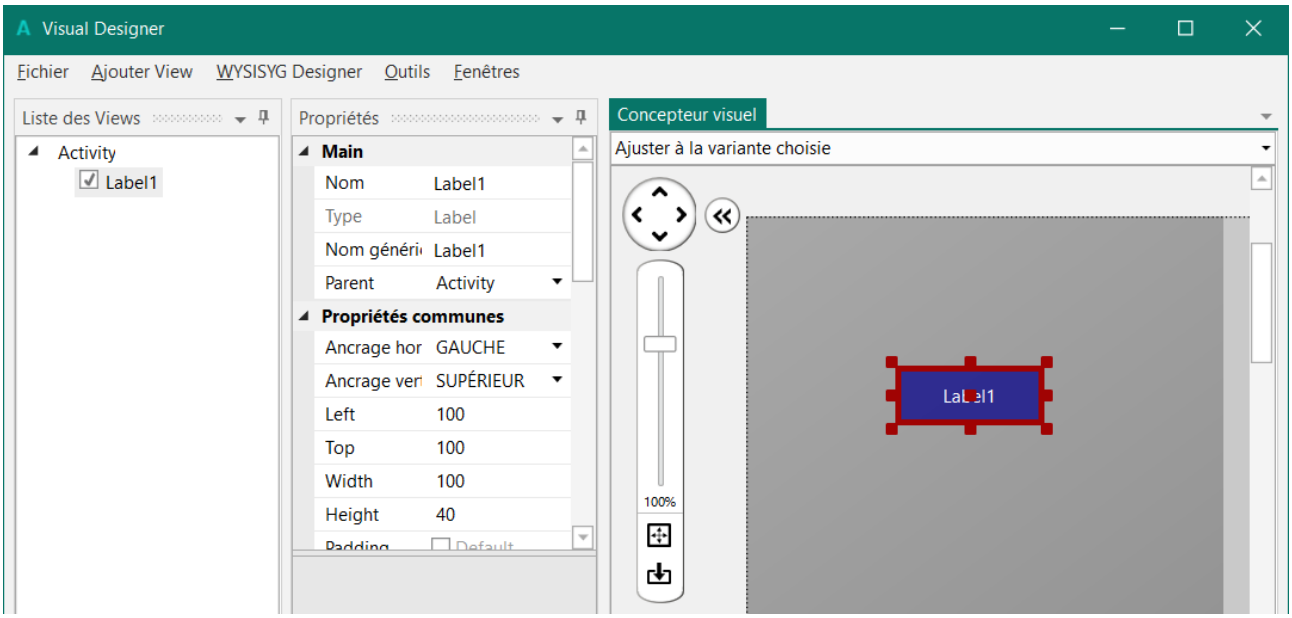
Dans le menu Ajouter View cliquez sur Label.

Nous voyons le Label avec son nom par défaut `Label1` dans les fenêtres suivantes :

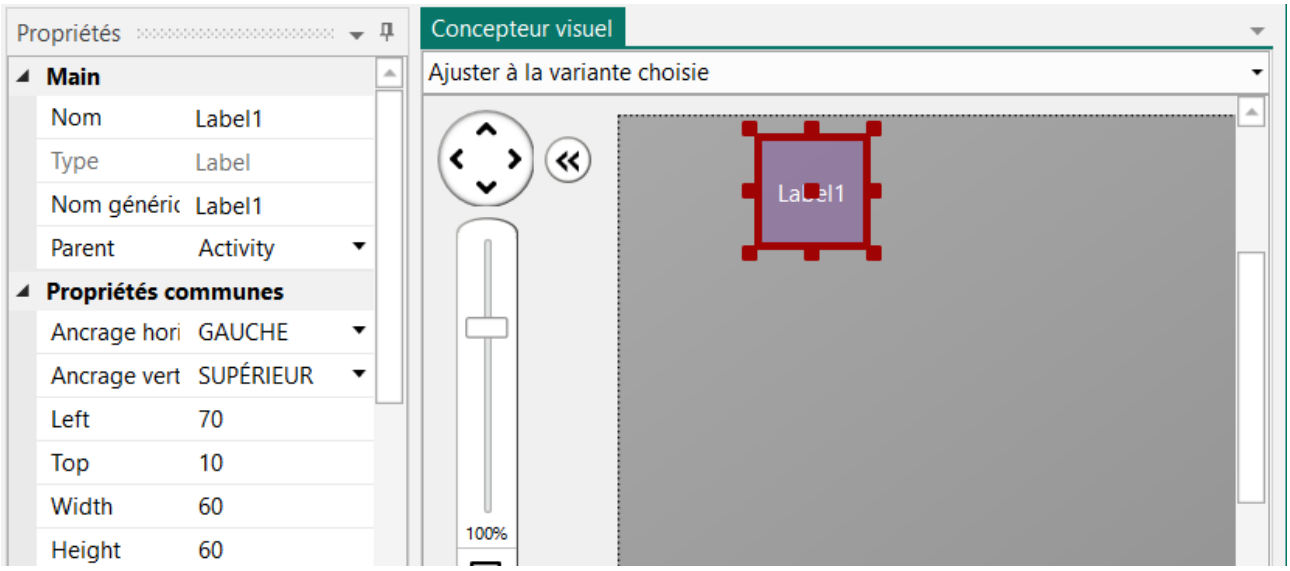
Liste des Views

Propriétés
Avec les propretés par défaut.

Concepteur visuel
à la position et aux dimensions par défaut.



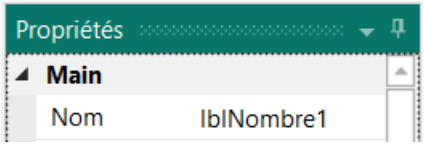
Redimensionnez et déplacez le Label avec les points rouges comme ci-dessous.



Les nouvelles propriétés Left, Top, Width et Height sont directement mises à jour dans la fenêtre Propriétés.
Vous pouvez aussi modifier ces propriétés directement dans la fenêtre Propriétés.

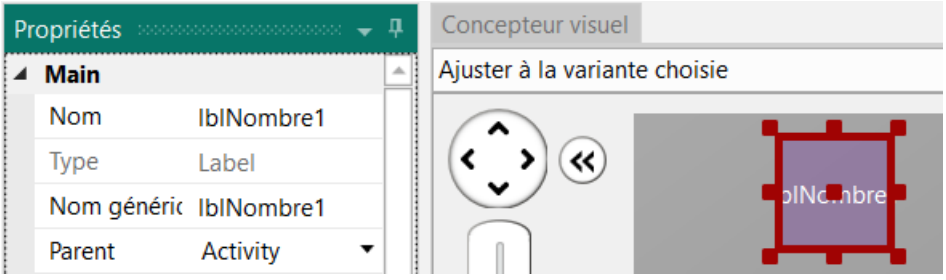
Nous allons changer les propriétés de ce premier Label pour nos besoins.

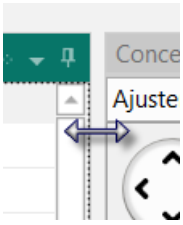
Le nom par défaut des Labels est Label suivi d'un nombre, dans notre cas Label1.
Nous changeons son nom en lblNombre1.
Les trois lettres au début 'lbl' correspondent à 'Label', et 'Nombre1' correspond au premier nombre.
Il est recommandé de donner des noms significatifs aux views, de cette manière nous savons directement le type de la view et sa fonction.



Pressez la touche Entrée ou cliquez quelque part ailleurs pour mettre à jour le nom dans les autres fenêtres et changer la propriété Nom générique des événements.

Main : Module Main.
Nom : Nom de la view.
Type : Type de la view. Dans notre cas, un Label, qui n'est pas éditable.
Nom générique … : Nom générique des routines qui gèrent les événements de la view.
Parent : View parent qui contient le Label.





Pour mieux voir les autres propriétés nous élargissons la fenêtre Propriétés.

Vérifions et modifions les autres propriétés :

Propriétés communes	
Ancrage horizontal	GAUCHE
Ancrage vertical	SUPÉRIEUR
Left	70
Top	10
Width	60
Height	60
Padding	Default
Enabled	☑
Visible	☑
Tag	
Text	5
FontAwesome Icons	
Material Icons	

Left, Top, Width et Height sont OK.
Ou, si les valeurs ne correspondent pas à l'image, veuillez les modifier.

Enabled, Visible sont OK
Tag, reste vide.
Text, nous définissons un nombre par défaut, 5.

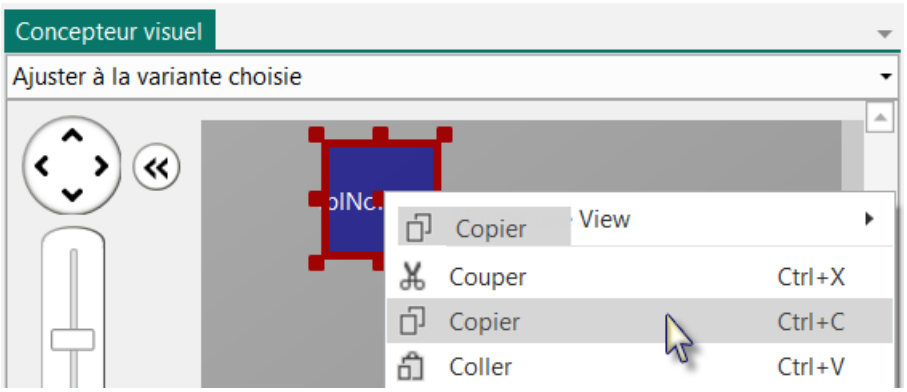
Propriétés texte	
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignment	CENTER_HORIZONTAL
Vertical Alignment	CENTER_VERTICAL
Size	36
Text Color	Default color
Single Line	☐
Ellipsize	NONE
Label Properties	
Drawable	ColorDrawable
Color	Default color
Niveau Alpha	0
Rayon des coins	0
Couleur du cadre	#FF000000
Épaisseur du cadre	0

Typeface, Style sont OK

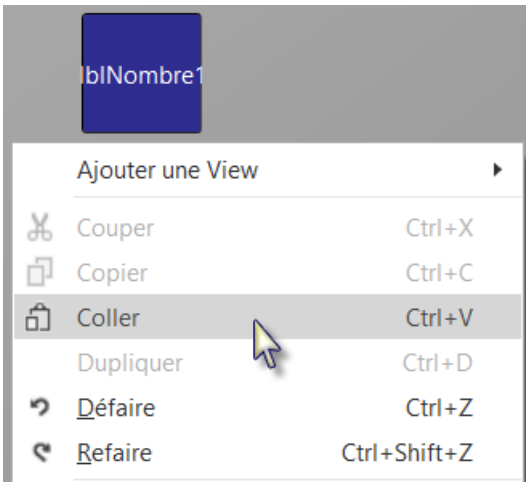
Horizontal Alignment, mis à CENTER_HORIZONTAL
Vertical Alignment, reste à CENTER_VERTICAL.
Size, nous mettons à 36

Les autres propriétés restent tel quel.

Nous avons besoin d'un deuxième Label similaire au premier pour le deuxième nombre. Au lieu d'ajouter une nouvelle view, nous copions le premier Label avec les mêmes propriétés. Seules les propriétés Nom et Left seront modifiées.

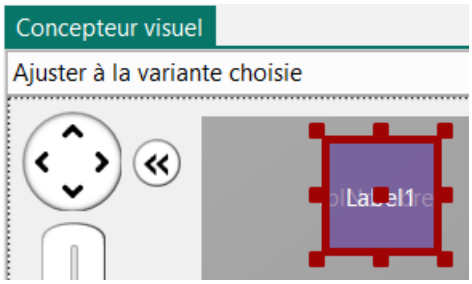


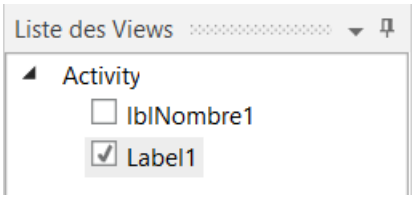
Dans le Concepteur visuel, cliquez avec le bouton droit sur lblNumber1 et cliquez sur Copier.



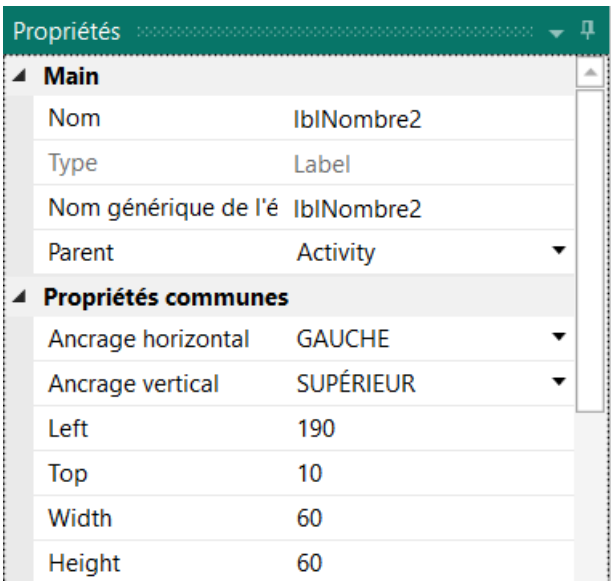
Cliquez quelque part ailleurs dans le Concepteur visuel et cliquez avec le bouton droit puis cliquez sur Coller.

Le nouveau Label couvre le précédent.



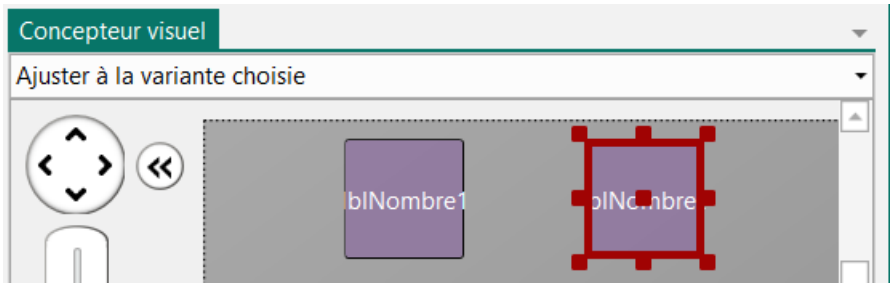


Nous voyons le nouveau Label a été ajouté dans la fenêtre Liste des Views.



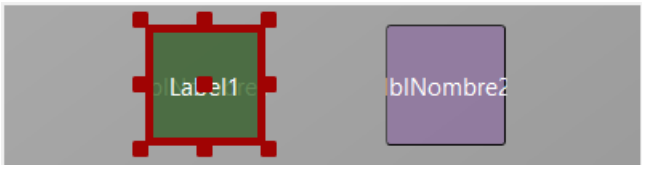
Changez le nom en lblNombre2.

Changez la propriété Left en 190.

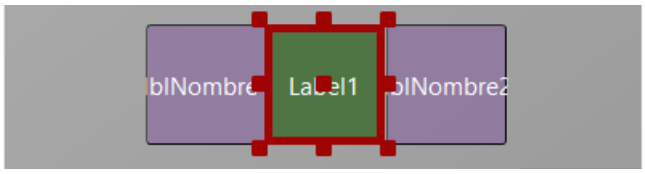


Le nouveau Label avec son nouveau nom se trouve dans sa nouvelle position.

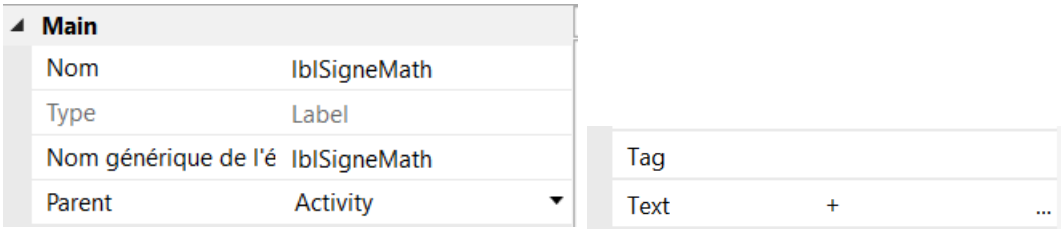
Maintenant, nous ajoutons un troisième Label pour le signe mathématique. Nous dupliquons lblNombre1. Une manière plus simple de copier une 'view'.
Dans le Concepteur visuel, cliquez avec le bouton droit sur lblNumber1 et cliquez sur Dupliquer.



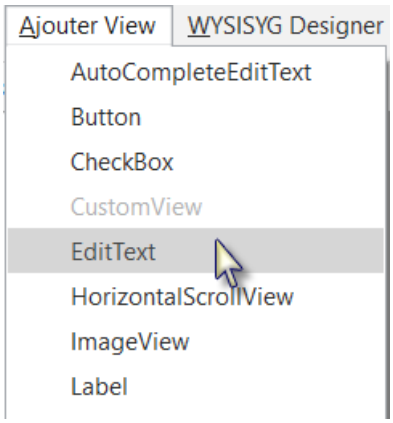
Le nouveau Label couvre lblNumber1.



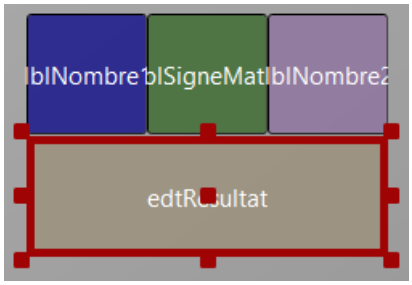
Positionnez-le entre les deux premier et changez son nom en lblSigneMath et sa propriété Text en '+'.



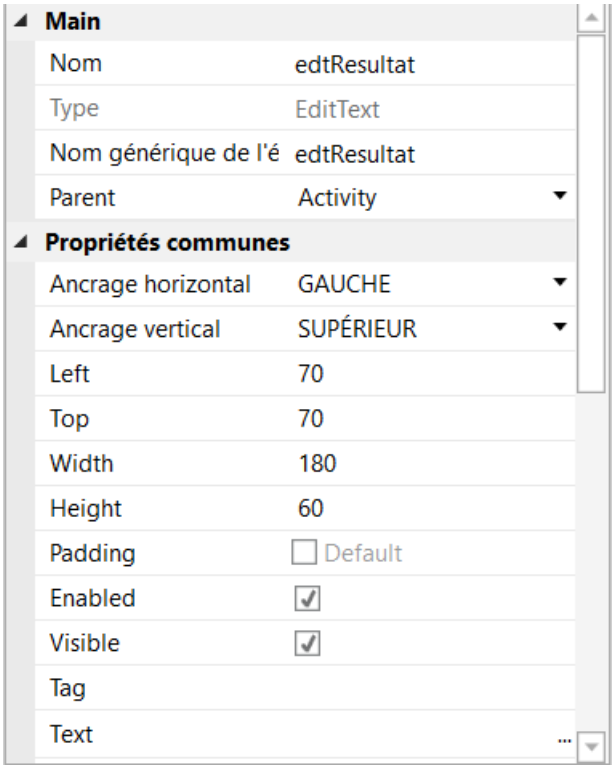
Maintenant nous ajoutons une view EditText.



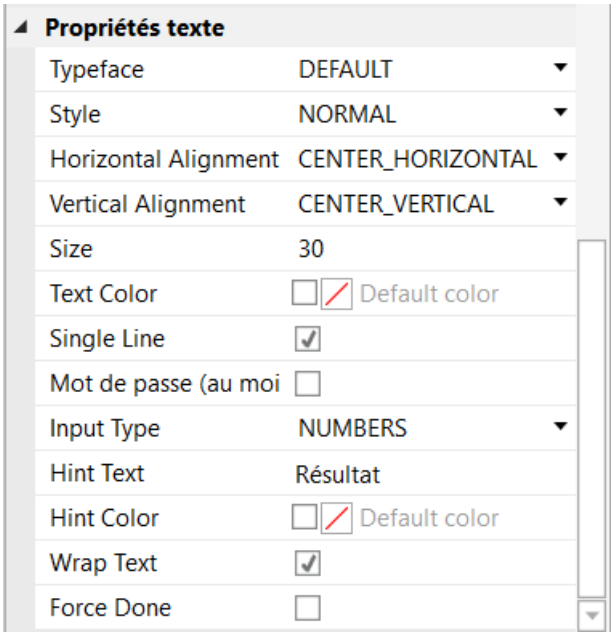
Dans le Designer, dans le menu Ajouter View.
Cliquez sur EditText.



Positionnez le sous les Labels et changez son nom en edtResultat. 'edt' signifie EditText et 'Resultat' sa fonction.



Nous modifions les propriétés ci-dessous.
Nom en edtResultat



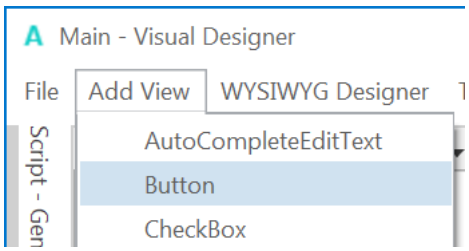
Horizontal Alignment en CENTER_HORIZONTAL

Size en 30

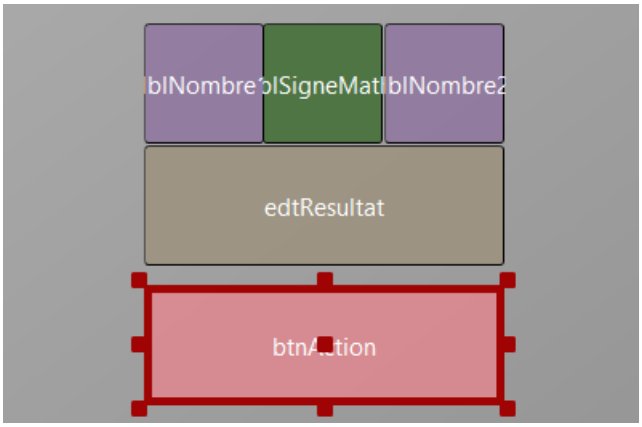
Input Type en NUMBERS
Hint Text en Résultat

Nous définissons Input Type en NUMBERS pour que l'utilisateur ne puisse entrer que des chiffres.

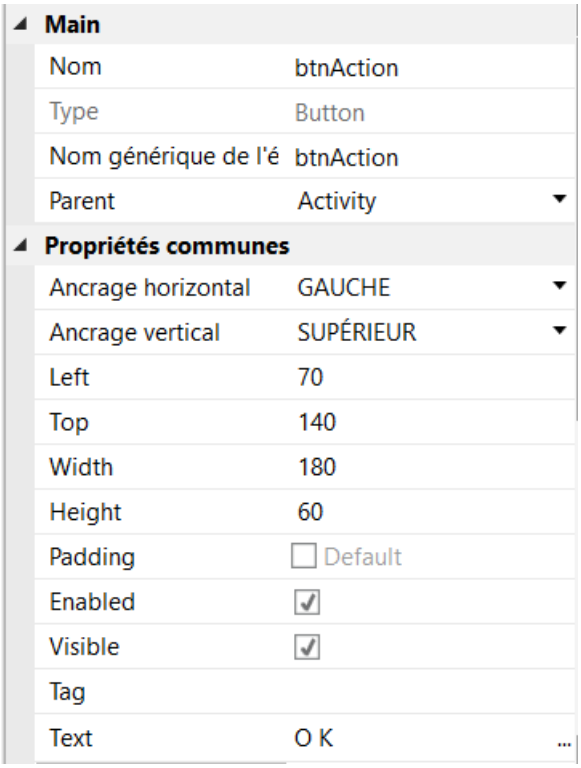
Hint Text représente le texte qui sera affiché dans la view EditText si aucun texte n'a encore été entré.



Maintenant nous ajoutons une view Button (bouton) qui, lorsque pressé, va soit vérifier le résultat que l'utilisateur a fourni, ou générer un nouveau problème d'arithmétique en fonction des réponses de l'utilisateur.



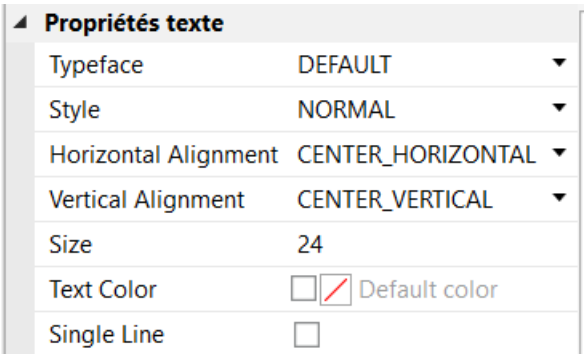
Positionnez la view Button et redimensionnez-la.



Modifiez les propriétés ci-dessous.

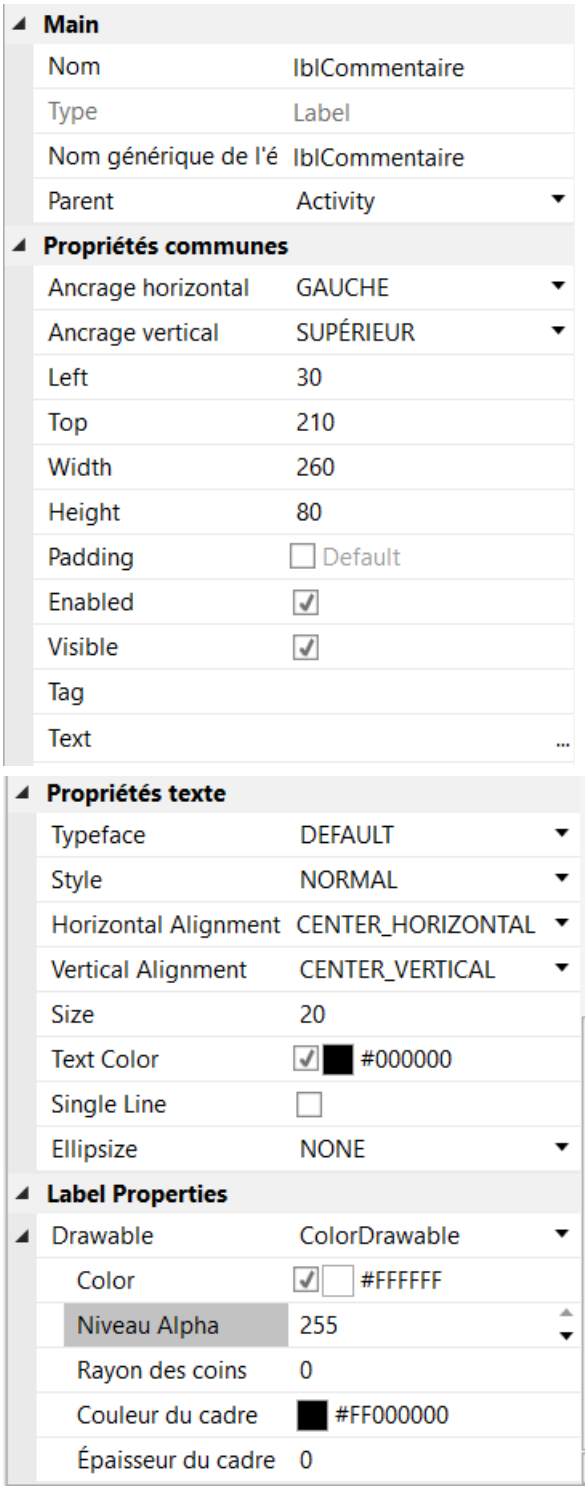
Nom en btnAction

Text en O K (avec un espace entre O et K)



Size en 24

Ajoutons encore un Label pour les commentaires.
Positionnez-le en dessous du Button btnAction et redimensionnez-le.



Modifiez les propriétés ci-dessous :
Nom en lblCommentaire

Left, Top Width et Height
comme à gauche.

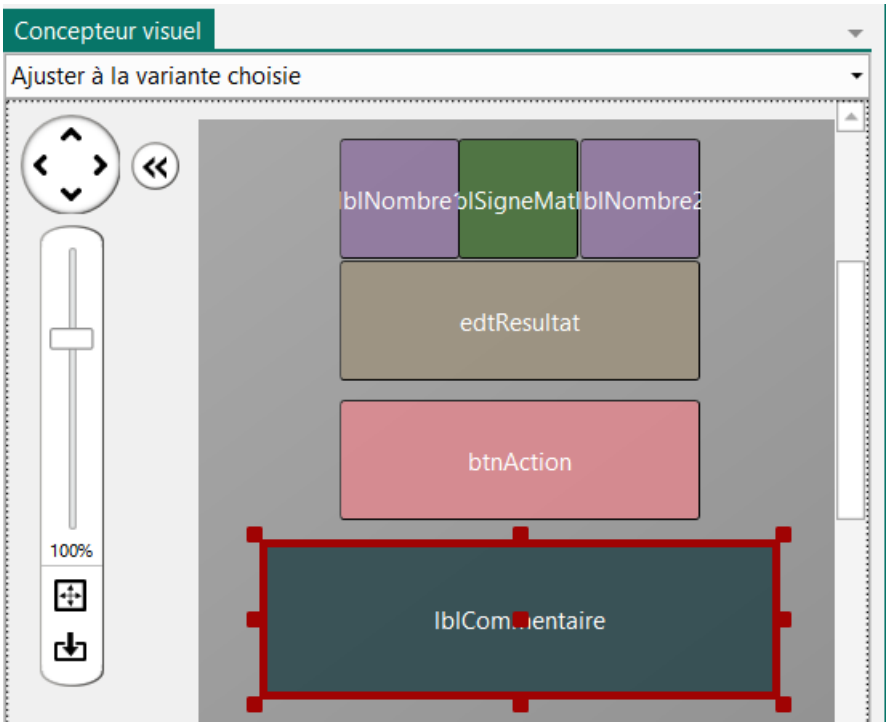
Horizontal Alignment CENTER_HORIZONTAL

Text Color en #000000
Nous définissons la couleur du texte (Text Color) en noir (#000000).

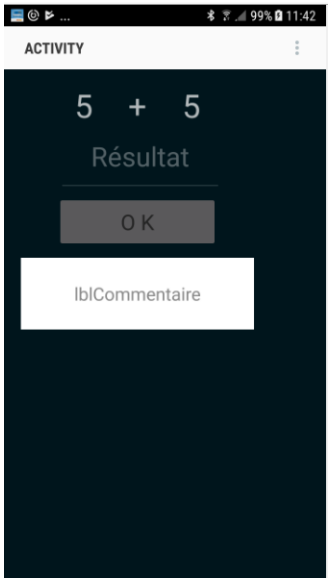
Color en #FFFFFF (blanc)
Alpha en 255

La couleur de fond par défaut d'un Label est noir et transparent. Nous la modifions en blanc.
Nous modifions sa propriété Alpha = 255, pour la rendre opaque.

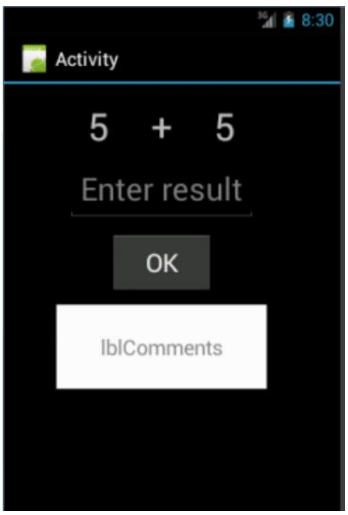
Le résultat ressemblera à l'image ci-dessous dans le Concepteur visuel.



Et sur un dispositif ou dans l'Emulateur.



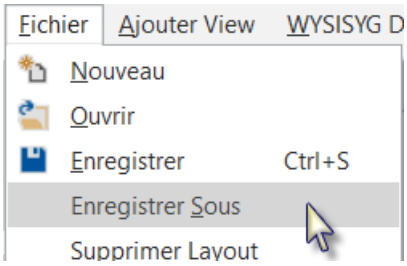
Samsung S6



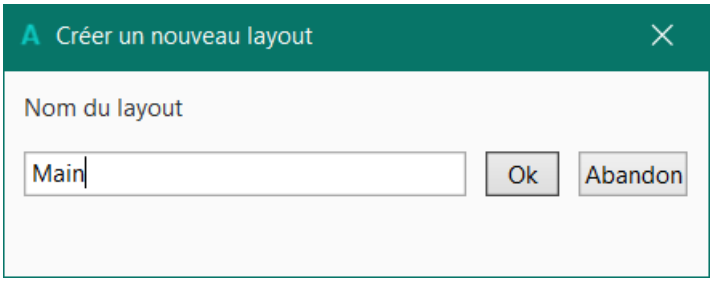
Android 4.2 Emulateur

Android 2.2 Emulateur

Il est temps d'enregistrer le layout.



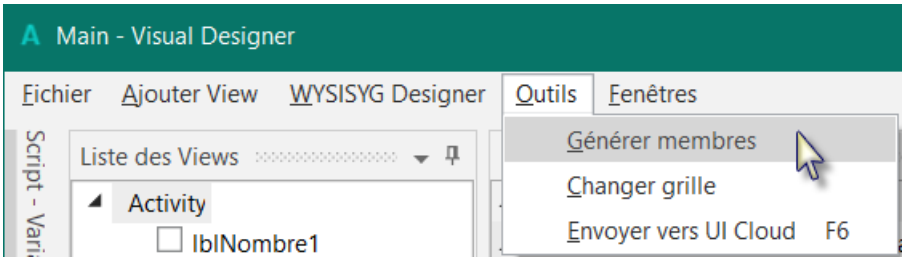
Dans le menu Fichier cliquez sur Save As... et enregistrez-le avec le nom 'Main'.



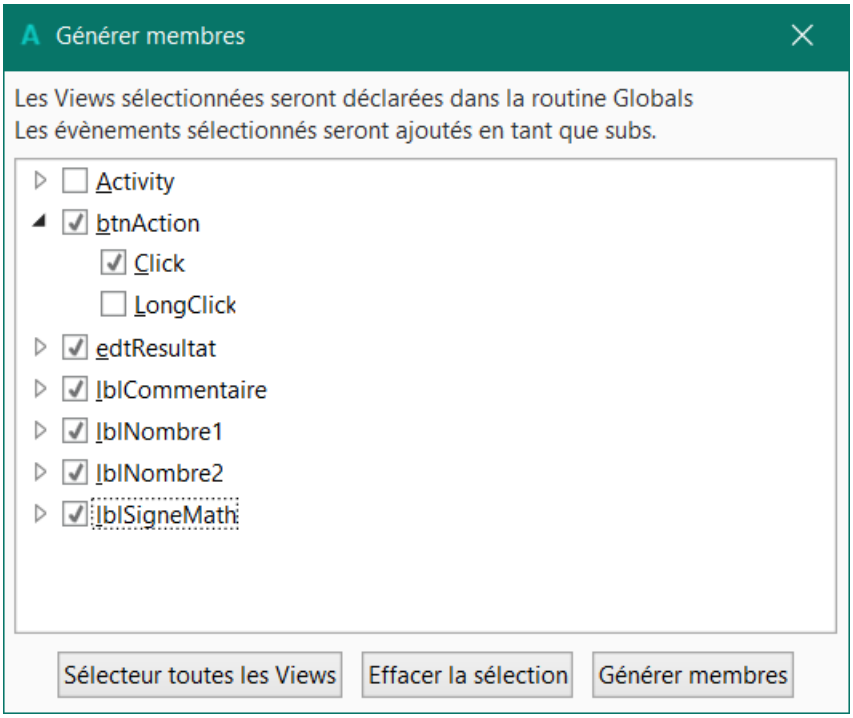
Cliquez sur Ok .

Pour écrire le code du projet nous avons besoin de références aux views définies dans le layout. Ceci se fait avec l'outil *Générer membres* dans le Designer.

L'outil *Générer membres* permet de générer automatiquement des références aux différentes views et de générer les cadres de routine d'événement.



Dans le menu Outils cliquez sur Générer membres pour ouvrir le générateur.



Dans cette fenêtre nous trouvons toutes les views du layout actuel.
Nous cochons toutes les views ainsi que l'événement Click pour le Button btnAction.
Cocher une view dans la liste, comme ▷ ☑ edtResultat, génère une référence à cette view dans la routine Sub Globals.
Ces déclarations sont nécessaires pour que le compilateur reconnaisse les views et également pour la fonction d'autocomplétion.

```
Private btnAction As Button
Private edtResultat As EditText
Private lblCommentaire As Label
Private lblSigneMath As Label
Private lblNombre1 As Label
Private lblNombre2 As Label
```

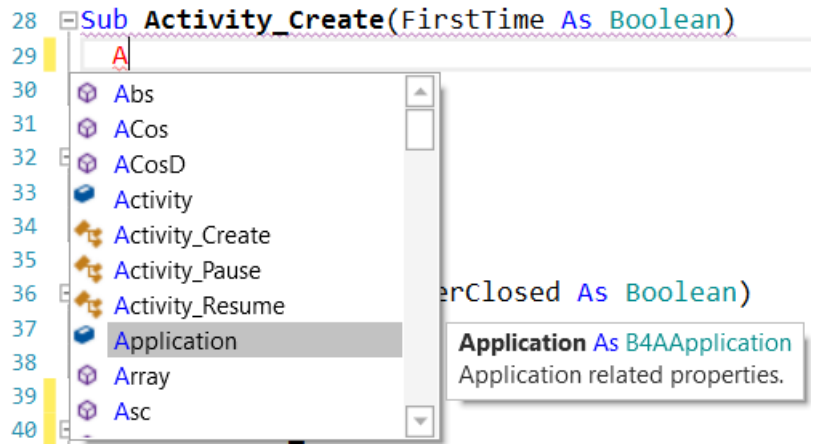
Cocher un événement, comme ☑ Click. Génère les cadres de la routine événement.

```
Sub btnAction_Click

End Sub
```

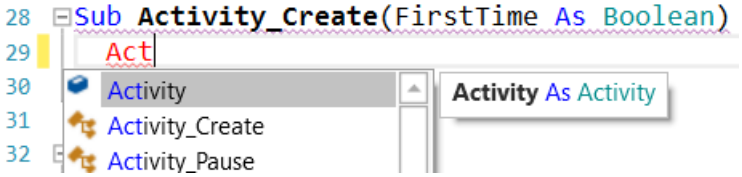
Cliquez sur Générer membres pour générer les références et les cadres de routines.

Retournons dans l'EDI pour écrire le code.
Tout d'abord, nous devons charger notre fichier layout. Au début de la routine "Activity_Create", procédez comme ci-dessous. Vous pouvez supprimer les deux premières lignes en vert.
Nous allons écrire la ligne de code suivante `Activity.LoadLayout("Main")`.

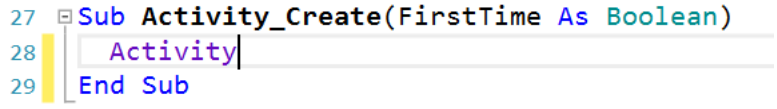
- Entrez 'A', dès que vous commencez à entrer du code, la fonction d'autocomplétion affiche les mots clé commençant par 'a', et le mot le plus approprié est présélectionné.



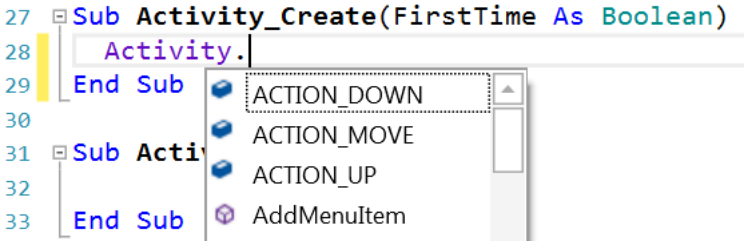
- Continuez et écrivez 'Act'.



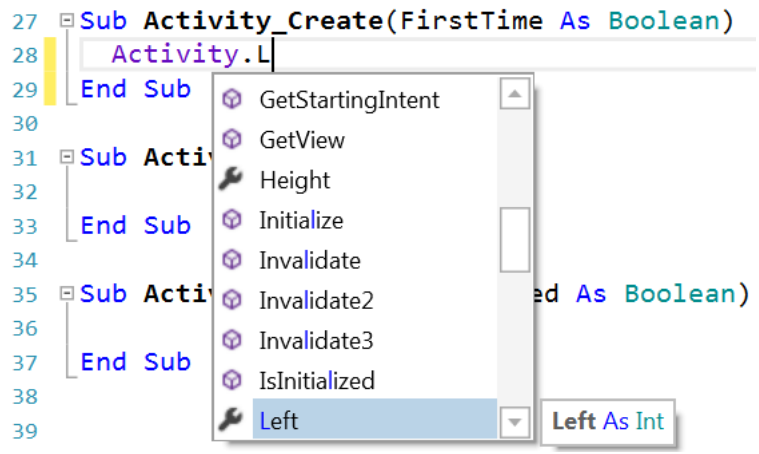
- Pressez la touche 'Entrée' ou cliquez sur Activity.



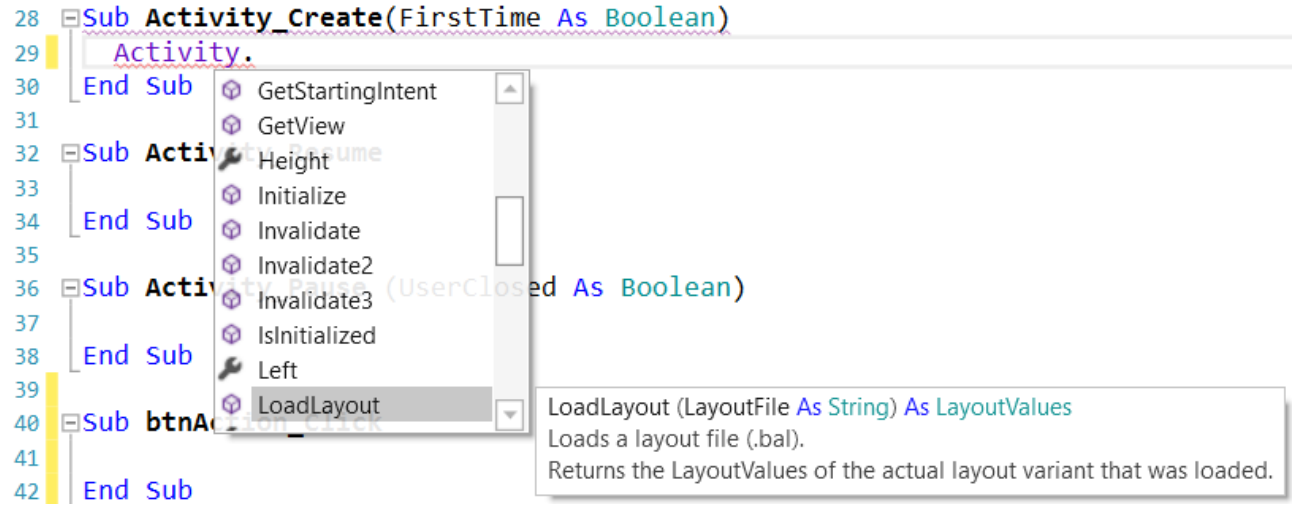
- Nous avons maintenant le mot clé Activity, écrivez un point.



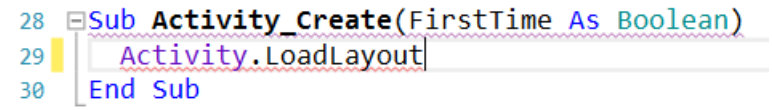
- La fonction d'autocomplétion affiche toutes les propriétés de la view.
- Écrivez 'L', la fonction d'autocomplétion affiche les propriétés commençant par 'L'



- Pressez la flèche du clavier vers le bas jusqu'à ce que LoadLayout est en surimpression avec son l'aide en ligne donnant une explication sur la propriété ou méthode.

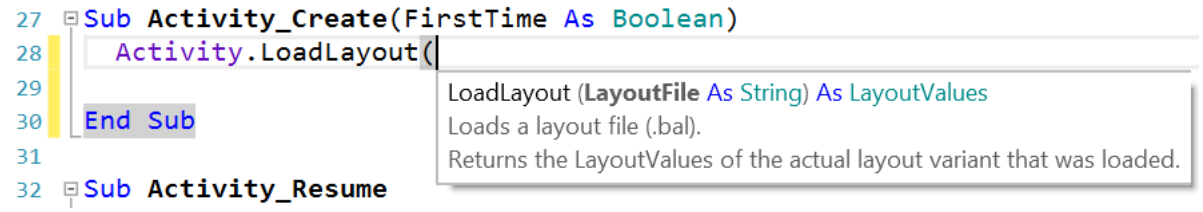


- Pressez la touche Entrée pour ajouter LoadLayout.



Remarquez que la ligne est soulignée en rouge indiquant une erreur, car la ligne est incomplète, il manque des paramètres.

- Écrivez '(', l'aide en ligne affiche les paramètres requis pour cette méthode.



- Écrivez "Main")

```
Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("Main")
End Sub
```

Nous voulons démarrer un nouveau problème dès le démarrage du programme. Pour ça, nous ajoutons un appel à la routine NouveauProbleme.

```
Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("Main")
   NouveauProbleme
End Sub
```

NouveauProbleme est en rouge indiquant une erreur 'Variable non déclarée …' car le compilateur ne connait pas encore ce mot.

Générer un nouveau problème d'addition consiste à générer aléatoirement deux nouveaux nombres entre 1 et 9 (inclus) pour Nombre1 et Nombre2, et afficher leur valeur dans les Labels en utilisant la propriété 'Text' de `lblNombre1` et `lblNombre2`.

Pour ce faire nous ajoutons le code ci-dessous :
Dans la routine Sub Globals nous déclarons deux variables pour les deux nombres.

```
   Public Nombre1, Nombre2 As Int
End Sub
```

Et ajoutons la routine 'NouveauProbleme' :

```
Private Sub NouveauProbleme
   Nombre1 = Rnd(1, 10)                ' Génère un nombre aléatoire entre 1 et 9
   Nombre2 = Rnd(1, 10)                ' Génère un nombre aléatoire entre 1 et 9
   lblNombre1.Text = Nombre1           ' Affiche Nombre1 dans le Label lblNombre1
   lblNombre2.Text = Nombre2           ' Affiche Nombre2 dans le Label lblNombre2
   lblCommentaire.Text = "Entrez le résultat" & CRLF & "et cliquez sur OK"
   edtResultat.Text = ""               ' Vide edtResult.Text
End Sub
```

Cette fonction, `Rnd(1, 10)`, génère un nombre aléatoire entre '1' (inclusif) et '10' (exclusif), donc entre '1' et '9'.

La ligne ci-dessous affiche un commentaire dans le Label `lblCommentaire` :
`lblComments.Text = "Entrez le résultat" & CRLF & "et cliquez sur OK"`
`CRLF` est le caractère NouvelleLigne.

Maintenant nous ajoutons le code pour l'événement 'Click' du Button `btnAction`.

Nous avons deux cas :
- Lorsque le texte du bouton est égal à "O K" (avec un espace entre le O et le K), ce qui signifie qu'un nouveau problème est affiché et que le programme attend que l'utilisateur entre le résultat et presse le bouton.
- Lorsque le texte du bouton est égal à "Nouveau", ce qui signifie que l'utilisateur a entré un résultat correct et lorsqu'il presse le bouton un nouveau problème est généré.

```
Sub btnAction_Click
   If btnAction.Text = "O K" Then
      If edtResult.Text = "" Then
         Msgbox("Il n'y a pas de résultat","E R R E U R")
      Else
         CheckResult
      End If
   Else
      New
      btnAction.Text = "O K"
   End If
End Sub
```

`If btnAction.Text = "O K" Then` vérifie que le texte du bouton est égal à "O K".
Si oui, nous vérifions si EditText est vide (pas de résultat).
Si oui, nous affichons un message indiquant qu'il n'y a pas de résultat dans EditText.
Si non, nous vérifions si le résultat est juste ou faux.
Si non, nous générons un nouveau problème, modifions le texte du bouton en "O K" et vidons EditText.

La dernière routine vérifie le résultat.

```
Sub TestResultat
   If edtResultat.Text = Nombre1 + Nombre2 Then
      lblCommentaire.Text = "B O N  résultat" & CRLF & "Cliquez sur Nouveau"
      btnAction.Text = "Nouveau"
   Else
      lblCommentaire.Text = "M A U V A I S  résultat" & CRLF & "Entrez un nouveau
résultat" & CRLF & "et cliquez sur O K"
   End If
End Sub
```

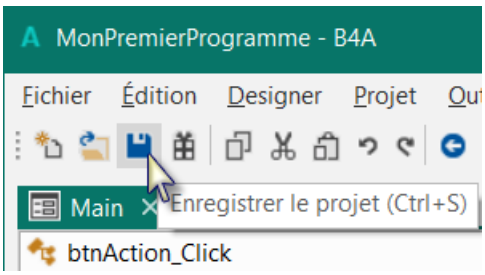
Avec `If edtResultat.Text = Nombre1 + Nombre2 Then` nous vérifions si le résultat est juste.

Si oui, nous affichons dans le Label `lblCommentaire` le texte ci-dessous :
'Résultat J U S T E'
'Cliquez sur Nouveau'
Et nous modifions le texte du bouton en " Nouveau ".
Si non, nous affichons dans le Label `lblCommentaire` le texte ci-dessous :
'Résultat F A U X'
'Entrez un nouveau résultat'
Et cliquez sur.

Dans la partie gauche de l'éditeur nous voyons une ligne jaune verticale, qui signifie que le code a été modifié.

```
65 Private Sub TestResultat
66    If edtResultat.Text = I
67      lblCommentaire.Text =
68      btnAction.Text = "Nou
69    Else
70      lblCommentaire.Text =
71    End If
72 End Sub
```

Si nous cliquons sur  pour enregistrer le projet, la ligne passe en vert indiquant que le code a été modifié mais déjà enregistré. Vous pouvez aussi presser Ctrl + S pour enregistrer le projet.

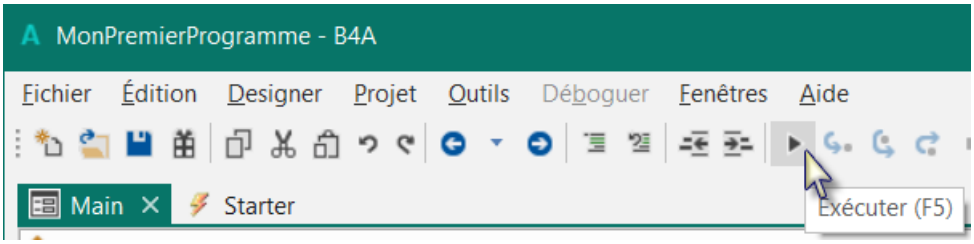



```
65 Private Sub TestResultat
66    If edtResultat.Text = I
67      lblCommentaire.Text =
68      btnAction.Text = "Nou
69    Else
70      lblCommentaire.Text =
71    End If
72 End Sub
```

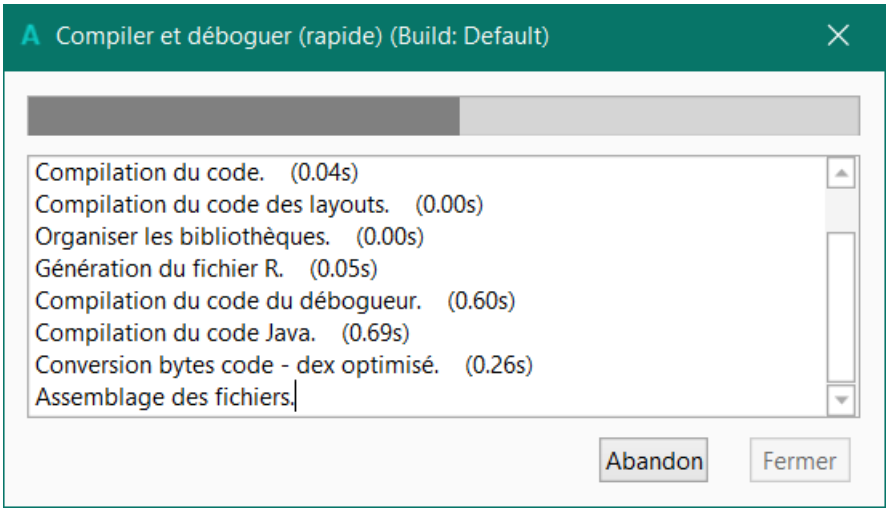
Lorsque nous quittons l'EDI et chargeons le projet à nouveau la ligne verte disparait.

Compilons le programme et transférons-le sur le dispositif.

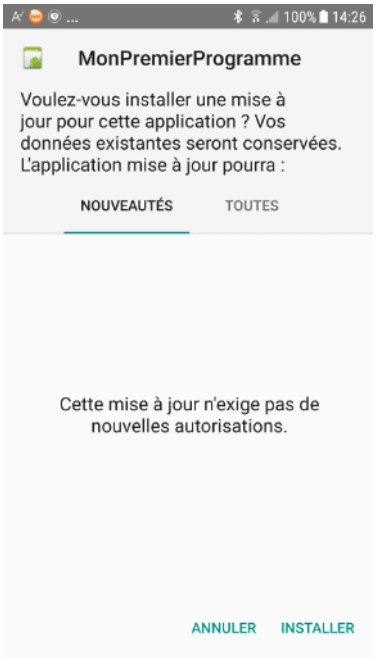
Dans l'EDI cliquez sur ▶ :

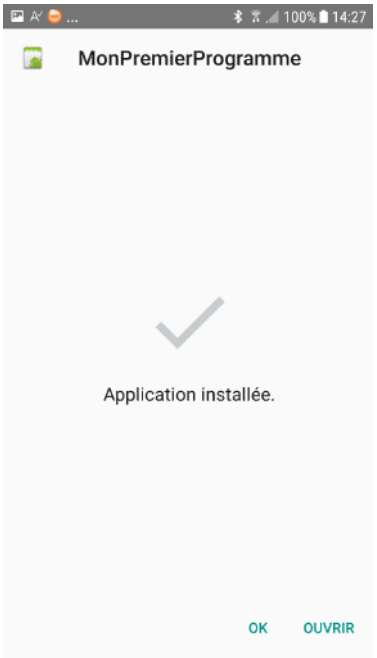


La compilation du projet démarre et la fenêtre ci-dessous affiche l'évolution de la compilation.



Lorsque le bouton Fermer est actif, la compilation et le transfert du projet sont terminés.
Sur le dispositifsi vous êtes connectés via B4A-Bridge, vous devez encore :



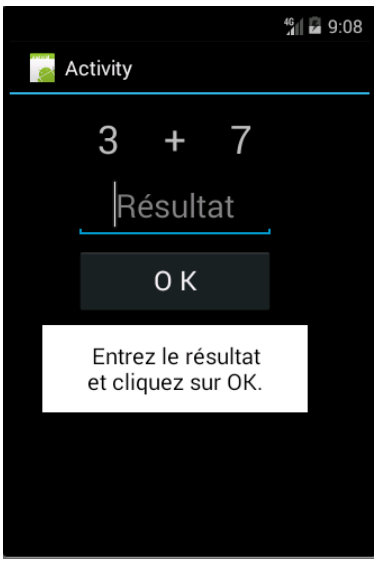


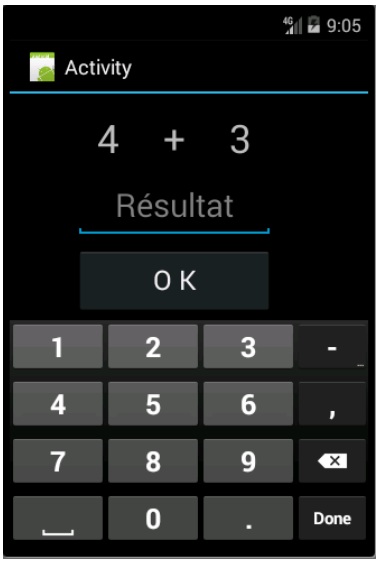
Confirmer l'installation puis ouvrir le projet.

Sur le dispositif vous obtenez un écran similaire à celui-ci-dessus, qui peut avoir une apparence différente selon la version d'Android, avec des nombres différents.

Sur le dispositif vous devez utiliser le clavier virtuel.
Pressez sur l'EditText pour afficher le clavier virtuel.

Nous pouvons évidemment faire des améliorations esthétiques dans la mise en page, mais ça n'était pas le but principal du premier projet et fait l'objet du chapitre suivant.





Sur certains dispositifs il se peut que les commentaires soient cachés par le clavier virtuel.

Cet inconvénient sera amélioré dans le chapitre suivant 'Second programme B4A', où nous créerons notre propre clavier.

2.7 Second programme B4A (SecondProgramme.b4a)

Ce projet est disponible dans le dossier SecondProgramme :
CodesSource\SecondProgramme\B4A\SecondProgramme.b4a

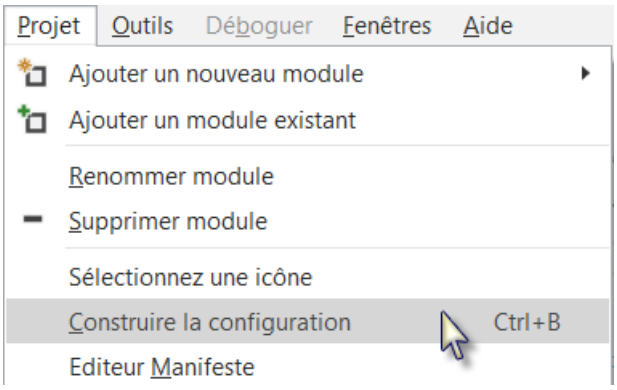
Améliorations par rapport à “MonPremierProgramme”.

Nous ajoutons un clavier numérique dans le layout pour éviter que le clavier virtuel du dispositif ne couvre les commentaires.

Créez un nouveau dossier “SecondProgramme”. Copiez tous les fichiers et sous-dossiers de MonPremierProgramme dans SecondProgramme et renommez les fichiers MonPremierProgramme.b4a en SecondProgramme.b4a et MonPremierProgramme.b4a.meta en SecondProgramme.b4a.meta.

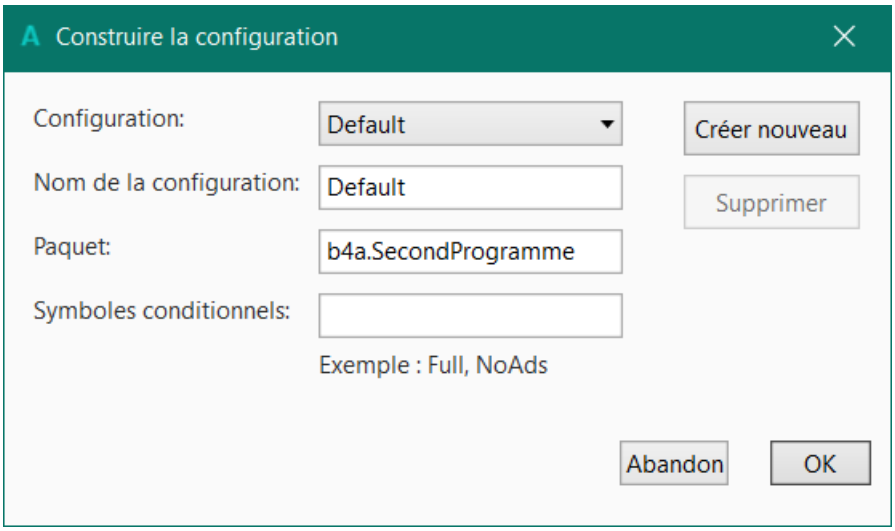
Chargez le nouveau projet dans l’EDI.

Nous devons modifier le nom du paquet.



Dans le menu Projet de l’EDI.

Cliquez sur Construire la configuration.



Changez le nom du Paquet en b4a.SecondProgramme.

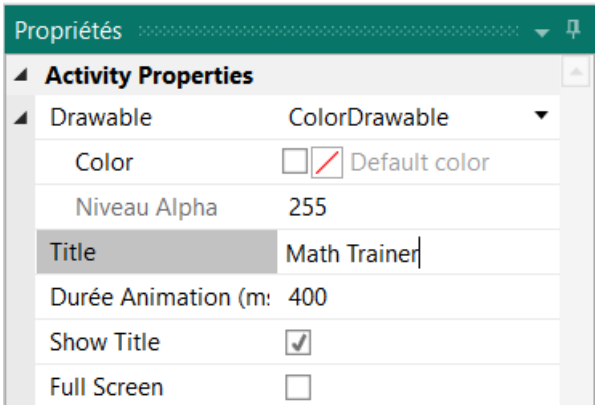
Cliquez sur OK.

Nous modifions aussi le paramètre ApplicationLabel tout en haut dans le code.

```
#Region  Project Attributes
   #ApplicationLabel: SecondProgramme
```

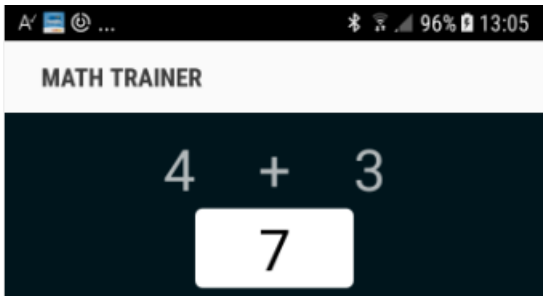
Lancez le Designer. Vous pouvez déjà connecter un dispositif ou un émulateur.

Dans le Concepteur visuel cliquez quelque part en dehors d'une view pour activer l'Activity.



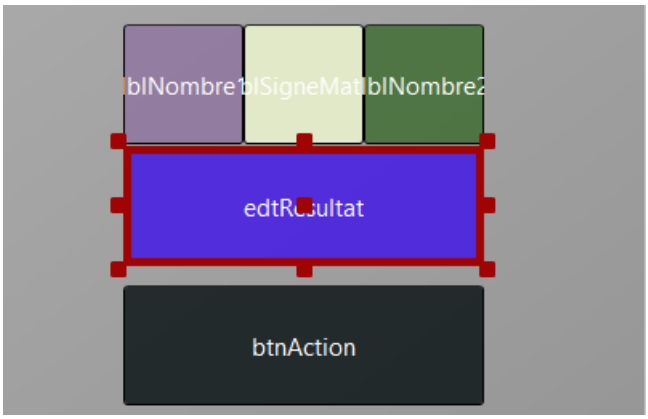
Modifiez le Titre en Math Trainer.

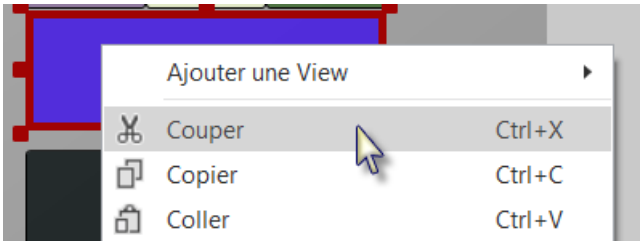
Ceci modifiera le titre de l'application lors de l'exécution.



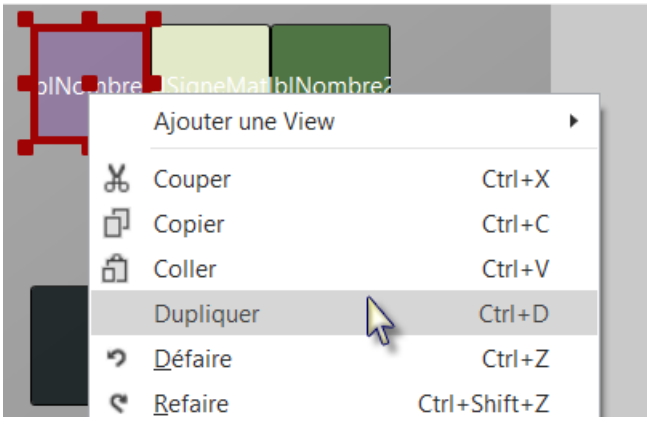
Nous remplaçons la view EditText `edtResultat` par un nouveau Label.

Dans le Concepteur visuel cliquez sur la view `edtResultat`.

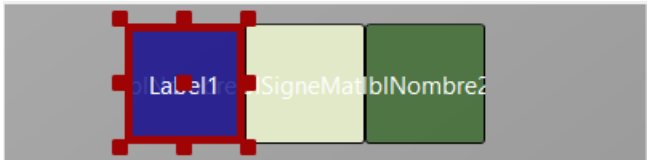




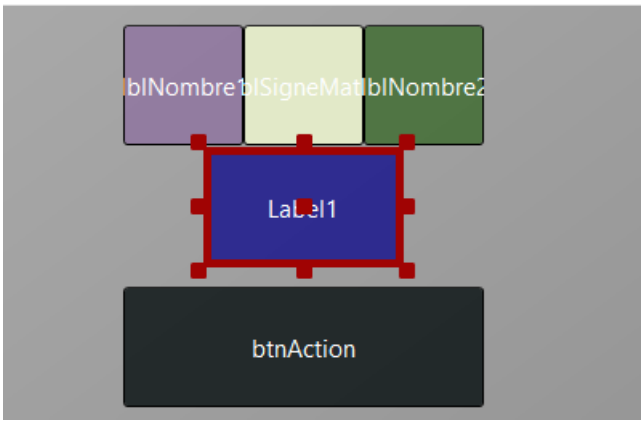
Cliquez avec le bouton droit sur `edtResultat` puis cliquez sur Couper.



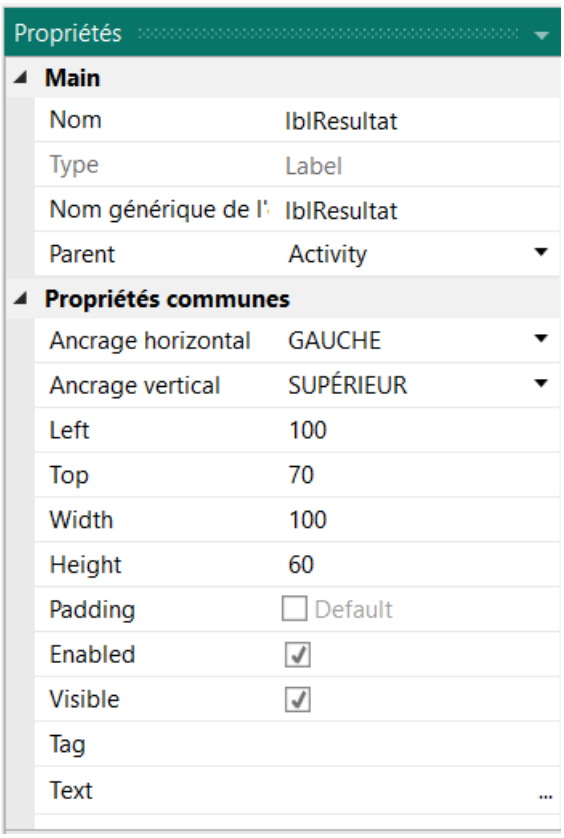
Cliquez avec le bouton droit sur lblNombre1 puis cliquez sur Dupliquer.



Le nouveau Label couvre lblNombre1.



Déplacez-le entre les Labels supérieurs et le bouton puis redimensionnez-le.

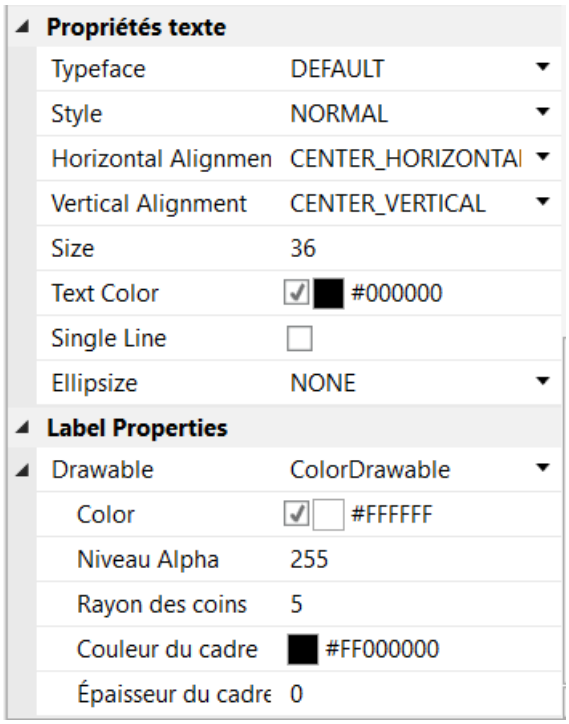


Modifiez les propriétés suivantes :

Nom en IblResultat

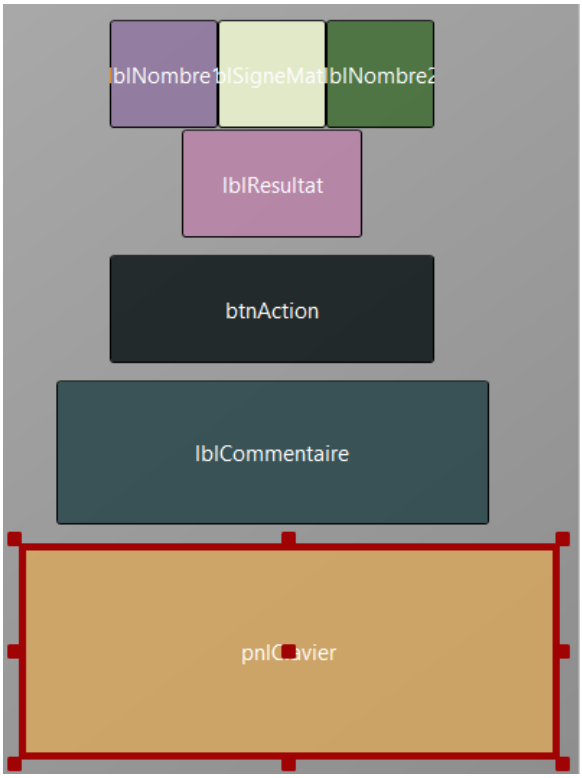
Modifiez les propriétés Left, Top, Width et Height si elles n'ont pas la même valeur que dans l'image.

Text en " " caractère espace.



Text Color en Black #000000 (noir)

Color en White #FFFFFF (blanc)
Niveau Alpha en 255
Rayon des coins en 5



Maintenant nous ajoutons un Panel pour les boutons du clavier.

Positionnez et redimensionnez-le selon l'image.

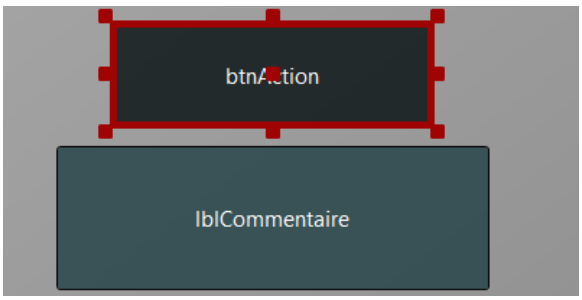
Main	
Nom	pnlClavier
Type	Panel
Nom générique de l'	pnlClavier
Parent	Activity

Modifiez son Nom en pnlClavier
"pnl" pour Panel, le type de view.

Panel Properties	
Elevation	0
Drawable	ColorDrawable
Color	#8C8C8C
Niveau Alpha	255
Rayon des coins	0
Couleur du cadre	#FF000000
Épaisseur du cadre	0

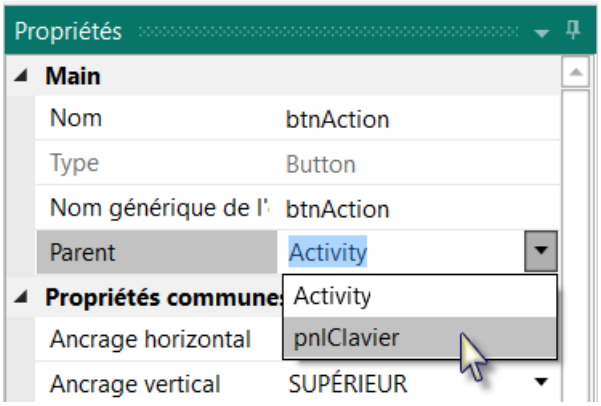
Modifiez

Color	en #8C8C8C
Niveau Alpha	en 255
Rayoin des coins	en 0

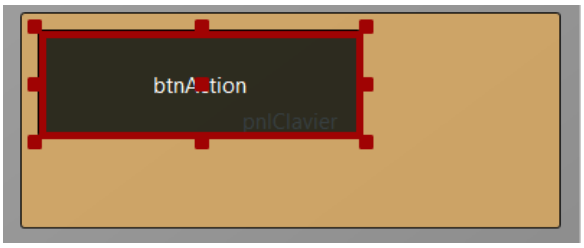


Nous déplaçons le bouton btnAction de l'Activity sur le Panel pnlClavier.

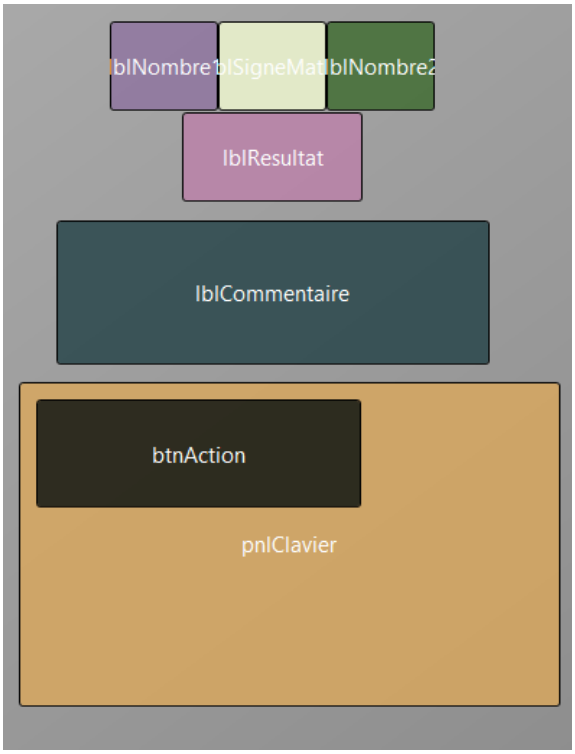
Cliquez sur btnAction.



Et dans la liste Parent cliquez sur pnlClavier.

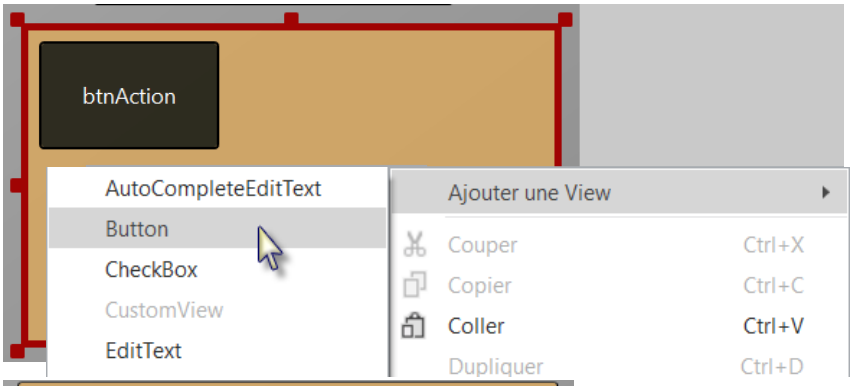


Le bouton appartient maintenant au Panel pnlClavier.

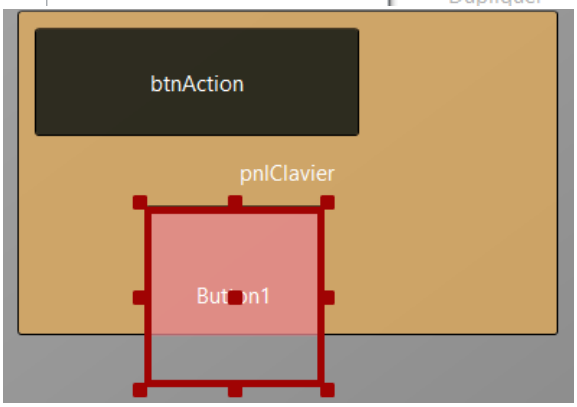


Nous réarrangeons les différentes views pour obtenir plus d'espace pour le clavier.

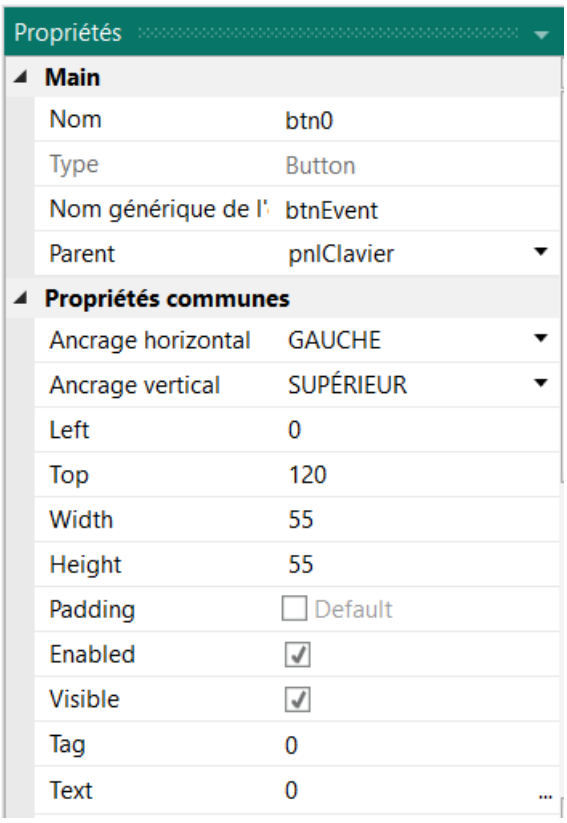
Modifiez la propriété Height des 4 Labels de 60 à 50.
Modifiez la propriété Top de lblResultat à 60.
Modifiez la propriété Top de lblCommentaire à 120.
Modifiez la propriété Top de pnlClavier à 210.
Modifiez la propriété Height de pnlClavier à 180.



Cliquez avec le bouton droit sur pnlClavier
puis cliquez sur Ajouter une View
et cliquez sur Button
pour ajouter un nouveau bouton.



Le nouveau bouton est ajouté.



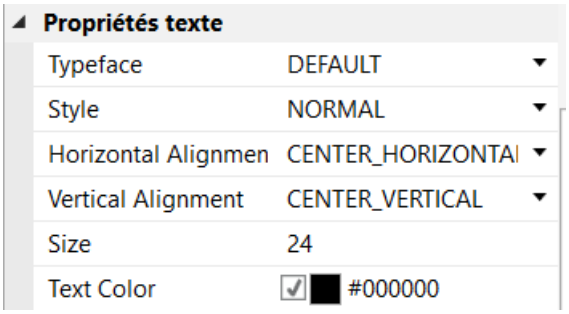
Modifiez les propriétés suivantes :

Nom en btn0

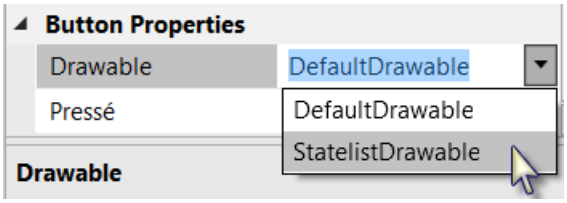
Nom générique de l'événement en btnEvent

Left en 0
Top en 120
Width en 55
Height en 55

Tag en 0
Text en 0

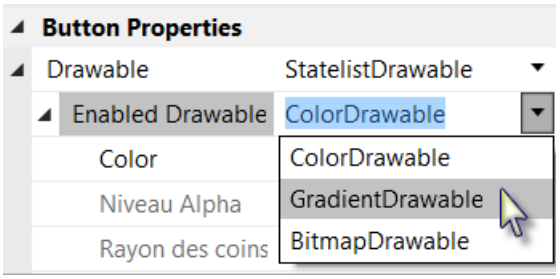


Size to 24
Text Color en Black #000000

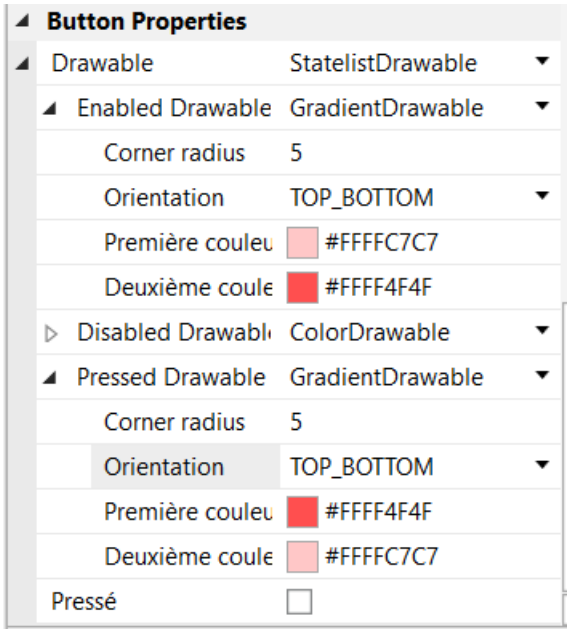


Nous changeons les couleurs du bouton.

Cliquez sur StatelistDrawable.



Dans Enabled Drawable
Cliquez sur GradientDrawable.



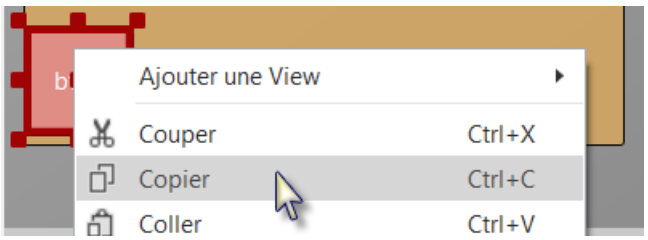
Modifiez les propriétés ci-dessous :

Orientation en TOP_BOTTOM
Première couleur en #FFFFC7C7
Deuxième couleur en #FFFF4F4F

Pressed Drawable en GradientDrawable

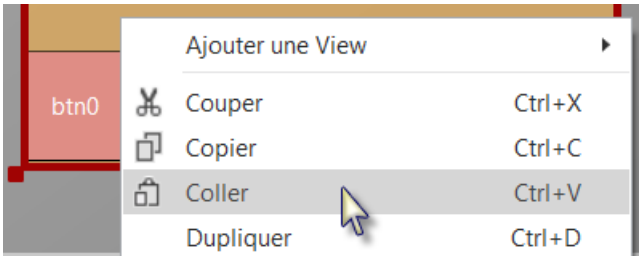
Orientation en TOP_BOTTOM
Première couleur en #FFFF4F4F
Deuxième couleur en #FFFFC7C7

Si vous avez connecté un dispositif vous verrez le bouton comme à gauche.



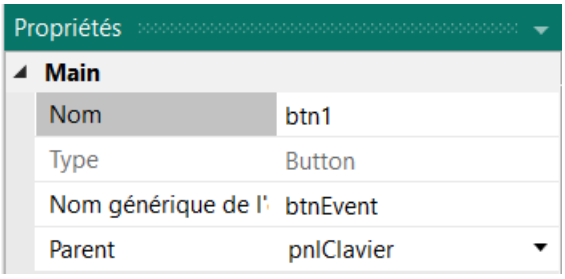
Nous dupliquons btn0 et positionnons le nouveau bouton à côté du précédent avec un léger espace.

Cliquez avec le bouton droit sur btn0 et cliquez sur Copier.



Cliquez avec le bouton droit sur pnlClavier et cliquez sur Coller.

Déplacez le nouveau bouton à côté du précédent.



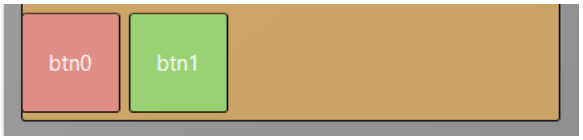
Modifiez les propriétés suivantes :

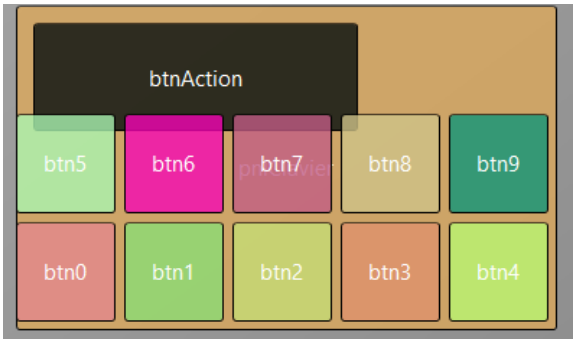
Nom en btn1

Tag en 1
Text en 1

Et le résultat.

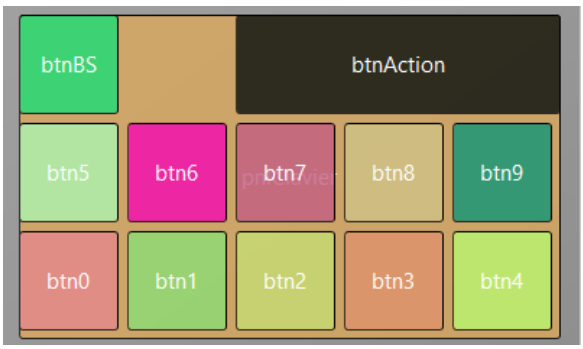
Dans le Concepteur visuel et sur le dispositif.





Ajoutons 8 boutons supplémentaires comme dans l'image.

Modifiez les propriétés suivantes :
Nom btn2 , btn3 , btn4 etc.
Tag 2 , 3 , 4 etc.
Text 2 , 3 , 4 etc.



Pour créer le bouton BackSpace, dupliquez un des boutons de numéro et positionnez-le comme dans l'image.

Redimensionnez et positionnez btnAction.

Modifiez Color de pnlClavier en Black #000000.

Modifiez leur propriétés Nom, Tag, Text and Color comme ci-dessous.

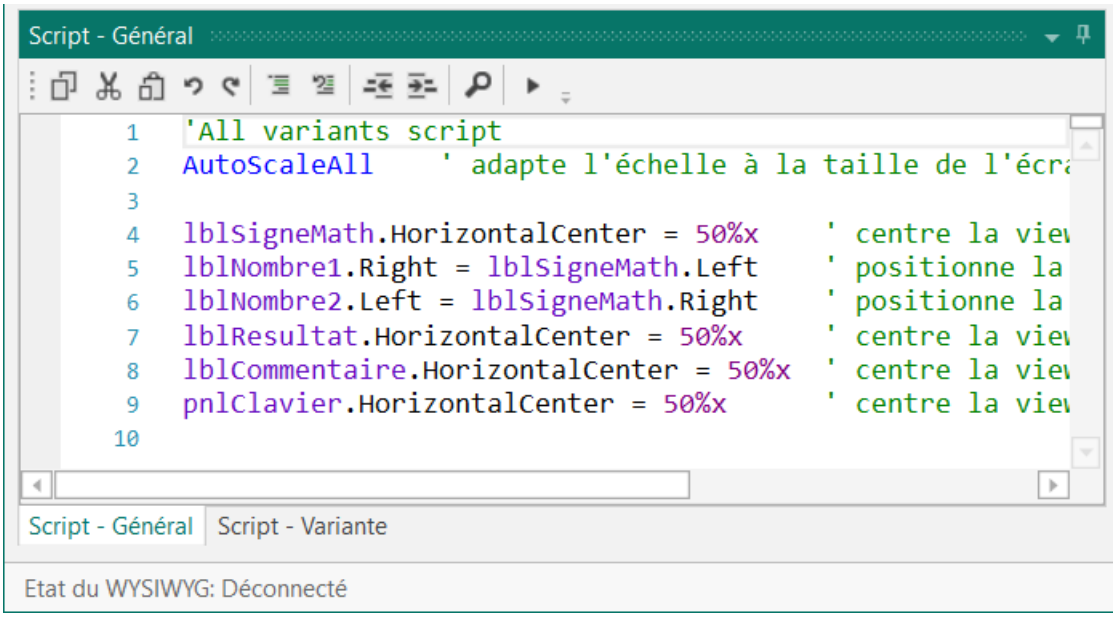
btnAction O K

Propriétés	
Main	
Nom	btnAction
Type	Button
Nom générique de l'	btnAction
Parent	pnlClavier
Propriétés communes	
Ancrage horizontal	GAUCHE
Ancrage vertical	SUPÉRIEUR
Left	120
Top	0
Width	180
Height	55
Padding	Default
Enabled	☑
Visible	☑
Tag	
Text	O K
Button Properties	
Drawable	StatelistDrawable
Enabled Drawable	GradientDrawable
Corner radius	5
Orientation	TOP_BOTTOM
Première couleu	#FF8EFF8E
Deuxième coule	#FF0A800A
Disabled Drawabl	ColorDrawable
Pressed Drawable	GradientDrawable
Corner radius	5
Orientation	TOP_BOTTOM
Première couleu	#FF0A800A
Deuxième coule	#FF8EFF8E

btnBS <

Propriétés	
Main	
Nom	btnBS
Type	Button
Nom générique de l'	btnEvent
Parent	pnlClavier
Propriétés communes	
Ancrage horizontal	GAUCHE
Ancrage vertical	SUPÉRIEUR
Left	0
Top	0
Width	55
Height	55
Padding	Default
Enabled	☑
Visible	☑
Tag	BS
Text	<
Button Properties	
Drawable	StatelistDrawable
Enabled Drawable	GradientDrawable
Corner radius	5
Orientation	TOP_BOTTOM
Première couleu	#FFC7C7FF
Deuxième coule	#FF4F4FFF
Disabled Drawabl	ColorDrawable
Pressed Drawable	GradientDrawable
Corner radius	5
Orientation	TOP_BOTTOM
Première couleu	#FF4F4FFF
Deuxième coule	#FFC7C7FF

Une autre amélioration consiste à centrer les différents objets horizontalement sur l'écran.
Pour cela, nous ajoutons le code ci-dessous dans le Designer, dans la fenêtre Script-Général.




```
'All variants script
AutoScaleAll ' adapte l'échelle à la taille de l'écran

lblSigneMath.HorizontalCenter = 50%x   ' centre la view au mileu de l'écran
lblNombre1.Right = lblSigneMath.Left   ' aligne le bord droit sur le bord gauche
lblNombre2.Left = lblSigneMath.Right   ' aligne le bord gauche sur le bord droit
lblResultat.HorizontalCenter = 50%x    ' centre la view au mileu de l'écran
lblCommentaire.HorizontalCenter = 50%x ' centre la view au mileu de l'écran
pnlClavier.HorizontalCenter = 50%x     ' centre la view au mileu de l'écran
```

Les deux premières lignes existent par défaut, nous les laissons.

```
'All variants script
AutoScaleAll ' adapte l'échelle à la taille de l'écran
```

```
lblSigneMath.HorizontalCenter = 50%x
```

`HorizontalCenter` centre une view horizontalement à la valeur définie, `50%x` dans notre cas donc le milieu de l'écran.

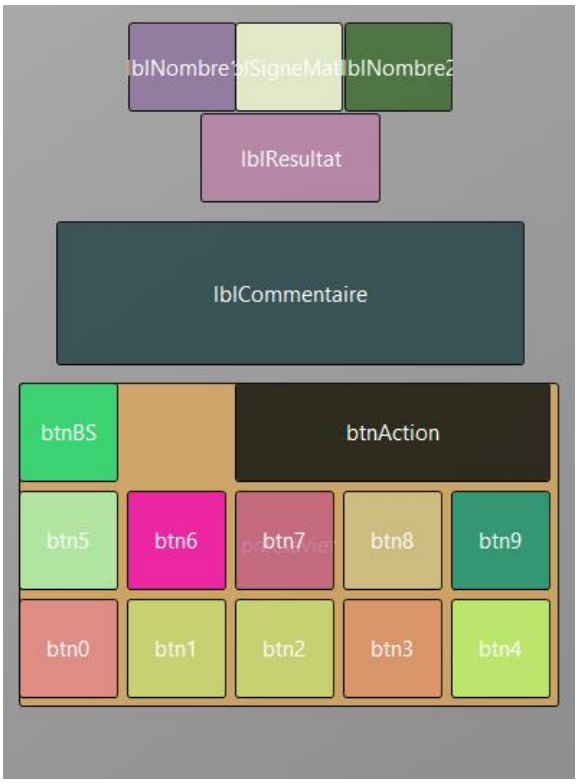
```
lblNombre1.Right = lblSigneMath.Left
```

Aligne le bord droit de `lblNombre1` au bord gauche de `lblSigneMath`, positionne `lblNombre1` juste à côté de `lblSigneMath` à gauche.

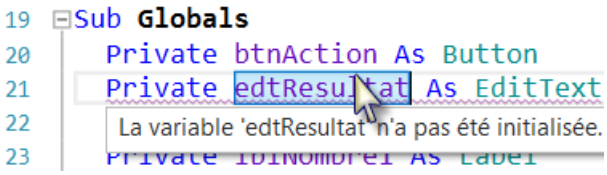
```
lblNombre2.Left = lblSigneMath.Right
```

Aligne le bord gauche de `lblNombre2` au bord droit de `lblSigneMath`, positionne `lblNombre2` juste à côté de `lblSigneMath` à droite.

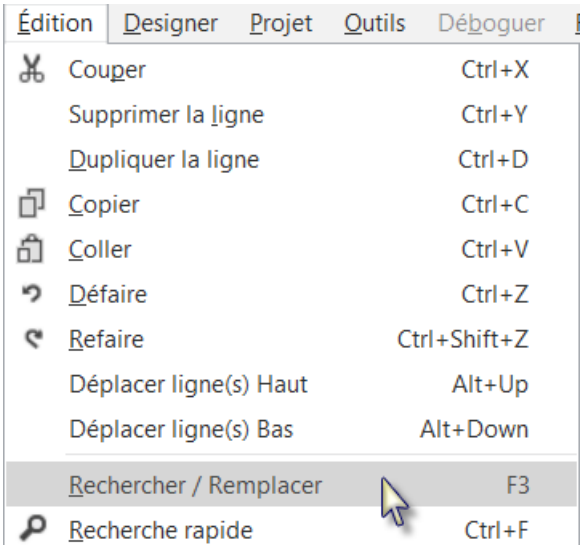
Le nouveau layout est terminé.
Dans le Concepteur visuel et sur le dispositif.



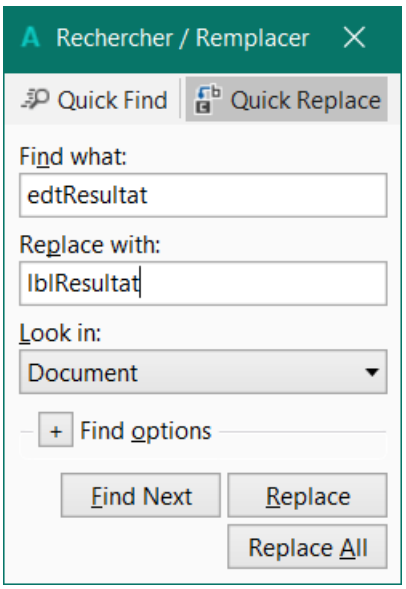
Nous allons modifier le code.
Tout d'abord nous devons remplacer edtResultat par lblResultat car nous avons remplacé l'EditText par un Label.



Double cliquez sur edtResultat pour le sélectionner.



Dans le menu Édition cliquez sur Rechercher / Remplacer ou pressez F3.



Entrez ‘lblResult’ dans la case Replace with.

Cliquez sur Replace All.

Nous devons aussi modifier le type de la view de EditText en Label.

```
Private lblResultat As Label
```

Nous écrivons maintenant la routine qui gère les événements Click des boutons.
Le nom générique des événements de tous les boutons est "btnEvent", sauf le bouton btnAction
La routine associée aux événement Click sera btnEvent_Click.
Écrivez le code suivant :

```
Private Sub btnEvent_Click

End Sub
```

Nous devons définir quel bouton a généré l’événement. Pour cela, nous utilisons l’objet Sender qui est un objet spécial contenant la référence à l’objet qui a générée l’événement.

```
Private Sub btnEvent_Click
   Private btnSender As Button
   btnSender = Sender

   Select btnSender.Tag
   Case "BS"
   Case Else
   End Select
End Sub
```

Pour avoir accès aux propriétés de la view qui a généré l’événement nous déclarons une variable locale.
`Private btnSender As Button`.
Et attribuons `btnSender = Sender`.

Puis, pour différencier entre le bouton retour arrière et les boutons numériques nous utilisons une structure Select / Case / End Select et la propriété Tag des boutons.
Rappelez-vous, lorsque nous avons ajouté les différents boutons nous avons défini leur propriété Tag en BS, 0, 1, 2 etc.

```
   Select btnSender.Tag
   Case "BS"
   Case Else
```

définit la variable à comparer.
vérifie si c’est le bouton avec la valeur Tag "BS".
gère tous les autres boutons.

Ajoutons le code pour les boutons numériques.
Nous ajoutons le texte du bouton au texte dans le Label lblResultat.

```
   Select btnSender.Tag
   Case "BS"
   Case Else
      lblResultat.Text = lblResultat.Text & btnSender.Text
   End Select
End Sub
```

Ce qui est fait dans cette ligne

```
   lblResult.Text = lblResult.Text & btnSender.Text
```

Le caractère "&" signifie concaténation, donc nous ajoutons simplement au texte déjà existant le contenu de la propriété Text du bouton qui a généré l'événement.

Ajoutons le code pour le bouton BackSpace.

```
   Select btnSender.Tag
   Case "BS"
      If lblResultat.Text.Length >0 Then
         lblResultat.Text = lblResultat.Text.SubString2(0, lblResultat.Text.Length - 1)
      End If
   Case Else
      lblResultat.Text = lblResultat.Text & btnSender.Text
   End Select
End Sub
```

Lorsqu'on presse le bouton BS nous devons supprimer le dernier caractère du texte dans lblResult.
Cependant, ceci est seulement valable si la longueur du texte est plus grande que 0.
Ce qui est vérifié avec :

```
If lblResultat.Text.Length > 0 Then
```

Pour supprimer le dernier caractère nous utilisons la fonction SubString2.

```
lblResult.Text = lblResult.Text.SubString2(0,lblResult.Text.Length - 1)
```

SubString2(BeginIndex, EndIndex) extrait un nouvel objet String commençant par le caractère à l'index BeginIndex (inclusif) jusqu'au caractère à l'index EndIndex (exclusif).

La routine complète est maintenant terminée.

```
Sub btnEvent_Click
   Private btnSender As Button

   btnSender = Sender

   Select btnSender.Tag
   Case "BS"
      If lblResultat.Text.Length > 0 Then
         lblResultat.Text = lblResultat.Text.SubString2(0, lblResultat.Text.Length - 1)
      End If
   Case Else
      lblResultat.Text = lblResultat.Text & btnSender.Text
   End Select
End Sub
```

Nous pouvons améliorer l'interface utilisateur par l'adjonction de couleurs de fond du Label lblCommentaire :
- Jaune pour un nouveau problème
- Vert clair pour un résultat JUSTE
- Rouge clair pour un résultat FAUX.

Nous ajoutons dans routine NouveauProbleme la ligne :
`lblCommentaire.Color = Colors.RGB(255,235,128).`

```
Private Sub NouveauProbleme
   Nombre1 = Rnd(1, 10)                ' Génère un nombre aléatoire entre 1 et 9
   Nombre2 = Rnd(1, 10)                ' Génère un nombre aléatoire entre 1 et 9
   lblNombre1.Text = Nombre1           ' Affiche Nombre1 dans le Label lblNombre1
   lblNombre2.Text = Nombre2           ' Affiche Nombre2 dans le Label lblNombre2
   lblCommentaire.Text = "Entrez le résultat" & CRLF & "et cliquez sur OK."
   lblCommentaire.Color = Colors.RGB(255,235,128)   ' couleur jaune
lblResultat.Text = ""                  ' Vide edtResult.Text
End Sub
```

Dans la routine TestResultat nous ajoutons les deux lignes pour les couleurs.

```
Private Sub TestResultat
   If lblResultat.Text = Nombre1 + Nombre2 Then
      lblCommentaire.Text = "Résultat  J U S T E." & CRLF & "Cliquez sur Nouveau."
      lblCommentaire.Color = Colors.RGB(128,255,128)      ' couleur vert clair
      btnAction.Text = "Nouveau"
   Else
      lblCommentaire.Text = "Résultat  F A U X." & CRLF & "Entrez un nouveau résultat" & CRLF &
"et cliquez sur O K."
      lblCommentaire.Color = Colors.RGB(255,128,128)      '
   End If
End Sub
```

Une autre amélioration consiste à ne pas afficher la touche '0' pour éviter des résultats commençant par un '0'.
Pour cela, nous la cachons dans la routine NouveauProbleme avec `btn0.Visible = False`.

```
Private Sub NouveauProbleme
   Nombre1 = Rnd(1, 10)                ' Génère un nombre aléatoire entre 1 et 9
   Nombre2 = Rnd(1, 10)                ' Génère un nombre aléatoire entre 1 et 9
   lblNombre1.Text = Nombre1           ' Affiche Nombre1 dans le Label lblNombre1
   lblNombre2.Text = Nombre2           ' Affiche Nombre2 dans le Label lblNombre2
   lblCommentaire.Text = "Entrez le résultat" & CRLF & "et cliquez sur OK."
   lblCommentaire.Color = Colors.RGB(255,235,128)   ' couleur jaune
   lblResultat.Text = ""               ' Vide edtResult.Text
   btn0.Visible = False
End Sub
```

Dans la routine btnEvent_Click, nous cachons la touche ‘0’ si la longueur du texte dans lblResult est égale à zéro et l’affichons si la longueur est plus grande que zéro.

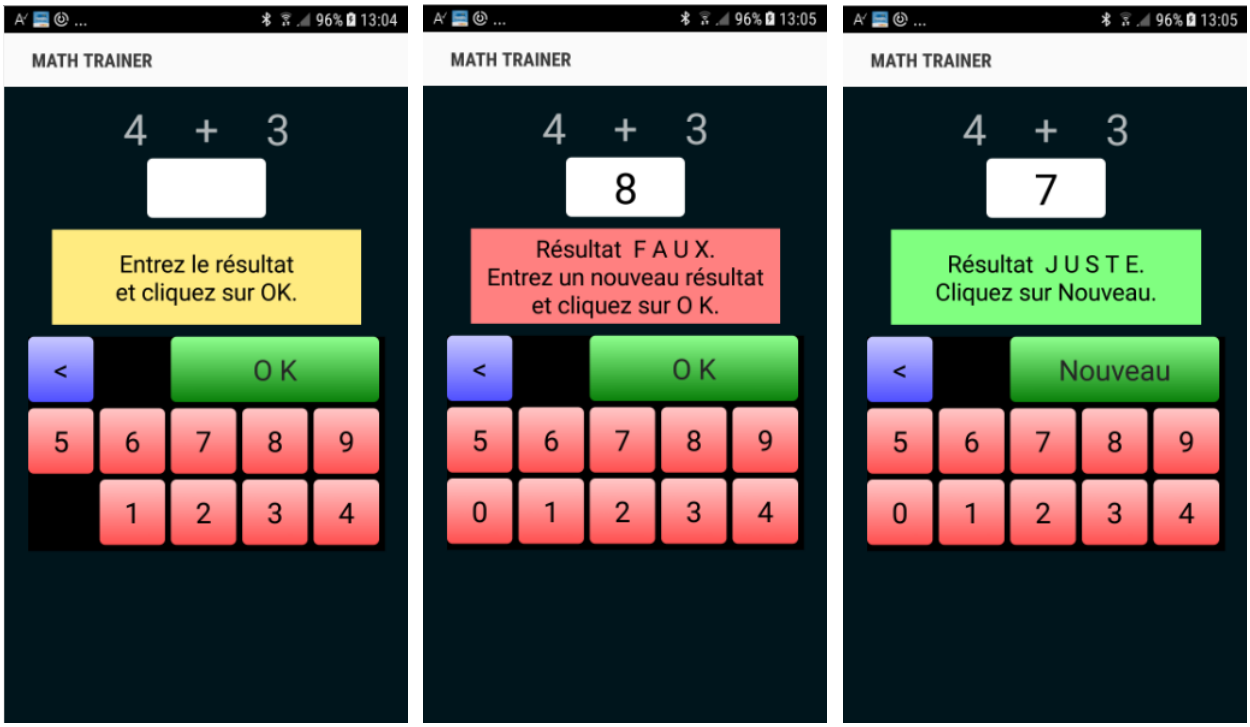
```
Sub btnEvent_Click
   Private btnSender As Button

   btnSender = Sender

   Select btnSender.Tag
   Case "BS"
      If lblResultat.Text.Length > 0 Then
         lblResultat.Text = lblResultat.Text.SubString2(0,lblResultat.Text.Length - 1)
      End If
   Case Else
      lblResultat.Text = lblResultat.Text & btnSender.Tag
   End Select

   If lblResultat.Text.Length = 0 Then
      btn0.Visible = False
   Else
      btn0.Visible = True
   End If
End Sub
```

Comme nous utilisons la référence `btn0` dans le code nous devons la déclarer dans la routine Globals.

Modifiez la ligne ci-dessous :

```
   Private btnAction, btn0 As Button
```

Exécutez le programme pour voir le résultat.



3 B4i Premiers pas

B4i est un environnement de développement simple et puissant et cible des dispositifs Apple (iPhone, iPad etc.).
Le langage B4i est similaire à B4A et Visual Basic.
Les applications compilées sont des application iOS natives; il n'y a pas d'autres dépendances.
Contrairement à d'autres EDIs, B4i est focalisé à 100% sur iOS.
B4i comprend un constructeur visuel pour les interfaces homme-machine, avec un support pour de multiples écrans et orientations.
Vous pouvez développer et déboguer avec un dispositif réel.

iOS 7 et supérieur sont supportés.

Ce dont vous avez besoin :

- Le programme B4i, qui est un programme s'exécutant sur un PC.
- Le Java SDK sur le PC, gratuit.
- Une licence développeur Apple, coût 99$ par an.
- Un dispositif pour les essais.
- Le programme Basi4i-Bridge sur le dispositif, gratuit.
- Un Mac Builder pour compiler le programme, qui peut être soit :
    - Un ordinateur Mac avec le programme Mac Builder, un wifi local.
    - Le service Mac Builder hébergé sur Internet, coût 26$ par an.
- Un ordinateur Mac ou un service MacInCloud pour distribuer le programme.

Liens vers des tutoriels (en anglais) dans le forum :
Local Mac Builder Installation
Creating a certificate and provisioning profile
Installing B4i-Bridge and debugging first app

3.1 Installation de B4i

3.1.1 Installation de Java JDK

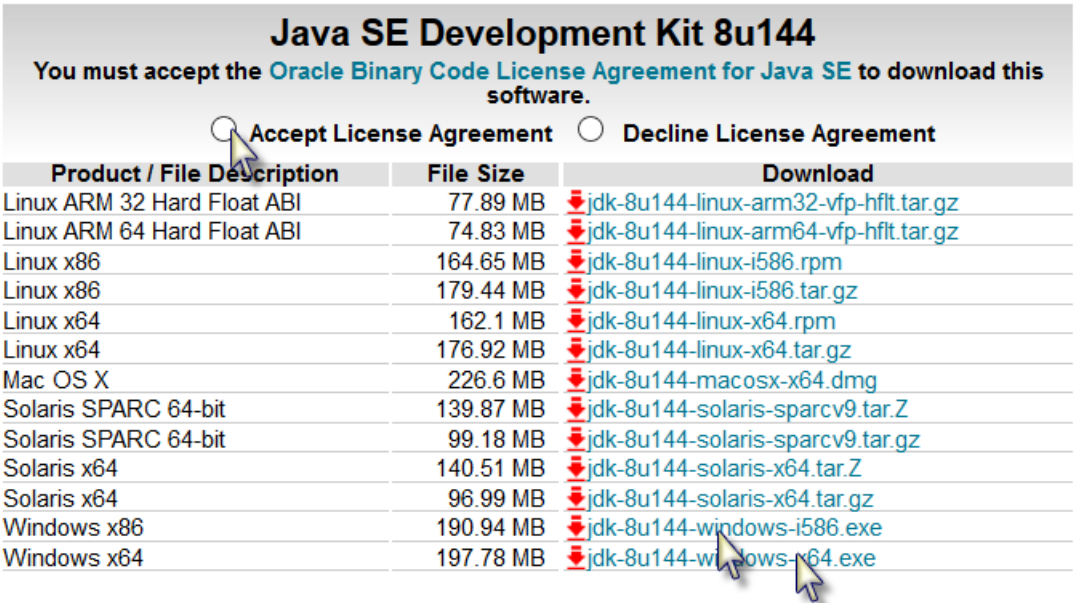
B4i dépend du composant gratuit Java JDK.

Si vous utilisez déjà B4A vous pouvez sauter ce chapitre.

Instructions d'installation :

En premier, vous devez installer **Java JDK**.
Notez qu'il n'y a aucun problème d'avoir plusieurs versions de Java installées sur un même ordinateur.

- Ouvrez ce lien Java 8 JDK download link.
- Cochez la case Accept License Agreement.



Java SE Development Kit 8u144

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ **Accept License Agreement** ○ **Decline License Agreement**

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.89 MB	jdk-8u144-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.83 MB	jdk-8u144-linux-arm64-vfp-hflt.tar.gz
Linux x86	164.65 MB	jdk-8u144-linux-i586.rpm
Linux x86	179.44 MB	jdk-8u144-linux-i586.tar.gz
Linux x64	162.1 MB	jdk-8u144-linux-x64.rpm
Linux x64	176.92 MB	jdk-8u144-linux-x64.tar.gz
Mac OS X	226.6 MB	jdk-8u144-macosx-x64.dmg
Solaris SPARC 64-bit	139.87 MB	jdk-8u144-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.18 MB	jdk-8u144-solaris-sparcv9.tar.gz
Solaris x64	140.51 MB	jdk-8u144-solaris-x64.tar.Z
Solaris x64	96.99 MB	jdk-8u144-solaris-x64.tar.gz
Windows x86	190.94 MB	jdk-8u144-windows-i586.exe
Windows x64	197.78 MB	jdk-8u144-windows-x64.exe

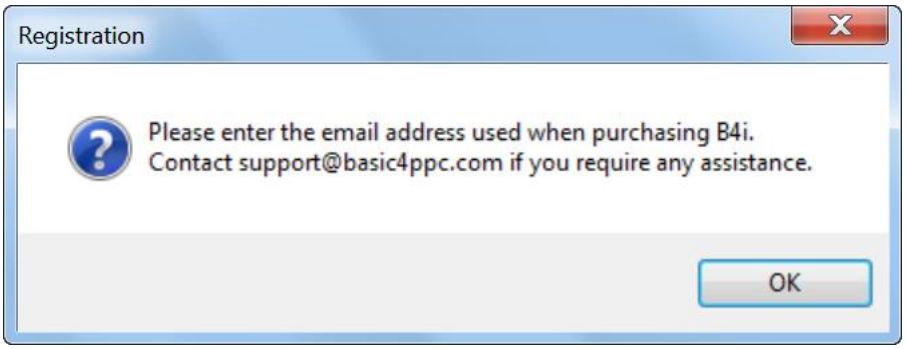
- Sélectionnez **"Windows x86"** ou **"Windows x64"** (pour des machines 64 bit) dans la liste des plateformes.
- Téléchargez le fichier et installez-le.

3.1.2 Installation de B4i

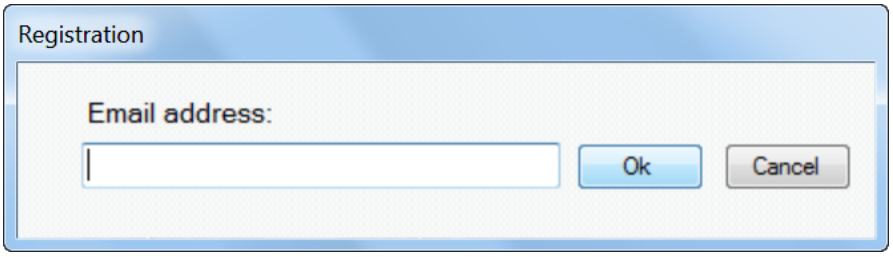
Téléchargez et installez le ficher B4i sur votre ordinateur.

Copiez le fichier de licence *b4i-license.txt* dans le dossier B4i et dans un autre endroit sûr pour sauvegarde. Notez que ce n'est pas un fichier texte, ne cherchez pas à l'ouvrir avec un éditeur de texte.

Lorsque vous exécutez B4i pour la première fois, le programme vous demande d'introduire votre adresse e-mail, celle que vous avez utilisé lors de l'achat de B4i.
Vous la trouvez aussi dans le mail que vous avez reçu avec le fichier B4i.



Entrez votre adresse e-mail.



Une fenêtre de confirmation que B4i a été enregistré sera affichée.

Contactez support@basic4ppc.com si vous avez besoin d'assistance.

3.1.3 Installation du Mac Builder

La compilation pour iOS nécessite le composant Mac Builder, il y deux options :

- Utiliser un ordinateur Mac connecté à un réseau local.
  Pour cela, vous devez télécharger et installer le Mac Builder.

- Utilise le service hébergé. Installation du service Mac Builder hébergé.
  Le service hébergé vous permet de développer des applications iOS sans ordinateur Mac.
  Toutes les étapes de développement peuvent être effectuées avec ce service à l'exception de la dernière étape qui consiste à télécharger l'application sur Apple App Store. Cette étape nécessite un ordinateur Mac ou un service tel que MacInCloud.
  Notez que ce service est actuellement limité à des projets de 15Mb max.

Lien vers le tutoriel dans le forum : Local Mac Builder Installation.

Les instructions ci-dessous expliquent comment installer le Mac Builder sur une machine locale.

1. Installez Java JDK 8 : http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
2. Installez Xcode 6.
3. Téléchargez et dézipez le B4i-Builder.
4. Ouvrez un terminal et naviguez vers le dossier B4i-Builder.
5. Exécutez-le avec : java -jar B4iBuildServer.jar
6. Définissez l'adresse builder IP dans l'EDI sous Outils - Build Server - Définitions Server.

Notes & Conseils

- Par défaut, les ports 51041 (http) et 51042 (https) sont utilisés.
- Le pare-feu doit être désactivé ou autoriser des connexions d'entrée sur les deux ports.
- Pour vérifier que le serveur fonctionne allez vers ce lien : http://<server ip>:51041/test
- Vous pouvez le serveur avec : http://<server ip>:51041/kill
- Il est recommandé de définir l'adresse ip serveur du Mac comme adresse statique. Ceci peut être fait dans les réglages du routeur ou dans le Mac sous Réglages réseau.
- Un unique Mac builder peut être utilisé par plusieurs développeurs pour autant qu'ils soient tous connectés au même réseau local. Notez que vous n'êtes pas autorisé d'héberger des 'builders' pour des développeurs en dehors de votre organisation.

Multiple IPs.

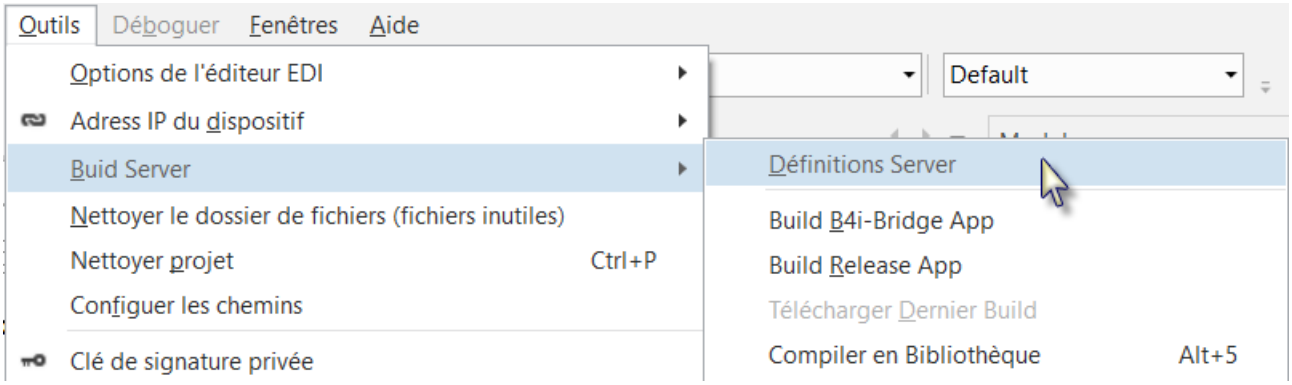
Lors de son démarrage, le serveur prend la première adresse IP fourni par le système et l'utilise pour sa propre adresse IP. Vous pouvez la voir dans les messages du serveur.
Dans la plupart des cas, c'est la bonne adresse. Néanmoins, si ce n'est pas le cas, alors le serveur est inutilisable.
Dans ce cas, vous devez définir explicitement l'adresse correcte :
- Ouvrez le dossier key et effacez tous les fichiers.
- Editez key.txt et changez-le en : `manual :<adresse ip correcte>`

Par exemple :
manual :192.168.0.199

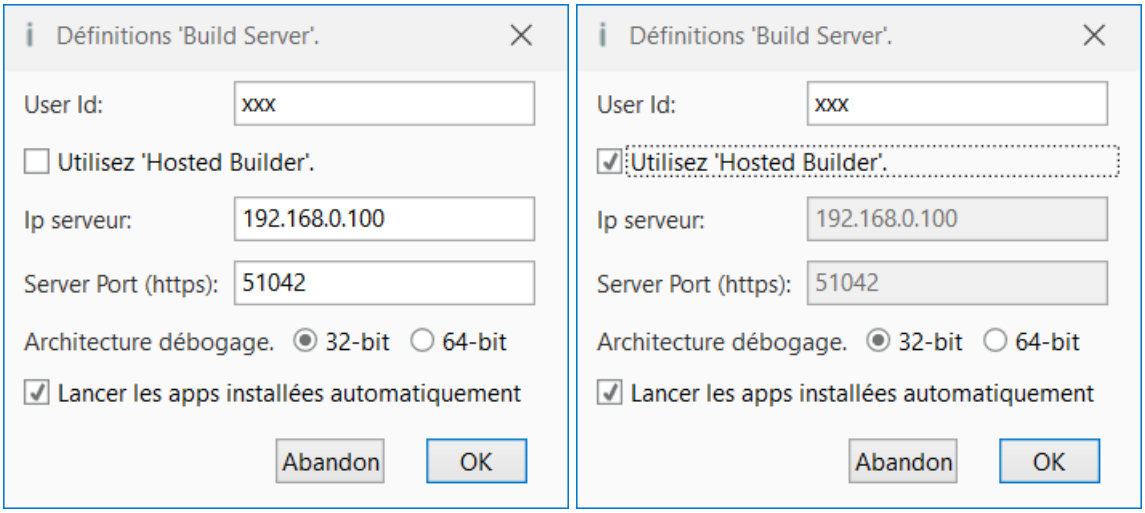
3.1.4 Mac builder hébergé (Hosted Mac builder) (optionnel)

Si vous avez acheté le service *Mac builder hébergé* vous avez reçu un mail avec votre user ID (identifiant utilisateur).

Vous devez le définir dans l'EDI.



Entrez le ID.

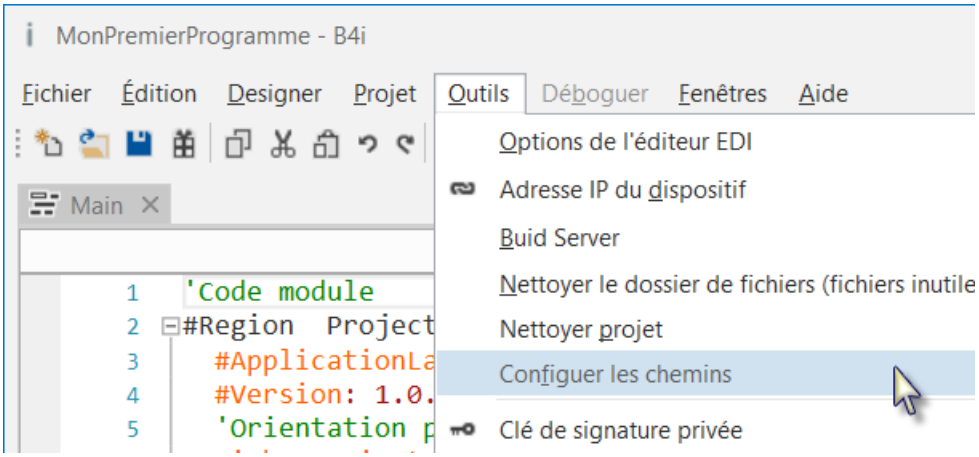


N'oubliez las de cocher 

Si vous utilisez le service Hosted Builder !

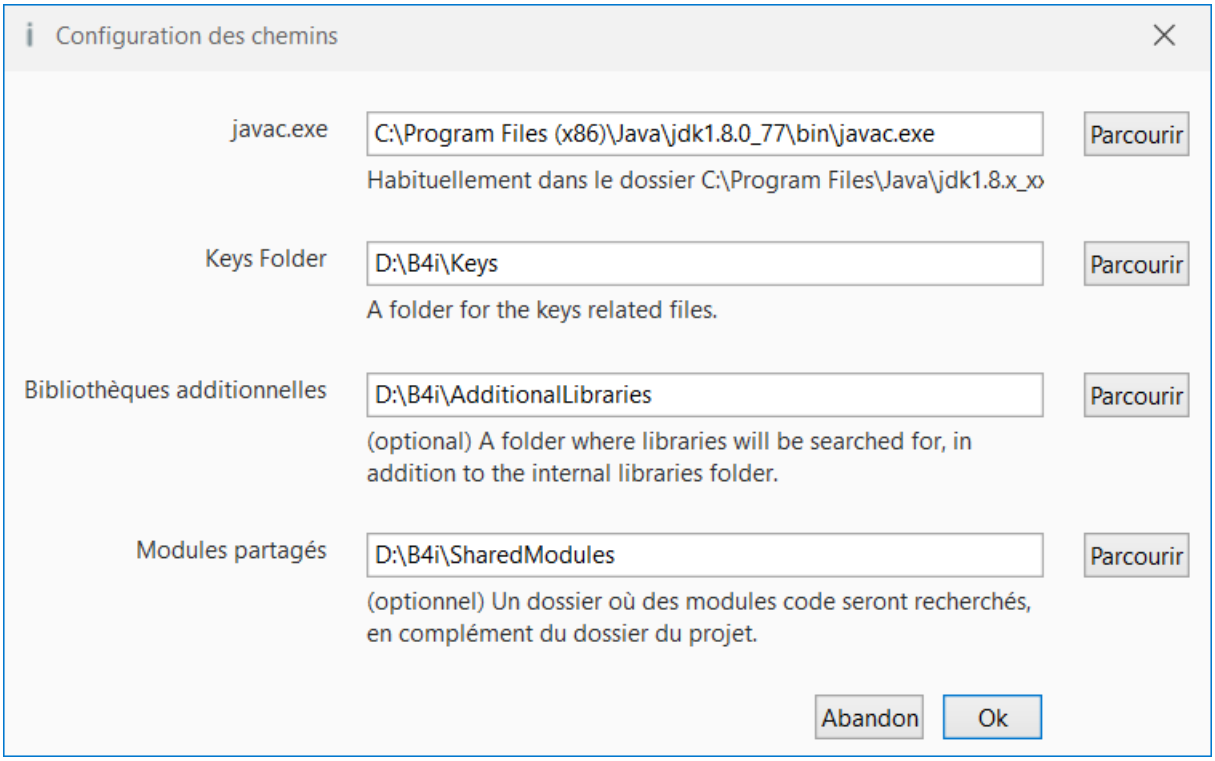
3.2 B4i Configuration des dossiers dans l'EDI

Vous devez configurer différents dossiers dans l'EDI.



Exécutez l'EDI.

Dans le menu Outils cliquez sur Configuer les chemins.



javac.exe :
Entrez le dossier dans lequel se trouve le fichier javac.exe.

Keys folder :
Créez un dossier spécifique pour les clés (Keys), par exemple C:\B4i\Keys.

Bibliothèques additionnelles :
Créez un dossier spécifique pour les bibliothèques additionnelles, par exemple C:\B4i\AdditionalLibraries.

Modules partagés :
Créez un dossier spécifique pour les modules partagés, par exemple C:\B4i\SharedModules.

3.3 Création d'un certificat et profil de provisionnement

Ne paniquez pas.
Alors que ce processus peut paraître un peu ennuyeux, il n'est pas trop compliqué et vous pouvez toujours supprimer les clés et recommencer à partir de zéro (ce qui n'est pas toujours le cas dans Android)

Notez que vous devez d'abord vous enregistrer auprès d'Apple en tant que développeur iOS (coût 99$ par an). Le processus est entièrement effectué sur un ordinateur Windows.

Pour pouvoir installer une application sur un dispositif iOS, vous devez créer un certificat et un profil de provisionnement.

Le certificat est utilisé pour signer les applications. Le profil de provisionnement, lié à un certificat spécifique, comprend une liste des dispositifs sur lesquels cette application peut être installée.

La vidéo montre les différentes étapes pour créer et télécharger un certificat et un profil de provisionnement.

Il y a deux étapes qui ne sont pas montrées dans la vidéo mais sont aussi nécessaires avant que vous ne puissiez créer un profil de provisionnement :

- Créez un App ID. Cette étape est simple. Assurez-vous de créer in id 'wildcard'.
- Ajoutez un ou plusieurs dispositifs. Pour cela, vous devez chercher les UDID des dispositifs.

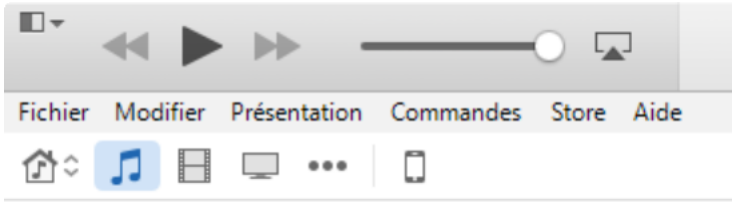
Lien pour le tutoriel dans le forum : Creating a certificate and provisioning profile.

3.3.1 UDID

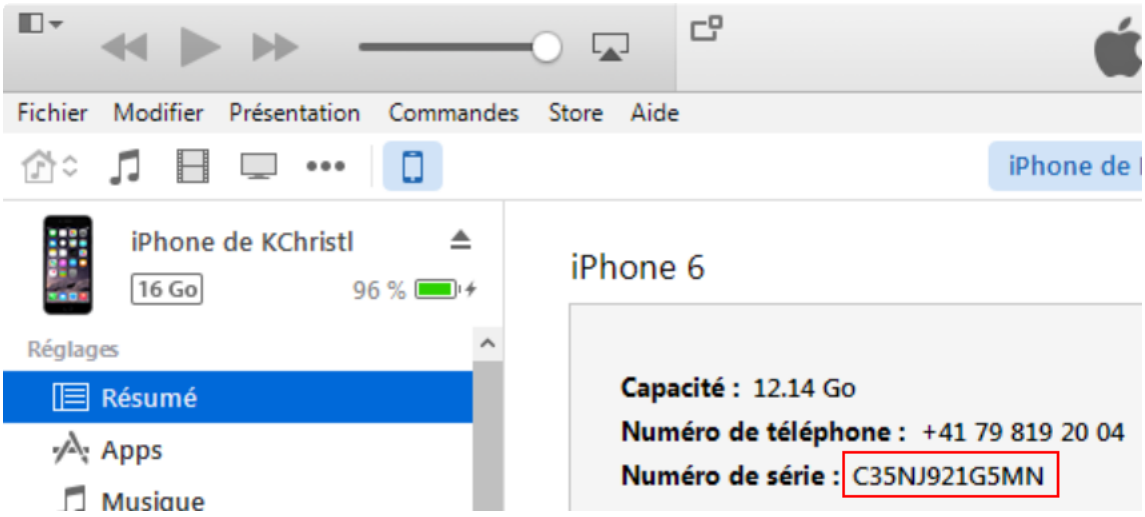
Les dispositifs sont reconnus par leur UDIDs. Il y a deux moyens pour obtenir ces UDID :

1. Si vous avez iTunes installé, vous pouvez les trouver dans iTunes.
   La première fois, connectez votre dispositif avec un câble USB à votre ordinateur.

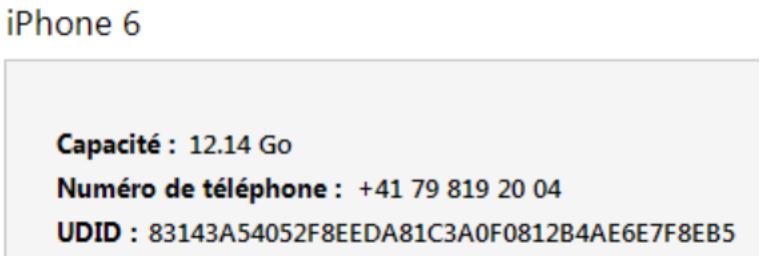
   Exécutez iTunes, vous devriez voir cette icône sur le haut. Ça peut prendre un moment avant que vous ne le voyiez.



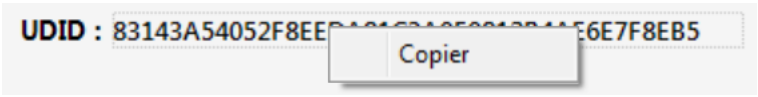
Cliquez sur et vous obtenez l'écran ci-dessous :



Maintenant, cliquez sur C35NJ921G5MN pour obtenir l'UDID.



Right click on 83143A54052F8EEDA81C3A0F0812B4AE6E7F8EB5 to copy the UDID.



2. Utilisez un service en ligne tel que celui-ci : http://get.udid.io/

3.3.2 Certificat et profil de provisionnement

Étapes principales :

1. Définissez un nouveau dossier clé (Key) dans l'EDI.
2. Créez une clé (Key) en sélectionnant dans Outils / Clé de signature privée.
3. Créez et téléchargez le certificat comme montré dans la vidéo. Vous devez télécharger le fichier CSR qui a été créé dans l'étape 2.
   Notez que vous pouvez choisir soit **iOS App Development** ou **App Store and Ad Hoc** dans la page de certificat.
4. Créez et téléchargez un profil de provisionnement.

3.4 Installation de B4i-Bridge

B4i-Bridge est une application que vous devez installer sur votre dispositif.
Elle a trois fonctions :

1. Lancer la procédure d'installation en cas de besoin.
2. Exécuter l'application (si une installation n'est pas nécessaire).
3. B4i-Bridge est aussi le Constructeur visuel WYSIWYG.

B4i-Bridge ne s'installe qu'une fois. L'installation se fait dans le navigateur du dispositif.

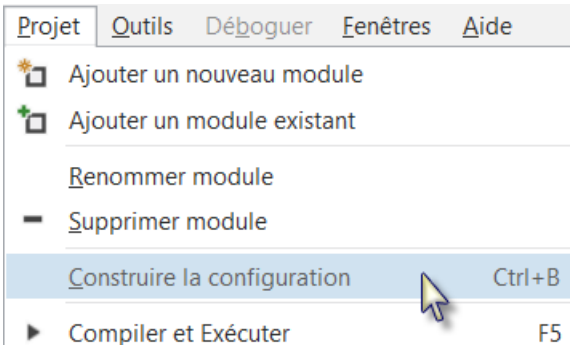
Lien vers le tutoriel dans le forum : Installing B4i-Bridge and debugging first app.

3.5 Installation du certificat B4I

Lancez Safari (navigateur du dispositif) et naviguez vers : www.b4x.com/ca.pem
Suivez les instructions.
Vous pouvez à tout moment voir le certificat dans Réglages / Général / Profil.

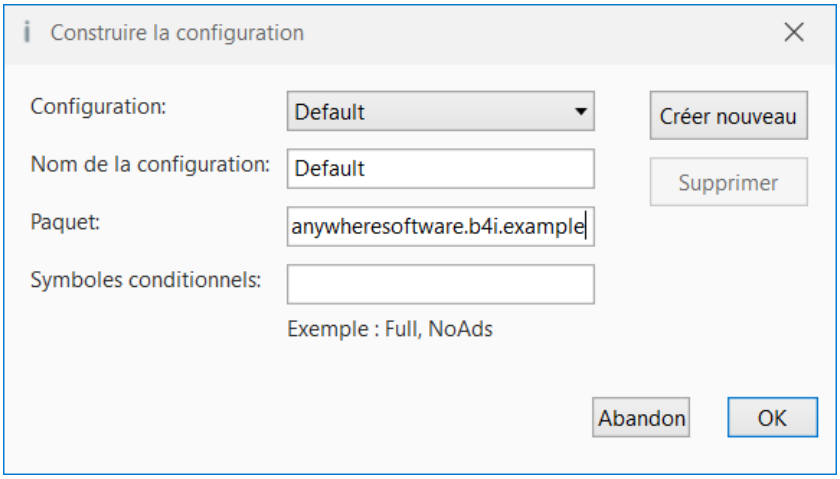
3.6 Définition du nom de Paquet

Exécutez B4i, chargez un projet ou utilisez le projet par défaut et définissez le nom du Paquet basé sur le 'provision app ID'.



Dans le menu Projet cliquez sur Construire la configuration.

La fenêtre ci-dessous sera affichée :

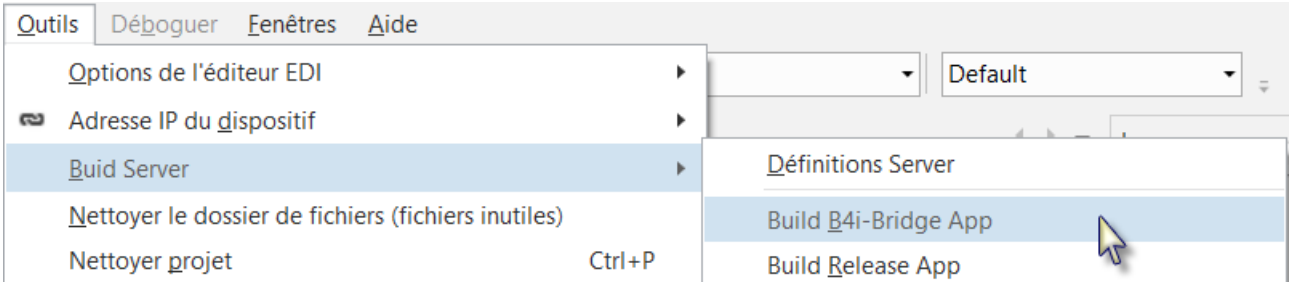


Modifiez le nom du Paquet en fonction du 'provision app ID'.

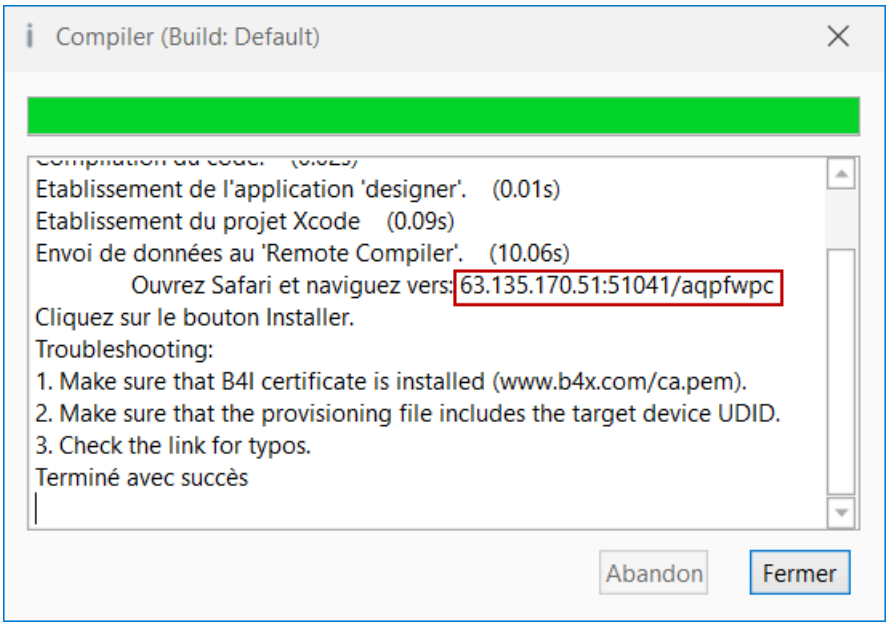
Exemple dans mon cas : Package: anywheresoftware.b4i.example
anywheresoftware.b4i. obligatoire, la suite en fonction de votre application.

3.7 Installation de Build B4i-Bridge

Dans le menu Outils cliquez sur Buid Server et cliquez sur Build B4i-Bridge App :



Vous obtenez la fenêtre ci-dessous dans laquelle vous trouvez le code pour Safari (chapitre suivant).



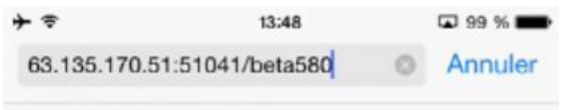
Le code dans le rectangle rouge est sûrement différent !

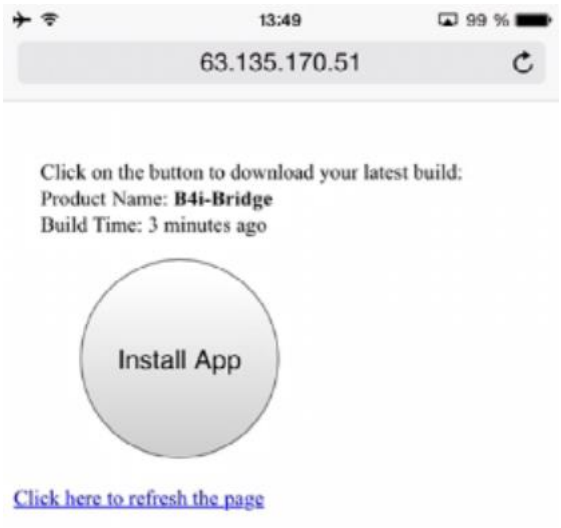
3.7.1 Chargez B4i-Bridge

Ouvrez Safari sur le dispositif 

.

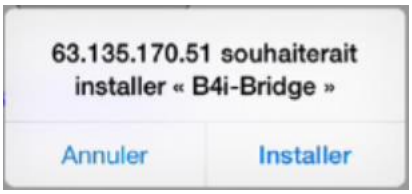
Entrez le code obtenu dans le chapitre précédent dans la case sur le haut de l'écran.





Cet écran sera affiché.

Cliquez sur.



Cliquez sur Installer pour l'installer

Fermez Safari.

3.7.2 Installation de B4i-Bridge et démarrage

Cliquez sur l'icône B4i-Bridge 

sur le dispositif, vous verrez l'animation d'installation et finalement l'icône de B4i-Bridge 

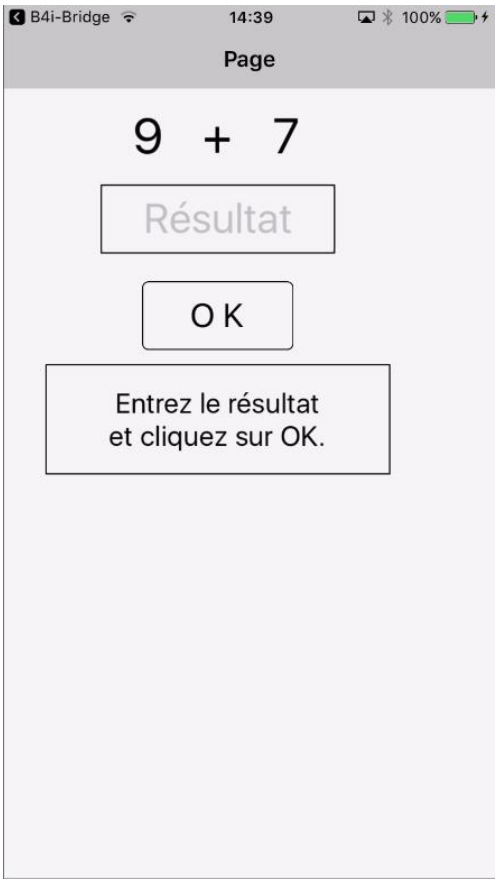
.

Conseils :
- Il n'est pas nécessaire d'attendre que l'animation d'installation soit terminée. Dès que l'animation a commencée vous pouvez presser l'icône.
- Si l'installation reste bloquée en 'attente' plus longtemps que 10 à 15 secondes, désinstallez et réinstallez l'application.
- B4i-Bridge doit être en avant-plan pour pouvoir démarrer une installation ou démarrer l'application. Dans la majorité des cas il est automatiquement en avant-plan. Si ça n'est pas le cas, vous devez presser l'icône pour l'amener en avant-plan.

3.8 Mon premier programme B4i (MonPremierProgramme.b4i)

Nous allons écrire notre premier programme B4i. C'est un programme d'entraînement de calcul pour enfants.

Le projet est disponible dans le dossier CodesSource qui est fourni avec ce livret :
CodesSource\MonPremierProgramme\B4i\ MonPremierProgramme.b4i



Nous aurons sur l'écran :
- 2 Labels affichant des nombres générés aléatoirement (entre 1 et 9).
- 1 Label avec le signe mathématique (+).
- 1 TextField dans lequel l'utilisateur devra entrer le résultat.
- 1 Button, utilisé soit pour confirmer le résultat entré ou pour générer un nouveau calcul.
- 1 Label avec un commentaire concernant le résultat.

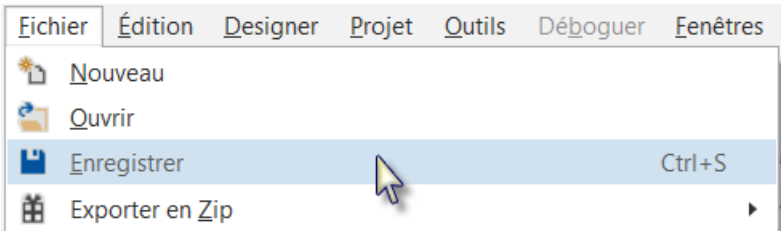
Dans iOS :
- Label est un objet pour afficher du texte.
- TextField est un objet permettant à l'utilisateur d'éditer du texte.
- Button est un objet permettant à l'utilisateur des actions, un clic.

Nous allons définir le layout (mise en page) de l'interface utilisateur avec le Concepteur visuel du Designer et passerons pas à pas au travers de tout le processus.
Le Designer gère les différents objets de l'interface.
Le Concepteur visuel montre les positions et dimensions des différents objets et permet de les déplacer ou de les redimensionner sur l'écran.
Sur un dispositif nous voyons l'aspect réel.

Exécutez B4i 

Enregistrez le projet.

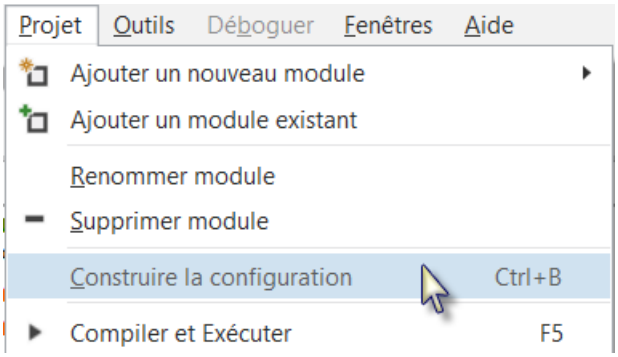
Vous devez enregistrer le projet avant de pouvoir utiliser le Designer.



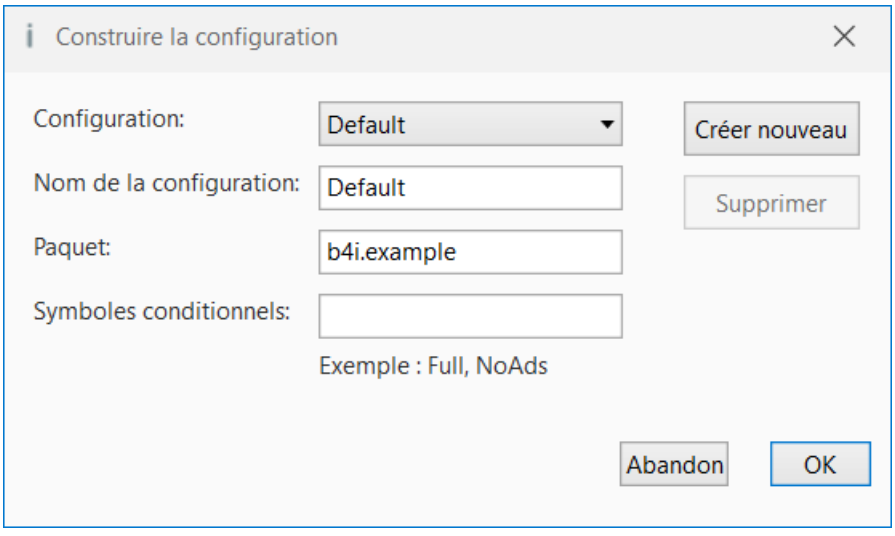
Créez un nouveau dossier MonPremierProgramme et enregistrez le projet avec le nom MonPremierProgramme.

Définissez le nom du Paquet.
Chaque programme nécessite un nom de Paquet.

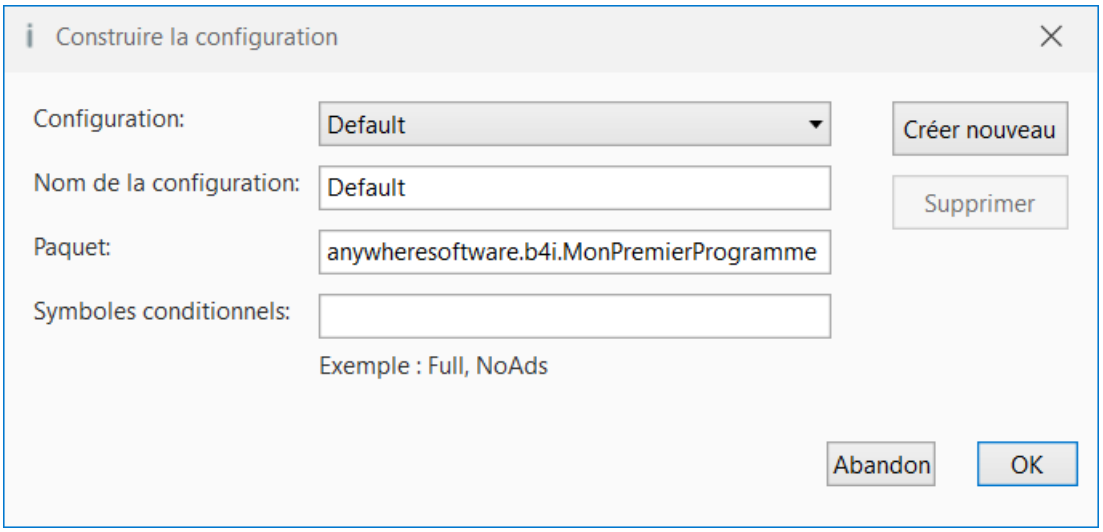
Dans le menu Projet cliquez sur Construire la configuration.



Cette fenêtre sera affichée :



Le nom par défaut est b4i.example. Nous le modifions en anywheresoftware.b4i.MonPremierProgramme.



Définissez l'attribut 'Application Label'.

L'attribut Application label est le nom du programme qui sera affiché sur le dispositif sous l'icône.

Sur le haut du code vous voyez la 'Region' Project Attributes.

Les Regions sont des parties de code qui peuvent être réduites ou étendues comme à droite.

```
1  'Code module
2  #Region  Project Attributes
9
```

```
1  'Code module
2  #Region  Project Attributes
3    #ApplicationLabel: B4i Exa
4    #Version: 1.0.0
5    'Orientation possible valu
6    #iPhoneOrientations: Portr
7    #iPadOrientations: Portrai
8  #End Region
```

Un clic sur ⊞ étend la Region.
Un clic sur ⊟ réduit la Region.
Les Regions sont expliquées dans le chapitre *Réduire une Région* dans le livret *B4x EDI*.

```
#Region  Project Attributes
   #ApplicationLabel: B4i Example
   #Version: 1.0.0
   'Orientation possible values: Portrait, LandscapeLeft, LandscapeRight and
PortraitUpsideDown
   #iPhoneOrientations: Portrait, LandscapeLeft, LandscapeRight
   #iPadOrientations: Portrait, LandscapeLeft, LandscapeRight, PortraitUpsideDown
#End Region
```

Le nom par défaut est `B4i Example`, nous le modifions en `MonPremierProgramme`.

Modifiez cette ligne :

```
   #ApplicationLabel: B4i Example
```

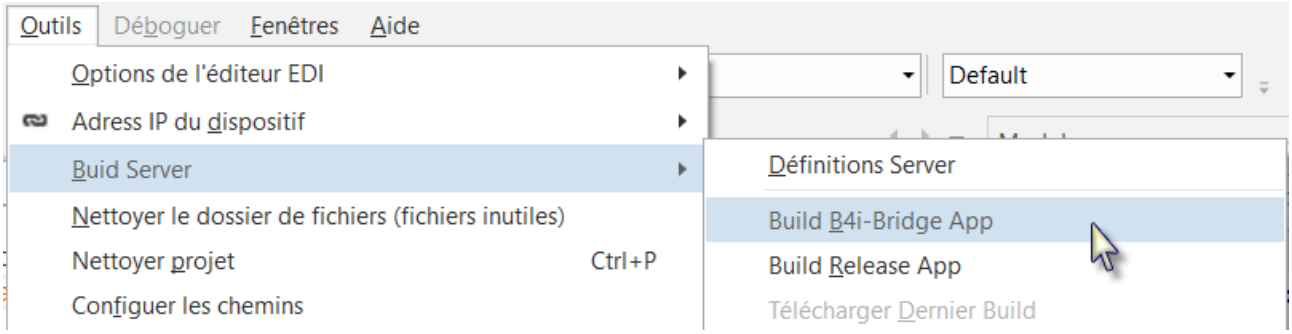
en

```
   #ApplicationLabel: MyFirstProgram
```

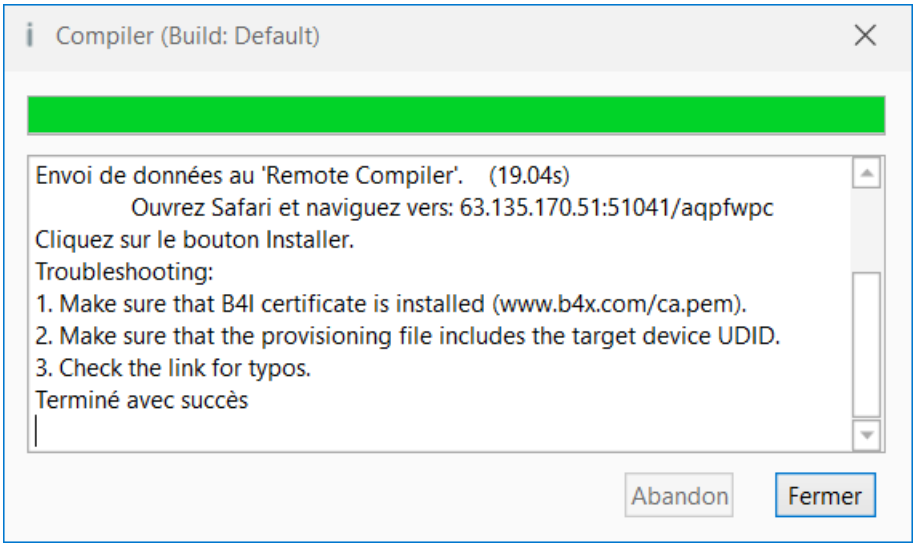
Les autres attributs sont expliqués dans le chapitre *Entêtes de code Project Attributes / Activity Attributes* dans le livret *B4x EDI*.

Dans l'EDI exécutez Build B4i-Bridge App.

Dans l'EDI, menu Outils / Build Server / Build B4i-Bridge App



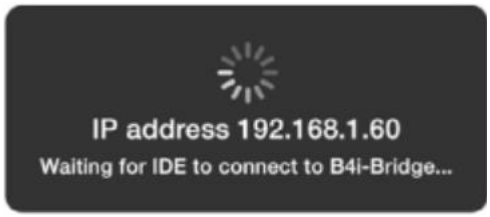
Vous obtenez cette fenêtre.



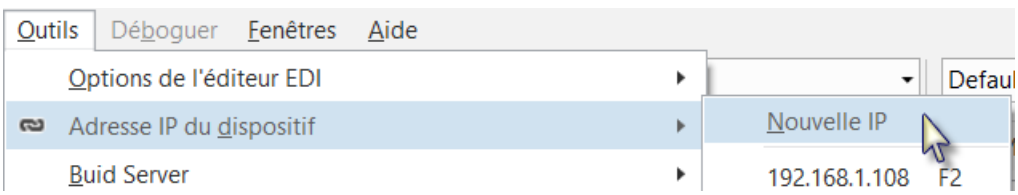
Sur le dispositif exécutez B4i-Bridge 

.

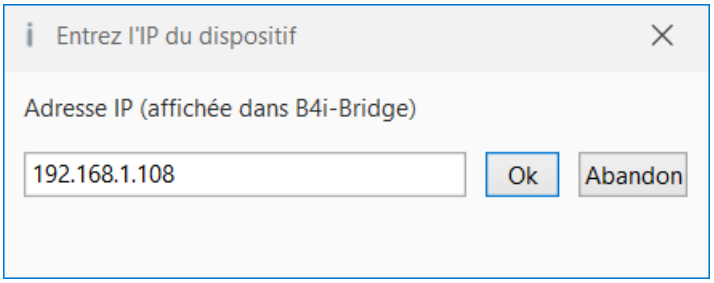
Sur son écran vous voyez l'adresse IP du dispositif.



Dans l'EDI cliquez sur Outils / Adresse IP du dispositif / Nouvelle IP



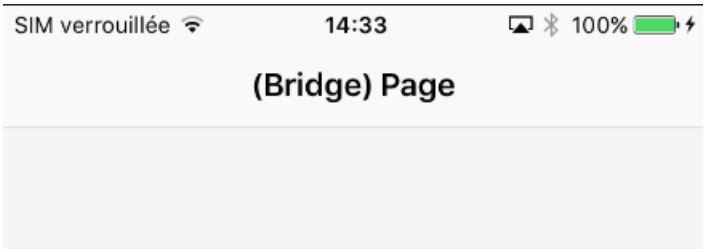
Entrez l'adresse IP :



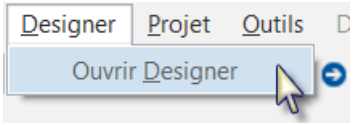
Cliquez sur 

.

Vous verrez cet écran sur le dispositif (seule la partie supérieure est affichée).

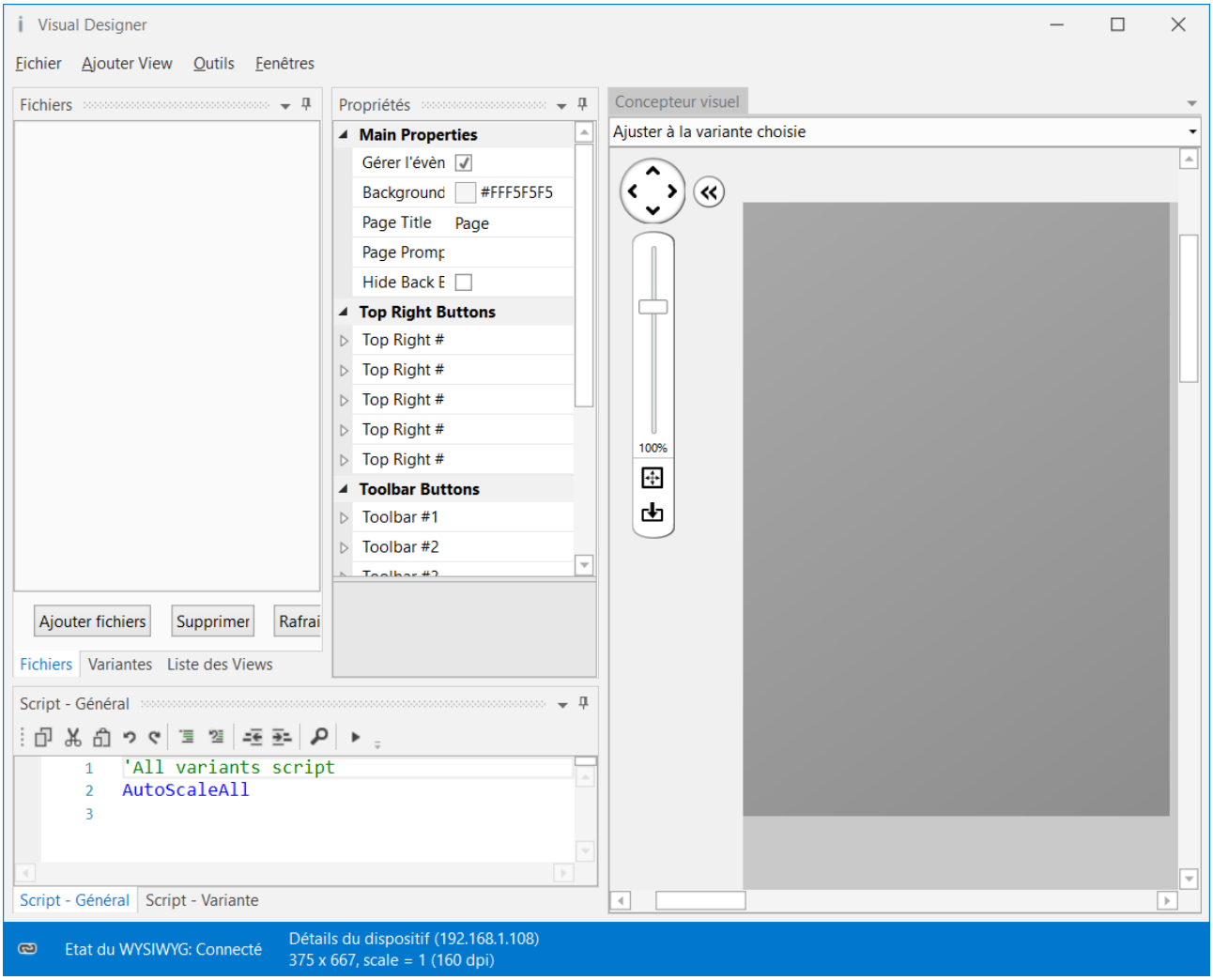


Dans l'EDI exécutez le Designer.

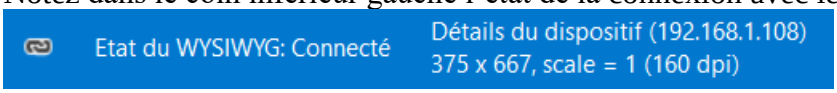


Attendez que le Designer soit prêt.

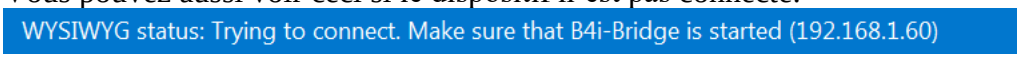
Le Designer ressemble à l'image ci-dessous.



Notez dans le coin inférieur gauche l'état de la connexion avec le dispositif :



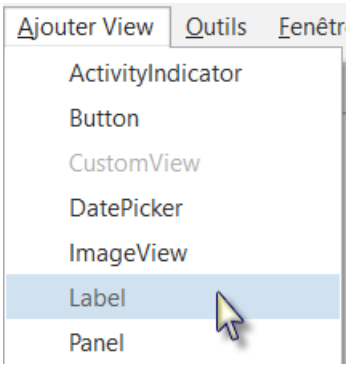
Vous pouvez aussi voir ceci si le dispositif n'est pas connecté.

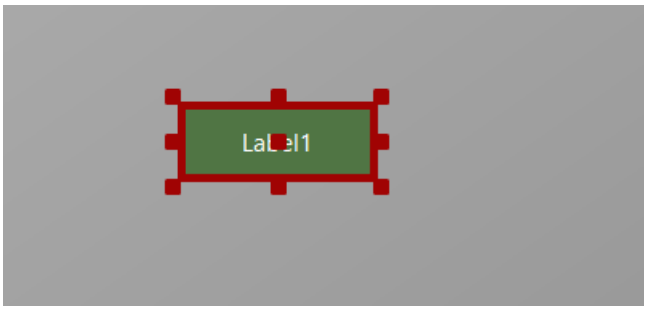


Dans le Designer il y aussi le Constructeur visuel qui affiche graphiquement le layout, pas exactement WYSIWYG, mais montre les positions et dimensions des différents objets.
Seul le haut de la fenêtre est montré ci-dessous.

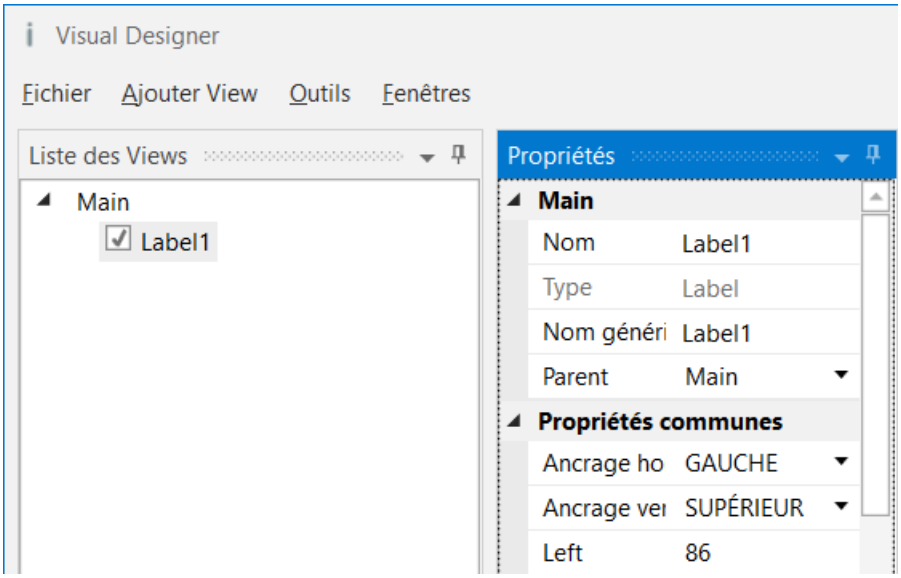
Le rectangle en gris foncé représente la surface du dispositif connecté.

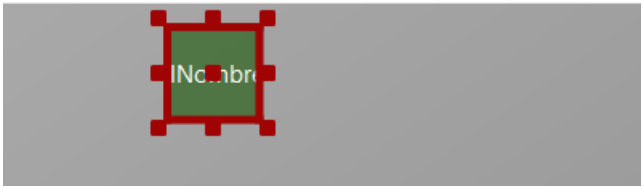
Maintenant, nous allons ajouter 2 Labels pour les nombres.
Dans le Designer, ajoutez un Label.



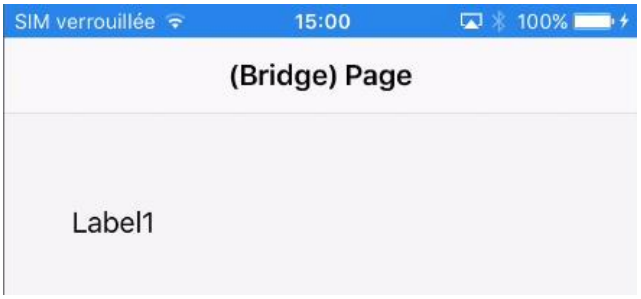


Le Label apparait dans le Concepteur visuel, dans la fenêtre Liste des views et les propriétés par défaut du Label sont affichées dans la fenêtre Propriétés.



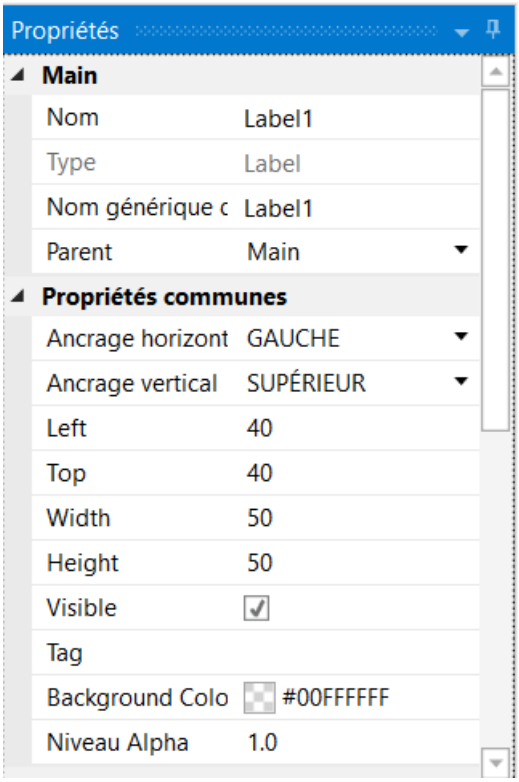


Redimensionnez et déplacez le Label avec les petits carrés rouges comme à gauche.



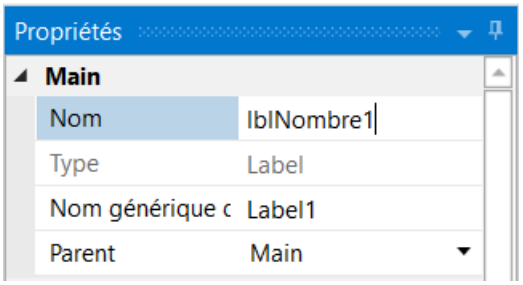
Vous pouvez suivre les layout sur le dispositif.

Pour le moment nous ne voyons que Label1.
La couleur de fond est transparente par défaut.
Label1 est le nom par défaut d'un Label.

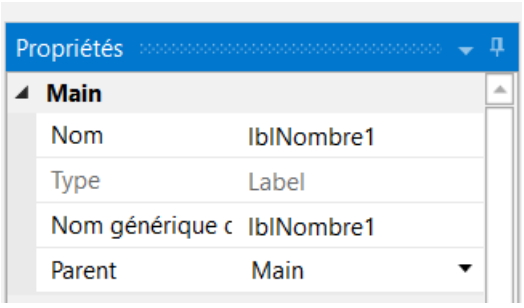


Les nouvelles propriétés Left, Top, Width et Height sont directement mises à jour dans la fenêtre Propriétés.
Vous pouvez aussi modifier les propriétés Left, Top, Width et Height directement dans la fenêtre Propriétés.

Nous modifions les propriétés du Label à nos besoins.
Par défaut, le nom est Label avec un nombre, ici Label1.
Nous modifions le nom en lblNombre1.
Les trois premières lettres 'lbl' pour 'Label', et 'Nombre1' pour le premier nombre.
Il est recommandé de donner des noms significatifs aux objets, de cette manière nous savons directement le type d'objet et sa fonction.



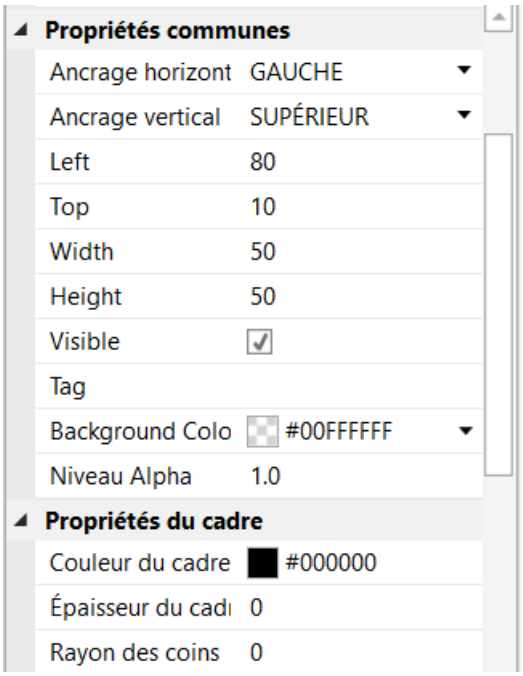
Presser la touche 'Entrée' ou cliquez quelque part ailleurs pour mettre à jour la propriété Event Name.



Nous avons maintenant :

Main :	Module Main.
Nom :	Nom de la view.
Type :	Type de la view. Dans ce cas, Label, affichage de texte non éditable.
Event Name :	Nom générique des routines qui gèrent les événements du Label.
Parent :	View parent à laquelle appartient le Label, Main dans ce cas.

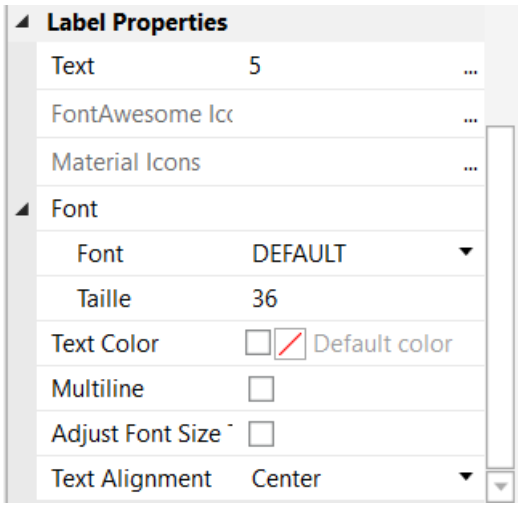
Vérifions et modifions les autres propriétés :



Définissez Left, Top, Width et Height avec les valeurs dans l'image.

Visible est coché.

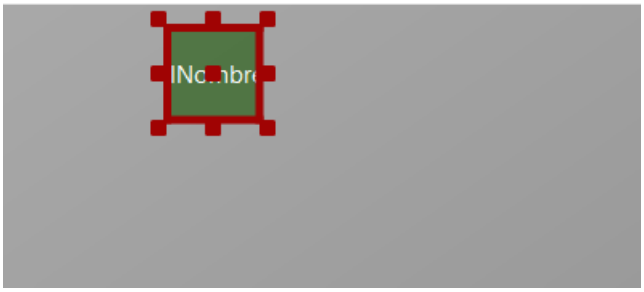
Nous gardons les couleurs par défaut.



Text mis à 5

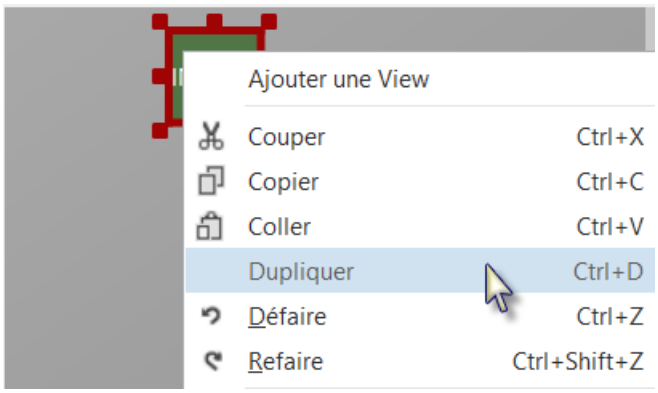
Nous gardons la police de caractères Font par default.
Taille, mis à 36.

Text Alignment mis à Center.



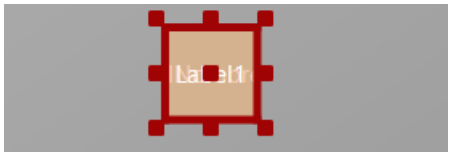
Et le résultat dans le Constructeur visuel et sur le dispositif.

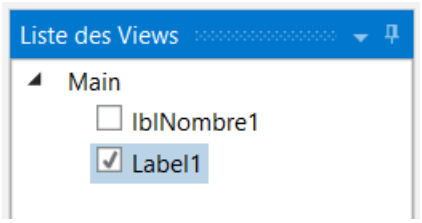
Nous avons besoin d'un deuxième Label similaire au premier. Au lieu d'en ajouter un nouveau, nous copions le premier avec les mêmes propriétés. Seule les propriétés Nom and Left seront modifiées.



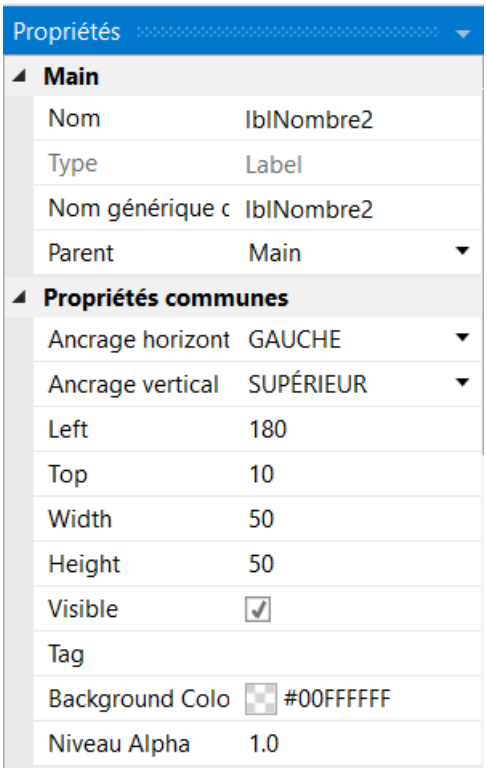
Cliquez avec le bouton droit sur lblNombre1 puis cliquez sur Dupliquer dans le menu contextuel.

Le nouveau Label couvre le Label dupliqué.





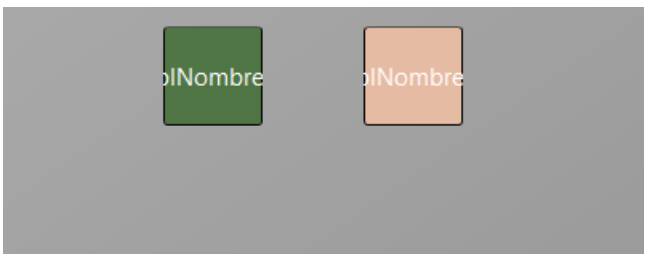
Sur la gauche, dans la fenêtre Liste des Views, vous voyez les différentes views.
Le nouveau Label Label1 est ajouté.

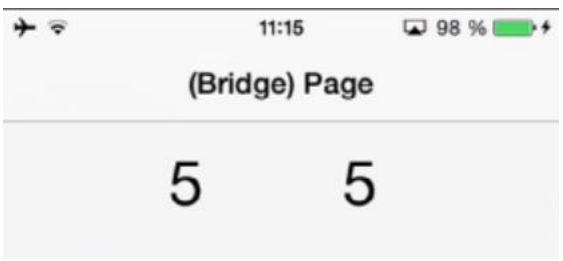


Nous positionnons maintenant le nouveau Label et modifions son nom en lblNombre2.

Modifiez le nom en lblNombre2.

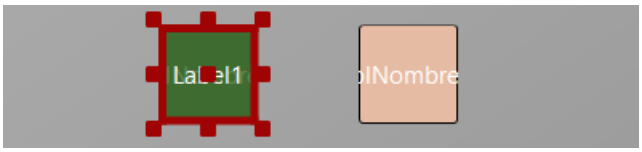
La propriété Left en 180.





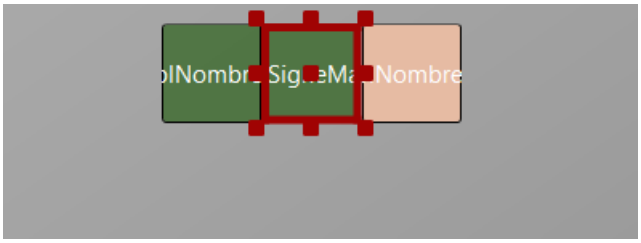
Et le résultat dans le Constructeur visuel et sur le dispositif.

Ajoutons un 3ème Label pour le signe mathématique. Nous dupliquons lblNombre1 encore une fois. Cliquez avec le bouton droit sur lblNombre1 puis cliquez sur Dupliquer dans le menu contextuel.



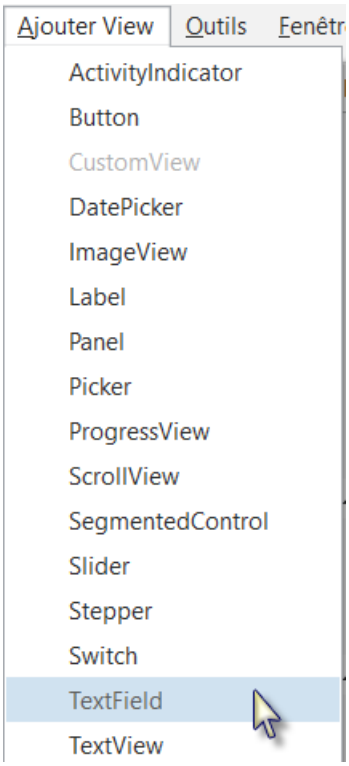
Le nouveau Label couvre lblNumber1.

Positionnez-le entre les deux autres Labels et modifiez son nom en lblSigneMath et sa propriété Text en '+'.



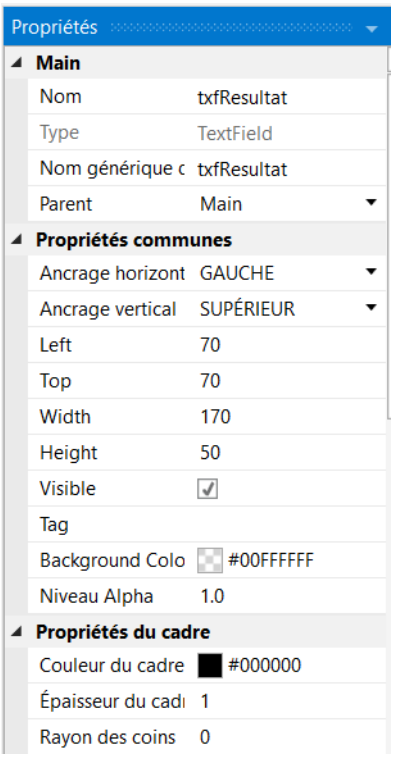
Et le résultat dans le Constructeur visuel et sur le dispositif.

Maintenant, ajoutons un objet TextField.



Dans le Constructeur visuel, dans le menu Ajouter View, cliquez sur TextField.

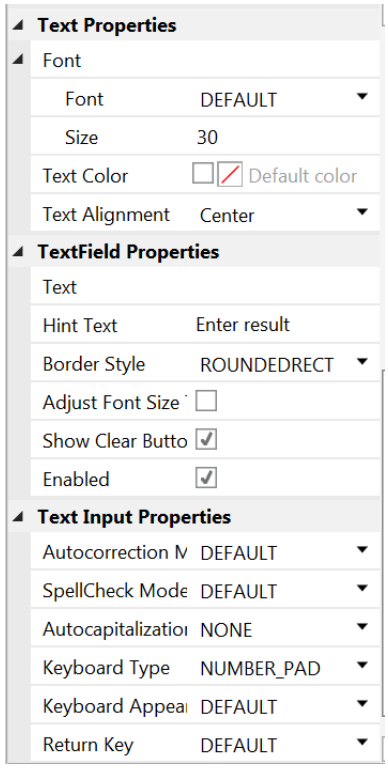
Positionnez le juste au-dessous des trois Labels et modifiez son nom en txfResultat.
'txf' signifie TextField et 'Resultat' sa fonction.



Modifiez ces propriétés.
Nom en txfResultat

Left, Top, Width and Height.

Épaisseur du cadre en 1

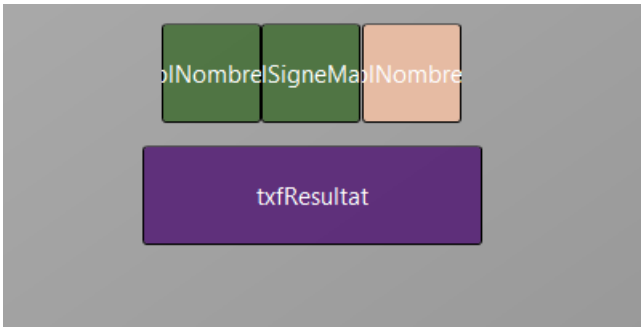


Size en 30

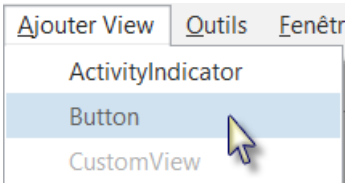
Text Alignment en Center

Hint Text en Résultat
Hint Text représente le texte affiché dans TextField s'il n'y a aucun texte écrit.

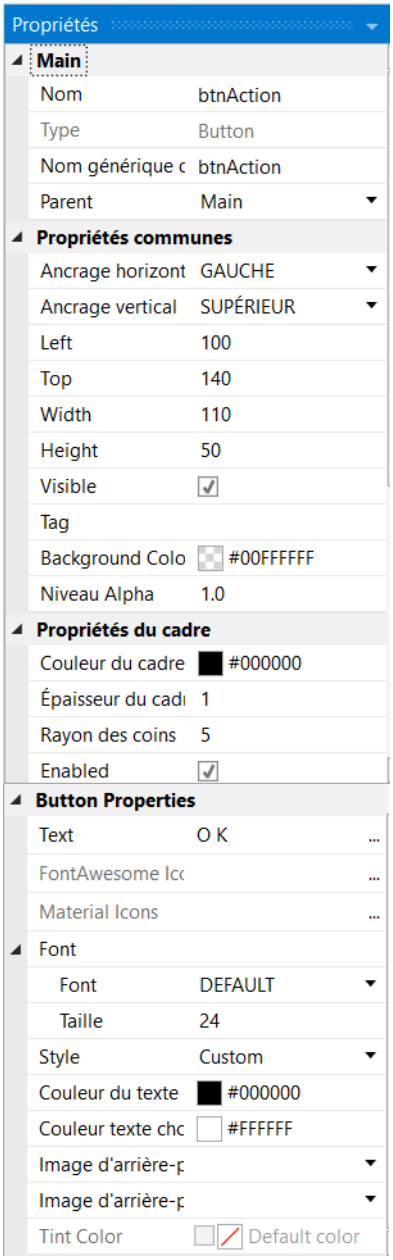
Keyboard Type en NUMBER_PAD
Définir Keyboard Type en NUMBER_PAD n'autorise que l'écriture de chiffres.



Et le résultat dans le Constructeur visuel et sur le dispositif.



Nous ajoutons un Button (bouton) qui, lorsqu'il est pressé vérifie le résultat fourni par l'utilisateur ou génère un nouveau problème dépendant de la réponse de l'utilisateur.



Positionnez le au-dessous du TextField txfResultat.
Redimensionnez-le et modifiez les propriétés ci-dessous :

Nom en btnAction

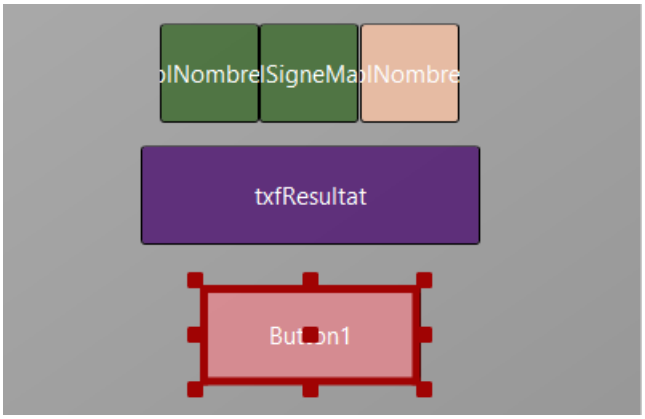
Left, Top, Width et Height.

Background Color en #FFBDBBBB

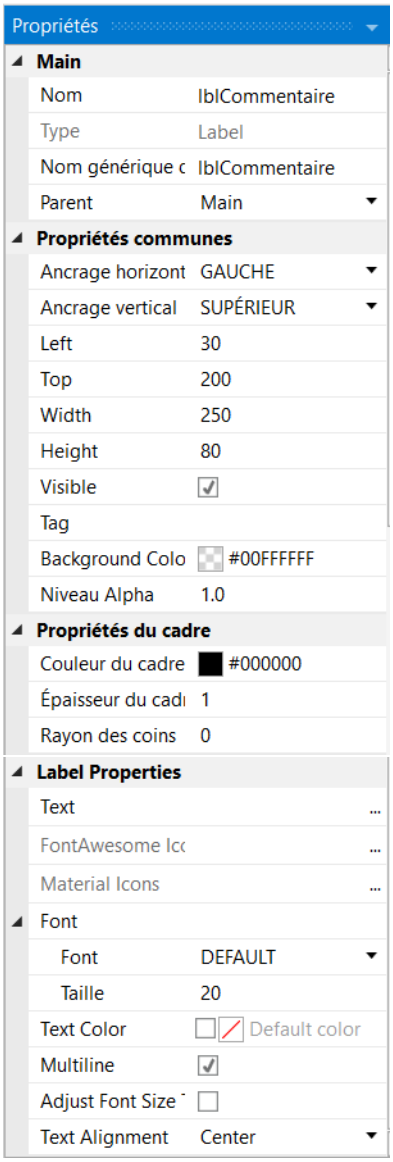
Épaisseur cadre en 1

Text en O K (avec un espace entre O et K)

Taille en 24



Et le résultat dans le Constructeur visuel et sur le dispositif.



Ajoutons encore le dernier Label pour afficher des commentaires. Positionnez le au-dessous du Button et redimensionnez-le.

Modifiez les propriétés ci-dessous :
Nom en lblCommentaire

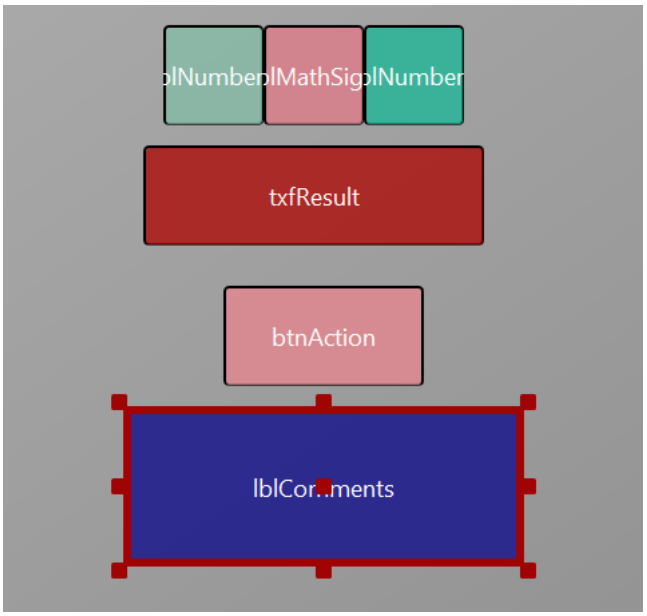
Left, Top, Width et Height.

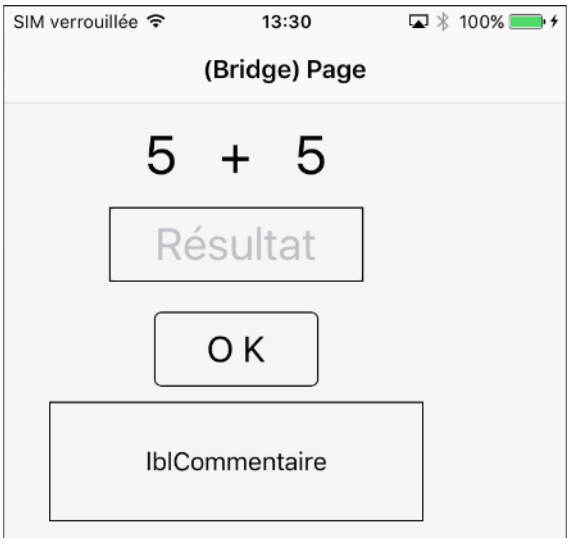
Epaisseur du cadre en 1

Taille en 20

Multilignes en True (coché)

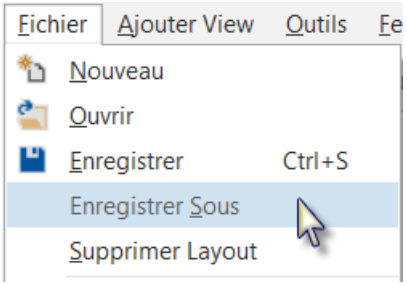
Text Alignment en Center



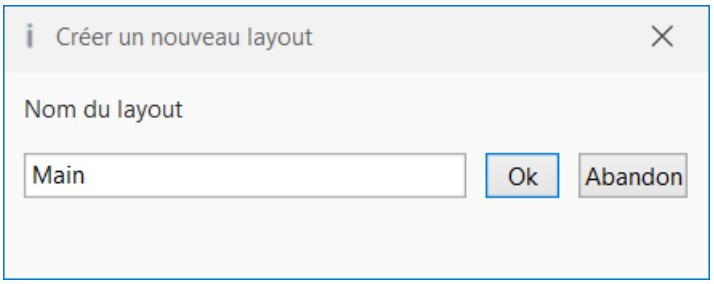


Et le résultat dans le Constructeur visuel et sur le dispositif.

Maintenant nous enregistrons le fichier layout.

Cliquez sur 

et enregistrez le fichier avec le nom 'Main'.

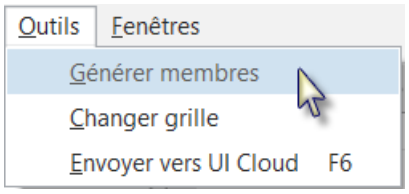


Cliquez sur 

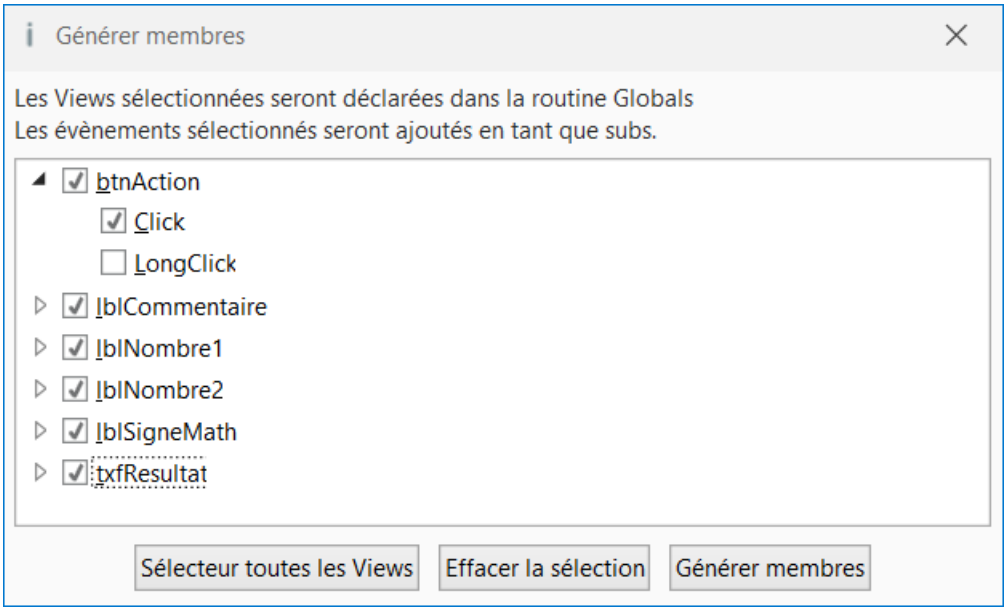
.

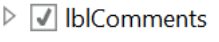
Pour le code du projet nous avons besoin de références pour les différentes views.
Nous générons ces références au moyen de la fonction *Générer Membres* du Designer.

La fonction *Générer Membres* génère automatiquement les références et cadres de routines d'événement pour les views.

Cliquez sur 

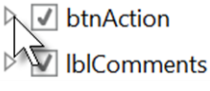
pour ouvrir le générateur.

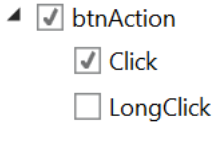


Dans cette fenêtre nous trouvons une liste de toutes les views définies dans le layout courant.
Nous cochons toutes les views et l'événement Click pour le bouton btnAction.
Le fait de cocher une view 

génère sa référence dans la routine Process_Globals.
Ces références sont nécessaires au compilateur et pour la fonction d'autocomplétion.

```
Private btnAction As Button
Private lblComments As Label
Private lblMathSign As Label
Private lblNumber1 As Label
Private lblNumber2 As Label
Private txfResult As TextField
```

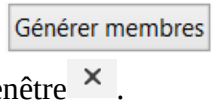
Un clic sur 

montre tous les événements de cette view 

.
Cocher un événement d'une view

générera le cadre pour cette routine d'événement.

```
Sub btnAction_Click

End Sub
```

Cliquez sur 

pour générer les références et le cadre des routines d'événement, puis fermez la fenêtre 
.

Maintenant nous retournons dans l'EDI pour écrire le code.

Dans le haut du code nous trouvons :

```
Sub Process_Globals
   'These global variables will be declared once when the application starts.
   'Public variables can be accessed from all modules.
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page

   Private btnAction As Button
   Private lblComments As Label
   Private lblMathSign As Label
   Private lblNumber1 As Label
   Private lblNumber2 As Label
   Private txfResult As TextField
End Sub
```

Ces lignes sont générées automatiquement dans l'EDI.

```
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page
```

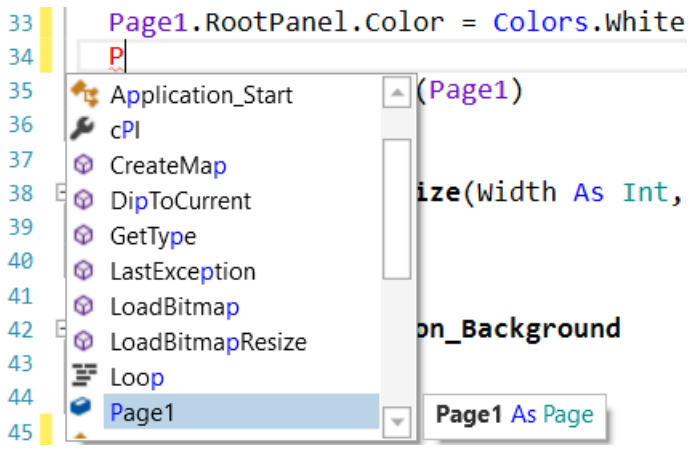
iOS nécessite au minimum ces objets : une Application, un NavigationControl et une Page, les détails sont expliqués dans le chapitre Flux du Programme / Cycle de vis dans le livret B4x Langage Basic.

Au-dessous nous trouvons la routine `Application_Start` qui est la première routine à être exécutée lors du démarrage du programme.
Le code ci-dessous est aussi généré automatiquement dans chaque nouveau projet.

```
Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   NavControl.ShowPage(Page1)
End Sub
```

```
NavControl = Nav                  > Définit NavControl comme NavigationController
Page1.Initialize("Page1")         > Initialise Page1, "Page1"  est le nom générique (EventName)
                                  pour  Page1.
Page1.Title = "Page 1"            > Définit le titre de la Page.
Page1.RootPanel.Color = Colors.White    > Définit la couleur de fond en blanc (white).
NavControl.ShowPage(Page1)        > Affiche Page1 sur le dispositif.
```

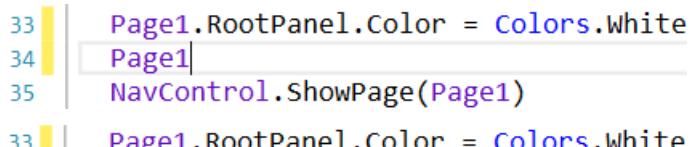
En premier, notre programme doit charger le fichier layout que nous avons défini dans les pages précédentes.
Le fichier doit être chargé sur RootPanel (panel racine) de Page1, nous le chargeons juste avant `NavControl.ShowPage(Page1)`
Nous profitons de la fonction d'autocompletion et de l'aide intégrée de B4i.



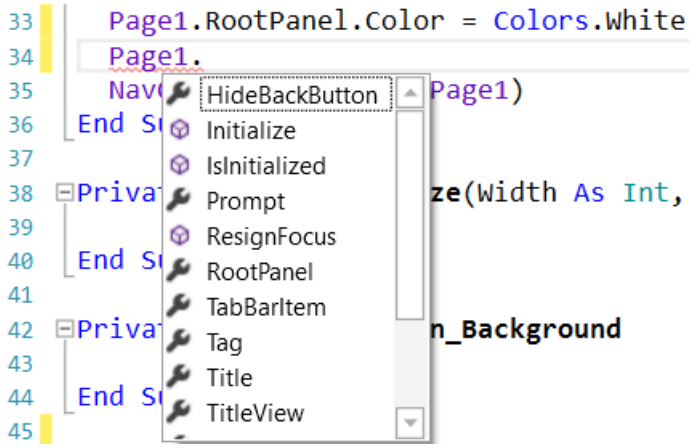
Entrez P dans une nouvelle ligne 34.

Une liste déroulante est affichée avec tous les mots clé, views et routines contenant 'P'.

Le premier élément commençant par 'P' est mis en évidence.



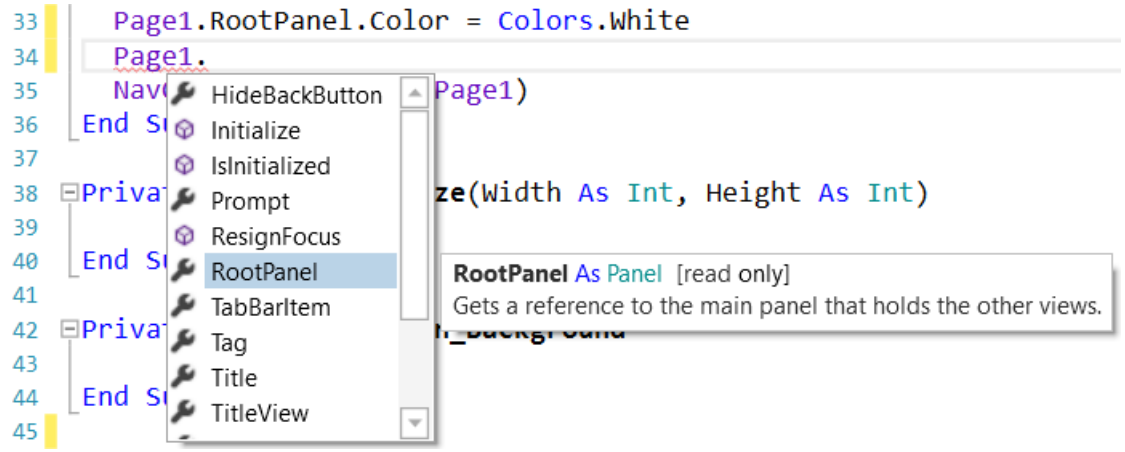
Pressez la touche Entrée pour valider.



`P` est complété en `Page1`.
Ajoutez un point "."

La liste déroulante contient toutes les propriétés de la view, Page dans notre cas.

Avec la touche flèche vers le bas, allez jusqu'à RootPanel.



Nous voyons RootPanel mis en évidence, et à côté de la liste l'aide en ligne avec la syntaxe pour les paramètres et une explication.

```
33    Page1.RootPanel.Color = Colors.White
34    Page1.RootPanel
35    NavControl.ShowPage(Page1)
```

Pressez la touche Entrée pour valider.

Ajoutez un point “.”

```
28    Page1.RootPanel.Color = Colors.White
29    Page1.RootPanel.
30    NavControl.Sh
31  End Sub
32
33  Private Sub Pag
34
35  End Sub
36
37  Private Sub App
38
39  End Sub
40
```

ACTION_DOWN
ACTION_MOVE
ACTION_UP
AddView
Alpha
BringToFront
Color
GetAllViewsRecursive
GetView

A nouveau, une liste déroulante avec les propriétés de la view concernée, celles d’un Panel dans notre cas, est affichée.

Avec la touche flèche vers le bas, allez jusqu’à LoadLayout.

A nouveau nous voyons la syntaxe et l’explication.

```
33    Page1.RootPanel.Color = Colors.White
34    Page1.RootPanel.
35    NavControl.Sh
36  End Sub
37
38  Private Sub Pag                    eight As Int)
39
40  End Sub
41
42  Private Sub App
43
44  End Sub
45
```

Initialize
IsFocused
IsInitialized
Left
LoadLayout
NumberOfViews
Parent
RemoveAllViews
RemoveViewAt
RemoveViewFromParent

LoadLayout (LayoutFile As String) As LayoutValues
Loads a layout file to the panel.

```
33    Page1.RootPanel.Color = Colors.White
34    Page1.RootPanel.LoadLayout
35    NavControl.ShowPage(Page1)
```

Pressez la touche Entrée pour valider.

Écrivez “(”.

```
33    Page1.RootPanel.Color = Colors.White
34    Page1.RootPanel.LoadLayout(
35    NavControl.ShowPage(Page1)
36  End Sub
```

LoadLayout (**LayoutFile** As String) As LayoutValues
Loads a layout file to the panel.

L’aide en ligne nous indique que faire avec une explication.

```
32    Page1.Title = "Page 1"
33    Page1.RootPanel.Color = Colors.White
34    Page1.RootPanel.LoadLayout("Main")
35    NavControl.ShowPage(Page1)
```

Complétez la ligne avec le nom du fichier layout.
L’extension du fichier n’est pas nécessaire.
Le nom du fichier `"Main"` est entre guillemets car c’est on objet String.

La ligne jaune dans la marge gauche indique que le code des lignes en question a été modifié.

```
33    Page1.RootPanel.Color = (
34    Page1.RootPanel.LoadLayou
35    NavControl.ShowPage(Page1
```

Dès que vous enregistrez le projet, la ligne jaune passe en vert.

Nous voulons générer un nouveau problème lors du démarrage. Pour cela, nous ajoutons un appel à la routine NouveauProbleme dans Application_Start.

```
Private Sub Application_Start (Nav As NavigationController)
   'SetDebugAutoFlushLogs(True) 'Uncomment if program crashes before all logs are
printed.
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   Page1.RootPanel.LoadLayout("Main")
   NavControl.ShowPage(Page1)

   NouveauProbleme
End Sub
```

NouveauProbleme est en rouge car la routine NouveauProbleme n'a pas encore été définie.
Générer un nouveau problème signifie générer aléatoirement deux nouveaux nombres, Nombre1 et Nombre2, entre 1 et 9 (inclusif) puis afficher ces deux nombres dans les propriétés 'Text' de lblNombre1 et lblNombre2.

Nous écrivons le code ci-dessous :
Dans la routine Process_Globals nous ajoutons les deux variables pour les deux nombres.

```
   Private Nombre1, Nombre2 As Int
End Sub
```

Puis nous écrivons la routine 'NouveauProbleme' :

```
Private Sub NouveauProbleme
   Nombre1 = Rnd(1, 10)          ' Génère un nombre aléatoire entre 1 et 9
   Nombre2 = Rnd(1, 10)          ' Génère un nombre aléatoire entre 1 et 9
   lblNombre1.Text = Nombre1     ' Affiche Nombre1 dans le Label lblNombre1
   lblNombre2.Text = Nombre2     ' Affiche Nombre2 dans le Label lblNombre2
   lblCommentaire.Text = "Entrez le résultat" & CRLF & "et cliquez sur OK."
   txfResultat.Text = ""         ' Vide edtResult.Text
End Sub
```

La fonction ci-dessous génère un nombre aléatoire entre '1' (inclusif) et '10' (exclusif) :
`Rnd(1, 10)`

Dans cette ligne `Nombre1 = Rnd(1, 10)     ' Génère un nombre aléatoire entre 1 et 9`
Le texte après l'apostrophe, `' Génère...`, est considéré comme un commentaire.
Il est conseillé d'ajouter des commentaires expliquant la fonctionnalité du code.

L'instruction ci-dessous affiche le commentaire dans le Label lblCommentaire :
`lblCommentaire.Text = "Entrez le résultat" & CRLF & "et cliquez sur OK."`
`CRLF` est le caractère Nouvelle Ligne.

Maintenant nous ajoutons le code pour l'événement Click du bouton Button.

Nous avons deux cas :
- Le texte de `btnAction` est égal à "O K", ce qui signifie qu'un nouveau problème est affiché, et que le programme attend que l'utilisateur entre un résultat et presse le bouton.
- Le texte de `btnAction` est égal à "Nouveau", ce qui signifie que l'utilisateur a entré un résultat correct et lorsqu'il presse `btnAction` un nouveau problème sera généré.

```
Sub btnAction_Click
   If btnAction.Text = "O K" Then
      If txfResultat.Text = "" Then
         lblCommentaire.Text = "Pas de résultat!" & CRLF & "Entrez le résultat" & CRLF &
"et cliquez sur OK."
      Else
         TestResultat
      End If
   Else
      NouveauProbleme
      btnAction.Text = "O K"
   End If
End Sub
```

`If btnAction.Text = "O K" Then` Vérifie si le texte du bouton est égal à "O K".
Si oui, nous vérifions si `txfResultat` est vide, pas de résultat entré par l'utilisateur.
Si oui nous affichons un message dans `lblCommentaire` indiquant qu'il n'y a pas de résultat dans `txfResultat`.
Si non, nous vérifions si le résultat est juste ou faux.
Si non, nous générons un bouveau problème, modifions le texte de `btnAction` en "O K" et vidons `txfResultat`.

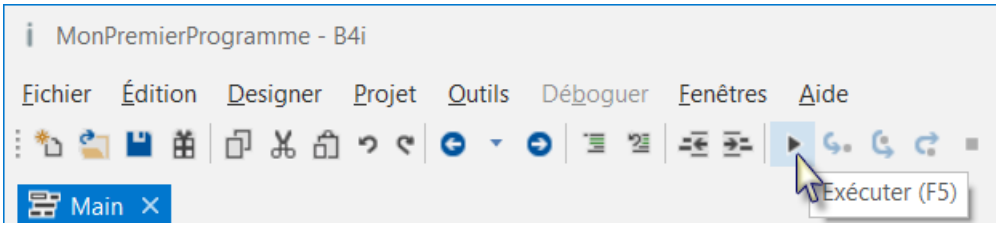
La dernière routine vérifie le résultat.

```
Private Sub TestResultat
   If txfResultat.Text = Nombre1 + Nombre2 Then
      lblCommentaire.Text = "Résultat  J U S T E." & CRLF & "Cliquez sur Nouveau."
      btnAction.Text = "Nouveau"
   Else
      lblCommentaire.Text = "Résultat  F A U X." & CRLF & "Entrez un nouveau résultat" & CRLF &
"et cliquez sur O K."
   End If
End Sub
```

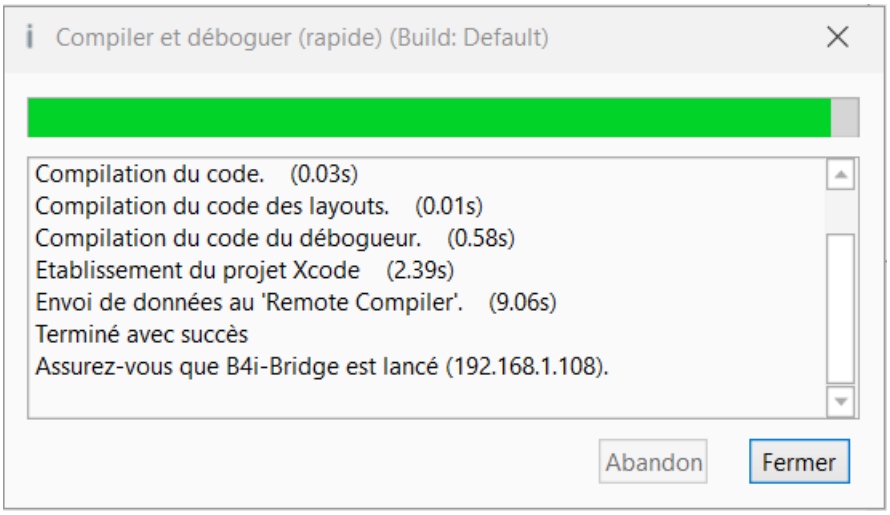
Avec `If txfResultat.Text = Nombre1 + Nombre2 Then` nous vérifions si le résultat est juste.

Si oui, nous affichons dans `lblCommentaire` le texte ci-dessous :
'Résultat J U S T E'
'Cliquez sur Nouveau'
Et modifions le texte de en "Nouveau".
Si non, nous affichons dans `lblCommentaire` le texte ci-dessous :
'Résultat F A U X'
'Entrez un nouveau résultat'
'et cliquez sur O K'

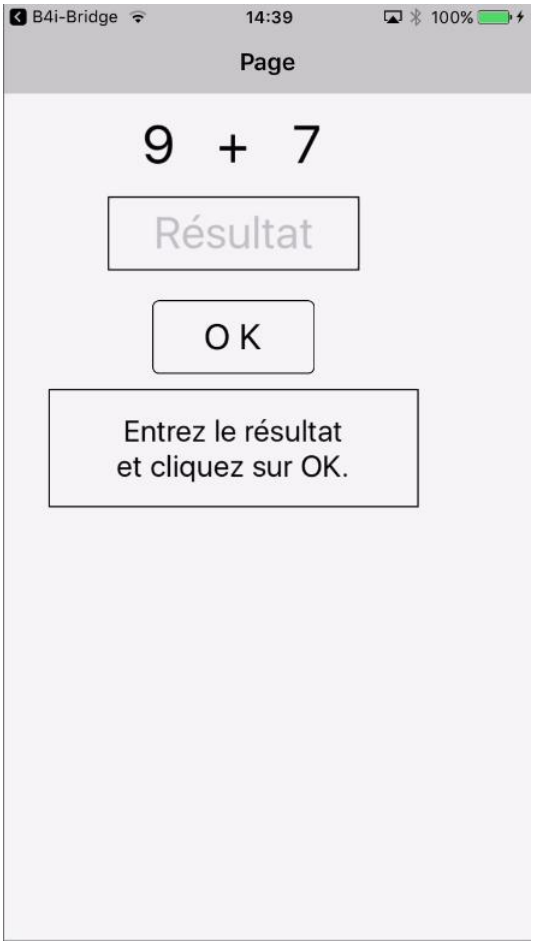
Nous allons compiler le programme et le transférer sur le dispositif.
Dans l'EDI cliquez sur ▶ ou pressez F5 :



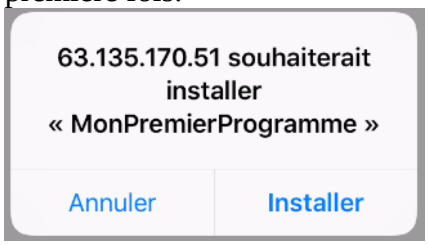
Le programme va être compilé.



Lorsque vous voyez 'Terminé avec succès.' Dans la fenêtre de compilation, la compilation et le transfert sont terminés.



Sur le dispositif vous verrez un message similaire à celui ci-dessous lorsque vous exécutez le programme la première fois.



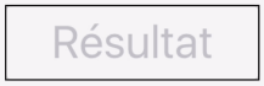
Touchez **Installer**.

Vous verrez quelque part sur votre dispositif l'icône du programme 

, touchez la pour l'exécuter.

Et vous verrez un écran similaire à celui à gauche, mais avec des nombres différents.

Nous pourrions, bien sûr, apporter des améliorations fonctionnelles et esthétiques au projet, mais ça n'était pas le but de ce premier projet, mais est l'objet du projet SecondProgramme.

Touchez 

pour activer le clavier.

Entrez le résultat, 16 dans l'exemple.

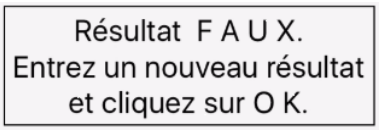
Vous verrez un écran similaire à celui à gauche.

Cliquez sur 

pour confirmer le résultat.

Si le résultat est juste, vous verrez un écran similaire à celui à gauche.

Si le résultat est faux, le message ci-dessous sera affiché.



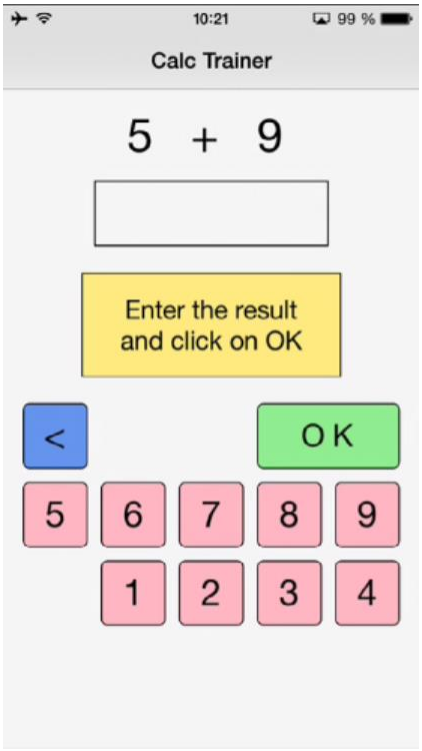
3.9 Second programme B4i (SecondProgram.b4i)

Le projet se trouve dans le dossier CodesSource :
CodesSource\SecondProgramme\B4i\SecondProgramme.b4i.

Améliorations par rapport au projet “MonPremierProgramme”.

- Clavier numérique indépendant évitant l’utilisation du clavier virtuel.
- Couleurs dans les commentaires.
- Centrage des différents objets.

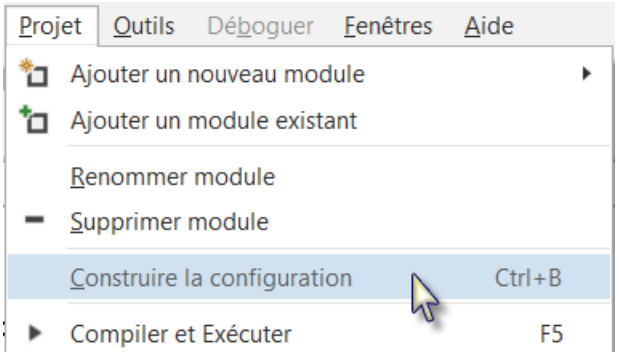
Créez un nouveau dossier et nommez-le “SecondProgramme”. Copiez tous les fichiers du projet MonPremierProgramme dans ce nouveau dossier et renommez les fichiers MonPremierProgramme.b4i en SecondProgramme.b4i et MonPremierProgramme.b4i.meta en SecondProgramme.b4i.meta.

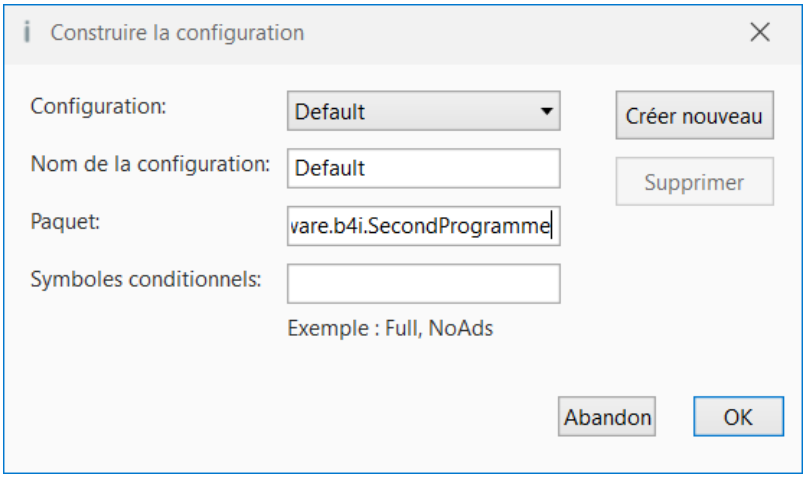


Chargez le nouveau programme dans l’EDI.

Nous devons modifier le nom du Paquet.

Dans l’EDI, menu Projet.
Cliquez sur Construire la configuration.





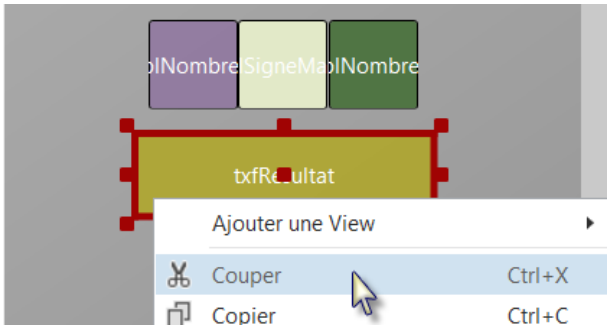
Modifiez le nom du paquet en anywheresoftware.b4i.SecondProgram et cliquez sur OK.

Nous devons encore modifier le paramètre ApplicationLabel sur le haut du code.

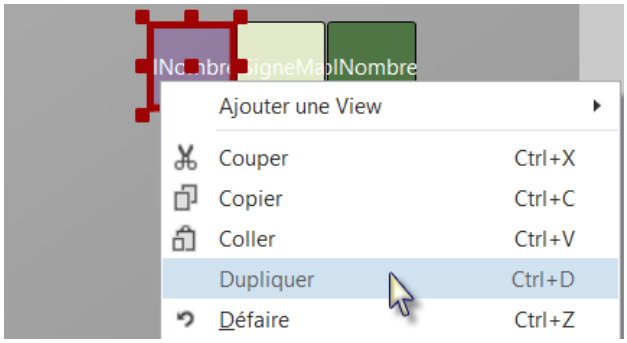
```
#Region  Project Attributes
   #ApplicationLabel: SecondProgramme
```

Exécutez le Designer.

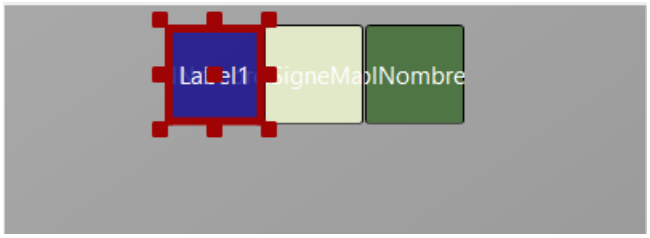
Nous remplaçons la view txfResultat TextField par un nouveau Label.
Exécutez le Designer et dans le Concepteur visuel cliquez sur la view txfResultat.



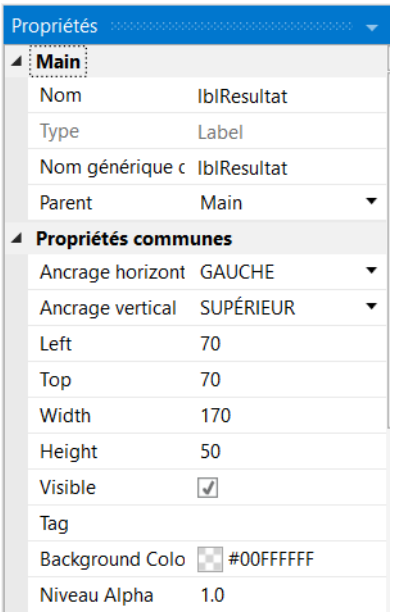
Cliquez avec le bouton droit sur txfResultat
et cliquez sur Couper.



Cliquez avec le bouton droit sur lblNombre1
et cliquez sur Dupliquer.



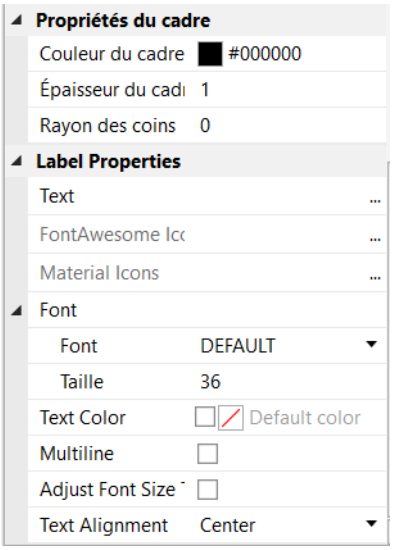
Le nouveau Label couvre lblNombre1.



Modifiez les propriétés ci-dessous :

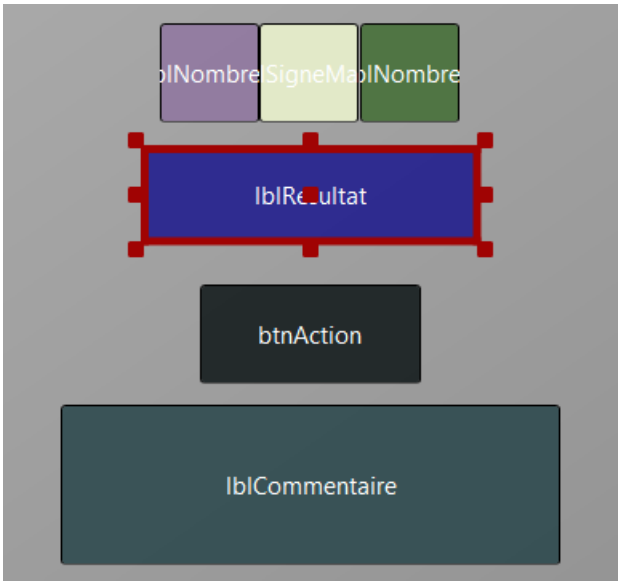
Nom en lblResultat

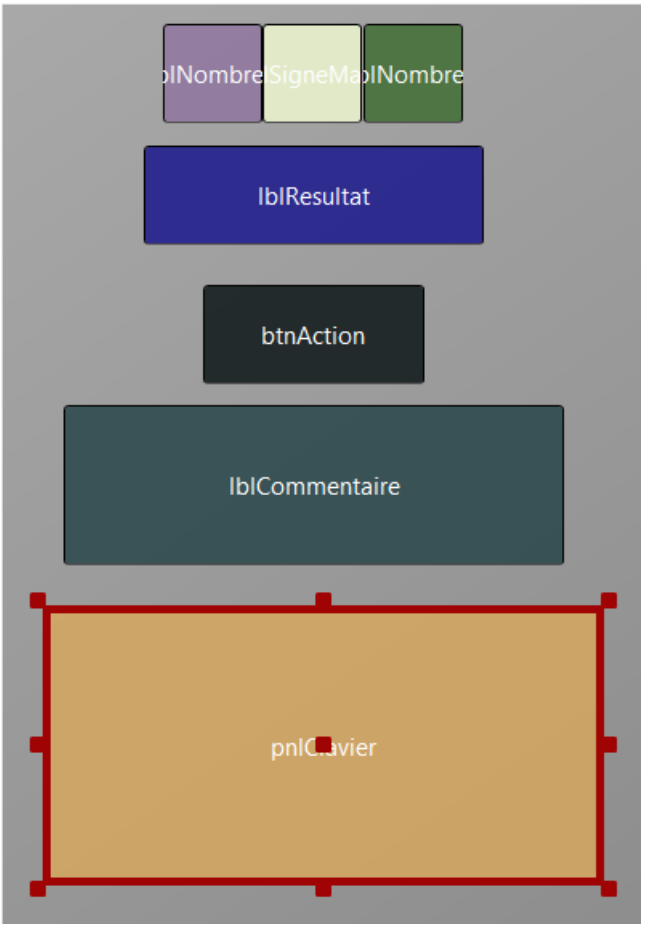
Left, Top, Width, Height



Épaisseur de cadre en 1

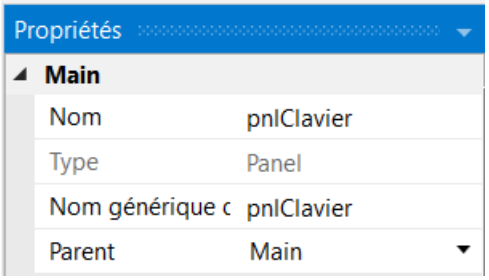
Text en "" vide, pas de caractères.





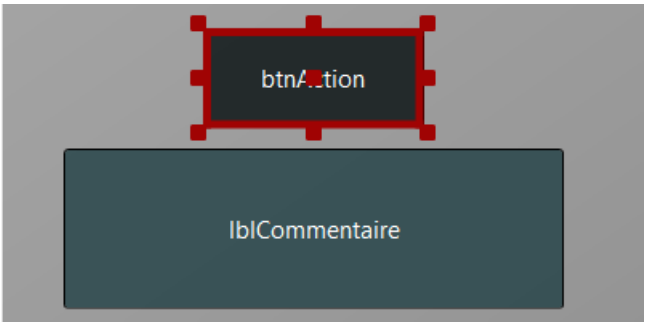
Ajoutons maintenant un Panel pour les boutons du clavier.

Positionnez et redimensionnez-le comme dans l'image.



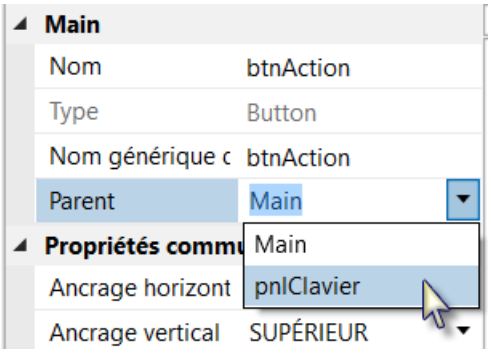
Modifiez son nom en pnlClavier
"pnl" pour Panel, le type de la view.

Modifiez
Rayon des coins en 0

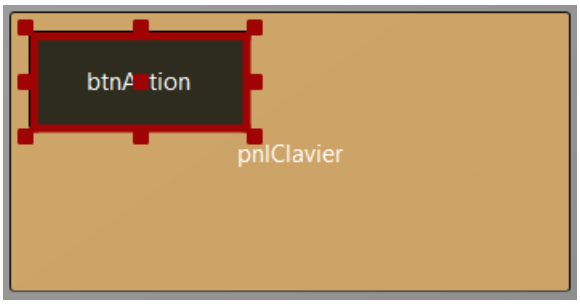


Nous déplaçons btnAction de Main sur le Panel pnlClavier.

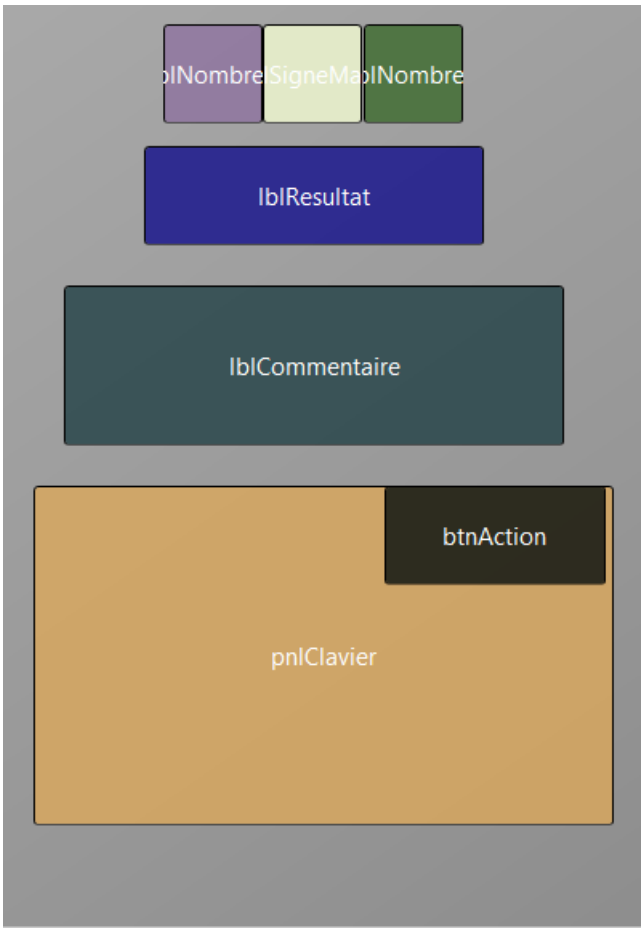
Cliquez sur btnAction.



Dans la liste Parent cliquez sur pnlClavier.



Le bouton btnAction appartient maintenant au Panel pnlClavier.



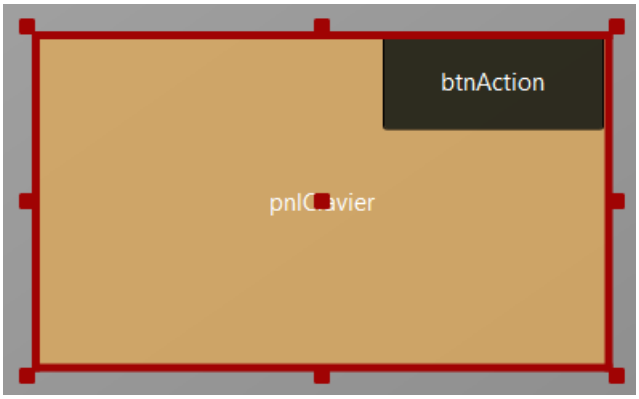
Nous réarrangeons les views pour obtenir plus de place pour le clavier

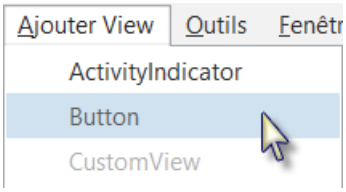
Modifiez les propriétés ci-dessous :

lblCommentaire	Top = 140
pnlClavier	Left = 15
pnlClavier	Top = 240
pnlClavier	Width = 290
pnlClavier	Height = 170
pnlClavier	Épaisseur de cadre = 0

Déplacez btnAction dans le coin supérieur droit de pnlClavier.

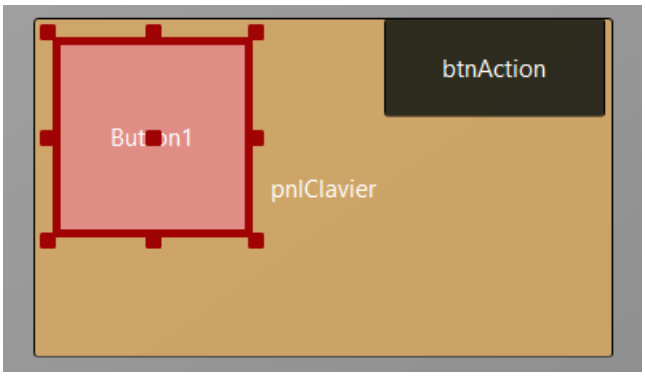
Cliquez sur le Panel pnlClavier pour le sélectionner.



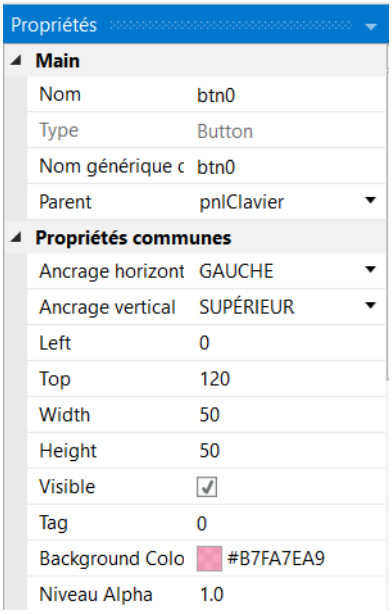


Cliquez sur

Pour ajouter un nouveau bouton.



Le nouveau bouton est ajouté.

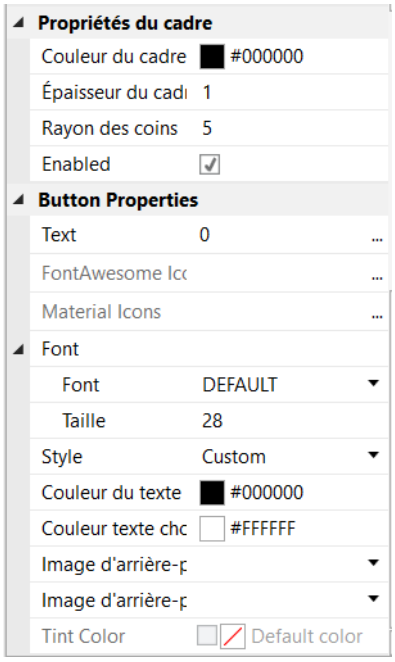


Modifiez les propriétés ci-dessous :

Nom en btn0
Nom générique de l'événement en btnEvent

Left en 0
Top en 120
Width en 50
Height en 50

Tag en 0
Background Color to #B7FA7EA9

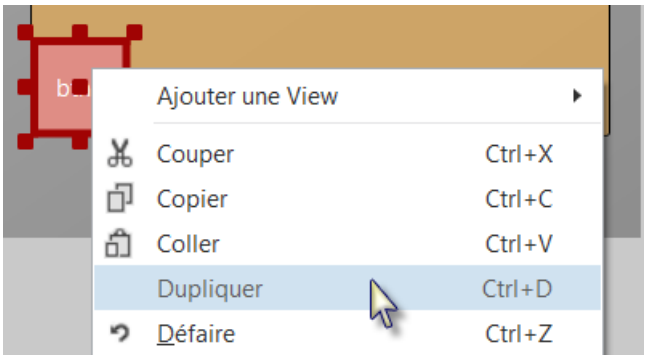


Épaisseur du cadre en 1
Rayon des coins en 5

Text en 0

Taille en 28

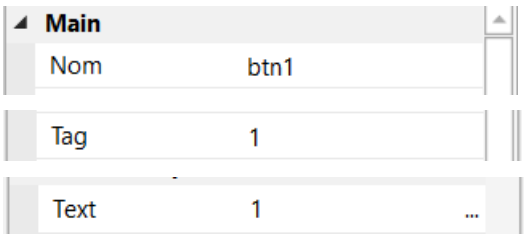
Le bouton ressemblera à ceci.



Duplquons btn0 et positionnons-le juste à côté de btn0.

Cliquez avec le bouton droit sur btn0 et cliquez sur Dupliquer.

Déplacez le nouveau bouton btn0 à côté de btn0 avec un petit espace.

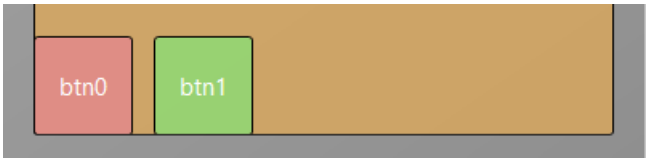


Modifiez les propriétés ci-dessous :

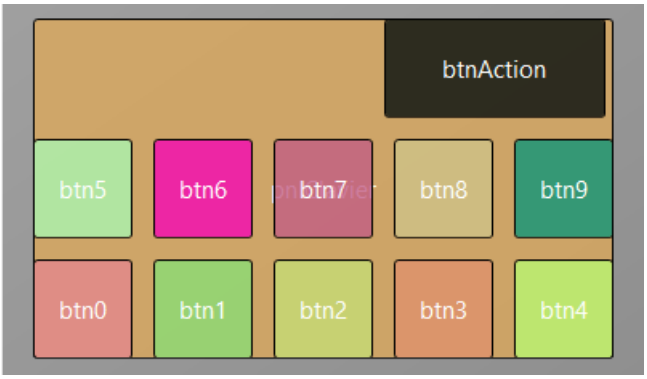
Nom en btn1

Tag en 1

Text en 1



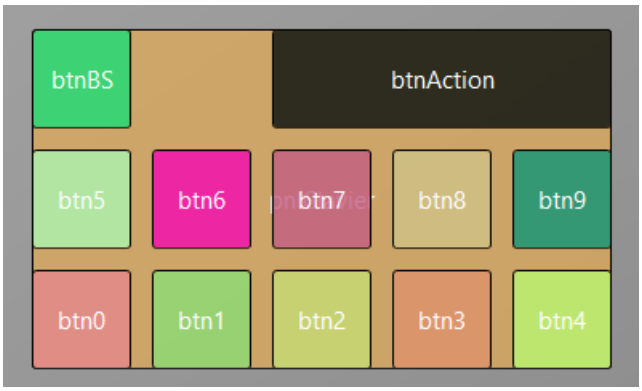
Et le résultat.



Ajoutez 8 boutons supplémentaires et positionnez-les comme dans l'image.

Modifiez leurs propriétés comme ci-dessous :

Nom	btn2,	btn3,	btn4	etc.
Tag	2 ,	3 ,	4	etc.
Text	2 ,	3 ,	4	etc.



Pour créer le bouton Retour Arrière (BackSpace), nous dupliquons un des boutons nombre et positionnons-le dans le coin supérieur gauche.

Redimensionnez et positionnez btnAction.

Modifiez les propriétés Name, Tag, Text et Color comme ci-dessous.

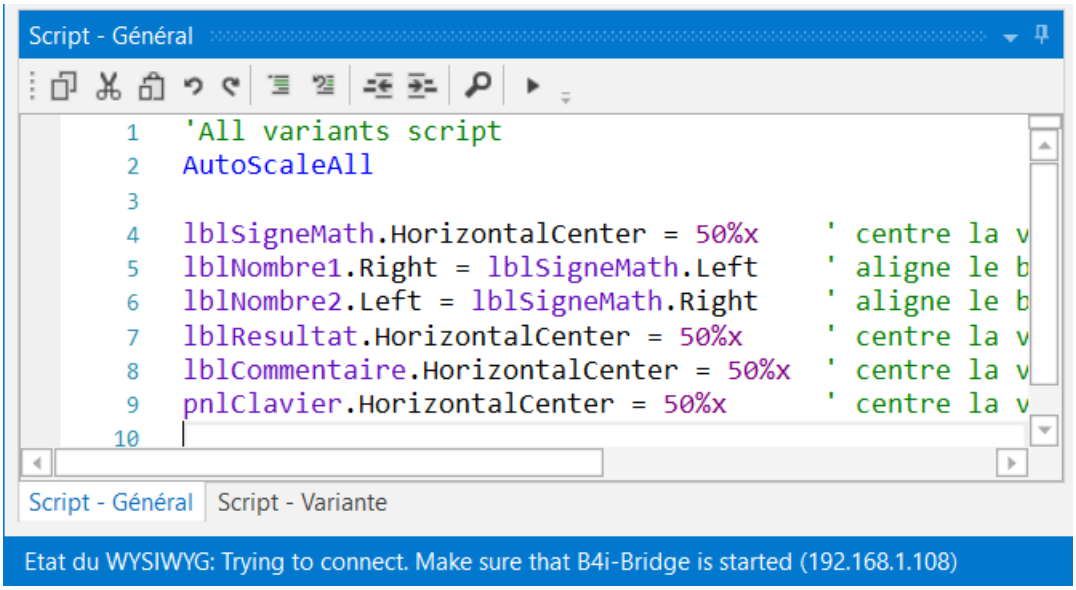
btnBS <

Propriétés	
Main	
Nom	btnBS
Type	Button
Nom générique c	btnBS
Parent	pnlClavier
Propriétés communes	
Ancrage horizont	GAUCHE
Ancrage vertical	SUPÉRIEUR
Left	0
Top	0
Width	50
Height	50
Visible	☑
Tag	BS
Background Colo	#FF7E88FA
Niveau Alpha	1.0
Propriétés du cadre	
Couleur du cadre	#000000
Épaisseur du cadı	1
Rayon des coins	5
Enabled	☑
Button Properties	
Text	< ...
FontAwesome Icc	...
Material Icons	...
Font	
Font	DEFAULT
Taille	28
Style	Custom
Couleur du texte	#000000
Couleur texte chc	#FFFFFF
Image d'arrière-ţ	
Image d'arrière-ţ	
Tint Color	Default color

btnAction O K

Propriétés	
Main	
Nom	btnAction
Type	Button
Nom générique c	btnAction
Parent	pnlClavier
Propriétés communes	
Ancrage horizont	GAUCHE
Ancrage vertical	SUPÉRIEUR
Left	120
Top	0
Width	170
Height	50
Visible	☑
Tag	
Background Colo	#FF03F86D
Niveau Alpha	1.0
Propriétés du cadre	
Couleur du cadre	#000000
Épaisseur du cadı	1
Rayon des coins	5
Enabled	☑
Button Properties	
Text	O K ...
FontAwesome Icc	...
Material Icons	...
Font	
Font	DEFAULT
Taille	24
Style	Custom
Couleur du texte	#000000
Couleur texte chc	#FFFFFF
Image d'arrière-ţ	
Image d'arrière-ţ	
Tint Color	Default color

Une autre amélioration consiste à centrer les différents objets horizontalement sur l'écran.
Pour cela, nous ajoutons le code ci-dessous dans le Designer, dans la fenêtre Script-Général.




```
'All variants script
AutoScaleAll

lblSigneMath.HorizontalCenter = 50%x   ' centre la view au mileu de l'écran
lblNombre1.Right = lblSigneMath.Left   ' aligne le bord droit sur le bord gauche
lblNombre2.Left = lblSigneMath.Right   ' aligne le bord gauche sur le bord droit
lblResultat.HorizontalCenter = 50%x    ' centre la view au mileu de l'écran
lblCommentaire.HorizontalCenter = 50%x ' centre la view au mileu de l'écran
pnlClavier.HorizontalCenter = 50%x     ' centre la view au mileu de l'écran
```

Les deux premières lignes existent par défaut, nous les laissons.

```
'All variants script
AutoScaleAll ' adapte l'échelle à la taille de l'écran
```

```
lblSigneMath.HorizontalCenter = 50%x
```

`HorizontalCenter` centre une view horizontalement à la valeur définie, `50%x` dans notre cas donc le milieu de l'écran.

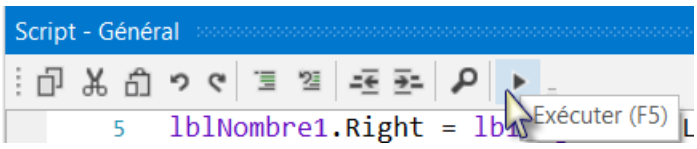
```
lblNombre1.Right = lblSigneMath.Left
```

Aligne le bord droit de `lblNombre1` au bord gauche de `lblSigneMath`, positionne `lblNombre1` à gauche juste à côté de `lblSigneMath`.

```
lblNombre2.Left = lblSigneMath.Right
```

Aligne le bord gauche de `lblNombre2` au bord droit de `lblSigneMath`, positionne `lblNombre2` à droite juste à côté de `lblSigneMath`.

Pour voir le résultat, cliquez sur  dans la fenêtre Script – Général.
Ce qui exécute le code.



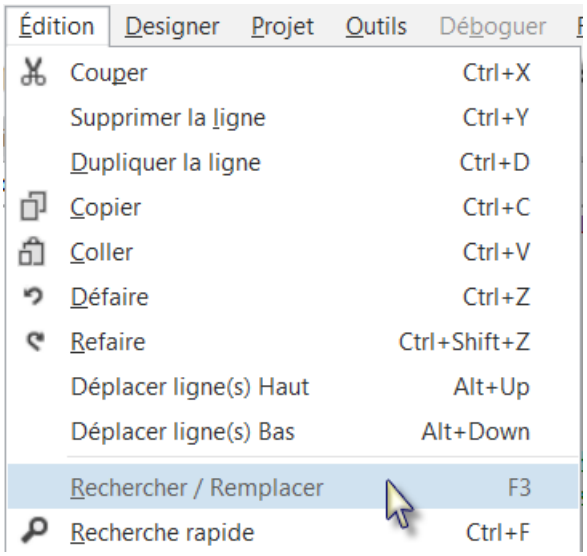
Le nouveau layout sur le dispositif.

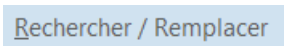
Si vous aviez connecté votre dispositif dès le début vous auriez pu suivre les évolutions du layout au fur et à mesure.

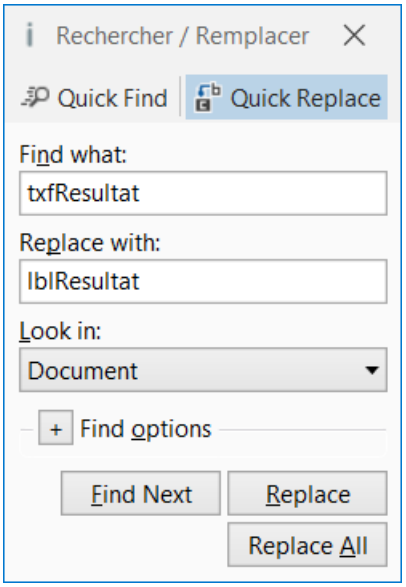
Maintenant nous allons mettre à jour le code.
En premier, nous devons remplacer txfResultat par `lblResultat` puisque nous remplaçons un objet TextField par un objet Label.

```
20    Private btnAction As Button
21    Private lblCommentaire As Label
22    Private lblNombre1 As Label
23    Private lblNombre2 As Label
24    Private lblSigneMath As Label
25    Private txfResultat As TextField
```

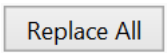
Double cliquez sur txfResultat pour le sélectionner.



Cliquez sur 



Le fenêtre Rechercher / Remplacer est affichée.

Cliquez sur 

et fermez la fenêtre.

Nous devons aussi modifier le type de la view de TextField à Label.

```
Private lblResult As Label
```

Maintenant, nous écrivons la routine qui gère les événements Click des boutons.
La propriété Nom générique des événements (Event Name) est "btnEvent" pour tous les boutons, sauf pour btnAction.
Le nom de la routine associée au événements Click est btnEvent_Click.

Écrivez le code ci-dessous :

```
Private Sub btnEvent_Click

End Sub
```

Nous avons besoin de savoir quel bouton a généré l'événement. Pour ça, nous utilisons un objet spécifique Sender qui contient la référence de la view qui a généré l'événement.

```
Private Sub btnEvent_Click
   Dim btnSender As Button

   btnSender = Sender

   Select btnSender.Tag
   Case "BS"

   Else
   End Select
End Sub
```

Pour pouvoir accéder aux propriétés de la view qui a généré l'événement, nous déclarions une variable locale.
`Dim btnSender As Button`
Et définissons `btnSender = Sender`.

Puis, pour différencier entre le bouton Retour Arrière (BS) et les boutons numériques nous utilisons une structure Select / Case / End Select et utilisons la propriété Tag des boutons.
Rappelez-vous, lorsque nous avons ajouté les différents boutons, nous avions défini leur propriété Tag à BS, 0, 1, 2 etc.

```
   Select btnSender.Tag
   Case "BS"
   Case Else
   End Select
End Sub
```

Sélectionne la variable à vérifier.
Vérifie si c'est le bouton avec la propriété Tag = "BS".
Gère tous les autres boutons.

Ajoutons le code pour les boutons numériques.
Nous ajoutons la valeur de la propriété Text du bouton qui a généré l'événement au texte dans la propriété Text du Label lblResultat.

```
   Select btnSender.Tag
   Case "BS"
   Case Else
      lblResultat.Text = lblResultat.Text & btnSender.Text
   End Select
End Sub
```

Ceci est effectué dans cette ligne.

```
   lblResultat.Text = lblResultat.Text & btnSender.Text
```

Le caractère "&" signifie concaténation, nous ajoutons simplement la valeur de la propriété Text du bouton qui a généré l'événement au texte déjà contenu dans la propriété Text du Label lblResultat.

Ajoutons, maintenant, le code pour le bouton Retour Arrière (BS).

```
   Select btnSender.Tag
   Case "BS"
      If lblResult.Text.Length > 0 Then
         lblResult.Text = lblResult.Text.SubString2(0, lblResult.Text.Length - 1)
      End If
   Case Else
      lblResult.Text = lblResult.Text & btnSender.Text
   End Select
End Sub
```

Lorsqu'on presse le bouton BS nous devons effacer le dernier caractère du texte contenu dans la propriété Text de lblResultat. Néanmoins, ceci n'est valable que si la longueur du texte est plus grande que 0. Ceci est vérifié avec :

```
If lblResult.Text.Length > 0 Then
```

Pour effacer le dernier caractère nous utilisons la fonction SubString2.

```
lblResultat.Text = lblResultat.Text.SubString2(0, lblResultat.Text.Length - 1)
```

`SubString2(BeginIndex, EndIndex)` extrait un nouvel objet String commençant à l'index BeginIndex (inclusif) jusqu'à l'indice EndIndex (exclusif). Dans notre cas, depuis 0 (premier caractère) jusqu'à la longueur du texte moins 1.

La routine est maintenant terminée.

```
Private Sub btnEvent_Click
   Private btnSender As Button

   btnSender = Sender
   Select btnSender.Tag
   Case "BS"
      If lblResultat.Text.Length > 0 Then
         lblResultat.Text = lblResultat.Text.SubString2(0, lblResult.Text.Length - 1)
      End If
   Case Else
      lblResultat.Text = lblResultat.Text & btnSender.Text
   End Select
End Sub
```

Dans la routine `Sub btnAction_Click` nous ajoutons, à la fin, `lblResult.Text = ""` pour vider le contenu.

```
   Else
      New
      btnAction.Text = "O K"
      lblResult.Text = ""
   End If
End Sub
```

Nous pouvons améliorer l'interface utilisateur en ajoutant des couleurs de fond dans `lblCommentaire` pour les commentaires.

Définissons :
- Jaune pour nouveau problème
- Vert clair pour résultat JUSTE
- Rouge clair pour résultat FAUX.

Nous ajoutons en premier la ligne ci-dessous dans la routine NouveauProbleme.
```
lblComments.Color = Colors.RGB(255,235,128)
```

```
Private Sub NouveauProbleme
   Nombre1 = Rnd(1, 10)             ' Génère un nombre aléatoire entre 1 et 9
   Nombre2 = Rnd(1, 10)             ' Génère un nombre aléatoire entre 1 et 9
   lblNombre1.Text = Nombre1        ' Affiche Nombre1 dans le Label lblNombre1
   lblNombre2.Text = Nombre2        ' Affiche Nombre2 dans le Label lblNombre2
   lblCommentaire.Text = "Entrez le résultat" & CRLF & "et cliquez sur OK."
   lblCommentaire.Color = Colors.RGB(255,235,128) ' couleur de fond jaune
   lblResultat.Text = ""            ' Vide edtResult.Text
End Sub
```

Et nous ajoutons les deux lignes dans la routine TestResultat avec `lblComments.Color =...`

```
Private Sub TestResultat
   If lblResultat.Text = Nombre1 + Nombre2 Then
      lblCommentaire.Text = "Résultat  J U S T E." & CRLF & "Cliquez sur Nouveau."
      lblCommentaire.Color = Colors.RGB(128,255,128)      ' couleur de fond vert clair
      btnAction.Text = "Nouveau"
   Else
      lblCommentaire.Text = "Résultat  F A U X." & CRLF & "Entrez un nouveau résultat" & CRLF &
"et cliquez sur O K."
      lblCommentaire.Color = Colors.RGB(255,128,128)      ' couleur de fond rouge clair
   End If
End Sub
```

Puis, nous donnons au programme un titre plus significatif en ajoutant
`Page1.Title = "Math Trainer"`
dans la routine Application_Start juste avant `NavControl.ShowPage(Page1)`.

```
Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   Page1.RootPanel.LoadLayout("Main")
   Page1.Title = "Math Trainer"
   NavControl.ShowPage(Page1)

   NouveauProbleme
End Sub
```

Autre amélioration, nous voulons cacher le bouton '0' pour éviter d'entrer des '0' comme premier caractère.
Pour ça, nous ajoutons cette ligne `btn0.Visible = False`. Dans la routine NouveauProjet.

```
Private Sub NouveauProbleme
   Nombre1 = Rnd(1, 10)                ' Génère un nombre aléatoire entre 1 et 9
   Nombre2 = Rnd(1, 10)                ' Génère un nombre aléatoire entre 1 et 9
   lblNombre1.Text = Nombre1           ' Affiche Nombre1 dans le Label lblNombre1
   lblNombre2.Text = Nombre2           ' Affiche Nombre2 dans le Label lblNombre2
   lblCommentaire.Text = "Entrez le résultat" & CRLF & "et cliquez sur OK."
   lblCommentaire.Color = Colors.RGB(255,235,128) ' couleur de fond jaune
   lblResultat.Text = ""               ' Vide edtResult.Text
   btn0.Visible = False
End Sub
```

Nous voyons que `btn0` est en rouge, ce qui signifie qu'il y a une erreur.
`btn0.Visible = False`
Regardons dans l'onglet Logs sur le haut de quelle erreur il s'agit.

Variable non déclarée, elle n'est donc pas reconnue par le système.

Pour que btn0 soit reconnu nous, le déclarons dans la routine Process_Globals.

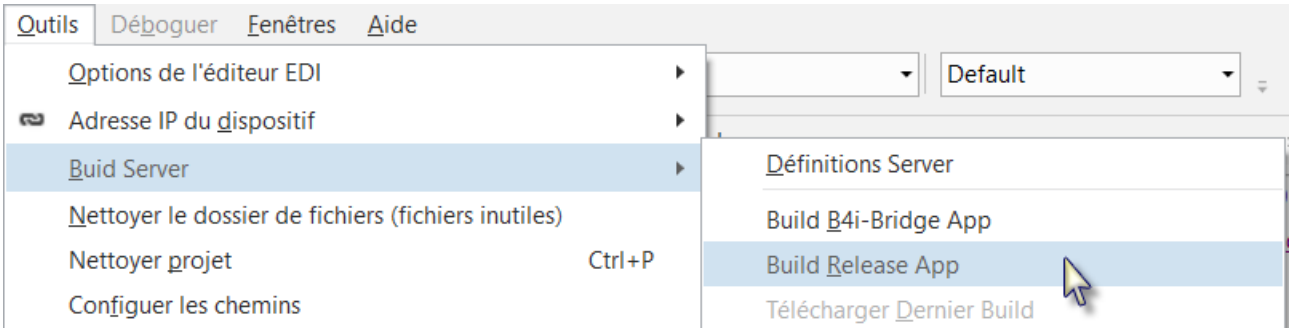
```
   Private btnAction, btn0 As Button
```

Maintenant, `btn0` n'est plus en rouge.
`btn0.Visible = False`

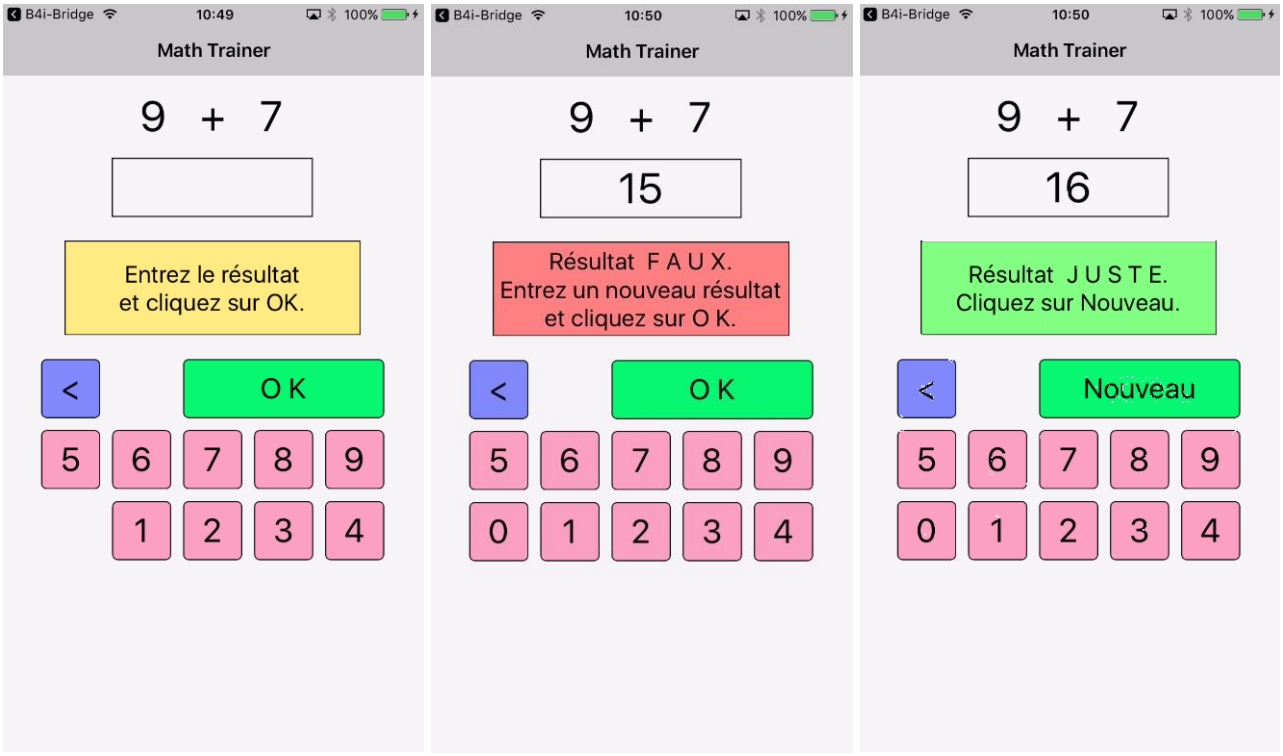
En plus, dans la routine btnEvent_Click, nous cachons le bouton `btn0` si le texte dans la propriété Text dans `lblResultat` est égale à zéro et affichons le bouton lorsque la longueur du texte est plus grande que zéro.

```
Private Sub btnEvent_Click
   Dim btnSender As Button

   btnSender = Sender

   Select btnSender.Tag
   Case "BS"
      If lblResult.Text.Length >0 Then
         lblResult.Text = lblResult.Text.SubString2(0,lblResult.Text.Length - 1)
      End If
   Case Else
      lblResult.Text = lblResult.Text & Send.Tag
   End Select

   If lblResult.Text.Length = 0 Then
      btn0.Visible = False
   Else
      btn0.Visible = True
   End If
End Sub
```

Dernier point, pour rendre le programme indépendant du Débogueur ou le diffuser sur AppStore vous devez le compiler en mode ‘Release’ :



Il ne reste plus qu’à tester le programme.



4 B4J Premiers pas

B4J, pour Basic for Java, est un outil de développement, **100% gratuit,** pour des applications bureau, serveur et des solutions IoT.

Les applications compilées peuvent être exécutés sur des plateformes Windows, Mac, Linux et ARM (tel que Raspberry Pi).

B4J est un langage similaire à B4A. B4i et Visual Basic.

B4J inclut un 'Concepteur visuel' puissant permettant de créer des interfaces utilisateur avec support pour des écrans et orientations multiples.

Vous avez besoin de :

- Le programme B4J, qui est un programme Windows tournant sur un PC.
- Le SDK Java, sur le PC, gratuit.

4.1 Installation de B4J

4.1.1 Installation de Java JDK

B4J est basé sur le composant SDK Java.

Si vous utilisez déjà B4A ou B4i vous pouvez sauter ce chapitre.

En premier, vous devez installer **Java JDK**.
Notez qu'il n'y a aucun problème d'avoir plusieurs versions de Java installées sur un même ordinateur.

- Ouvrez ce lien Java 8 JDK download link.
- Cochez la case Accept License Agreement.
- Sélectionnez **"Windows x86"** ou **"Windows x64"** (pour des machines 64 bit) dans la liste des plateformes.
- Téléchargez le fichier et installez-le.

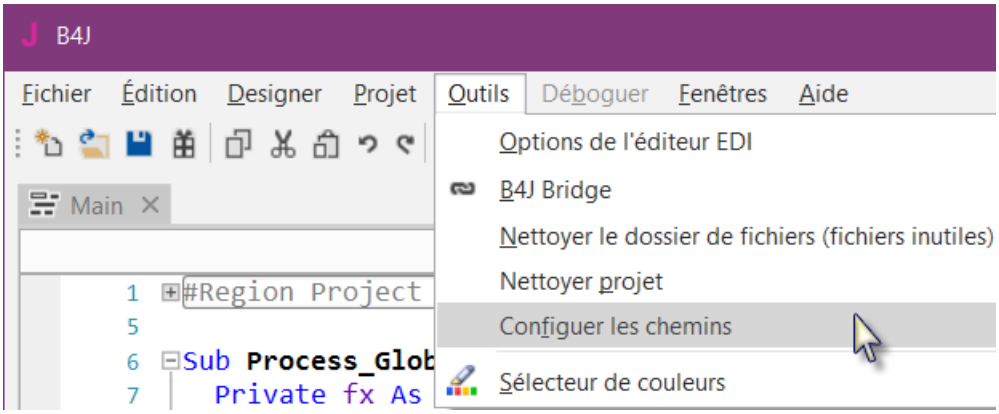
4.1.2 Installation de B4J

Téléchargez le fichier B4J et installez-le sur votre ordinateur.

Vu que B4J est gratuit, il n'y a donc pas de fichier de license comme pour B4A et B4i.

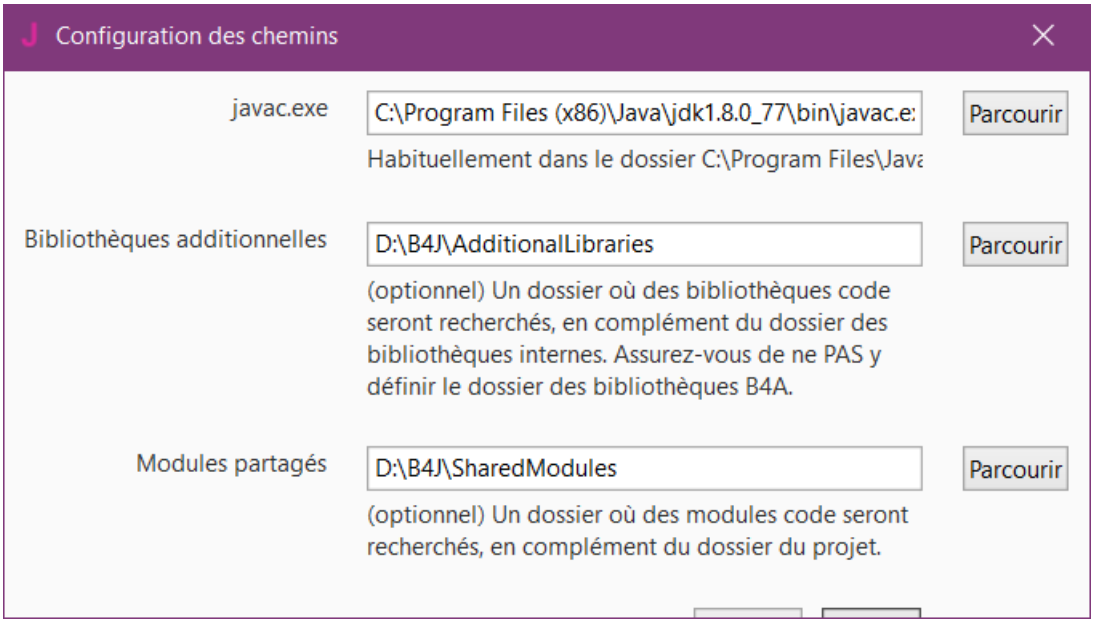
4.2 Configuration des dossiers dans l'EDI

Vous devez configurer différents dossiers dans l'EDI.



Exécutez l'EDI.

Dans le menu Outils cliquez sur Configuer les chemins.



javac.exe :
Entrez le chemin pour le fichier javac.exe file.

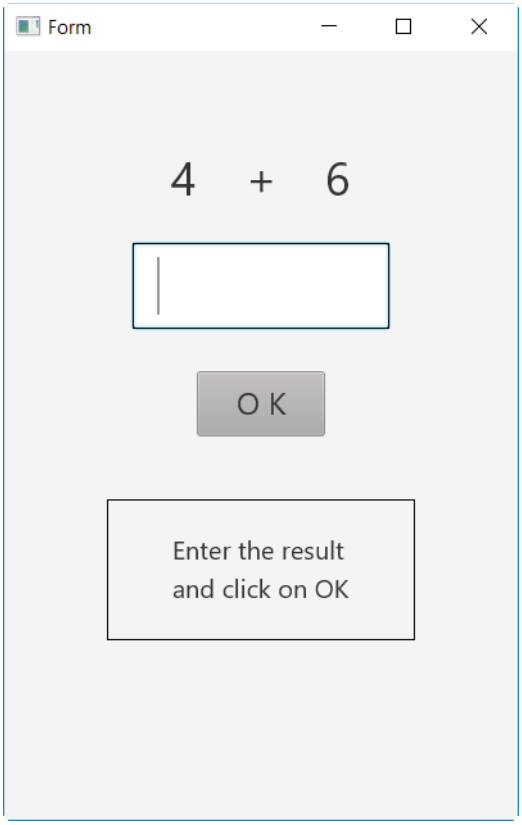
Bibliothèques Additionnelles :
Créez un dossier spécifique pour les bibliothèques additionnelles (Additional libraries), par exemple C:\B4J\AdditionalLibraries.

Modules partagés :
Créez un dossier spécifique pour les Modules partagés (Shared Modules), par exemple C:\B4J\SharedModules.

4.3 Mon premier programme B4J (MonPremierProgramme.b4j)

Nous allons écrire notre premier programme B4J. C'est un programme d'entraînement de calcul pour enfants.

Le projet est disponible dans le dossier des codes sources fourni avec le livret :
CodesSource\MonPremierProgramme\ B4J\ MonPremierProgramme.b4j



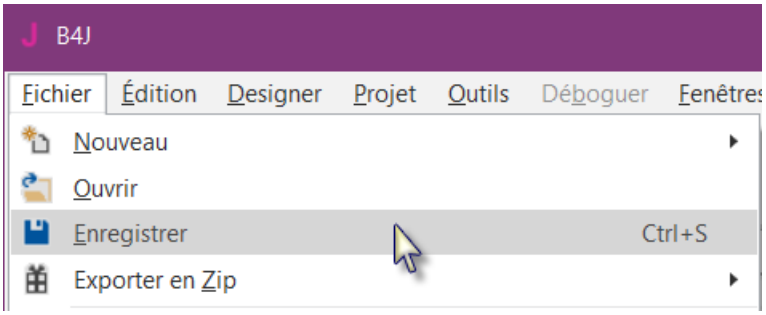
Dans la fenêtre nous aurons :
- 2 Labels affichant des nombres (de 1 à 9) générées aléatoirement.
- 1 Label avec le signe mathématique (+).
- 1 TextField dans lequel l'utilisateur devra entrer le résultat du calcul.
- 1 Button, utilisé soit pour confirmer le résultat soit pour générer un nouveau calcul.
- 1 Label avec des commentaires.

Dans B4J :
- Node est un objet d'interface homme – machine.
- Label est un objet (Node) pour afficher du texte.
- TextField est un objet (Node) permettant à l'utilisateur d'éditer du texte, similaire à EditText dans B4A.
- Button est un objet (Node) permettant des actions de l'utilisateur.

Nous allons définir le layout (mise en page) de l'interface utilisateur avec le Concepteur visuel du Designer et passerons pas à pas au travers de tout le processus.
Le Designer gère les différents objets de l'interface.
Le Concepteur visuel montre les positions et dimensions des différents objets et permet de les déplacer ou de les redimensionner sur l'écran.
Dans la fenêtre WYSIWYG nous voyons l'aspect réel.

Exécutez B4J 

Enregistrez le projet.

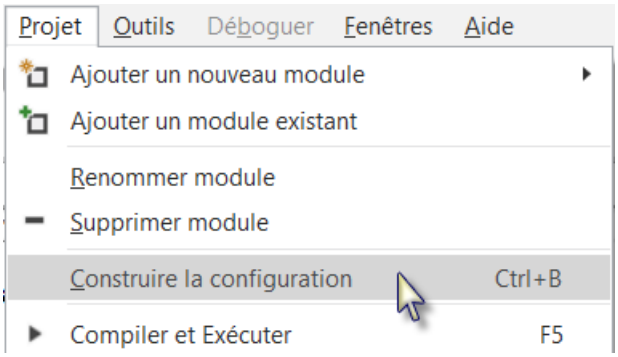


Vous devez enregistrer le projet avant de pouvoir utiliser le Designer.

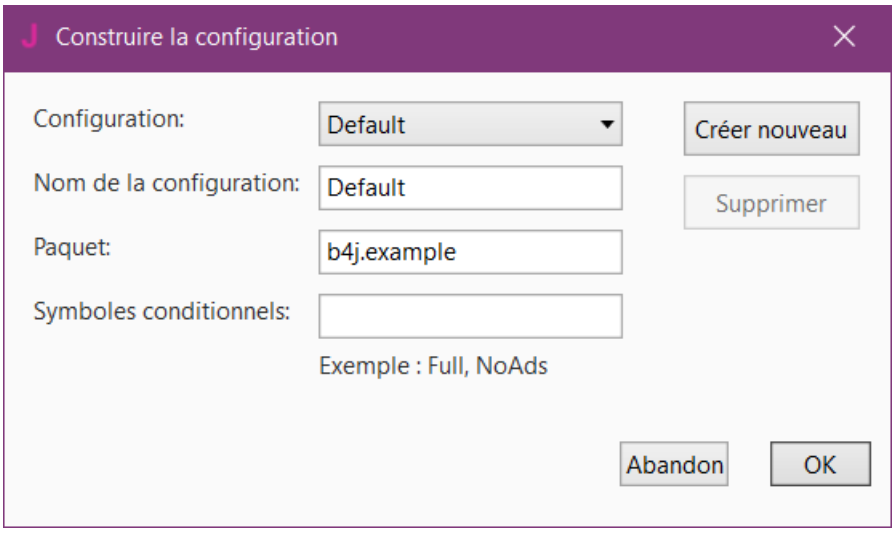
Créez un nouveau dossier MonPremierProgramme et enregistrez le projet avec le nom MonPremierProgramme.

Définissez le nom du paquet.
Chaque programme nécessite un nom de paquet.

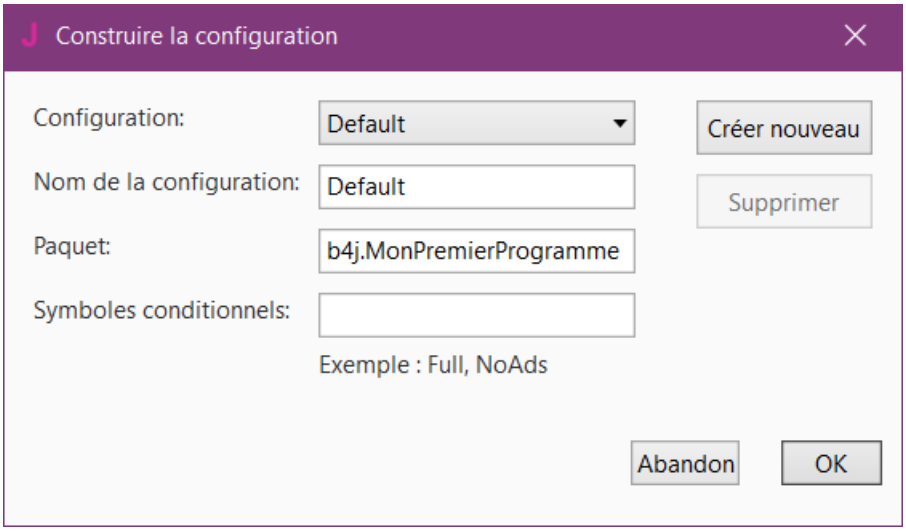
Dans le menu Projet cliquez sur Construire la configuration.



La fenêtre ci-dessous sera affichée :



Le nom par défaut est b4j.example. Nous le modifions en b4j.MonPremierProgramme.

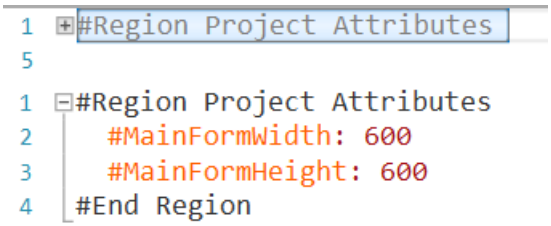


Définissez les dimensions de la Form.

Les dimensions de la Form (fenêtre) correspondent à la fenêtre principale affichée sur l'écran.

Sur le haut du code, vous voyez `Region Project Attributes`.

Regions sont des parties de code qui peuvent être réduits ou étendus.
Un clic sur ⊞ étend la Région.
Un clic sur ⊟ réduit la Région.
Les Regions sont expliquées dans le chapitre *Réduire une Région* dans le livret *B4x EDI*.



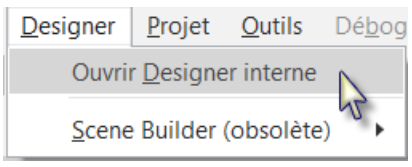
```
1  #Region Project Attributes
5
1  #Region Project Attributes
2     #MainFormWidth: 600
3     #MainFormHeight: 600
4  #End Region
```

Ici, nous pouvons définir les dimensions de la fenêtre principale appelée Form.
Les valeurs par défaut sont MainFormWidth (largeur) = 600 et MainFormHeight (hauteur) = 600.

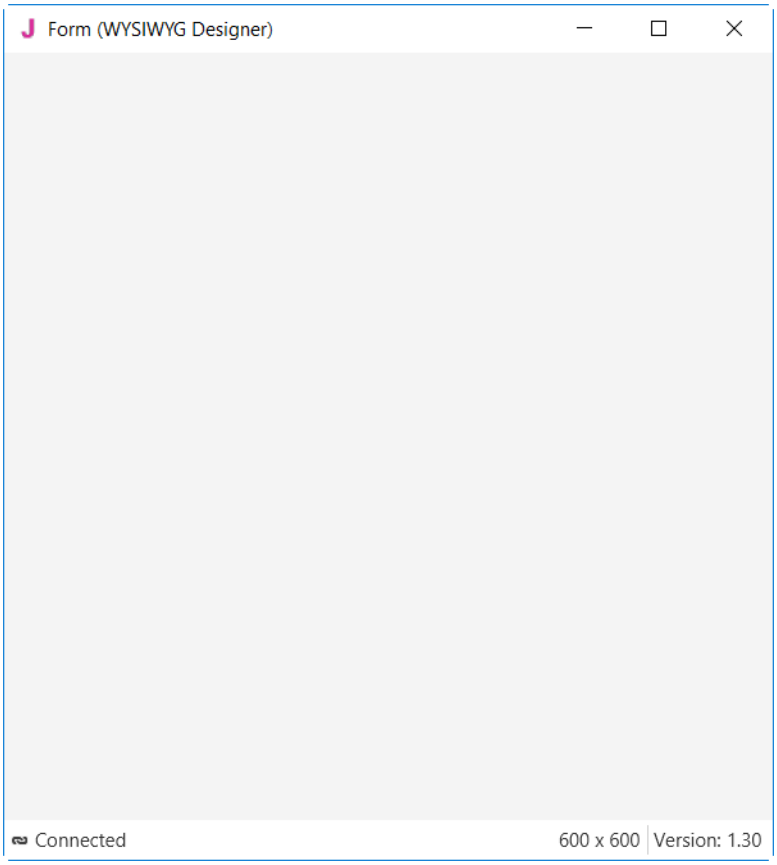
Nous modifions ces valeurs en MainFormWidth = 400 et MainFormHeight = 600.

```
#Region Project Attributes
   #MainFormWidth: 400
   #MainFormHeight: 600
#End Region
```

Dans l'EDI exécutez le Designer.

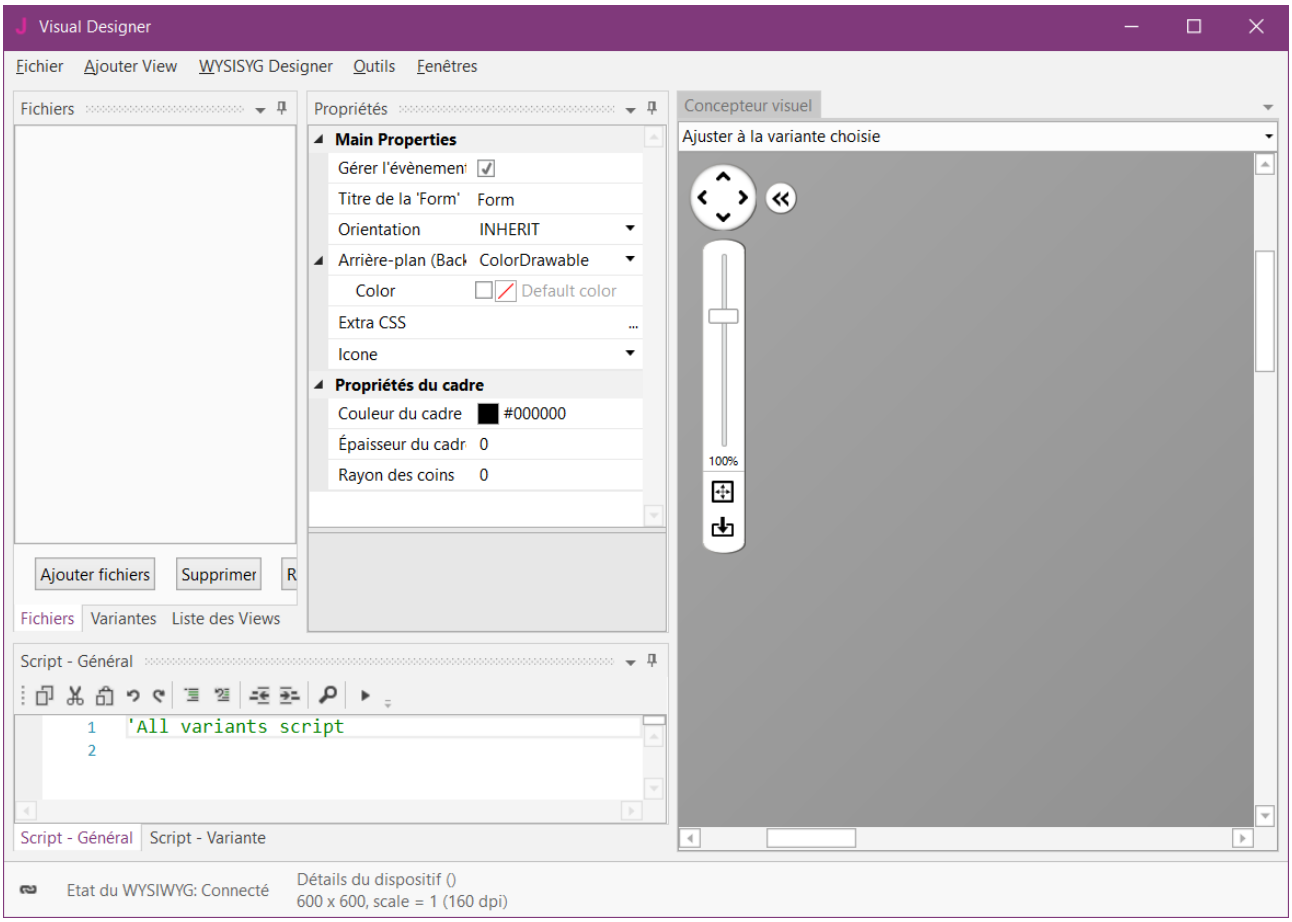


Attendez que le Designer soit prêt.



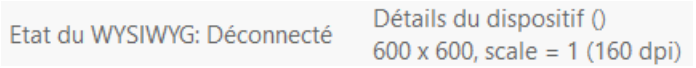
Nous obtenons une fenêtre WYSIWYG.

Et le Designer ressemble à l'image ci-dessous, les dimensions peuvent varier.

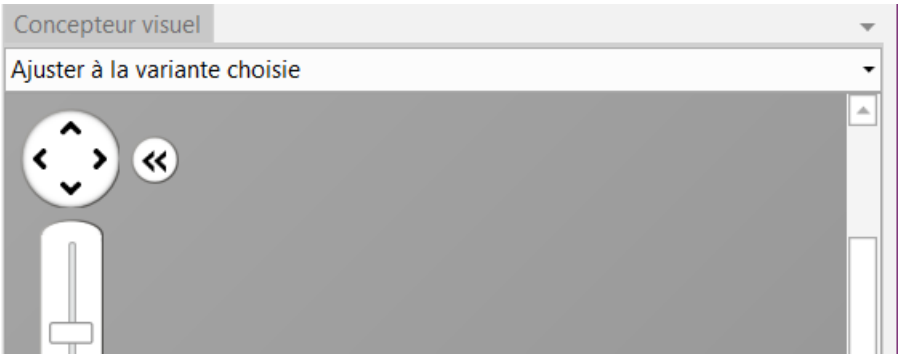


Notez que dans le coin inférieur droit du Designer vous voyez l'état de la connexion avec la fenêtre WYSIWYG :

Si vous fermez la Form WYSIWYG vous verrez cet état :



Dans le Designer, nous avons aussi le 'Concepteur visuel' qui affiche le layout pas exactement WYSIWYG mais les positions et dimensions des différents objets.
Seul le haut du 'Concepteur visuel' est montré.

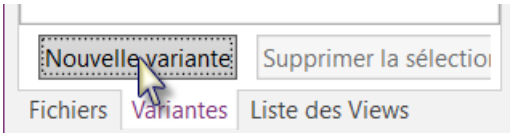


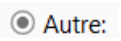
La partie en gris foncé correspond à la surface de la fenêtre connectée, qui est la Form WYSIWYG.

Les dimensions par défaut de la fenêtre sont 600 * 600, nous les modifions en 400 * 600, comme dans les Project Attributes.

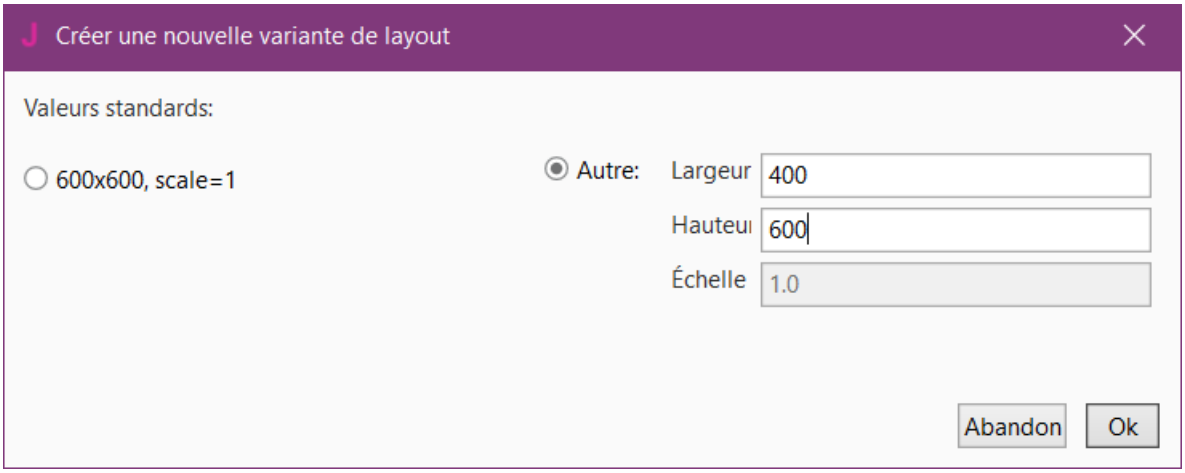
Cliquez sur 

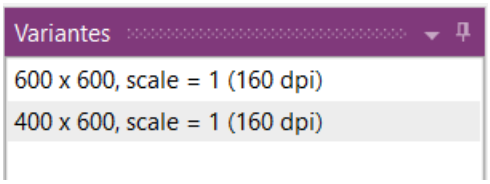
.



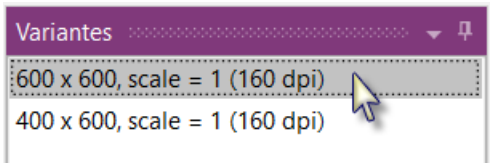
Dans la fenêtre ci-dessous cliquez sur 

et entrez les deux valeurs.



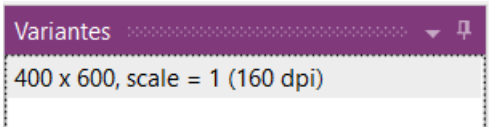


Dans la fenêtre Variantes la nouvelle variante est affichée.

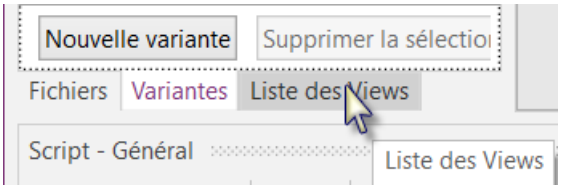


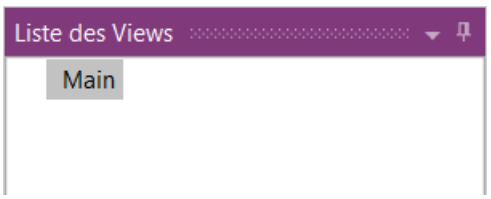
Cliquez sur 600 x 600, scale = 1 (160 dpi) pour sélectionner cette variante 600 * 600.

Cliquez sur Supprimer la sélectio pour supprimer la variante 600 * 600.

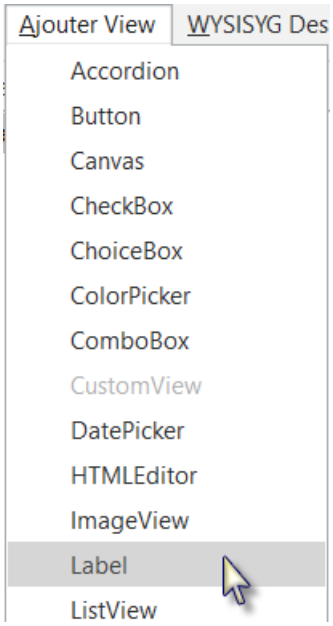


Cliquez sur Liste des Views pour afficher la fenêtre Liste des Views.

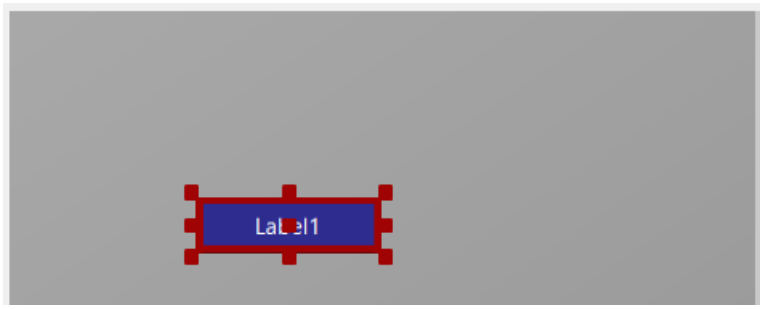




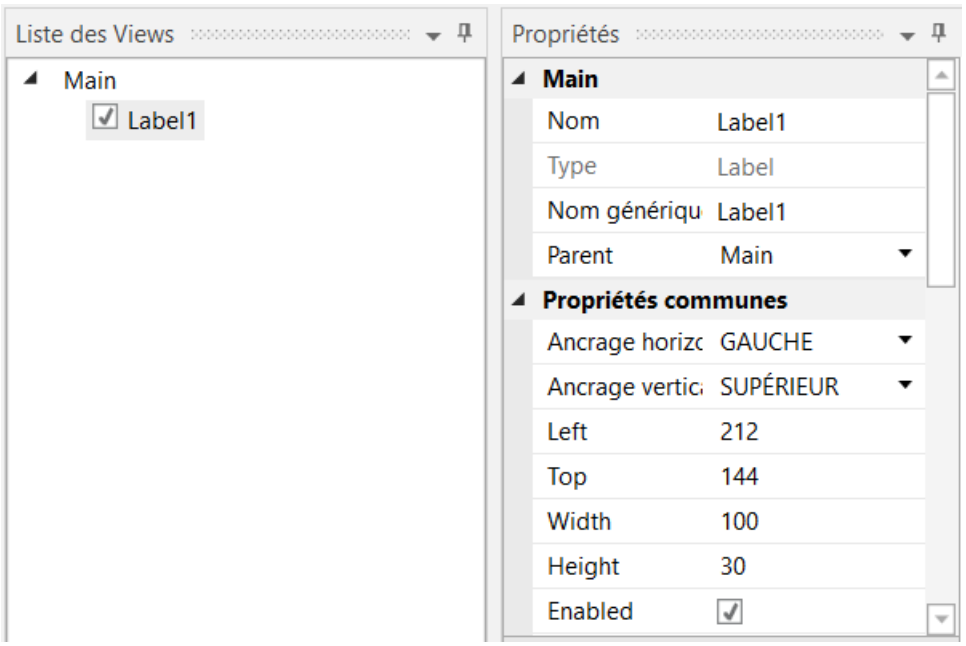
Nous allons ajouter 2 Labels pour afficher les nombres.
Dans le Designer, ajoutez un Label.

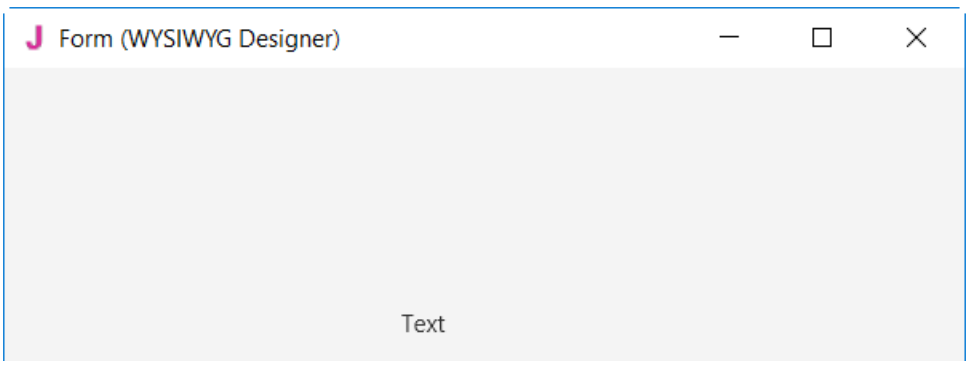


Dans le menu Ajouter View cliquez sur Label.



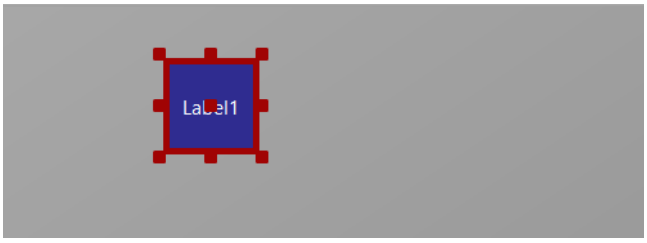
Le Label est affiché dans le Concepteur visuel, dans la fenêtre Liste des Views et ses propriétés par défaut sont listées dans la fenêtre Propriétés.

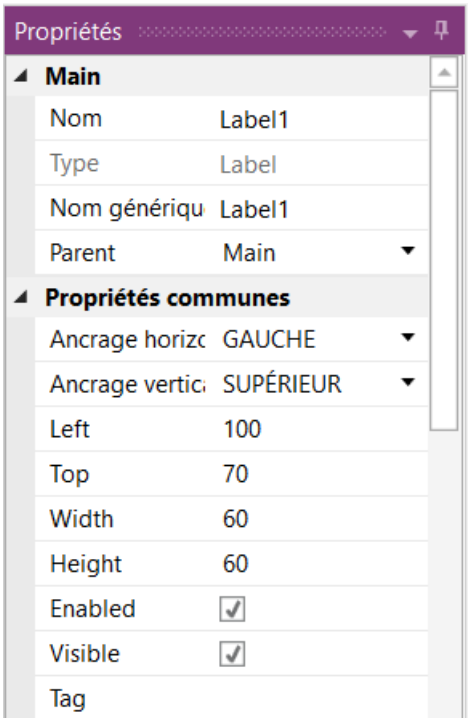




Et dans la Form WYSIWYG.

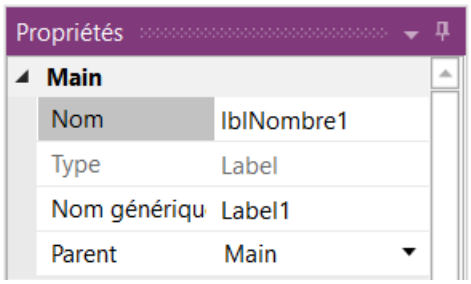
Redimensionnez et déplacez le Label au moyen des petits carrés rouges comme sur l'image.



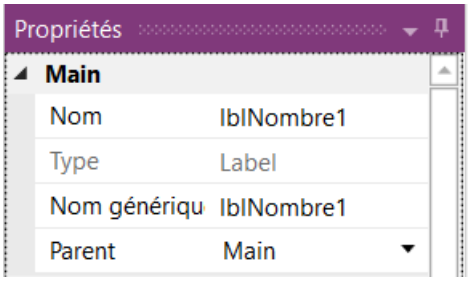


Les nouvelles valeurs des propriétés Left, Top, Width et Height sont automatiquement mises à jour dans la fenêtre Propriétés. Vous pouvez aussi modifier les propriétés directement dans la fenêtre Propriétés.

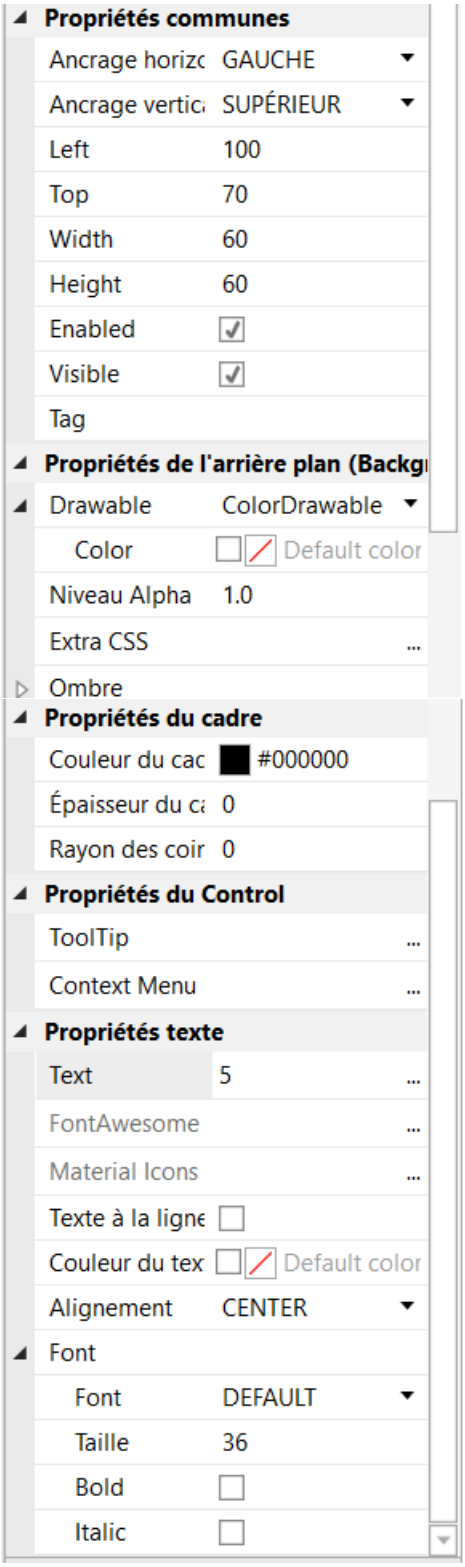
Nous modifions les propriétés du premier Label pour les adapter à nos besoins.
Par défaut son nom est, Label avec un nombre, ici Label1.
Nous modifions le nom en lblNombre1.
Les trois lettres au début 'lbl' signifient 'Label', et 'Nombre1' signifie premier nombre.
Il est recommandé d'utiliser des noms significatifs pour les objets, de cette manière nous savons directement de quel type d'objet il s'agit ainsi que sa fonction.



Presser la touche Entrée ou cliquer quelque part ailleurs dans le Concepteur visuel modifie aussi la propriété Nom générique des événements, qui est le nom générique des routines d'événements pour cet objet.



Main : Module Main.
Nom : Nom de l'objet (node).
Type : Type de l'objet (node). Dans notre cas, Label, qui n'est pas éditable.
Nom générique ...: Nom générique des routines qui gèrent les événements du Label.
Parent : Objet parent auquel appartient le Label.



Vérifions et modifions les autres propriétés ci-dessous :

Modifiez Left, Top, Width et Height avec les valeurs dans l'image.

Visible est coché.

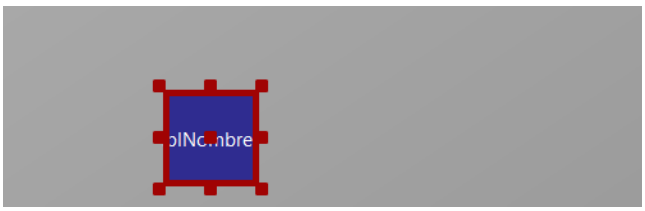
Nous gardons les couleurs par défaut.

Text en 5

Alignment en CENTER.

Nous gardons la valeur par défaut pour Font DEFAULT

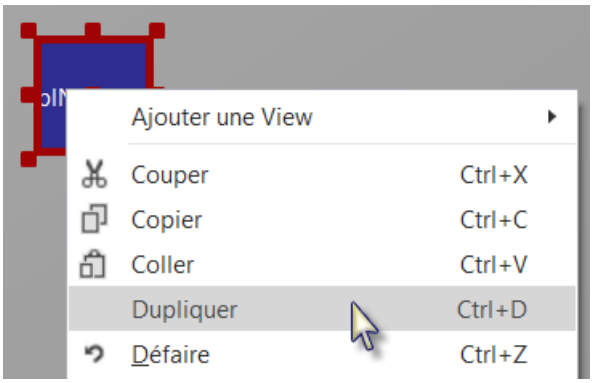
Taille en 36.



Et le résultat dans le Constructeur visuel

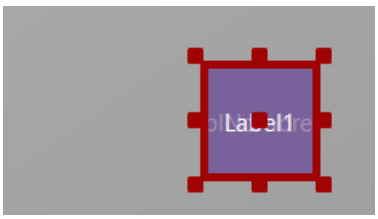
et dans la fenêtre WYSIWYG.

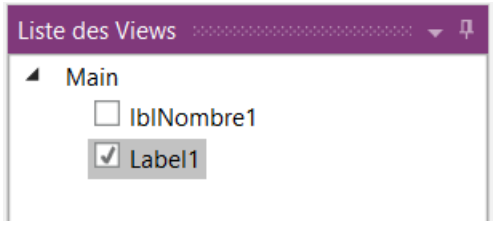
Nous avons besoin d'un deuxième Label, similaire au premier. Au lieu d'ajouter un nouveau, nous dupliquons le premier avec les mêmes propriétés. Seuls les propriétés Nom et Left seront modifiées.



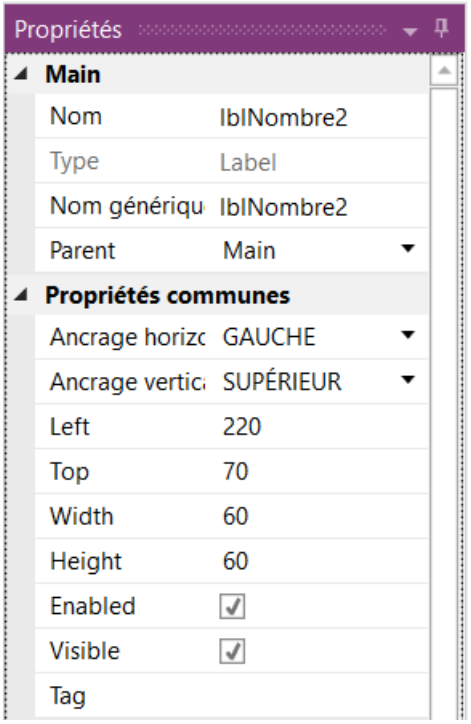
Cliquez avec le bouton droit sur lblNumber1 puis, cliquez sur Dupliquer dans le menu contextuel.

Le nouveau Label couvre le précédent.





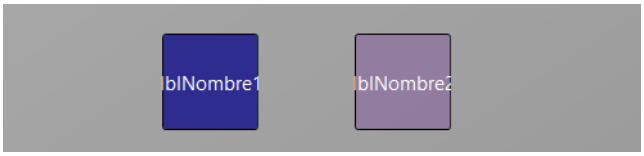
Dans la fenêtre Liste des Views à gauche, vous voyez les différents objets.
Un nouveau label Label1 a été ajouté.



Modifions la position du nouveau Label et son nom.

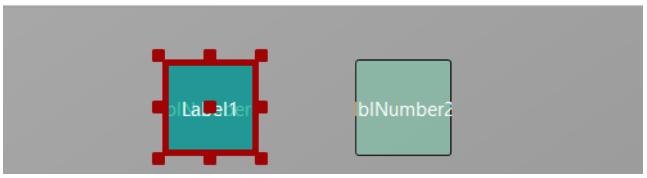
Modifions le nom en lblNombre2.

Et Left en 220.



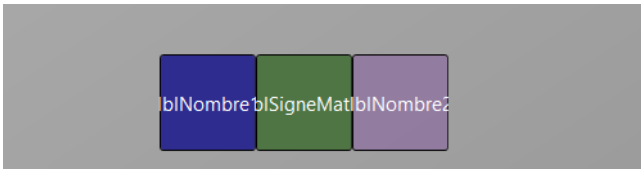
Et le résultat dans le Constructeur visuel et dans la fenêtre WYSIWYG.

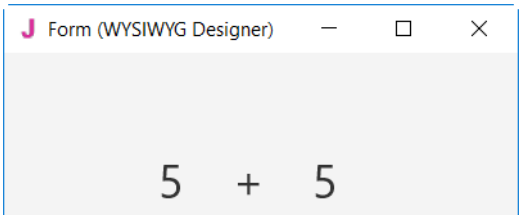
Ajoutons maintenant un troisième Label pour le signe mathématique. Nous dupliquons encore une fois lblNombre1.
Cliquez avec le bouton droit sur lblNumber1 puis, cliquez sur Dupliquer dans le menu contextuel.



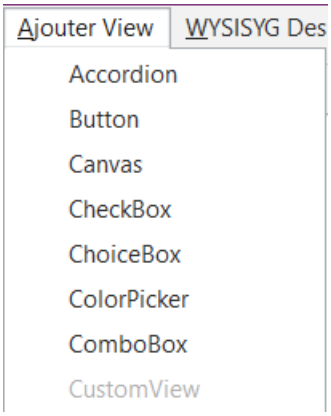
Le nouveau Label couvre lblNombre1.

Positionnez-le entre les deux premiers Labels, modifiez son nom en lblSigneMath et sa propriété Text en '+'.



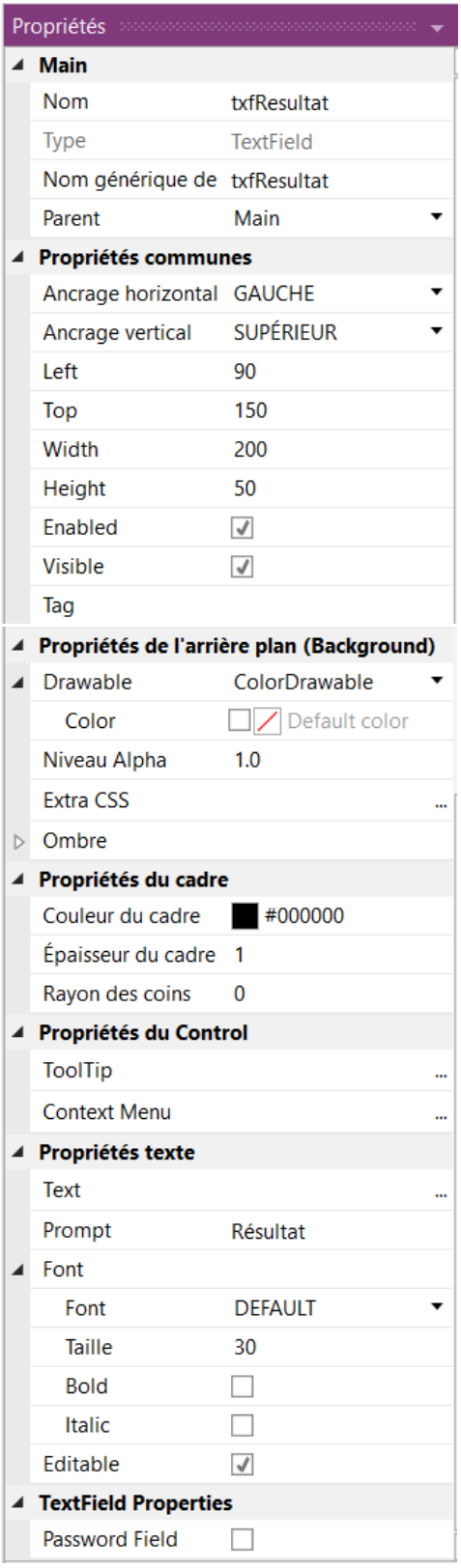


Et le résultat dans le Constructeur visuel et dans la fenêtre WYSIWYG.



Ajoutons un objet TextField.
Dans le menu Ajouter View cliquez sur TextField.

Positionnez-le au-dessous des trois Labels et modifiez son nom en txfResultat.
'txf' signifie TextField et 'Resultat' sa fonction.



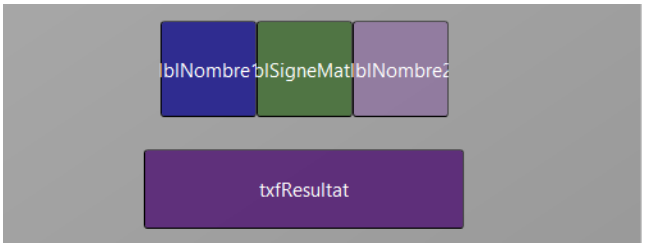
Modifiez ces propriétés.
Nom en txfResultat

Left, Top, Width et Height.

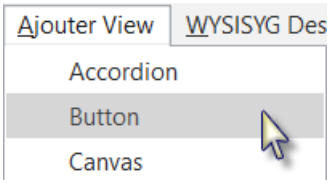
Épaisseur du cadre en 1

Prompt en Résultat
Prompt représente le texte affiché lorsqu'aucun texte n'est entré.

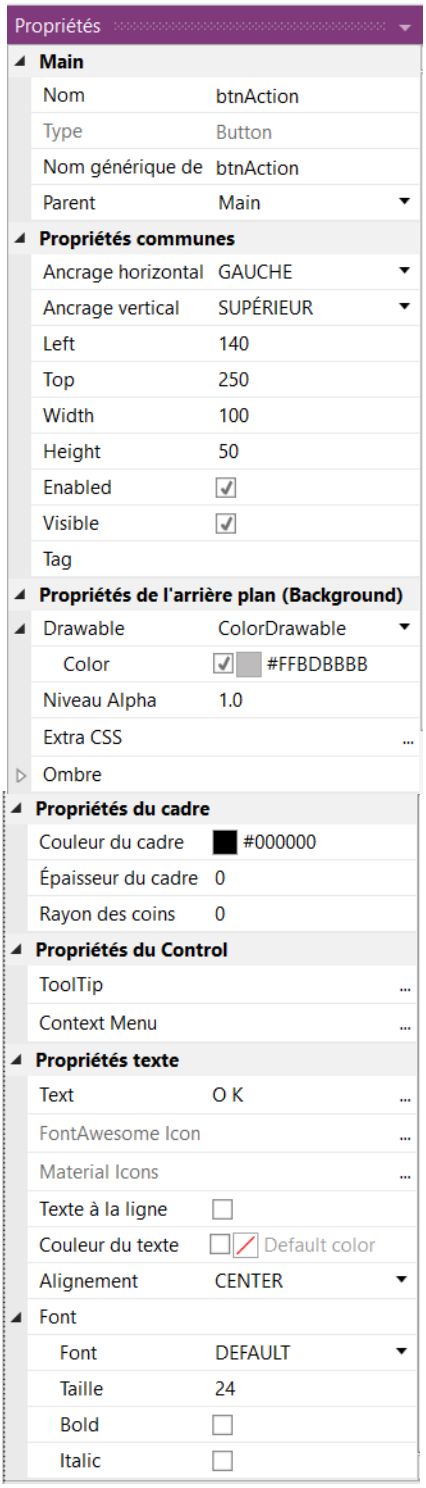
Taille en 30



Et le résultat dans le Constructeur visuel et dans la fenêtre WYSIWYG.



Maintenant, ajoutons un bouton (Button) qui, lorsque pressé, vérifie le résultat que l'utilisateur a entré, ou génère un nouveau problème en fonction des données entrées par l'utilisateur.



Positionnez le sous l'objet TextField. Redimensionnez-le et modifiez les propriétés ci-dessous :

Nom en btnAction

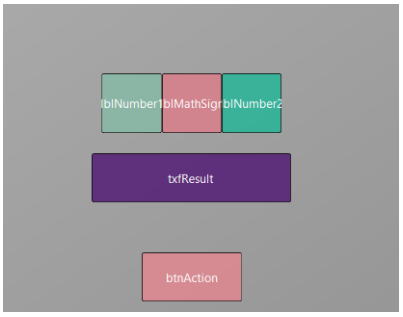
Left, Top, Width et Height.

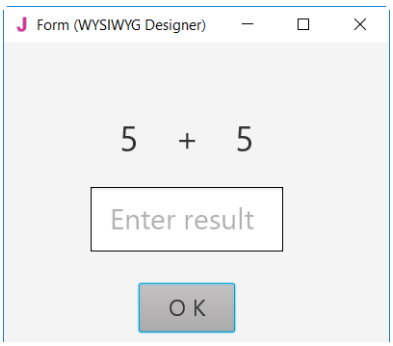
Color en #FFBDBBBB

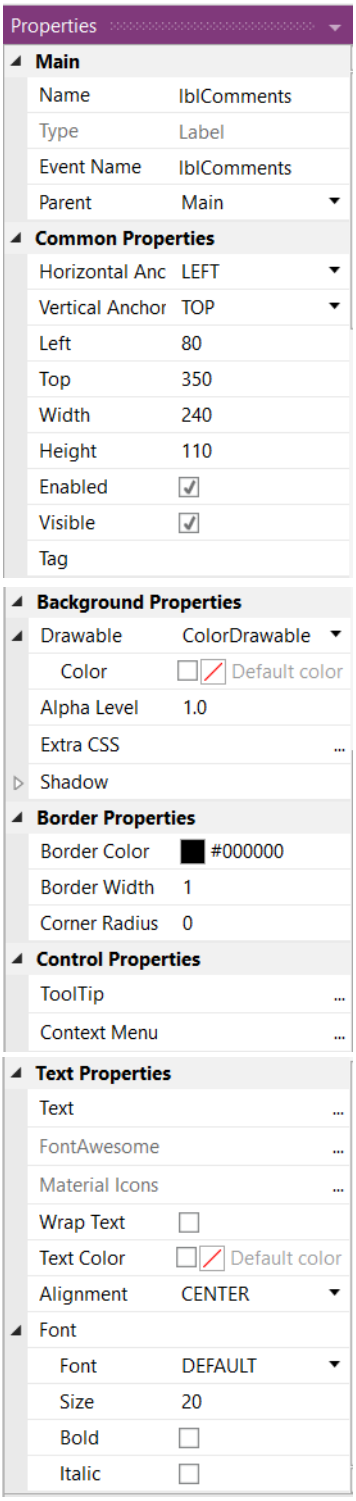
Text en O K (avec un espace entre O et K)

Taille en 24

Resultat (images réduites)







Ajoutons un dernier Label pour les commentaires. Positionnez le au-dessous du bouton et redimensionnez-le.

Modifiez les propriétés ci-dessous :

Nom en lblCommentaire

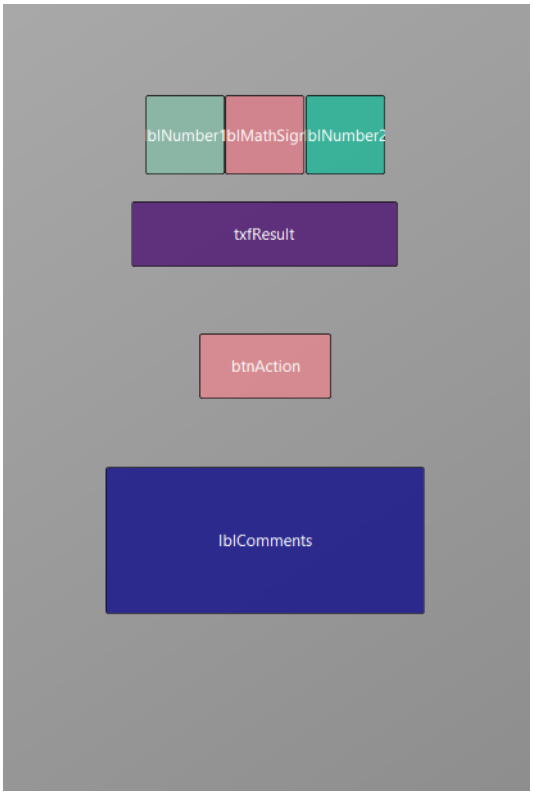
Left, Top, Width et Height.

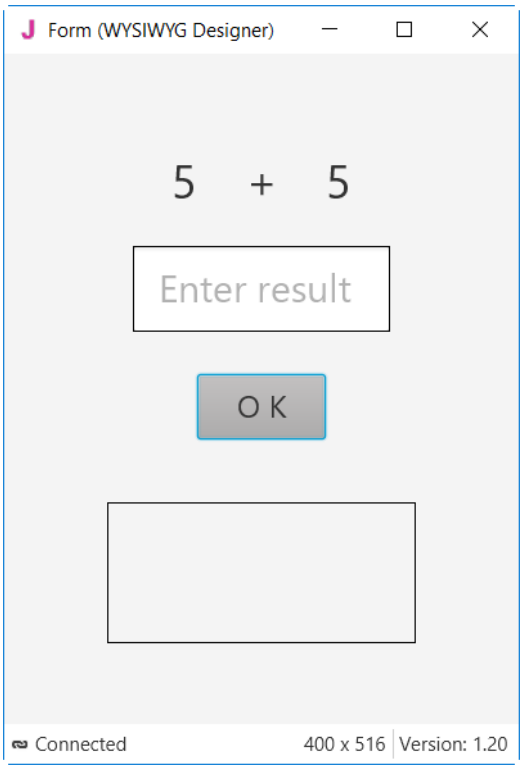
Épaisseur du cadre en 1

Text vide

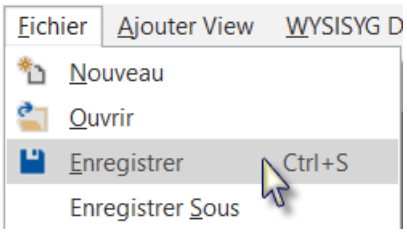
Taille en 20

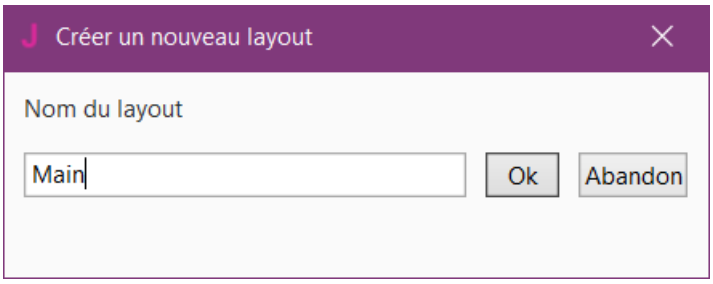
Et le résultat.





Il est temps d'enregistrons le fichier layout.

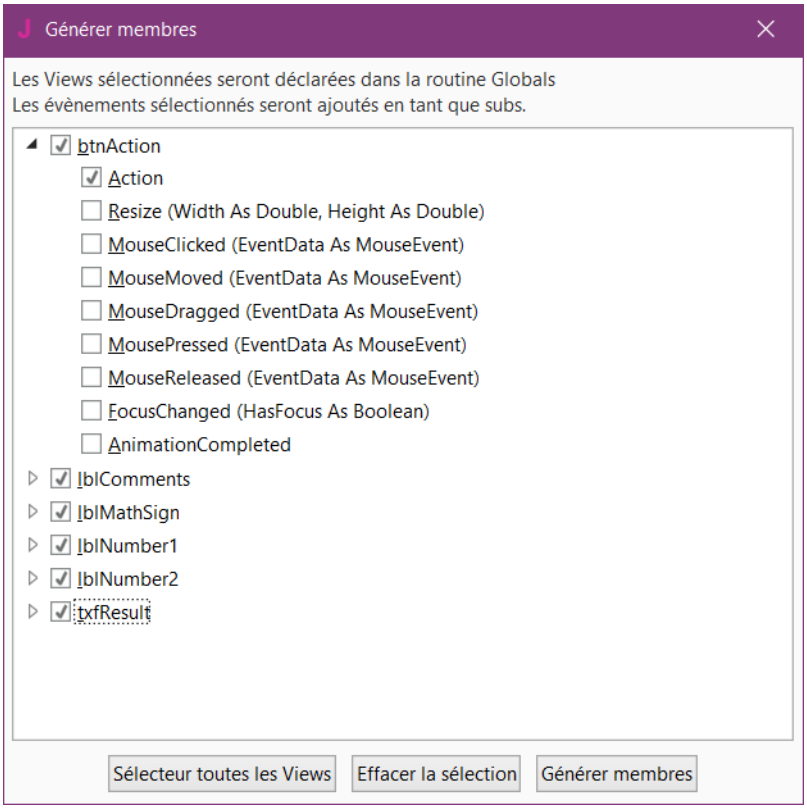
Cliquez sur  et enregistrez-le avec le nom 'Main'.



Cliquez sur  .

Pour écrire les routines pour le projet, nous devons référencer les objets dans le code.
Nous le faisons avec l'outil *Générer membres* du Designer.

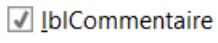
L'outil *Générer membres* génère automatiquement les références des objets ainsi que des cadres pour les routines d'événement.



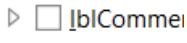
Cliquez sur

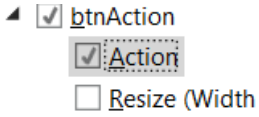
pour ouvrir le générateur.

Dans cette fenêtre nous trouvons tous les objets ajoutés dans le layout.
Nous cochons tous les objets ainsi que l'événement Action pour btnAction.

Le fait de cocher un objet  génère sa référence dans la routine Process_Globals. Ces références sont nécessaires pour que les objets soient reconnus par le compilateur et aussi pour la fonction d'autocomplétion.

```
Private btnAction As Button
Private lblCommentaire As Label
Private lblNombre1 As Label
Private lblNombre2 As Label
Private lblSigneMath As Label
Private txfResultat As TextField
```

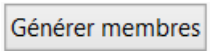
Un clic sur  affiche tous les événements pour l'objet sélectionné



.

Le fait de cocher un événement d'un objet  génère le cadre de la routine événement.

```
Sub btnAction_Action

End Sub
```

Cliquez sur  pour générer les références at les cadres des routines d'événement, puis fermez le fenêtre  .

Nous retournons maintenant dans l'EDI pour écrire le code.

Sur le haut du programme nous avons :

```
Sub Process_Globals
   Private fx As JFX
   Private MainForm As Form
   Private btnAction As Button
   Private lblCommentaire As Label
   Private lblNombre1 As Label
   Private lblNombre2 As Label
   Private lblSigneMath As Label
   Private txfResultat As TextField
End Sub
```

Les deux lignes ci-dessous sont nécessaire et figurent systématiquement dans le code d'un projet.

```
   Private fx As JFX
   Private MainForm As Form
```

B4J nécessite un objet MainForm, ainsi que la bibliothèque JFX pour les objets (nodes), détails dans le chapitre *Flux du programme / cycle de vie* dans livret B4x Langage Basic.

En dessous nous trouvons la routine `AppStart` qui est la première routine à être exécutée au démarrage de programme.
Le contenu ci-dessous est aussi systématiquement ajouté dans chaque projet.

```
Sub AppStart (Form1 As Form, Args() As String)
   MainForm = Form1
   'MainForm.RootPane.LoadLayout("Layout1") 'Load the layout file.
   MainForm.Show
End Sub
```

```
MainForm = Form1                    > Attribue Form1 à la variable MainForm.
'MainForm.RootPane.LoadLayout("Layout1")    > Charge un fichier layout si besoin.
MainForm.Show                       > Affiche MainForm
```

En premier, nous devons charger le fichier layout que nous avons défini dans les pages précédentes.
Le fichier doit être chargé dans MainForm.RootPane, pour ça, nous décommentons cette ligne
`'MainForm.RootPane.LoadLayout("Layout1")`
et modifions le nom du fichier.

```
Sub AppStart (Form1 As Form, Args() As String)
   MainForm = Form1
   MainForm.RootPane.LoadLayout("Main") 'Load the layout file.
   MainForm.Show
End Sub
```

Nous voulons générer un nouveau problème dès le démarrage du programme. Pour ça, nous ajoutons un appel à la routine NouveauProbleme dans la routine AppStart.

```
Sub AppStart (Form1 As Form, Args() As String)
   MainForm = Form1
   MainForm.RootPane.LoadLayout("Main") 'Load the layout file.
   MainForm.Show

   NouveauProbleme
End Sub
```

NouveauProbleme est affiché en rouge car la routine 'NouveauProbleme' n'existe pas encore.
Générer un nouveau problème consiste à générer aléatoirement deux nombres de 1 à 9 (inclusif) pour les variables Nombre1 et Nombre2 et afficher ces valeurs dans les propriétés Text des deux Labels lblNombre1 et lblNombre2.

Pour cela, nous écrivons le code suivant :
Dans la routine Process_Globals nous ajoutons les références pour les variables des deux nombres.

```
   Private Nombre1, Nombre2 As Int
End Sub
```

Et la routine 'NouveauProbleme' :

```
Private Sub NouveauProbleme
   Nombre1 = Rnd(1, 10)          ' Génère un nombre aléatoire entre 1 et 9
   Nombre2 = Rnd(1, 10)          ' Génère un nombre aléatoire entre 1 et 9
   lblNombre1.Text = Nombre1     ' Affiche Nombre1 dans le Label lblNombre1
   lblNombre2.Text = Nombre2     ' Affiche Nombre2 dans le Label lblNombre2
   lblCommentaire.Text = "Entrez le résultat" & CRLF & "et cliquez sur O K"
   txfResultat.Text = ""         ' Vides la propriété Text de edtResult
End Sub
```

La fonction ci-dessous génère un nombre entier aléatoire de '1' (inclusif) à '10' (exclusif) :
`Rnd(1, 10)`

Dans cette ligne `Nombre1 = Rnd(1, 10)     ' Génère un nombre aléatoire entre 1 et 9`
Le texte après l'apostrophe, `' Génère...`, est considéré comme un commentaire.
Il est recommandé d'ajouter des commentaires expliquant le but du code.

La ligne ci-dessous affiche le commentaire dans le Label `lblCommentaire`, en attribuant le texte à la propriété Text de lblCommentaire :
`lblCommentaire.Text = " Entrez le résultat" & CRLF & " et cliquez sur O K"`
`CRLF` est le caractère NouvelleLigne.

Maintenant, ajoutons le code pour l'événement Click du bouton `btnAction`.

Nous avons deux cas :
- Lorsque le texte du bouton est égal à "O K", ça signifie qu'un nouveau problème est affiché, et que le programme attend une réponse de l'utilisateur et qu'il presse le bouton.
- Lorsque le texte du bouton est égal à "Nouveau", ça signifie que l'utilisateur a donné une réponse juste et lorsqu'il presse le bouton un nouveau problème doit être généré.

```
Private Sub btnAction_Action
   If btnAction.Text = "O K" Then
      If txfResultat.Text="" Then
        lblCommentaire.Text = "Il n'y a pas de résultat" & CRLF & "Entrez un résultat" & CRLF &
"et cliquez sur O K"
      Else
        TestResultat
      End If
   Else
      NouveauProbleme
      btnAction.Text = "O K"
   End If
End Sub
```

`If btnAction.Text = "O K" Then` vérifie si le texte du bouton est égal à "O K".
Si oui, nous vérifions si la propriété Text du TextField `txfResultat` est vide.
    Si oui, nous affichons un message indiquant à l'utilisateur qu'il n'a pas entré de résultat.
    Si non, nous vérifions si le résultat est juste ou faux.
Si non, nous générons un nouveau problème, modifions le texte du bouton à "O K" et vidons la propriété Text de l'objet TextField.

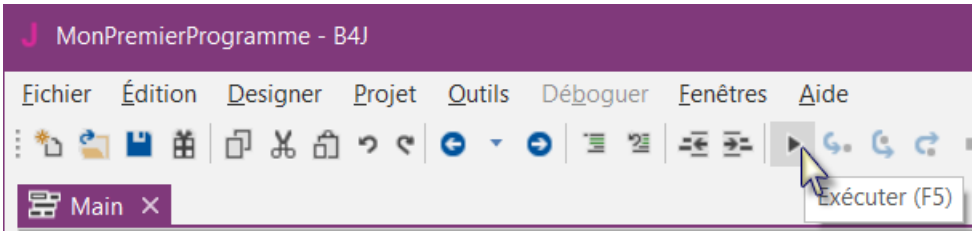
La dernière routine vérifie si le résultat est juste ou faux.

```
Sub TestResultat
   If txfResultat.Text = Nombre1 + Nombre2 Then
        lblCommentaire.Text = "Résultat  J U S T E" & CRLF & "Cliquez sur Nouveau"
        btnAction.Text = "Nouveau"
   Else
        lblCommentaire.Text = "Résultat  F A U X" & CRLF & "Entrez un nouveau résultat" & CRLF &
"et cliquez sur O K"
   End If
End Sub
```

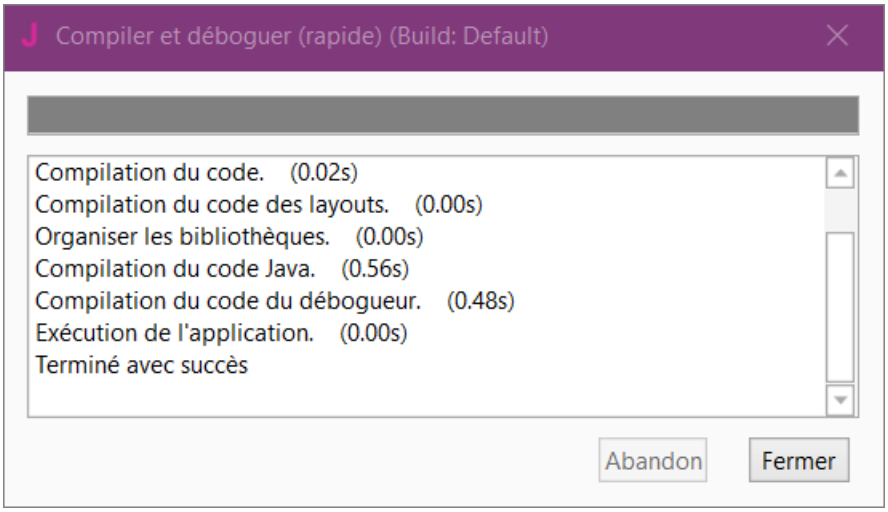
Avec `If txfResultat.Text = Nombre1 + Nombre2 Then` nous vérifions si le résultat fourni par l'utilisateur est juste.

Si oui, nous affichons dans le Label lblCommentaire le texte ci-dessous :
    'Résultat J U S T E'
    'Cliquez sur Nouveau'
    Et nous modifions le texte du bouton en "Nouveau".
Si non, nous affichons dans le Label lblCommentaire le texte ci-dessous :
    'Résultat F A U X'
    'Entrez un nouveau résultat'
    'et cliquez sur O K'

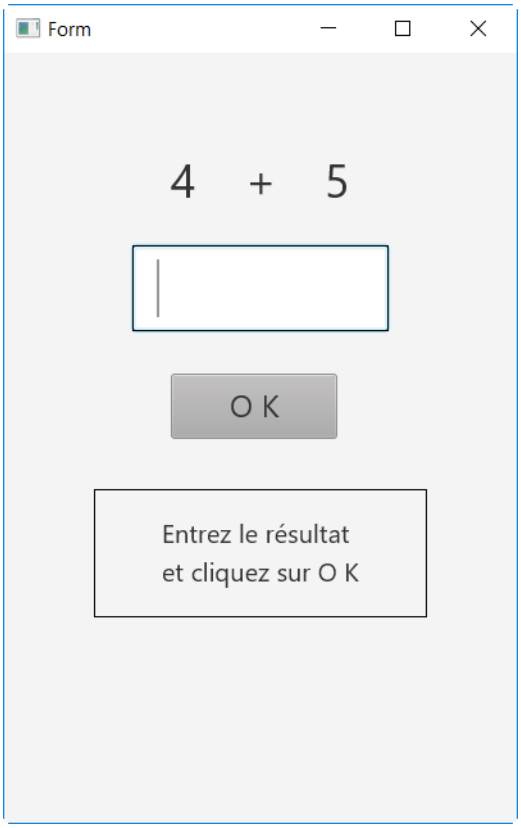
Compilons et exécutons le programme.
Dans la barre d'icônes cliquez sur  :



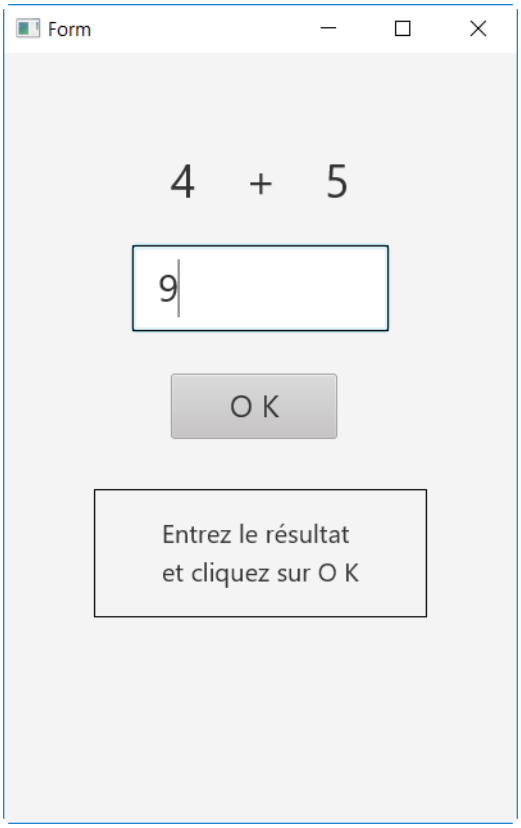
Le programme va être compilé.



Lorsque vous voyez `'Terminé avec succès'` dans la fenêtre de compilation, la compilation et le transfert sont terminés et le programme sera exécuté.

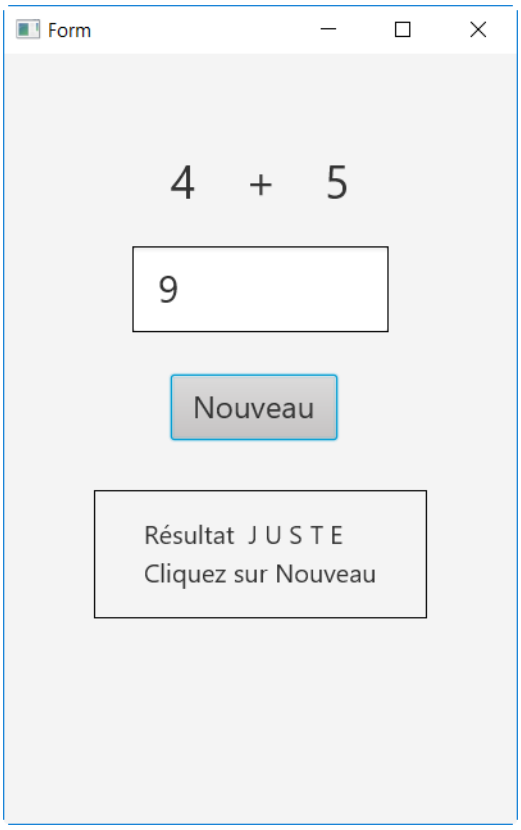


Vous verrez une fenêtre similaire à celle à gauche, avec des nombres différents.



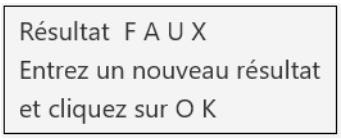
Entrez 9.

Cliquez sur  pour confirmer le résultat.



Si le résultat est juste, vous verrez un écran similaire à celui à gauche.

Si le résultat est faux, le message ci-dessous sera affiché :



Cliquez sur  pour générer un nouveau problème.

Bien sûr, l'esthétique n'est pas la meilleure, mais ça n'était pas le but du premier programme.
Des améliorations font l'objet du second programme.

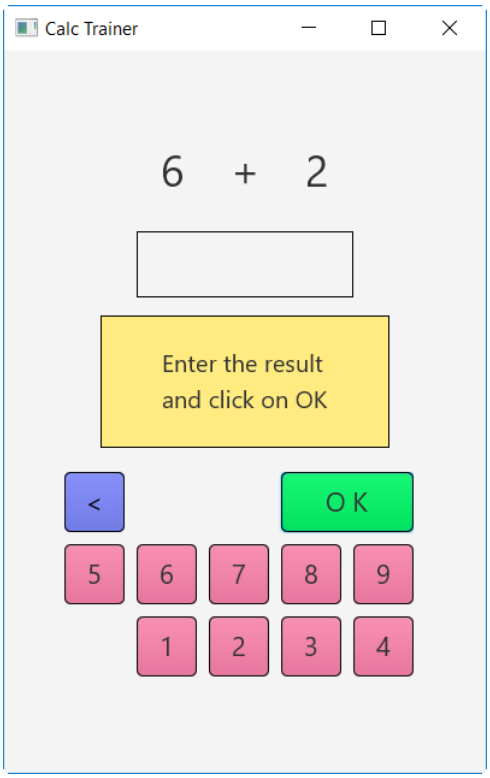
4.4 Second programme B4J (SecondProgramme.b4j)

Le projet est disponible dans le dossier CodesSource :
CodesSource\SecondProgramme\B4J\SecondProgramme.b4j.

Améliorations par rapport à “Mon premier programme”.

- Clavier numérique intégré pour éviter l’utilisation du clavier du PC.
- Couleurs dans les commentaires.

Créez un nouveau dossier avec le nom “SecondProgramme”. Copiez tous les fichiers et sous-dossiers de MonPremierProgramme vers SecondProgramme et renommez le fichier MonPremierProgramme.b4j en SecondProgramme.b4j et MonPremierProgramme.b4j.meta en SecondProgramme.b4j.meta.

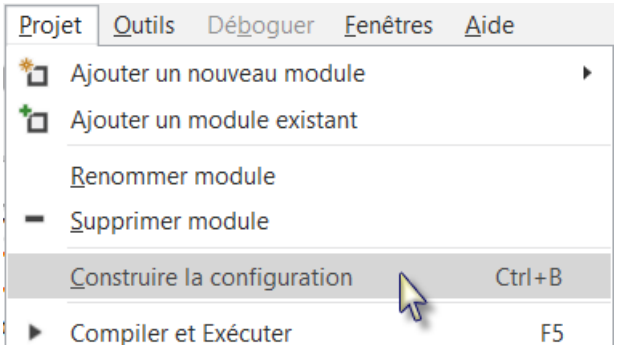


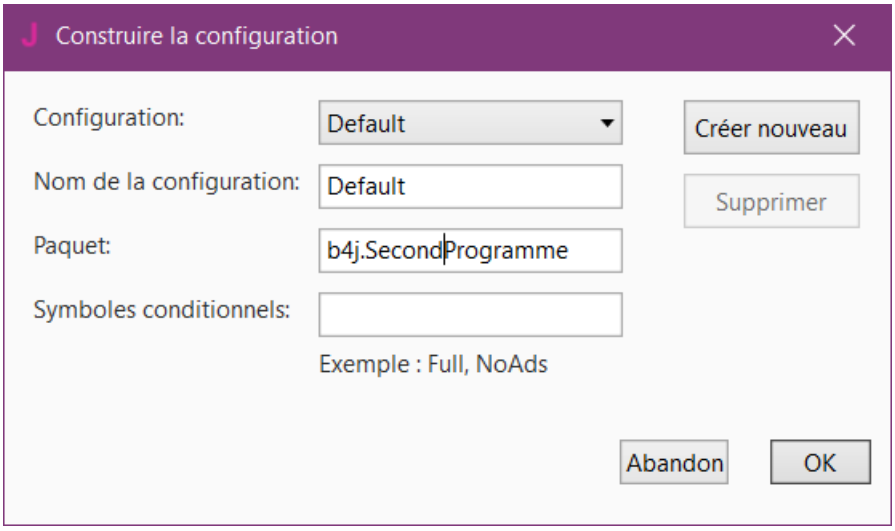
Chargez le nouveau programme dans l'EDI.

Exécutez le Designer.

Nous devons changer le nom du Paquet.

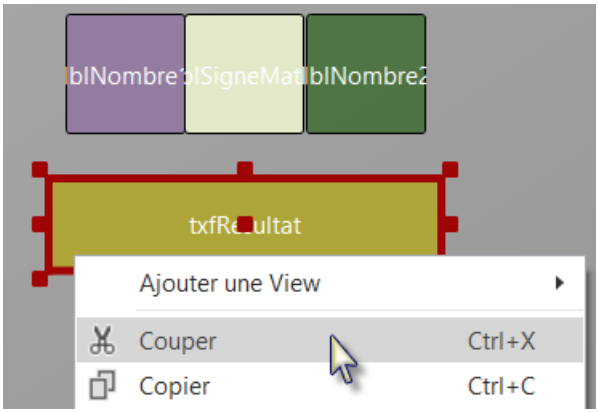
Dans l'EDI, menu Projet.
Cliquez sur Construire la configuration.



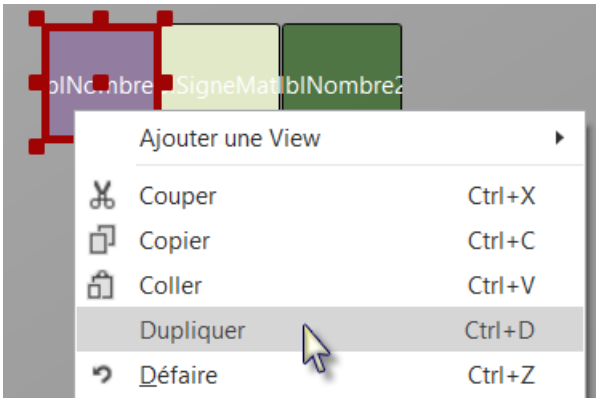


Modifiez le nom du Paquet en b4j.SecondProgramme et cliquez sur OK.

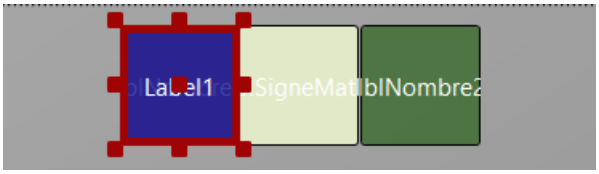
Nous allons remplacer l'objet TextField txfResultat par un nouveau Label.
Dans le concepteur visuel, cliquez sur l'objet txfResultat.



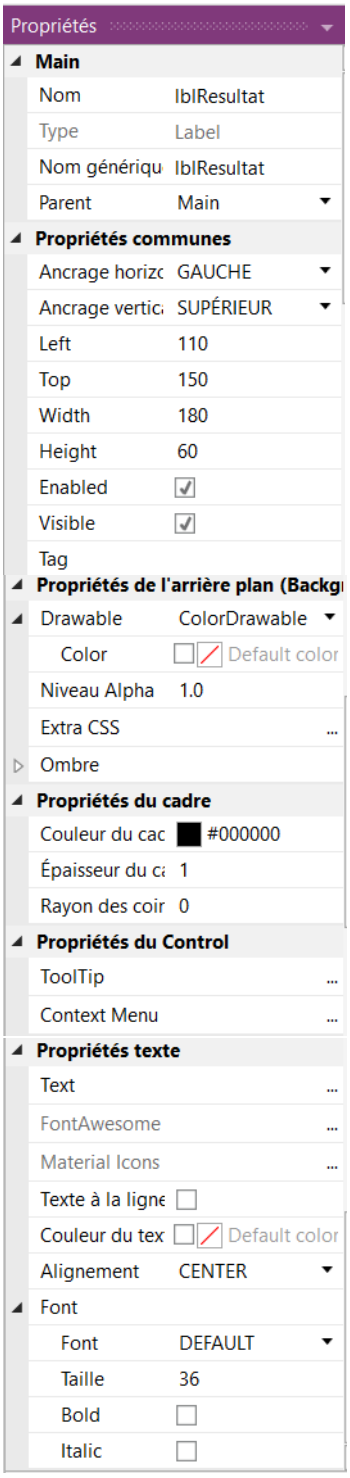
Cliquez avec le bouton droit sur txfResultat et cliquez sur Couper.



Cliquez avec le bouton droit sur lblNombre1et cliquez sur Dupliquer.



Le nouveau label couvre lblNombre1.



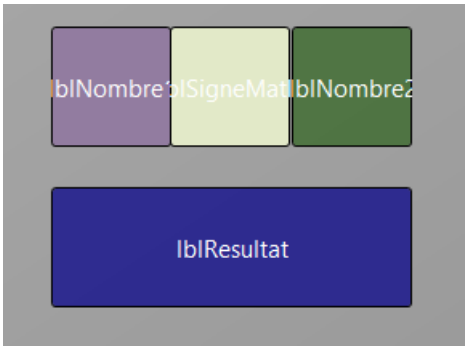
Modifiez les propriétés suivantes :

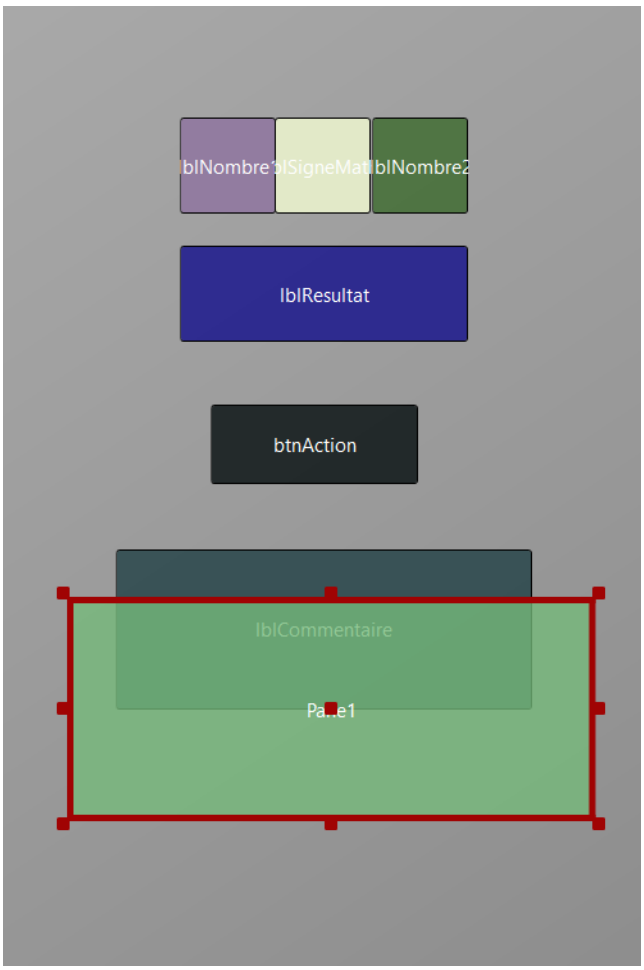
Nom en lblResultat

Left, Top, Width, Height

Épaisseur du cadre en 1

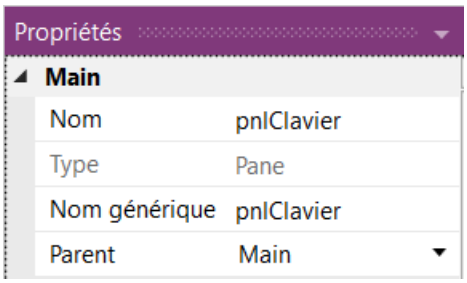
Text en "" pas de texte



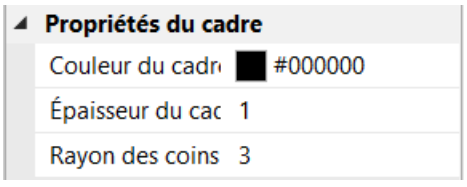


Ajoutez on objet Pane pour les boutons du clavier.
Dans B4J l'objet Pane est similaire à l'objet Panel dans B4A et B4i.

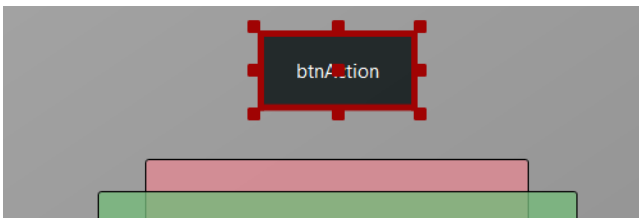
Positionnez et redimensionnez-le comme dans l'image.



Modifiez son Nom en pnlClavier
"pnl" pour Pane (habitude de B4A pour Panel), le type de l'objet.

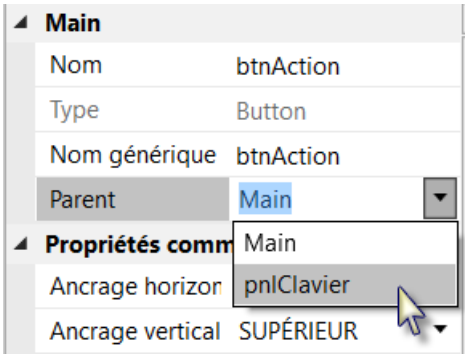


Modifiez
Rayon des coins en 0

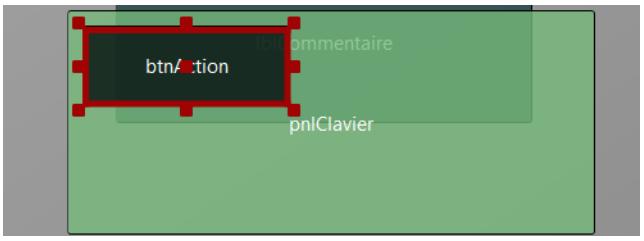


Déplacez le bouton btnAction de Main sur l'objet pnlClavier.

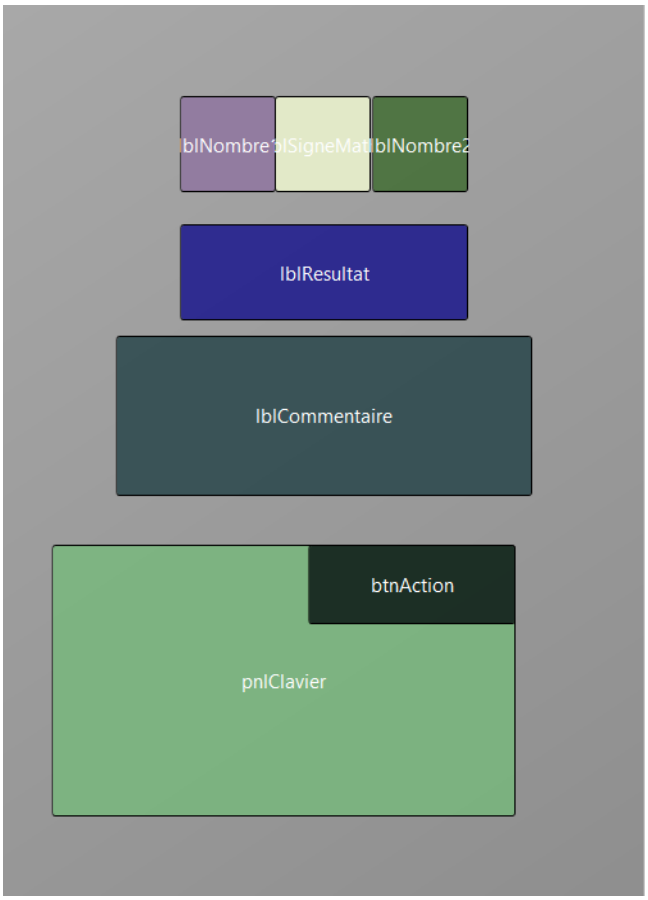
Cliquez sur btnAction.



Dans la liste Parent cliquez sur pnlClavier.



Le boutons appartient maintenant à l'objet pnlClavier.

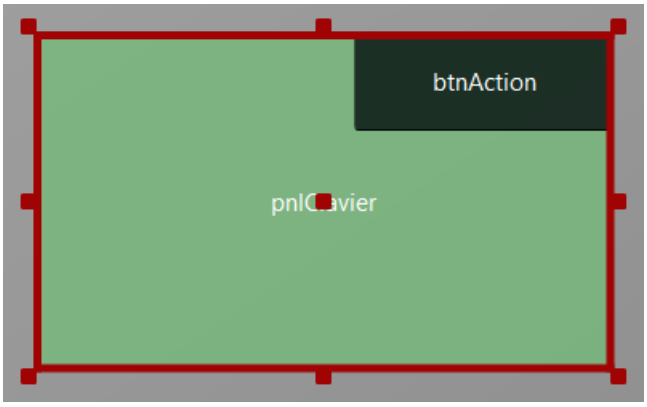


Réarrangeons les différents objets pour avoir plus de place pour le clavier.

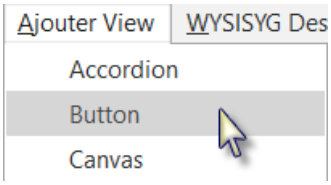
Modifiez les propriétés ci-dessous :

lblCommentaire	Top = 220
pnlClavier	Left = 50
pnlClavier	Top = 350
pnlClavier	Width = 290
pnlClavier	Height = 170
pnlClavier	Épaisseur de cadre = 0

Déplacez btnAction dans le coin supérieur droit de pnlClavier.

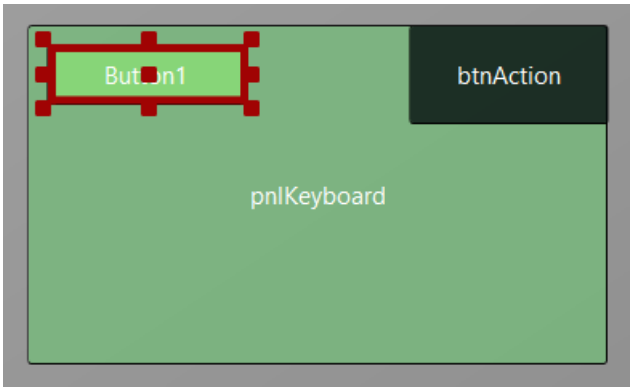


Cliquez sur pnlClavier pour le sélectionner.

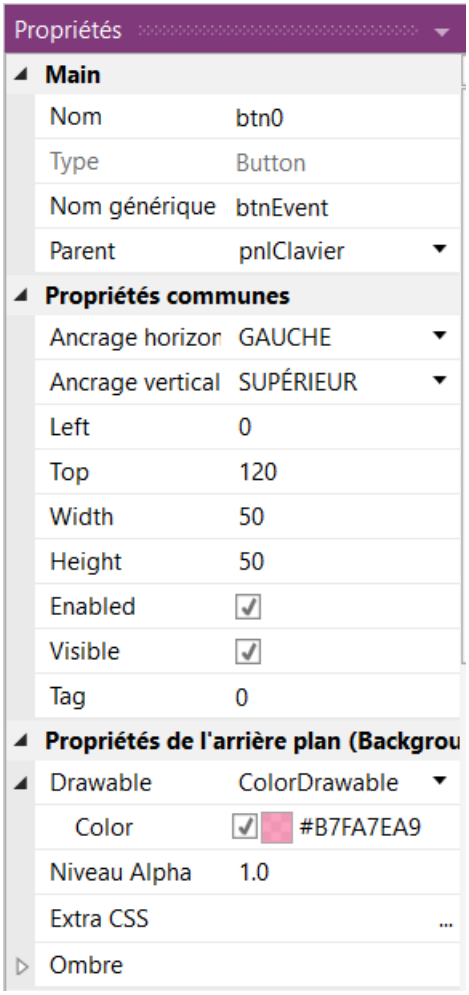


Cliquez sur

Pour ajouter un nouveau bouton.



Le nouveau bouton est ajouté.



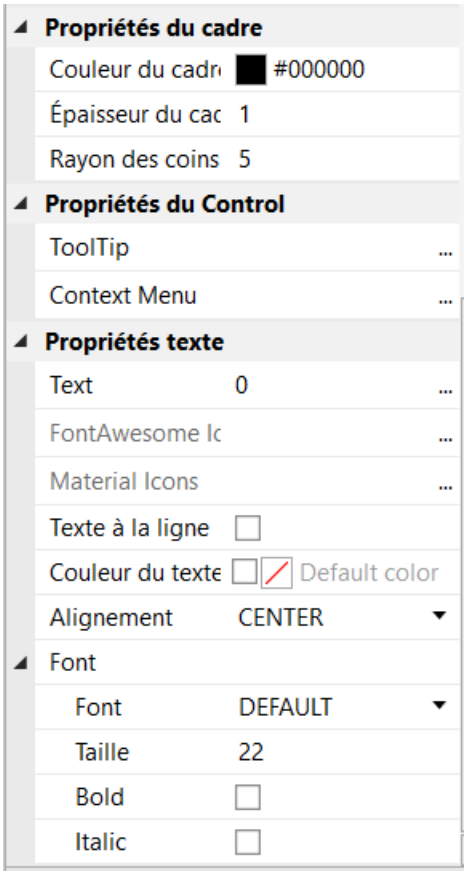
Modifiez les propriétés ci-dessous :

Nom en btn0
Nom générique… en btnEvent

Left en 0
Top en 120
Width en 50
Height en 50

Tag en 0

Color en #B7FA7EA9

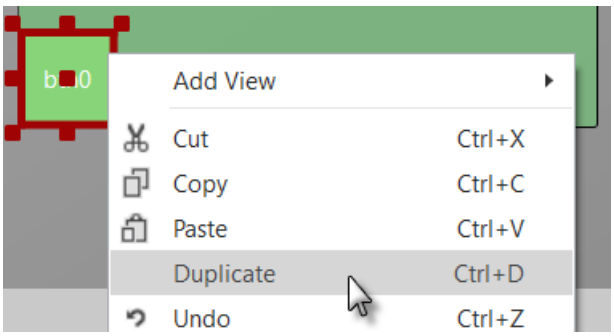


Épaisseur du cadre en 1
Rayon des coins en 5

Text en 0

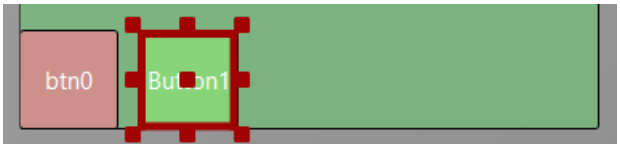
Taille en 22

Le bouton ressemblera à ceci.



Dupliquez btn0.

Cliquez avec le bouton droit sur btn0 et cliquez sur Dupliquer.



Déplacez le nouveau bouton juste à côté du précédent avec un petit espace.

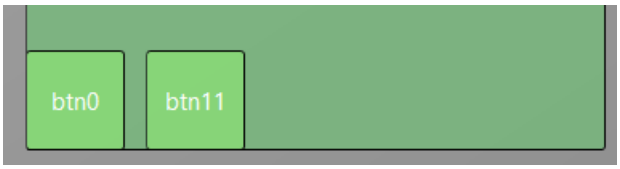
Main	
Nom	btn1
Tag	1
Text	1 ...

Modifiez les propriétés suivantes :

Nom en btn1

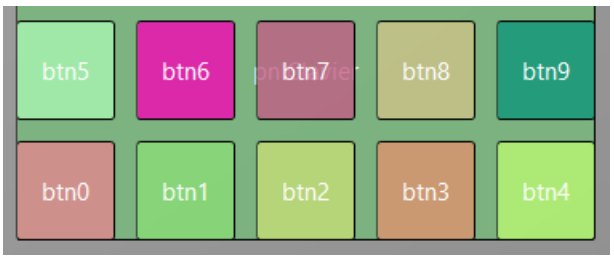
Tag en 1

Text en 1



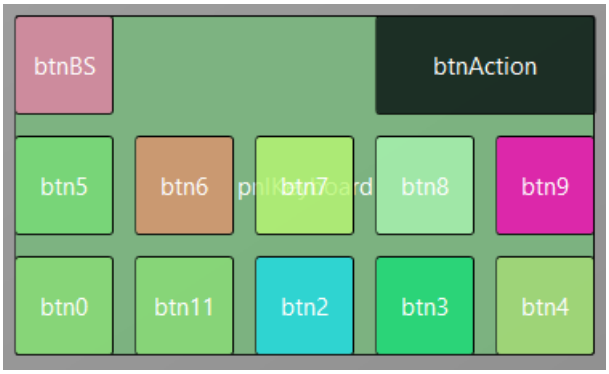
Et le résultat.

Ajoutez 8 boutons supplémentaires comme dans l'image.



Modifiez les propriétés ci-dessous :

Nom	btn2,	btn3,	btn4	etc.
Tag	2 ,	3 ,	4	etc.
Text	2 ,	3 ,	4	etc.

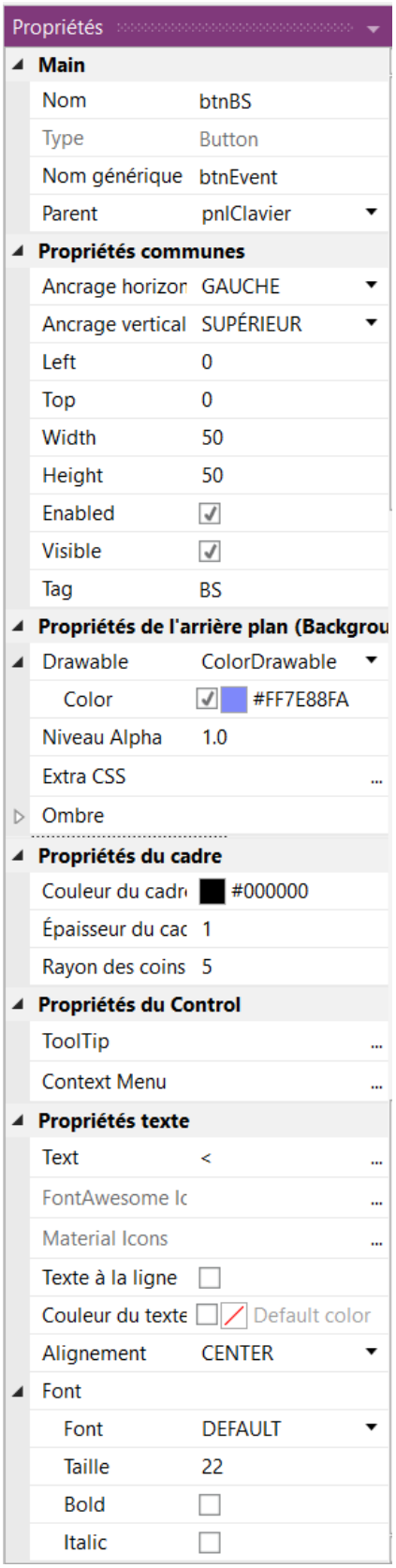


Pour créer le bouton Retour arrière (BackSpace), dupliquez un des boutons numériques, et positionnez-le dans le coin supérieur gauche.

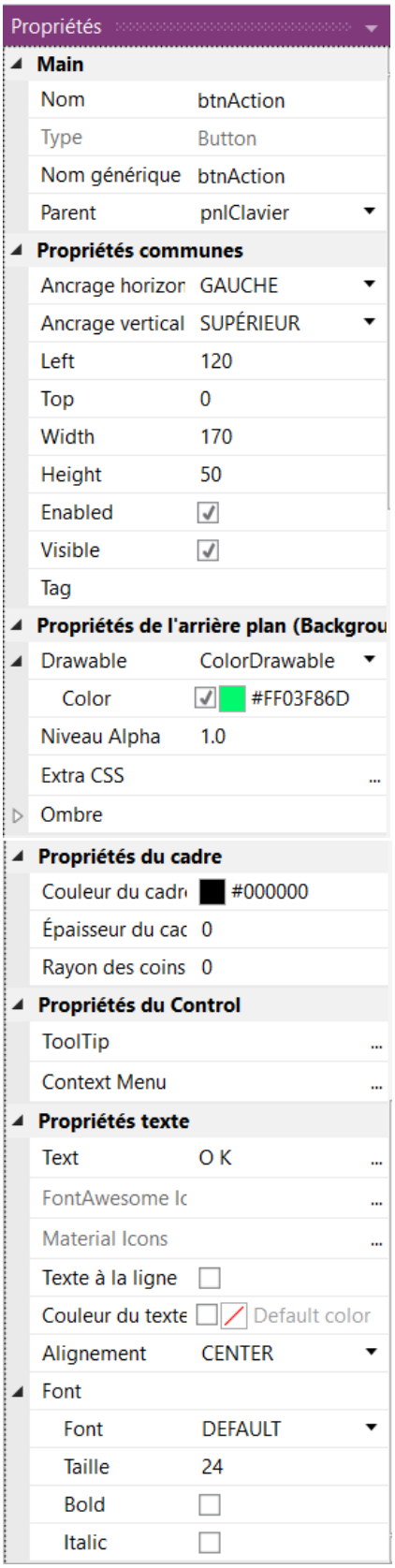
Redimensionnez et positionnez btnAction.

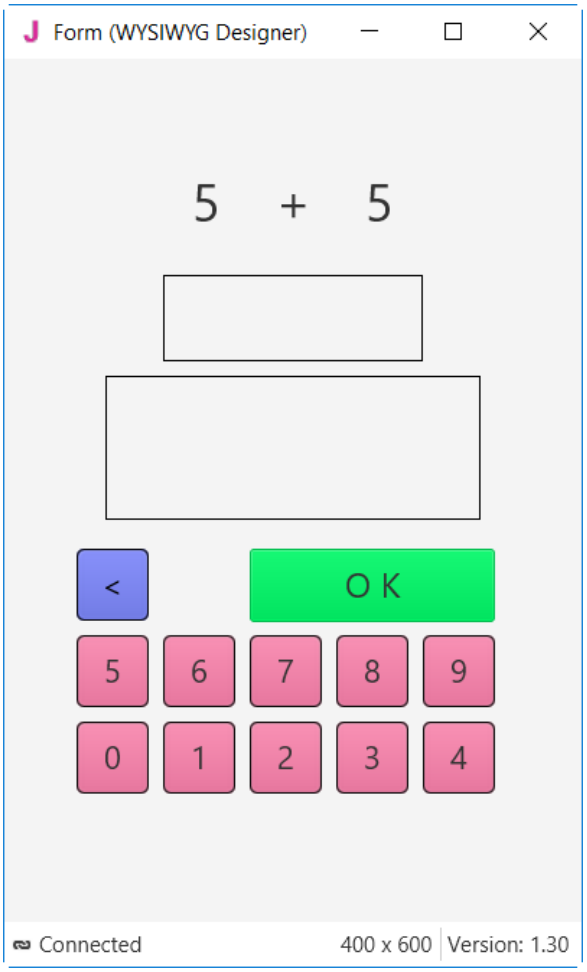
Modifiez leur Nom, Tag, Text et Color comme ci-dessous.

btnBS <



btnAction O K





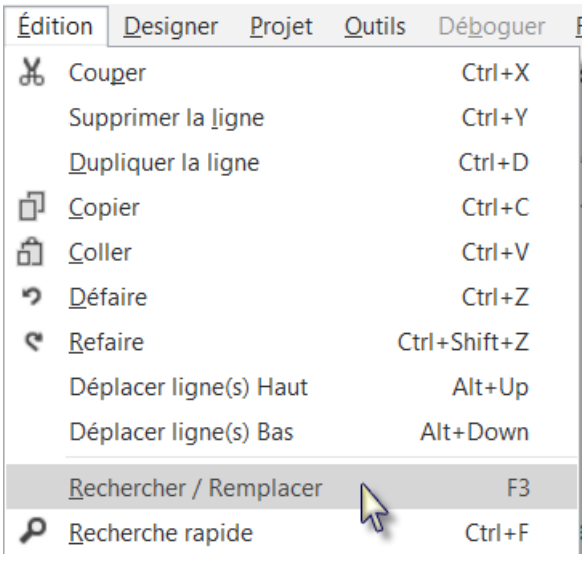
Le layout terminé dans la fenêtre WYSIWYG.

Vous pouviez aussi suivre toutes les évolutions de la définition du layout dans la fenêtre WYSIWYG.

Nous allons maintenant mettre à jour le code.
Tout d'abord, nous devons remplacer txfResultat par lblResultat puisque que nous avons remplacé un objet TextField par un objet Label.

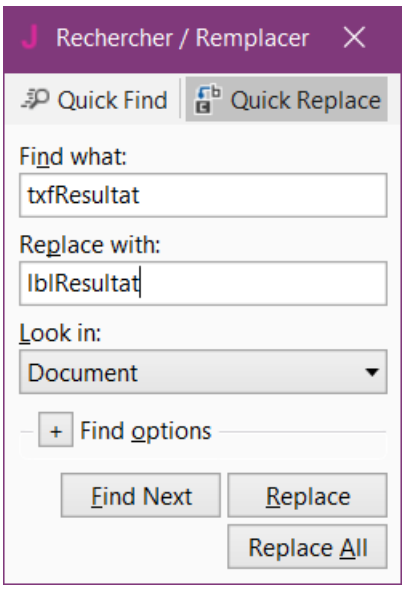
```
10    Private lblCommentaire As Label
11    Private lblNombre1 As Label
12    Private lblNombre2 As Label
13    Private lblSigneMath As Label
14    Private txfResultat As TextField
```

Double cliquez txfResultat pour le sélectionner.



Dans le menu Édition.

Cliquez sur Rechercher / Remplacer.



La fenêtre Rechercher / Remplacer est affichée.

Cliquez sur Replace All et fermez la fenêtre.

Nous devons aussi modifier le type de l'objet de TextField en Label.

```
Private lblResultat As Label
```

Maintenant nous écrivons la routine qui gère les événements Click des boutons.
Le nom générique des événements pour tous les boutons est "btnEvent", sauf pour btnAction.
Le nom de la routine associée à l'événement Click sera donc btnEvent_Click.
Écrivez le code ci-dessous :

```
Private Sub btnEvent_Action

End Sub
```

Nous devons connaitre le bouton qui a généré l'événement. Pour ça, nous utilisons l'objet spécial Sender qui contient, dans la routine d'événement, la référence de l'objet qui a généré l'événement.

```
Private Sub btnEvent_Action
   Private btnSender As Button

   btnSender = Sender

   Select btnSender.Tag
   Case "BS"
   Case Else
   End Select
End Sub
```

Pour avoir accès aux propriétés de l'objet qui a généré l'événement, nous déclarons une variable locale `Private btnSender As Button`.
et attribuons `btnSender = Sender`.

Puis, pour différentier entre le bouton Retour arrière et les boutons numériques nous utilisons une structure Select / Case / End Select et la propriété Tag des boutons.
Rappelez-vous, lorsque nous avions ajouté les boutons nous avions défini leur propriété Tag en BS, 0, 1, 2 etc.

```
   Select btnSender.Tag
   Case "BS"
   Case Else
```

`Select` définit la variable à tester.
Vérifie si c'est le bouton avec la valeur Tag "BS".
Gère tous les autres boutons.

Ajoutons le code pour les boutons numériques. Nous ajoutons le contenu de la propriété Text du bouton au texte dans la propriété Text du Label lblResultat dans la ligne :
`lblResultat.Text = lblResultat.Text & btnSender.Text`

```
   Select btnSender.Tag
   Case "BS"
   Case Else
      lblResultat.Text = lblResultat.Text & btnSender.Text
   End Select
End Sub
```

Le caractère "&" signifie concaténation, nous ajoutons donc au texte existant le texte du bouton qui a généré l'événement.

Écrivons maintenant le code pour le bouton Retour arrière (BS).

```
   Select btnSender.Tag
   Case "BS"
      If lblResult.Text.Length > 0 Then
         lblResult.Text = lblResult.Text.SubString2(0, lblResult.Text.Length - 1)
      End If
   Case Else
      lblResult.Text = lblResult.Text & btnSender.Text
   End Select
End Sub
```

Lorsqu'on presse le bouton Retour arrière nous voulons supprimer le dernier caractère du texte dans lblResultat. Ceci n'est possible que si la longueur du texte est > 0. Ceci est vérifié avec :
`If lblResultat.Text.Length > 0 Then`

Pour supprimer le dernier caractère nous utilisons la fonction SubString2.
`lblResultat.Text = lblResultat.Text.SubString2(0, lblResultat.Text.Length - 1)`

SubString2(BeginIndex, EndIndex) extrait un nouvel objet String commençant à l'index défini dans BeginIndex (inclusif) jusqu'à l'index défini dans EndIndex (exclusif).

La routine est terminée.

```
Private Sub btnEvent_Action
   Private btnSender As Button

   btnSender = Sender
   Select btnSender.Tag
      Case "BS"
         If lblResultat.Text.Length >0 Then
            lblResultat.Text = lblResultat.Text.SubString2(0,lblResultat.Text.Length - 1)
         End If
      Case Else
         lblResultat.Text = lblResultat.Text & btnSender.Text
   End Select
End Sub
```

Dans la routine `Sub btnAction_Action` nous ajoutons à la fin, `lblResult.Text = ""` pour vider le texte.

```
   Else
      New
      btnAction.Text = "O K"
      lblResultat.Text = ""
   End If
End Sub
```

Nous améliorons l'interface utilisateur en ajoutant des couleurs au Label lblCommentaire.
Définissons :
- Jaune pour un nouveau problème.
- Vert clair pour une réponse JUSTE.
- Rouge clair pour une réponse FAUX.

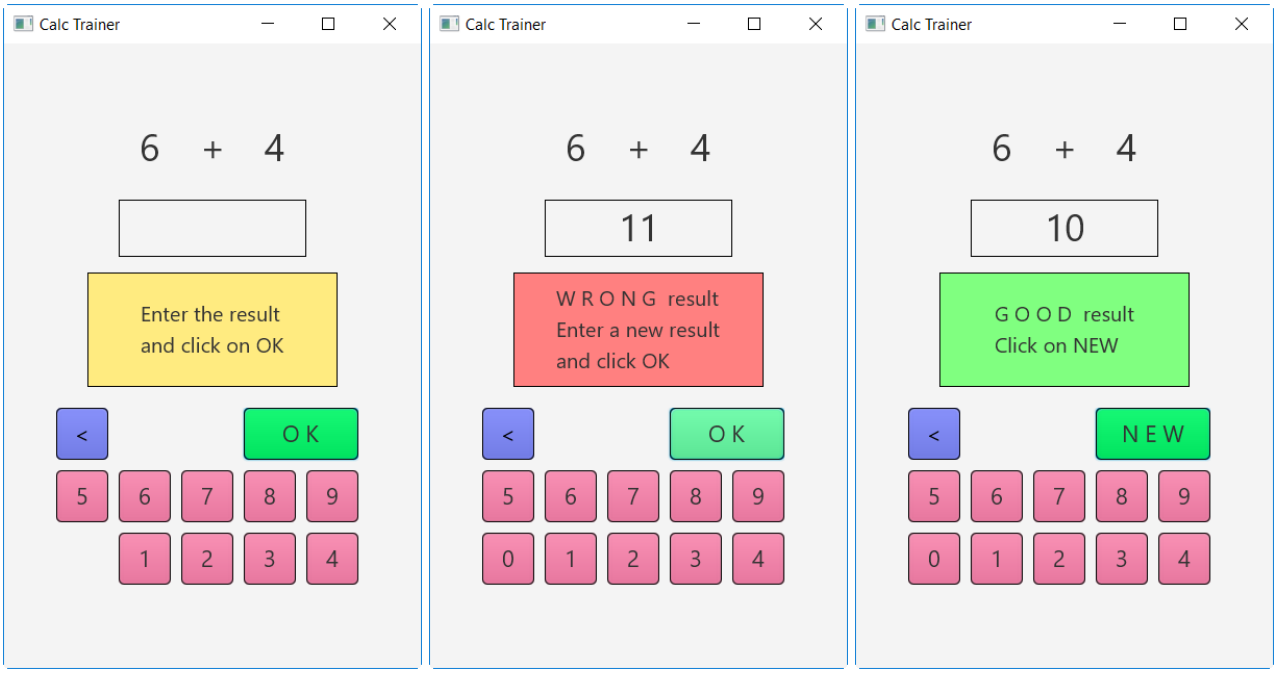
Nous ajoutons la ligne ci-dessous dans la routine NouveauProbleme :
`CSSUtils.SetBackgroundColor(lblCommentaire, fx.Colors.RGB(255,235,128))`
Nous avons besoin de bibliothèque CSSUtils pour modifier la couleur de fond d'un objet !
Voir page suivante comment ajouter une bibliothèque.

```
Private Sub NouveauProbleme
   Nombre1 = Rnd(1, 10)              ' Génère un nombre aléatoire entre 1 et 9
   Nombre2 = Rnd(1, 10)              ' Génère un nombre aléatoire entre 1 et 9
   lblNombre1.Text = Nombre1         ' Affiche Nombre1 dans le Label lblNombre1
   lblNombre2.Text = Nombre2         ' Affiche Nombre2 dans le Label lblNombre2
   lblCommentaire.Text = "Entrez le résultat" & CRLF & "et cliquez sur O K"
   CSSUtils.SetBackgroundColor(lblCommentaire, fx.Colors.RGB(255,235,128))  ' couleur jaune
   lblResultat.Text = ""             ' Vide la propriété Text de edtResult
End Sub
```

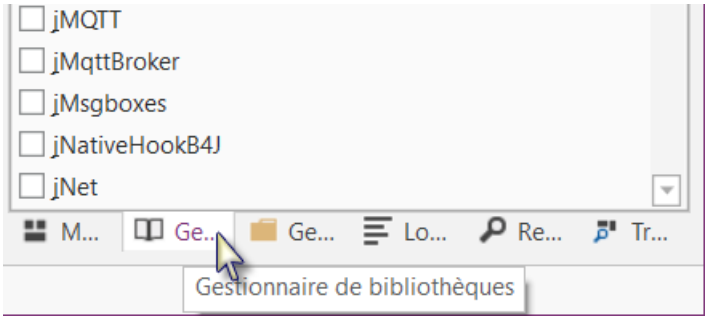
Et dans la routine TestResultat nous ajoutons les deux lignes avec
`CSSUtils.SetBackgroundColor...`

```
Private Sub TestResultat
   If lblResultat.Text = Nombre1 + Nombre2 Then
      lblCommentaire.Text = "Résultat  J U S T E" & CRLF & "Cliquez sur Nouveau"
      CSSUtils.SetBackgroundColor(lblCommentaire, fx.Colors.RGB(128,255,128))   ' couleur vert clair
      btnAction.Text = "Nouveau"
   Else
      lblCommentaire.Text = "Résultat  F A U X" & CRLF & "Entrez un nouveau résultat" & CRLF & "et cliquez
sur O K"
      CSSUtils.SetBackgroundColor(lblCommentaire, fx.Colors.RGB(255,128,128))   ' couleur rouge clair
   End If
End Sub
```

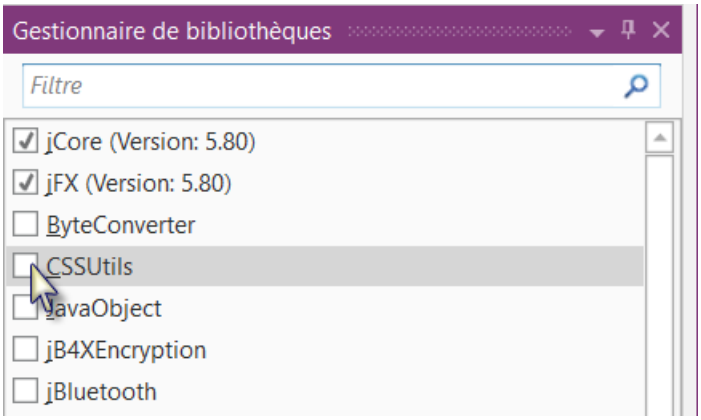
Nous donnons encore un titre significatif au programme en ajoutant
`MainForm.Title = "Math Trainer"` dans la routine AppStart juste après `MainForm.Show`.



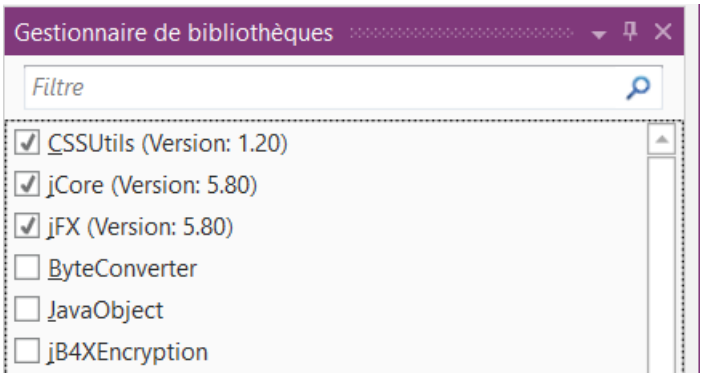
Ajout de la bibliothèque CSSUtils.



Dans le coin inférieur droit, cliquez sur l'onglet Gestionnaire de bibliothèques.



Cliquez sur CSSUtils dans la liste des bibliothèques.



Et le résultat.
La bibliothèque CSSUtils est ajoutée au projet.

Les bibliothèques sont classées par ordre alphabétique.

L'utilisation de bibliothèques est expliquée dans le chapitre *Bibliothèques* dans le livret *B4x Langage Basic*.

Une autre amélioration est de rendre le bouton ‘0’ invisible lorsque l’utilisateur doit entrer le premier chiffre.
Pour ça, nous cachons le bouton dans la routine NouveauProbleme avec `btn0.Visible = False`.

```
Private Sub NouveauProbleme
   Nombre1 = Rnd(1, 10)            ' Génère un nombre aléatoire entre 1 et 9
   Nombre2 = Rnd(1, 10)            ' Génère un nombre aléatoire entre 1 et 9
   lblNombre1.Text = Nombre1       ' Affiche Nombre1 dans le Label lblNombre1
   lblNombre2.Text = Nombre2       ' Affiche Nombre2 dans le Label lblNombre2
   lblCommentaire.Text = "Entrez le résultat" & CRLF & "et cliquez sur O K"
   CSSUtils.SetBackgroundColor(lblCommentaire, fx.Colors.RGB(255,235,128))     ' couleur jaune
   lblResultat.Text = ""           ' Vide la propriété Text de edtResult
   btn0.Visible = False
End Sub
```

Nous voyons que `btn0` est affiché en rouge, ce qui signifie une erreur, et dans ce cas que l’objet n’est pas reconnu par l’EDI.

Pour y remédier, nous ajoutons `btn0` dans la routine Globals dans la ligne ci-dessous :

```
   Private btnAction, btn0 As Button
```

Maintenant, `btn0` n’est plus en rouge.

```
btn0.Visible = False
```

En plus, dans la routine btnEvent_Action, nous cachons le bouton si la longueur du texte dans lblResultat est égale à zéro et l’affichons si la longueur est plus grande que zéro.

```
Private Sub btnEvent_Action
   Dim btnSender As Button

   btnSender = Sender

   Select btnSender.Tag
   Case "BS"
      If lblResultat.Text.Length > 0 Then
         lblResultat.Text = lblResultat.Text.SubString2(0,lblResultat.Text.Length - 1)
      End If
   Case Else
      lblResultat.Text = lblResultat.Text & Send.Tag
   End Select

   If lblResultat.Text.Length = 0 Then
      btn0.Visible = False
   Else
      btn0.Visible = True
   End If
End Sub
```

5 Premiers pas avec B4R

B4R – Le moyen le plus simple pour développer des applications natives et puissantes pour des microcontrôleurs Arduino.

B4R suit les mêmes concepts que les autres outils B4X (B4A, B4i, B4J), fournissant un outil de développement simple et puissant.

Les applications compilées fonctionnent sur des circuits compatibles Arduino.

5.1 Installation de l'EDI Arduino

B4R nécessite l'installation de l'EDI Arduino version 1.6.7+.

Téléchargez l'EDI Arduino depuis ce lien : https://www.arduino.cc/en/Main/Software et installez-le.

N'installez pas Arduino 1.6.12 car il a un défaut.

Ne téléchargez pas l'EDI Arduino depuis ce site : http://www.arduino.org/downloads
Il ne fonctionne pas.

5.2 Installation de Microsoft .Net Framework

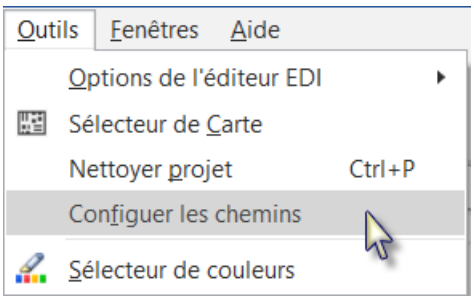
Sur la majorité des ordinateurs Microsoft .Net Framework est déjà préinstallé.

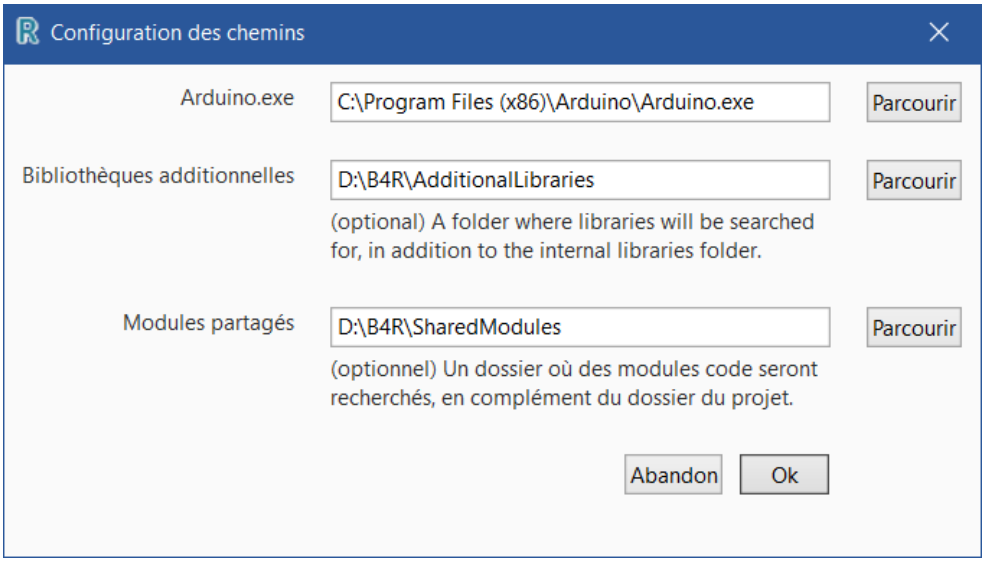
Si ce n'est pas le cas, vous devez installer soit :

- .Net Framework 4.5.2 pour Windows Vista +
- .Net Framework 4.0 pour Windows XP

5.3 Installation et configuration de B4R

- Téléchargez et installez B4R.
- Lancez B4R.
- Cliquez dans le menu Outils sur Configuer les chemins.





Utilisez les bouton Parcourir pour localiser le dossier contenant le fichier "arduino.exe".
Le chemin dépend de l'endroit où vous avez installé l'EDI Arduino.

Il est recommandé de créer un dossier spécifique pour les bibliothèques additionnelles.
B4R utilise deux types de bibliothèques :

- Bibliothèques standard, qui sont fournies avec B4R et se trouvent dans le dossier Libraries de B4R.
  Ces bibliothèques sont automatiquement mises à jour lorsque vous installez une nouvelle version de B4R.
- Bibliothèques additionnelles, qui ne font pas partie de B4R, et doivent être enregistrées dans un dossier spécifique différent de celui de B4R.
  Plus de détails dans le chapitre *Bibliothèques additionnelles* du livret *B4x Langage Basic*.

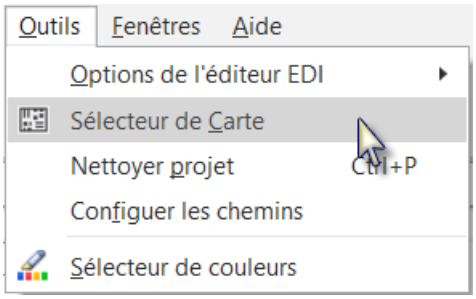
Vous devriez aussi créer un dossier spécifique pour les modules partagés (Shared Modules).
Un même module code peut être utilise par différents projets sans avoir à l'enregistrer dans le dossier du projet.

5.4 Connection d'un circuit

Lorsque vous connectez un circuit au PC avec un câble USB, Windows charge le pilote et affiche le port série utilisé.

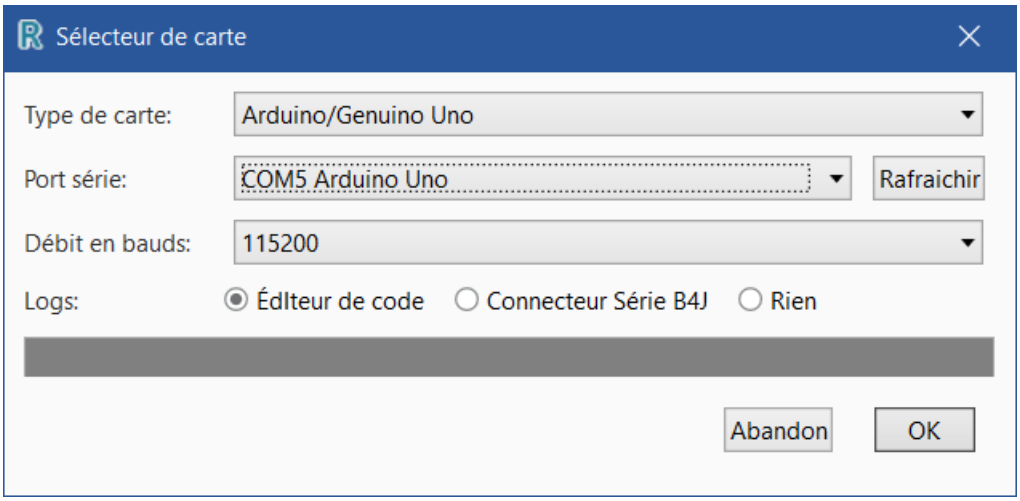
5.5 Sélection d'un circuit

Lancez B4R.



Dans le menu Outils, cliquez sur Sélecteur de Carte.

Une fenêtre similaire à celle ci-dessous sera affichée.

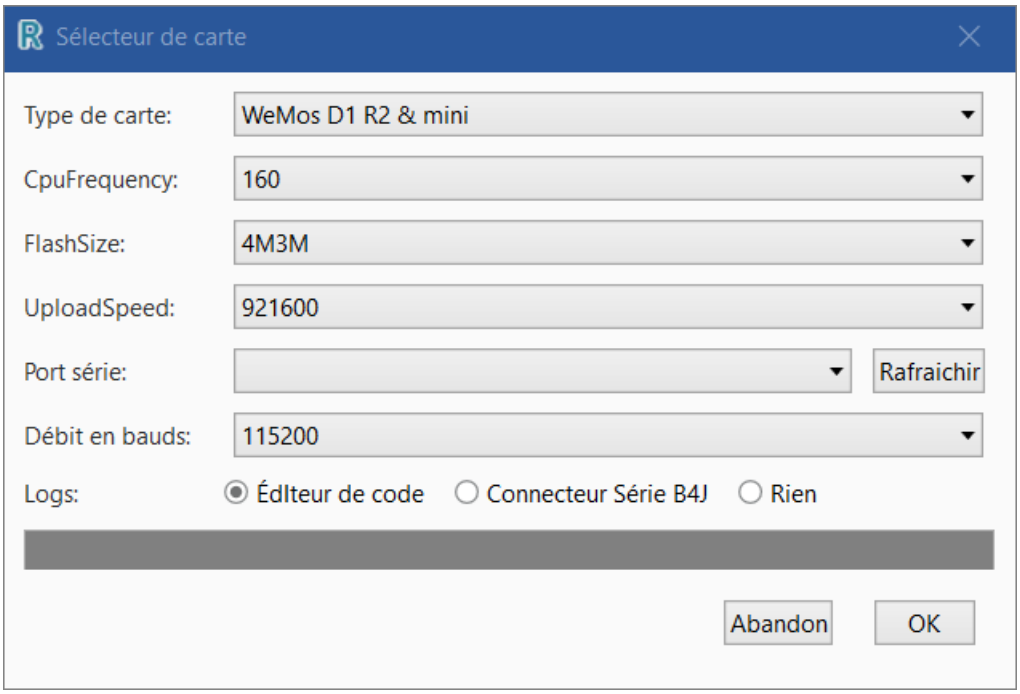


Sélectionnez le type de carte dans la liste déroulante.
Sélectionnez le port série et le débit en bauds.

Si un seul circuit est connecté il sera reconnu automatiquement.

Seul les circuits connectés sont affichés.

Selon le type de circuit d'autres paramètres peuvent être définis.



5.6 Le circuit Arduino UNO

Dans ce chapitre vous trouvez des explications sur quelques fonctions de base de Arduino UNO qui peuvent être utiles aux débutants.
Le circuit Arduino UNO est le circuit de base de la famille Arduino.

Il existe d'autres modules plus avancés.

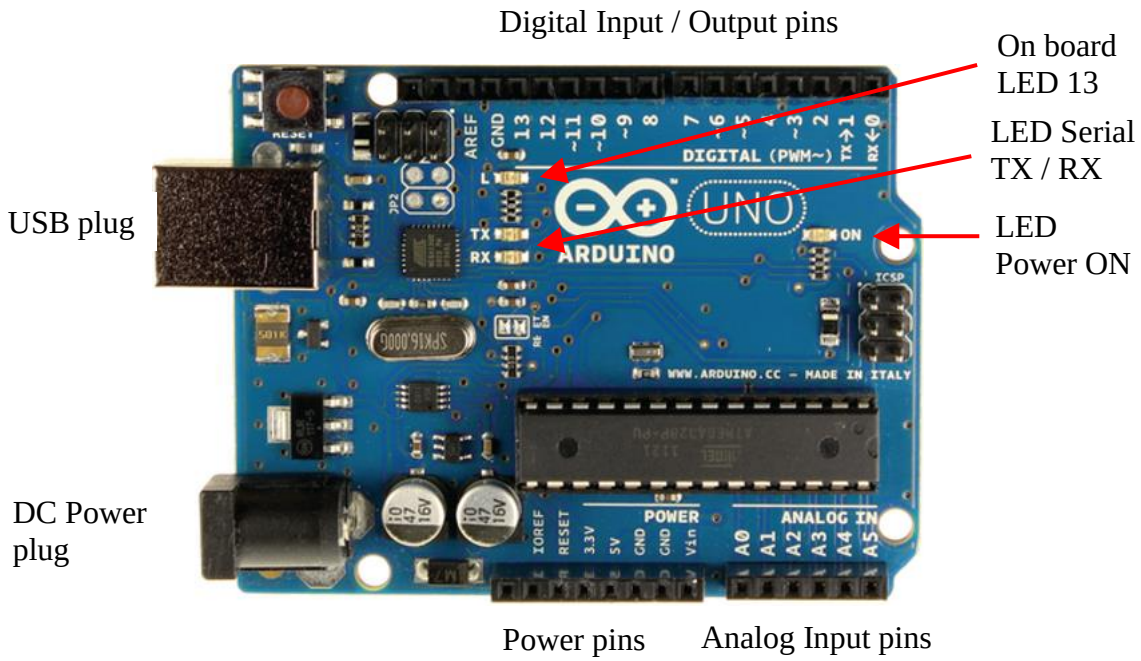
- Arduino DUE
- Arduino MEGA
- Arduino MICRO
- etc. voir Compare board specs.

Des circuits additionnels, appelés 'Shields' peuvent être clipés sur les circuits Arduino.

- Arduino Wi-Fi Shield 101
- Arduino Ethernet Shield
- etc.

L'Arduino UNO :

La source des informations dans ce chapitre est un résumé du site Arduino.



Digital Input / Output pins	Broches entrée / sortie digitales.
USB plug	Connecteur USB.
DC Power plug	Connecteur alimentation DC.
On board LED 13	LED 13 sur le circuit.
Serial TX / RX	Signaux série TX / RX.
LED Power ON	LED alimentation ON.
Power pins	Broches d'alimentation.
Analog Input pins	Broches entrées analogiques.

J'ai intentionnellement gardé les dénominations Anglaises sur le schéma.

5.6.1 Alimentation en courant

Le circuit peut être alimenté en courant soit par le connecteur DC power (7 - 12V), le connecteur USB (5V), ou la broche VIN du circuit (7-12V).
Une alimentation par les broches 5V ou 3.3V by-passe le régulateur, et peut endommager votre circuit (voir Broches ci-dessous). A déconseiller.

5.6.2 Broches

L'Arduino UNO a 3 connecteurs :

- Broches d'alimentation.
- Broches d'entrée / sortie digitaux (Digital Input / Output pins).
- Broches d'entrée analogiques (Analog Input pins).

5.6.3 Broches d'alimentation

Les Broches d'alimentations sont :

- **GND** Masse de l'alimentation, 2 broches (GND ground).
- **VIN** Broche d'alimentation (VIN V entrée).
  Tension d'alimentation lorsque le circuit UNO est alimenté par une source extérieure (par opposition à une alimentation de 5 volts depuis le connecteur USB ou une autre source régulée). Vous pouvez alimenter le circuit par cette broche, ou, si le circuit est alimenté par une source externe par le connecteur (DC Power plug), y accéder par cette broche.
  Tension 7 – 12 V.
- **5V** 5 Volt, tension de référence. **N'alimentez pas cette broche !**
  Cette broche fournit une tension régulée de 5V du régulateur du circuit.
- **3.3V** 3.3 Volt, tension de référence. **N'alimentez pas cette broche !**
  Une source de 3.3 volt générée par le régulateur du circuit. Courant maximum 50 mA.
- **RESET**
  Mettez cette broche sur LOW pour reseter microcontrôleur. Typiquement utilisé pour ajouter un bouton reset à des circuits 'shields' qui bloquent celui du circuit.
- **IOREF**
  Cette broche, sur le circuit UNO, fournit la tension de référence avec laquelle le microcontrôleur fonctionne. Un circuit shield, correctement configure, peut lire la tension de la broche IOREF et sélectionner la source appropriée pour travailler avec 5V ou 3.3V.

5.6.3.1 Broches d'entrée / sortie digitales (Digital Input / Output pins)

Chacune des broches digitales de l'UNO peut être utilisée comme entrée ou sortie, en spécifiant son mode *pinMode*, avec *DigitalRead* ou *DigitalWrite*. Elles opèrent avec 5 volts. Chaque broche peut fournir ou recevoir un courant de 20 mA comme conditions de fonctionnement recommandées et comprend une résistance 'pull-up' interne (déconnectée par défaut) de 20-50 kohm. Une valeur de courant maximale de 40mA ne doit pas être dépassée sur aucune des broches pour éviter un endommagement permanent du microcontrôleur.

En complément, quelques broches ont des fonctions spécialisées :

- Serial: 0 (RX) et 1 (TX). Utilisées pour recevoir (RX) et transmettre (TX) TTL des données série TTL. Ces broches sont reliées aux broches correspondantes du circuit ATmega8U2 USB-to-TTL Serial.
- PWM: ~3, ~5, ~6, ~9, ~10, et ~11. Ces numéros ont "~" pour préfixe.
  Elles peuvent fournir des signaux de sortie en PWM (**P**ulse **W**idth **M**odulation) (modulation de largeur d'impulsions) avec la fonction *AnalogWrite* permettant la modulation de la luminosité de LEDs (**L**ight **E**mitting **D**iode) (DELs Diodes Electro-Luminescentes) ou alimenter un moteur CC à différentes vitesses.
  Une valeur de 0 correspond à OFF et 255 correspond à toujours ON.
  Utilisation:
  pinTest3.AnalogWrite(Value As UInt)
  ```
  pinTest3.AnalogWrite(196)
  ```
  Après *AnalogWrite,* la broche va générer un signal rectangulaire constant avec un rapport cyclique spécifié jusqu'à l'appel suivant de *AnalogWrite* (ou un appel à *DigitalRead* ou *DigitalWrite* sur la même broche). La fréquence du signal PWM est approximativement de 490 Hz sur la plupart des broches. Sur l'UNO et les circuits similaires, les broches 5 et 6 ont une fréquence d'approximativement 980 Hz. Les broches 3 et 11 du circuit Leonardo fournissent également un signal à 980 Hz.
- LED 13 : Il y a une LED sur le circuit gérée par la broche 13. Lorsque le signal sur la broche est HAUT, la LED est allumée, si le signal est BAS elle est éteinte.

5.6.3.2 Broches d'entrée analogiques

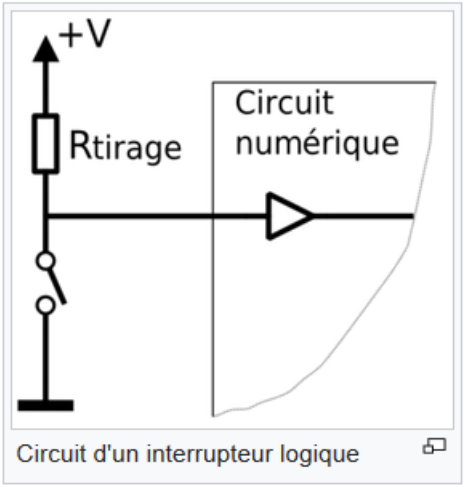
L'Arduino UNO contient 6 broches d'entée analogiques (Analog input pins) A0 à A5 avec un convertisseur analogique vers numérique de 10 bit, donc une résolution de 0 à 1023.
La tension de référence est de 5 Volt donnant une résolution de 4.9 mV par unité.
Alors que la fonction principale des broches analogiques pour la plupart des utilisateurs d'Arduino est de lire des capteurs analogiques, les broches analogiques ont aussi toutes les fonctionnalités des broches numériques 0 - 13.

5.6.4 Modes d'entrée INPUT / INPUT_PULLUP

Si vous avez configuré une broche avec le mode INPUT, et que vous lisez l'état d'un interrupteur, l'état de la broche sera "flottant" lorsque l'interrupteur est ouvert, donnant des résultats imprévisibles. Pour assurer une lecture correcte lorsque l'interrupteur est ouvert, une résistance de rappel ('pull-up') doit être ajoutée. La fonction de cette résistance est de mettre la broche dans un état connu lorsque l'interrupteur est ouvert. Une résistance de 10 K ohm est généralement utilisée, c'est une valeur suffisamment basse pour éviter un point flottant et en même temps une valeur suffisamment haute pour ne pas soutirer un courant trop élevé lorsque l'interrupteur est fermé.

Le mode INPUT_PULLUP ajoute une résistance de rappel ('pull up') interne pour éviter de devoir en ajouter une en externe.

Avec une résistance de rappel, la broche renvoie False lorsque l'interrupteur est fermé car elle est mise à 0 Volt.



Circuit d'un interrupteur logique

Source Wikipedia

5.6.5 Fonctions de base des broches

5.6.5.1 Initialize

Initialise une broche.

Pin.Initialize(Pin As Byte, Mode As Byte)

Pin est le numéro de la broche.
- 0, 1, 2, 3 etc. pour les broches digitales
- Pin.A0, Pin.A1 , Pin.A2 etc. pour les roches analogiques.

Mode est un des trois modes de connexion :
- MODE_INPUT
- MODE_INPUT_PULLUP ajoute une résistance ‘pull up’ interne.
- MODE_OUTPUT

Exemple1 : Initialisation de la broche digitale 3 en entrée.
```
Private pinTest1 As Pin
pinTest1.Initialize(3, pinTest1.MODE_INPUT)
```

Exemple2 : Initialisation de la broche digitale 3 en entrée avec une résistance ‘pull up’.
```
Private pinTest2 As Pin
pinTest2.Initialize(3, pinTest2.MODE_INPUT_PULLUP)
```

Exemple3 : Initialisation de la broche digitale 3 en sortie.
```
Private pinTest3 As Pin
pinTest3.Initialize(3, pinTest3.MODE_OUTPUT)
```

Exemple4 : Initialisation de la broche analogique 4 en entrée.
```
Private pinTest4 As Pin
pinTest4.Initialize(pinTest4.A4, pinTest4.MODE_INPUT)
```

Les broches analogiques, sur l’Arduino UNO, peuvent aussi être référencées par des numéros, comme ci-dessous :
```
pinTest4.Initialize(18, pinTest4.MODE_INPUT)
```

Pin.A0 = 14
Pin.A1 = 15
Pin.A2 = 16
Pin.A3 = 17
Pin.A4 = 18
Pin.A5 = 19

L’initialisation d’une broche analogique en sortie fonctionne comme une broche digitale en sortie.

5.6.5.2 DigitalRead

DigitalRead lit la valeur digitale actuelle d'une broche.
Les valeurs renvoyées sont True ou False.

Pin.DigitalRead renvoie une valeur du type Boolean.

Il y a deux modes dépendant du type de signal d'entrée.

- Pin.MODE_INPUT
- Pin. MODE_INPUT_PULLUP ajoute une résistance 'Pulp' à utiliser avec in interrupteur.

Exemple :

```
Private pinTest1 As Pin
pinTest1.Initialize(3, pinTest.MODE_INPUT)

Private Value As Boolean
Value = pinTest1.DigitalRead
```

L'Arduino utilise en interne 0 et 1 pour les valeurs booléennes.
`Log("State: ", Value)`
Va afficher soit 0 pour False ou 1 pour True dans les Logs.
Dans le code vous pouvez utiliser False et True.

5.6.5.3 DigitalWrite

DigitalWrite écrit une valeur booléenne sur la broche spécifiée.
Peut-être utilise sur toutes les broches digitales comme analogiques.

Pin.DigitalWrite (Value As Boolean)

Exemple:

```
Private pinTest3 As Pin
pinTest3.Initialize(3, pinTest3.MODE_OUTPUT)
```

`pinTest3.DigitalWrite(True)` directement avec une valeur.
`pinTest3.DigitalWrite(Value)` avec une variable.

5.6.5.4 AnalogRead

AnalogRead lit la valeur actuelle sur une broche analogique.
La valeur renvoyée est du type UInt avec des valeurs entre 0 et 1023 (10 bits).
La tension de référence est 5V.

Exemple :

```
Private pinPot As Pin
pinPot.Initialize(pinPot.A4, pinPot.MODE_INPUT)

Private Value As UInt
Value = pinPot.AnalogRead
```

5.6.5.5 AnalogWrite

AnalogWrite attribue un octet (Byte) à la broche définie pour une modulation de largeur d'impulsions.
AnalogWrite n'a rien faire avec les broches analogiques ni avec AnalogRead.

AnalogWrite peut être utilise seulement avec les broches numériques ~3, ~5, ~6, ~9, ~10, et ~11 sur le circuit Arduino UNO, les broches avec le préfixe ~.

Pin.AnalogWrite (Value As UInt)

Exemple : nous utilisons la broche ~3 qui permet la modulation PWM.

```
Private pinTest3 As Pin
pinTest3.Initialize(3, pinTest3.MODE_OUTPUT)
```

`pinTest3.AnalogWrite(145)` directement avec une valeur.
`pinTest3.AnalogWrite(Value)` avec une variable, `Value` doit être une variable du type UInt.

5.7 Premiers programmes

Tous les projets ont été réalisés avec le Arduino Starter Kit.

Les diagrammes de montage ont été réalisés avec le programme Fritzing.

Lorsque nous exécutons B4R nous obtenons le code par défaut ci-dessous.

La `#Region Project Attributes` est normalement réduite; ces attributs sont expliqués dans le chapitre *Entêtes de code Project Attributes / Activity Attributes* dans le livret *B4x EDI*.

```
#Region Project Attributes
   #AutoFlushLogs: True
   #CheckArrayBounds: True
   #StackBufferSize: 300
#End Region

Sub Process_Globals
   'These global variables will be declared once when the application starts.
   'Public variables can be accessed from all modules.
   Public Serial1 As Serial
End Sub

Private Sub AppStart
   Serial1.Initialize(115200)
   Log("AppStart")
End Sub
```

`#AutoFlushLogs:`	AutoFlushLogs: True assure que les Logs sont renvoyés sans problèmes. AutoFlushLogs: False peut provoquer des problèmes.
`#CheckArrayBounds:`	Vérifie les limites des tableaux (arrays) ou pas.
`#StackBufferSize:`	Définit la dimension par défaut du tampon de la pile (Stack buffer size).

`Public Serial1 As Serial`	Définit l'interface sérielle avec l'ordinateur.
`Serial1.Initialize(115200)`	Initialise le port série avec un débit en baud de 115200 Hertz.
`Log("AppStart")`	Affiche `AppStart` dans l'onglet Logs au démarrage du programme. Les Logs sont expiliqués dans le chapitre *Logs* dans le livret *B4x EDI*.

5.7.1 Bouton.b4r

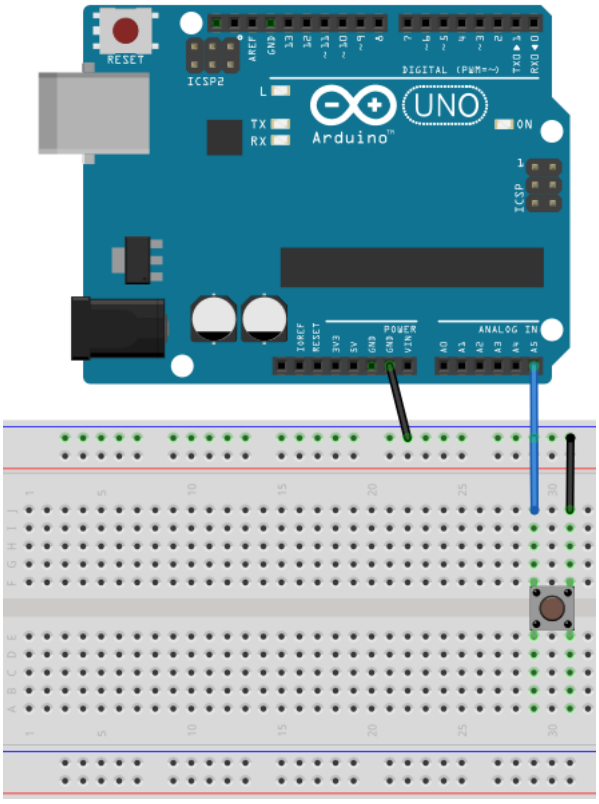
Écrivons le premier programme.

Il est similaire à l'exemple Button d'Erel dans le forum. Nous utilisons un commutateur à bouton poussoir et la LED 13 se trouvant sur la carte de l'Arduino UNO.

Le projet Bouton.b4r est disponible dans le dossier CodesSource\B4R\Bouton.

- Lancez B4R.
- Sauvez le projet sous Bouton dans un dossier avec le nom Bouton.
- Réalisez le montage du circuit avec le bouton et le câblage.
- Connectez l'Arduino au PC avec un câble USB.
- Écrivez le code.
- Exécutez le programme.

5.7.1.1 Schéma



Matériel :

- 1 commutateur à bouton poussoir

Reliez une des broches **GND** (masse / ground) de l'Arduino à une des lignes **GND** du breadboard.
Puis, reliez une des broches du commutateur à la ligne **GND** du breadboard.
Et, reliez l'autre broche du commutateur à la broche analogique **A5** de l'Arduino.

Nous aurions pu relier la première broche du commutateur directement à une des broches **GND** de l'Arduino, mais le montage est déjà prêt pour les exemples suivants.

Nous aurions aussi pu utiliser une des broches digitales au lieu d'une broche analogique.

5.7.1.2 Code

```
Sub Process_Globals
   Public Serial1 As Serial
   Private pinBouton As Pin      'broche (pin) pour le bouton
   Private pinLED13 As Pin       'broche (pin) pour la LED 13 de l'Arduino
End Sub
```

Nous déclarons la broche du commutateur et la LED 13 de l'Arduino.

```
Private Sub AppStart
   Serial1.Initialize(115200)
   Log("AppStart")

   pinBouton.Initialize(pinBouton.A5, pinBouton.MODE_INPUT_PULLUP)
   'Utilisation d'une résistance pull up pour éviter que la broche soit flottante.
   pinBouton.AddListener("pinBouton_StateChanged")

   pinLED13.Initialize(13, pinLED13.MODE_OUTPUT)
End Sub
```

Nous initialisons `pinBouton`, comme broche analogique **A5**, avec `pinBouton.A5` et définissons le mode d'entrée à `pinBouton.MODE_INPUT_PULLUP`. Nous avons besoin d'une résistance 'pull up' pour éviter que la broche ne soit flottante, `MODE_INPUT_PULLUP` connecte une résistance 'pull up' interne.

Nous ajoutons `pinBouton.AddListener("pinBouton_StateChanged")`, pour générer un événement StateChanged lorsque l'état de la broche `pinBouton` change, ce qui veut dire lorsque le bouton est pressé ou relâché.

Nous initialisons `pinLED13`, comme broche digitale 13, la LED 13 interne de l'Arduino, et définissons son mode de sortie `pinLED13.MODE_OUTPUT`.

```
Sub pinBouton_StateChanged (State As Boolean)
   Log("State: ", State)
   'State sera False lorsque le bouton est pressé à cause du mode PULLUP
   pinLED13.DigitalWrite(Not(State))
End Sub
```

Nous ajoutons un Log, `Log("State: ", State)`, pour afficher l'état.

Nous attribuons la valeur de State à la broche digitale LED 13, `pinLED13.DigitalWrite(Not(State))`.

Nous écrivons `Not(State)` car State sera False lorsque le bouton poussoir est pressé à cause du mode PULLUP.

Cliquez sur ▶ ou pressez F5 pour exécuter le code.

Lorsque nous pressons le bouton poussoir, la LED 13 de l'Arduino UNO s'allume et lorsque nous relâchons le bouton poussoir la LED s'éteint.

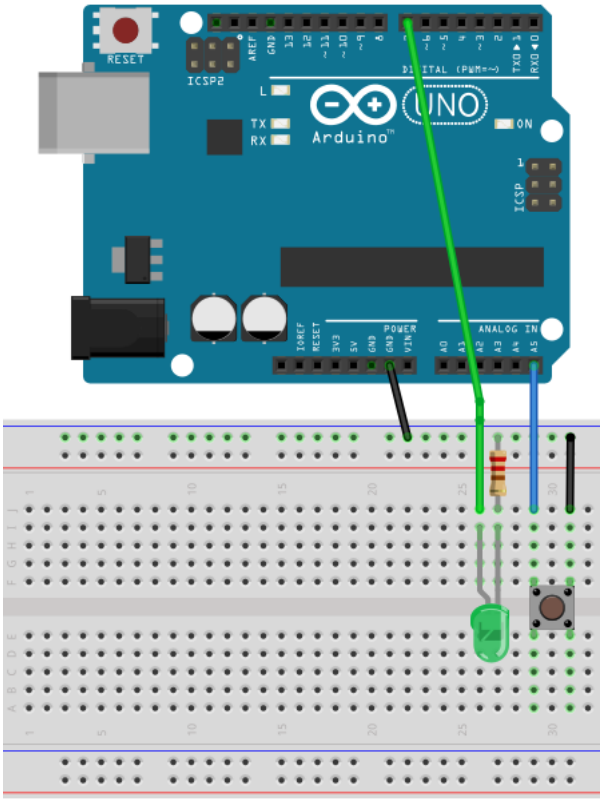
5.7.2 LedVerte.b4r

Pour ce projet nous utilisons une copie du projet Bouton.
Créez un nouveau dossier LedVerte, copiez-y tous les fichiers du projet Bouton et renommez les fichiers Bouton.xxx en LedVerte.xxx.

Nous ajoutons une LED verte à notre circuit qui peut être allumée ou éteinte avec le commutateur à bouton poussoir du premier projet.

Le projet LedVerte.b4r est disponible dans le dossier CodesSource\B4R\LedVerte.

5.7.2.1 Schéma



Matériel :

- 1 commutateur à bouton poussoir
- 1 LED verte
- 1 résistance 220 Ω

Nous gardons le montage du commutateur de premier exemple.
Une broche sur la ligne **GND** du breadboard.
L'autre sur la broche analogique **A5** de l'Arduino.

Et, nous
- Ajoutons une LED verte sur le breadboard.
- Relions la cathode (-) via une résistance de 220 Ω à la ligne **GND** du breadboard.
- Relions l'anode (+) à la broche digitale **7**.

5.7.2.2 Code

```
Sub Process_Globals
   Public Serial1 As Serial
   Private pinBoutton As Pin           'broche (pin) pour le bouton
   Private pinLEDVerte As Pin          'broche (pin) pour la Led verte
   Private Allumé = False As Boolean
End Sub
```

Nous gardons la définition de `pinBouton`.

Nous changeons la définition
`Private pinLED13 As Pin`
en
`Private pinLEDVerte As Pin`
Pour la LED verte.

Nous ajoutons une variable booléenne globale `Allumé` qui est True quand la LED est allumée.

```
Private Sub AppStart
   Serial1.Initialize(115200)

   pinBouton.Initialize(pinButton.A5, pinBouton.MODE_INPUT_PULLUP)
   'Utilisation d'une résistance pull up pour éviter que la broche soit flottante.
   pinBouton.AddListener("pinBouton_StateChanged")

   pinLEDVerte.Initialize(7, pinLEDVerte.MODE_OUTPUT)
End Sub
```

Nous gardons le code pour `pinBouton`.
Nous initialisons `pinLEDVerte` comme broche digitale **7** et définissons le mode de sortie à `pinLEDVerte.MODE_OUTPUT`.

```
Private Sub pinBouton_StateChanged (State As Boolean)
   Log("State: ", State)
   'State sera False lorsque le bouton est pressé à cause du mode PULLUP.
   If State = False Then
      Allumé = Not(Allumé)
      pinLEDVerte.DigitalWrite(Allumé)
   End If
End Sub
```

Chaque fois que la variable State est False, bouton poussoir pressé, nous changeons la variable `Allumé` et l'attribuons à la broche `pinLEDVerte`.

5.7.3 LedVerteSansRebond.b4r

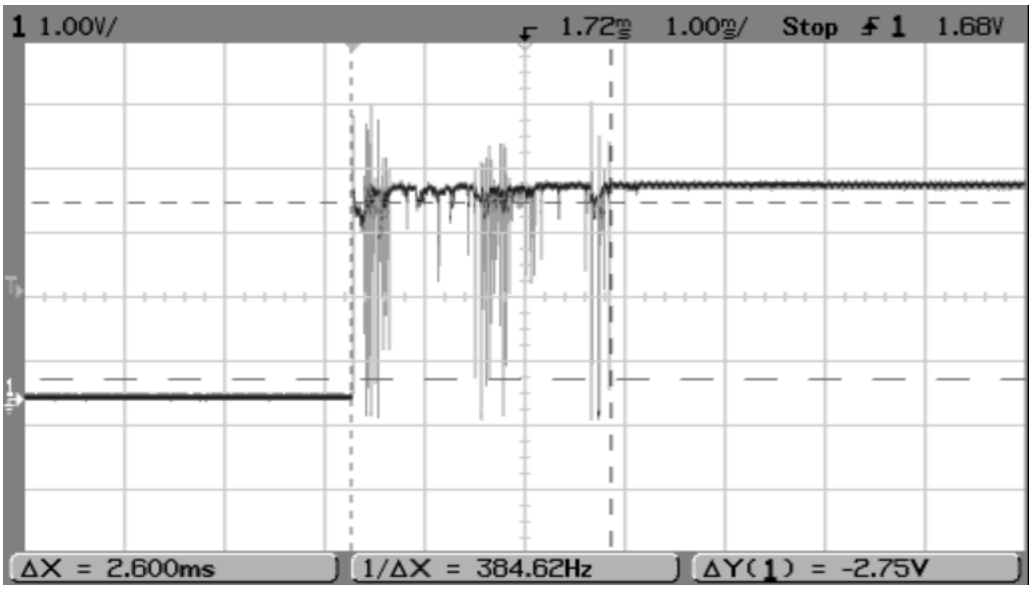
Pour ce projet nous utilisons exactement le même circuit que pour le projet LedVerte.b4r.

La seule différence se trouve dans le code.

Le projet LedVerteSansRebond.b4r est disponible dans le dossier CodesSource\B4R\ LedVerteSansRebond.

Le commutateur à bouton poussoir a un problème appelé rebond.
Le signal d'un commutateur mécanique n'est pas propre, le contact rebondit plusieurs fois ce qui est interprété comme plusieurs changements d'état. Si nous avons un nombre pair de changements d'état c'est comme si nous n'avions rien fait.
Mais nous ne voulons qu'un seul changement d'état par appui sur le bouton.

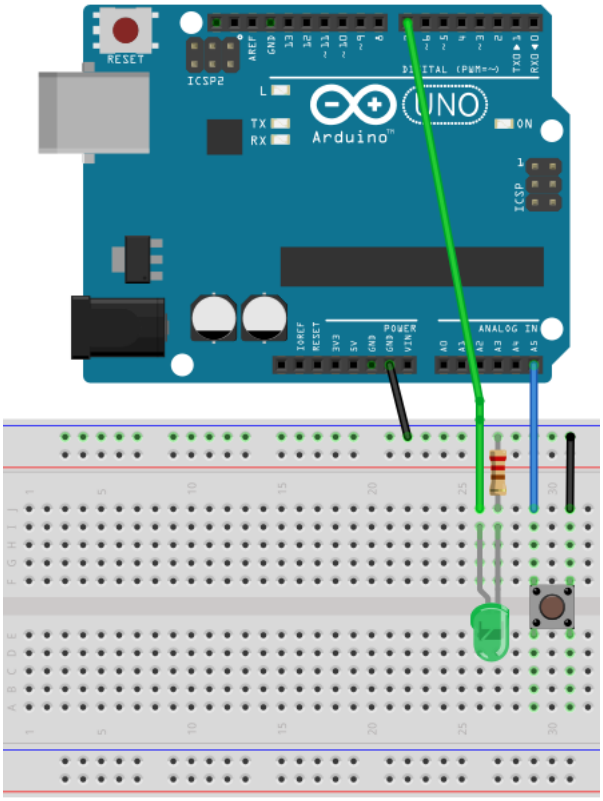
Image de rebonds d'un commutateur (source Wikipedia) :



Le commutateur rebondit plusieurs fois générant plusieurs changements d'état avant de rester dans un état stable.

Pour résoudre ce problème, nous ne réagissons pas, dans la routine `pinButton_StateChanged`, à des changements d'état pendant un temps donné (10 millisecondes).

5.7.3.1 Schéma



Matériel:

- 1 commutateur à bouton poussoir
- 1 LED verte
- 1 résistance 220 Ω

Nous gardons le montage du bouton poussoir du premier exemple.
Une broche liée à la ligne **GND** du circuit.
L'autre à la broche digitale **7** de l'Arduino.

Et nous
- ajoutons une LED verte sur le circuit.
- relions la cathode (-) à la ligne **GND** du circuit via une résistance de 220 Ω.
- relions l'anode (+) to digital pin **7**.

5.7.3.2 Code

Le code est pratiquement le même que LedGreen.b4r.

```
Sub Process_Globals
   Public Serial1 As Serial
   Private pinBouton As Pin            'broche (pin) pour le bouton
   Private pinLEDVerte As Pin          'broche (pin) pour la Led verte
   Private Allumé = False As Boolean
   Private RebondTemps As ULong
   Private RebondRetard = 10 As ULong
End Sub
```

Nous ajoutons deux nouvelles variables : `RebondTemps` et `RebondRetard`.

```
Private Sub AppStart
   Serial1.Initialize(115200)

   pinBouton.Initialize(pinBouton.A5, pinBouton.MODE_INPUT_PULLUP)
   'Utilisation d'une résistance pull up pour éviter que la broche soit flottante.
   pinBouton.AddListener("pinBouton_StateChanged")

   pinLEDVerte.Initialize(7, pinLEDVerte.MODE_OUTPUT)
End Sub
```

Même que LedVerte.b4r

Nouvelle routine `pinButton_StateChanged` :

```
Private Sub pinBouton_StateChanged (State As Boolean)
   Log("State: ", State)
   'State sera False lorsque le bouton est pressé à cause du mode PULLUP.
   If State = False Then
      If Millis - RebondTemps < RebondRetard Then
         Return            'Retour, dans le rebond !
      Else
         Allumé = Not(Allumé)
         pinLEDVerte.DigitalWrite(Allumé)
         RebondTemps = Millis  'remettre RebondTemps au temps actuel
      End If
   End If
End Sub
```

Chaque fois que State est False, bouton poussoir pressé :

- Nous comparons le temps entre le changement d'état actuel par rapport au premier changement d'état.
  Si le temps est plus court que `RebondRetard`, c'est un rebond, nous ne faisons rien.
  Si le temps est plus long que `RebondRetard`, c'est un changement d'état réel, nous exécutons le code.
- Nous modifions la variable `Allumé` et attribuons sa valeur à `pinLEDVerte`.
- Définissons un nouveau `RebondTemps`.

5.7.4 FeuxSignalisation.b4r

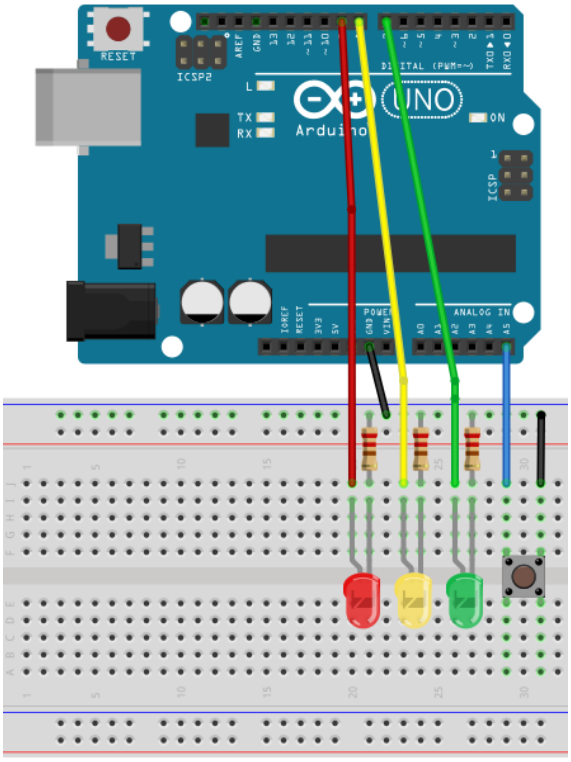
Ce projet est une evolution du projet LedVerteSansRebond et simule des feux de signalisation.

Nous utilisons une copie du projet LedVerteSansRebond.
Créez un nouveau dossier FeuxSignalisation, copiez-y tous les fichiers du projet LedVerteSansRebond et renommez les fichiers *LedVerteSansRebond .xxx* en *FeuxSignalisation.xxx*.

Les feux peuvent être actives ou désactivés avec le commutateur à bouton poussoir comme dans les projets précédents.
Le cycle rouge – vert et vert – rouge est géré par un Timer (minuteur) et le cycle jaune est régi par une routine appellee avec un retard correspondant à la duré de cycle jaune.

Le projet FeuxSignalisation.b4r est disponible dans le dossier CodesSource\B4R\ FeuxSignalisation.

5.7.4.1 Sketch



Matériel:

- 1 commutateur à bouton poussoir
- 1 LED verte
- 1 LED jaune
- 1 LED rouge
- 3 résistances de 220 Ω

Nous
- ajoutons une LED jaune et une rouge similaire à la verte.
- relions le (+) de la LED jaune à la broche digitale **8**.
- relions le (+) de la LED rouge à la broche digitale **9**.

5.7.4.2 Code

```
Sub Process_Globals
   Public Serial1 As Serial

   Public pinButton As Pin                              'broche pour le bouton
   Public pinLEDVert, pinLEDJaune, pinLEDRouge As Pin 'broches pour les LEDs
   Public TimerVertRouge As Timer
   Public FeuxON = False As Boolean
   Public LEDVerte = False As Boolean
   Public RebondTemps  As ULong
   Public RebondRetard  = 10 As ULong
End Sub
```

Nous déclarons le commutateur, les trois broches pour les LEDs, le Timer et qutre variables globales LightOn, LightGreen, RebondTemps et RebondRetard.

FeuxOn = False > Feux OFF
LEDVerte = True > LED verte ON

```
Private Sub AppStart
   Serial1.Initialize(115200)

   TimerVertRouge.Initialize("TimerVertRouge_Tick", 2000)

   'Utilisation de la résistance interne.
   pinButton.Initialize(pinButton.A5, pinButton.MODE_INPUT_PULLUP)
   pinButton.AddListener("pinButton_StateChanged")

   pinLEDVert.Initialize(7, pinLEDVert.MODE_OUTPUT)
   pinLEDJaune.Initialize(8, pinLEDJaune.MODE_OUTPUT)
   pinLEDRouge.Initialize(9, pinLEDRouge.MODE_OUTPUT)
End Sub
```

Nous initialisons TimerVertRouge avec le nom d'événement "TimerVertRouge" et un intervalley de 2000 ce qui signifie que l'événement Tick sera généré toutes les 2 secondes (2000 milli-secondes).
Nous gardons le code pour le bouton et la LED verte, et attribuons la broche digitale 8 en sortie à pinLEDJaune et attribuons la broche digitale 9 en sortie à pinLEDRouge.

Le code est, j'espère, auto-explicatif.

```
Private Sub pinButton_StateChanged (State As Boolean)
   Log("État: ", State)                          'Log la valeur de State (état)

   If State = False Then                         'si State = False
      If Millis - RebondTemps  < RebondRetard  Then
         Return
      Else
         pinLEDRouge.DigitalWrite(True)          'allume la LED rouge
         FeuxON = Not(FeuxON)                    'change la valeur de FeuxON
         RebondTemps  = Millis
         Log("Feux: ", FeuxON)                   'Log la valeur de FeuxON

         TimerVertRouge.Enabled = FeuxON         'active le Timer TimerVertRouge

         If FeuxON = False Then                  'si FeuxON = False
            pinLEDVert.DigitalWrite(False)       'éteint la LED verte
            pinLEDJaune.DigitalWrite(False)      'allume la LED jaune
            pinLEDRouge.DigitalWrite(False)      'éteint la LED rouge
         End If
      End If
   End If
End Sub

Private Sub TimerVertRouge_Tick
   If LEDVerte = True Then                       'si LEDVerte = True
      Log("TimerVertRouge_Tick LEDJaune ON")     'écrit le Log
      CallSubPlus("FinJaune", 500, 0)
      pinLEDVert.DigitalWrite(False)             'switch OFF LED Green
      pinLEDJaune.DigitalWrite(True)             'allume la LED jaune
      LEDVerte = False                           'met LEDVerte à False
   Else
      Log("TimerVertRouge_Tick LEDVerte ON")     'écrit le Log
      pinLEDRouge.DigitalWrite(False)            'éteint la LED rouge
      pinLEDVert.DigitalWrite(True)              'allume la LED verte
      LEDVerte = True                            'met LEDVerte à True
   End If
End Sub

Private Sub FinJaune(Tag As Byte)
   Log("LEDRouge ON")                            'écrit le Log
   Log(" ")
   pinLEDJaune.DigitalWrite(False)               'éteint la LED jaune
   pinLEDRouge.DigitalWrite(True)                'allume la LED rouge
End Sub
```

Nous utilisons la routine FinJaune pour passer du feu jaune au feu rouge.
Cette routine est appellee avec le mot clé CallSubPlus qui permet d'appeler la routine définie avec un retard donné.

```
CallSubPlus("FinJaune", 500, 0)
```

`FinJaune` = Nom de le routine
`500` = retard, 0.5 seconde (500 milli-secondes)
`0` = Tag, pas utilize dans notre cas.

Nous pouvons activer ou désactiver les feux avec le bouton commutateur.

5.8 Glossaire

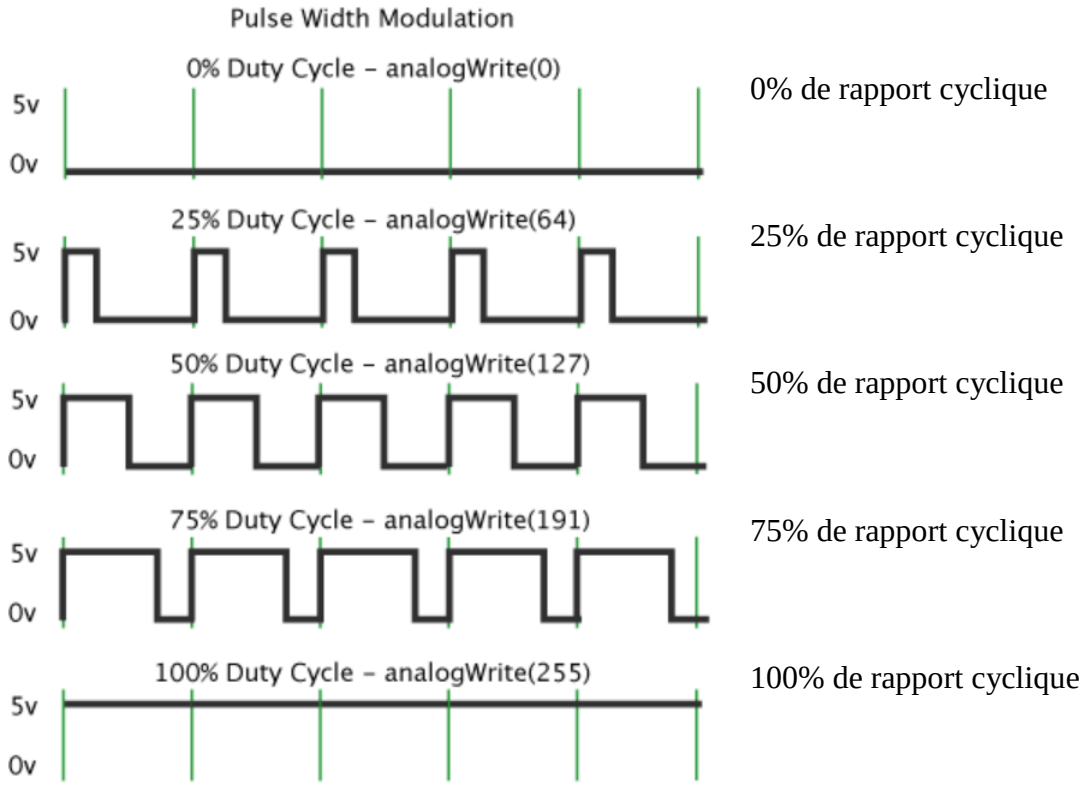
5.8.1 Bases d'électricité

Bases d'électricité (en anglais) : Electricity Basics.

5.8.2 PWM Pulse Width Modulation

Pulse Width Modulation, ou PWM (modulation de largeur d'impulsions), est une technique pour obtenir des résultats analogiques avec des moyens numériques. Un contrôle numérique est utilisé pour créer un signal rectangulaire, un signal qui est commute entre ON et OFF. Ce genre de commutations peut simuler des tensions entre 5 Volt (toujours ON) et 0 Volt (toujours OFF) en modifiant la durée pendant laquelle la tension est à 5 V par rapport à la durée pendant laquelle la tension est à 0 V, appelé le 'rapport cyclique'. La durée pendant laquelle le signal est à 5 V est appelée largeur d'impulsion (pulse width). Pour obtenir des valeurs analogiques variables, on modifie ou module cette largeur d'impulsion. Si on répète ce schéma de commutations suffisamment rapidement, avec une LED par exemple, le résultat est comme si on avait une tension stable entre 0 et 5V modulant la luminosité de la LED.

Dans le graphique ci-dessous, les lignes vertes représentent une périodicité régulière dans le temps. Cette durée ou période est l'inverse de la fréquence de modulation. En d'autres termes, avec une fréquence de modulation d'environ 500Hz, (celle des cartes Arduino), les lignes vertes auraient un espacement de 2 millisecondes. La fonction analogWrite() est basée sur 0 - 255, donc analogWrite(255) fournit un rapport cyclique de 100% (toujours ON), analogWrite(127) est à 50% de rapport cyclique (moitié du temps) et analogWrite(0) 0% donc toujours OFF.



0% de rapport cyclique

25% de rapport cyclique

50% de rapport cyclique

75% de rapport cyclique

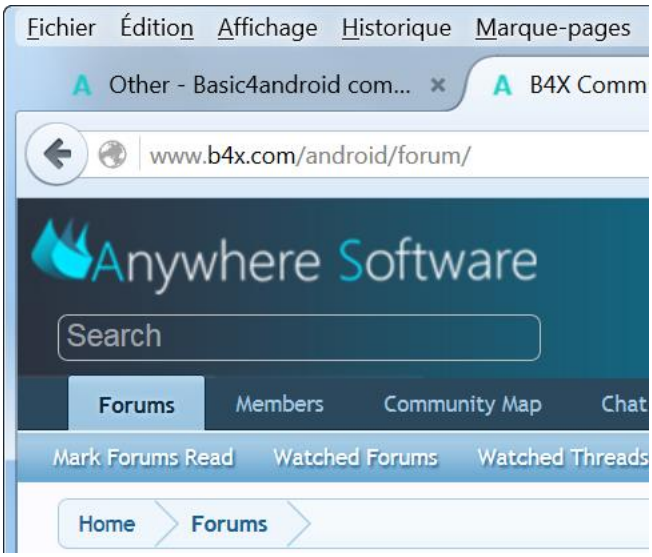
100% de rapport cyclique

Source Android Site Tutorials.

6 Outils d'aide

Les outils suivants sont utiles pour trouver des réponses à beaucoup de vos questions.

6.1 Fonction recherche dans le forum / Search

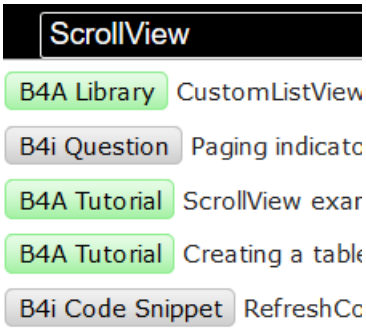


Dans le coin supérieur gauche vous trouvez le champ de recherche 'Search' pour le forum. Selon la largeur de la fenêtre, le champ 'Search' peut se trouver dans le coin supérieur droit.

Entrez une question ou un mot clé puis pressez 'Entrée'.

La fonction montre les posts qui correspondent à votre requête.

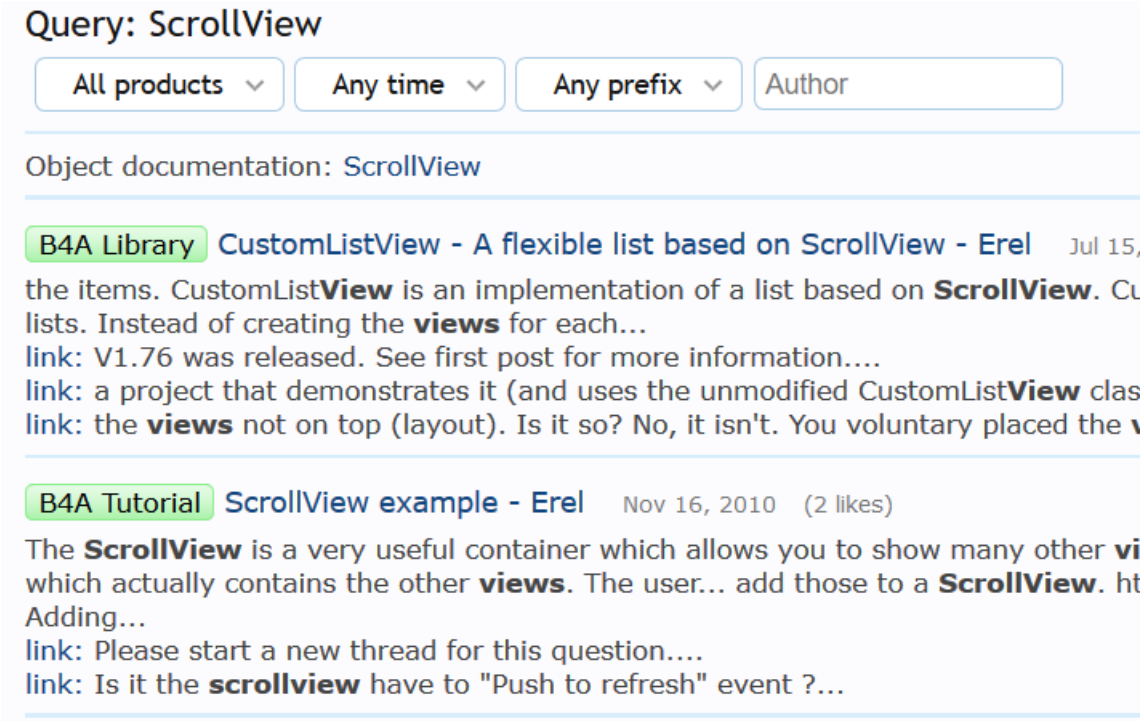
Exemple : Entrez le mot clé ScrollView :



Une liste de résultats est affichée juste en dessous du champ de recherche.

Cliquez sur un élément de la liste pour afficher le 'post' complet.

Et le résultat :



Sur le haut vous trouvez des boutons permettant un filtrage selon différents critères.

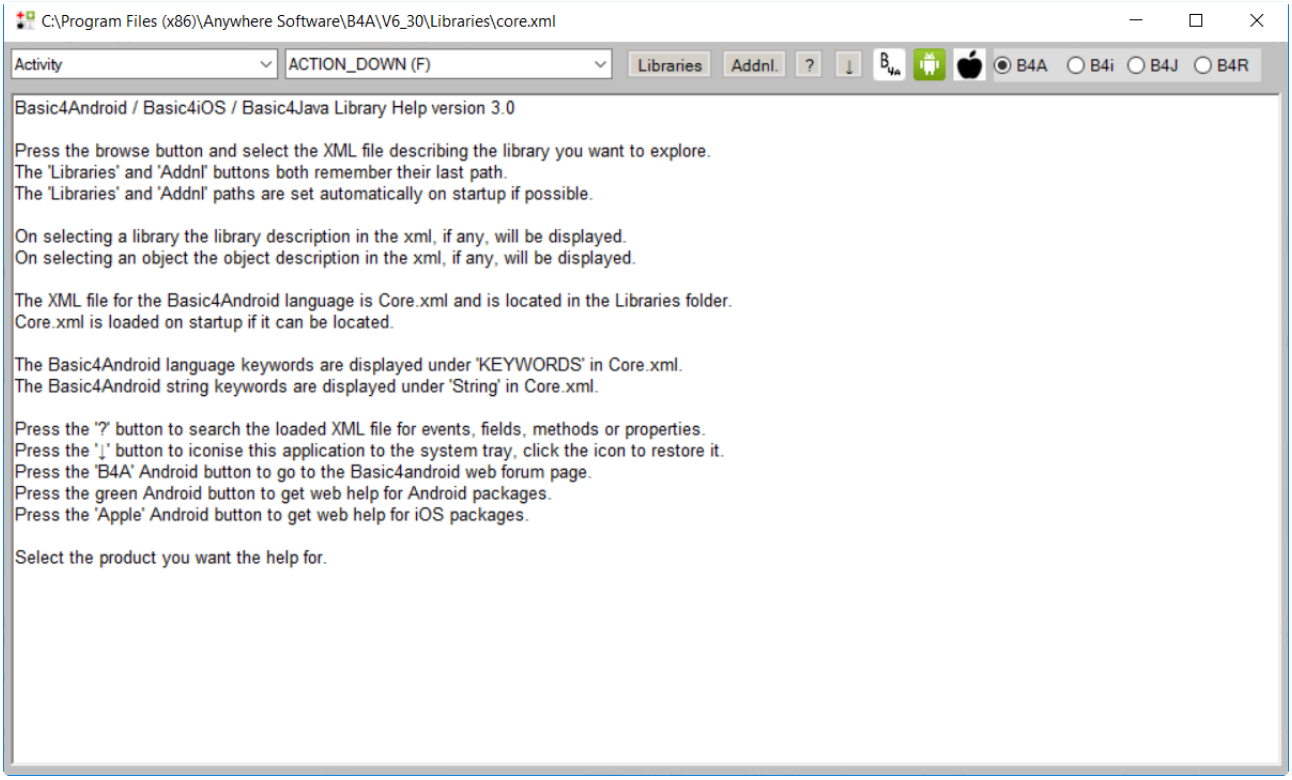
Cliquez sur le titre pour afficher le 'post' complet.

Exemple de filtrage sur un produit :

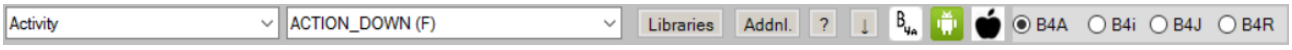
6.2 B4x Help Viewer

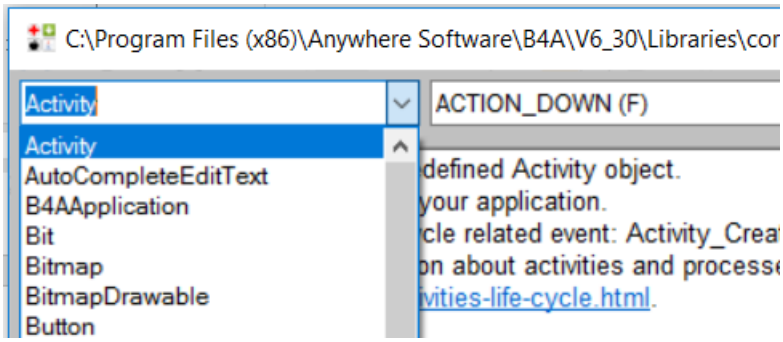
Ce programme affiche les fichiers d'aide xml. Il était écrit à l'origine par Andrew Graham (agraham) pour B4A. Je l'ai modifié, avec l'agrément d'Andrew, pour afficher les fichier xml de toutes les plateformes B4A, B4J, B4i et B4R.

Le programme peut être téléchargé depuis le forum.

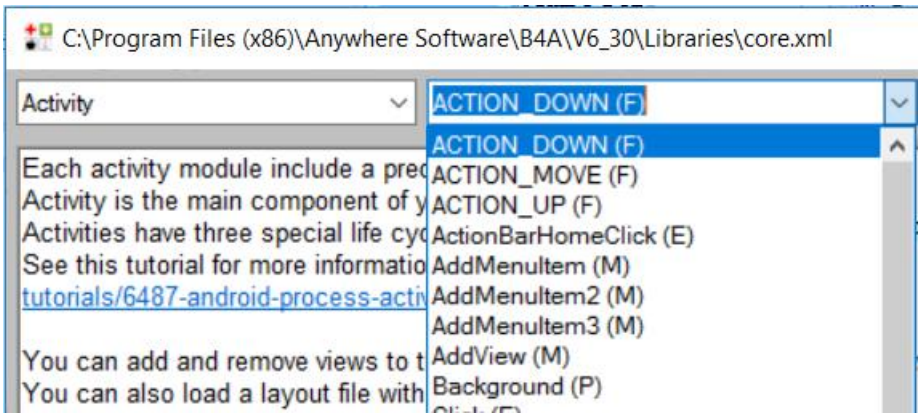


Sur le haut, on trouve :





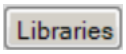
Dans le coin supérieur gauche se trouve une liste déroulante qui affiche les différents objets contenus dans la bibliothèque sélectionnée.



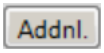
A côté de la liste des objets vous trouvez une autre liste déroulante avec

- méthodes(M)
- événements(E)
- propriétés(P)
- champs(F)

pour l'objet sélectionné.



Sélectionne les bibliothèques standards (celles livrées avec B4x).



Sélectionne les bibliothèques additionnelles.

Moteur de recherche pour trouver des objets avec des mots clé.

Ferme B4AHelp

Lance le forum 'Online Community'.

Lance le site Android Developers.

Lance le site iOS Developers.

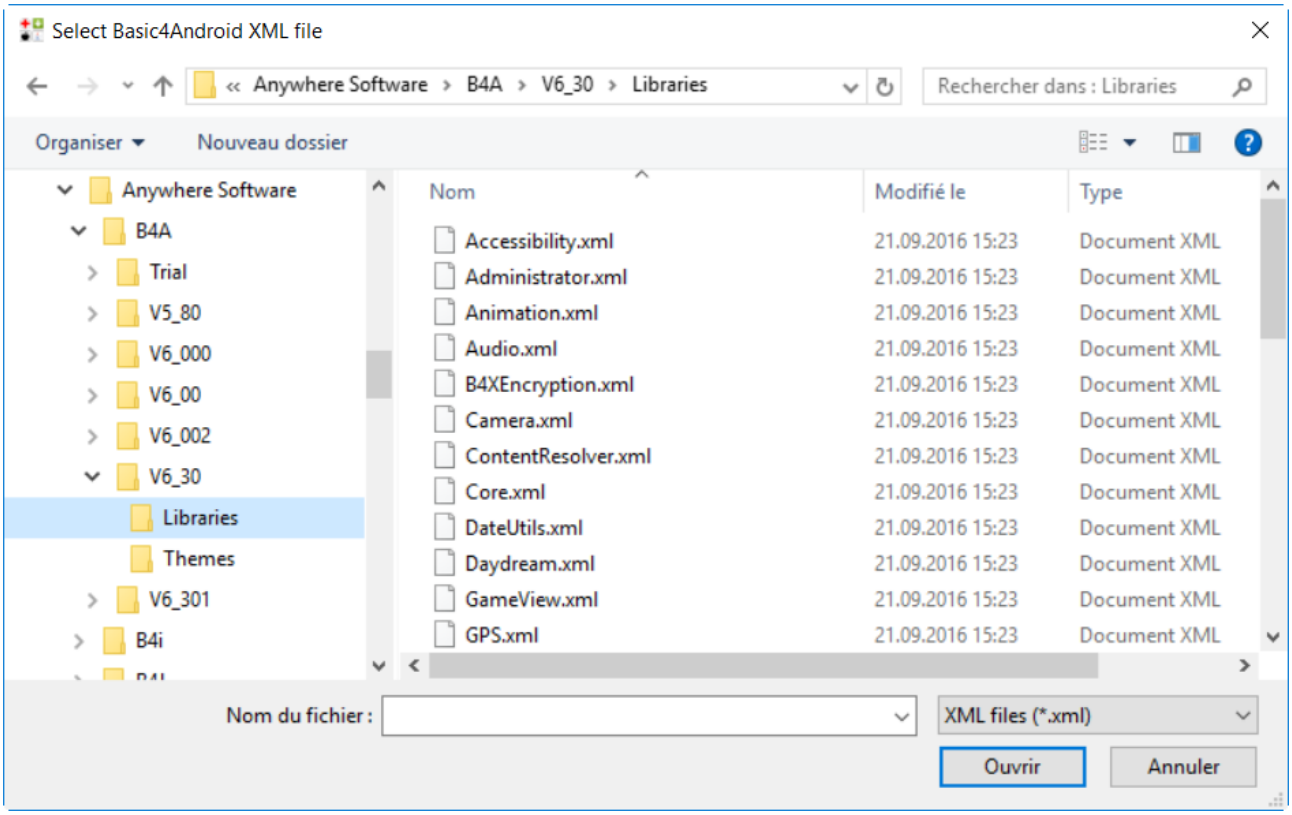
Fichiers d'aide B4A.

Fichiers d'aide B4i.

Fichiers d'aide B4J.

Fichiers d'aide B4R.

Libraries Bibliothèques standard.



Sélectionnez la bibliothèque et cliquez sur Ouvrir .

Ici « Anywhere Software › B4A › V6_30 › Libraries vous pouvez sélectionner le dossier dans lequel les bibliothèques standard sont enregistrées.

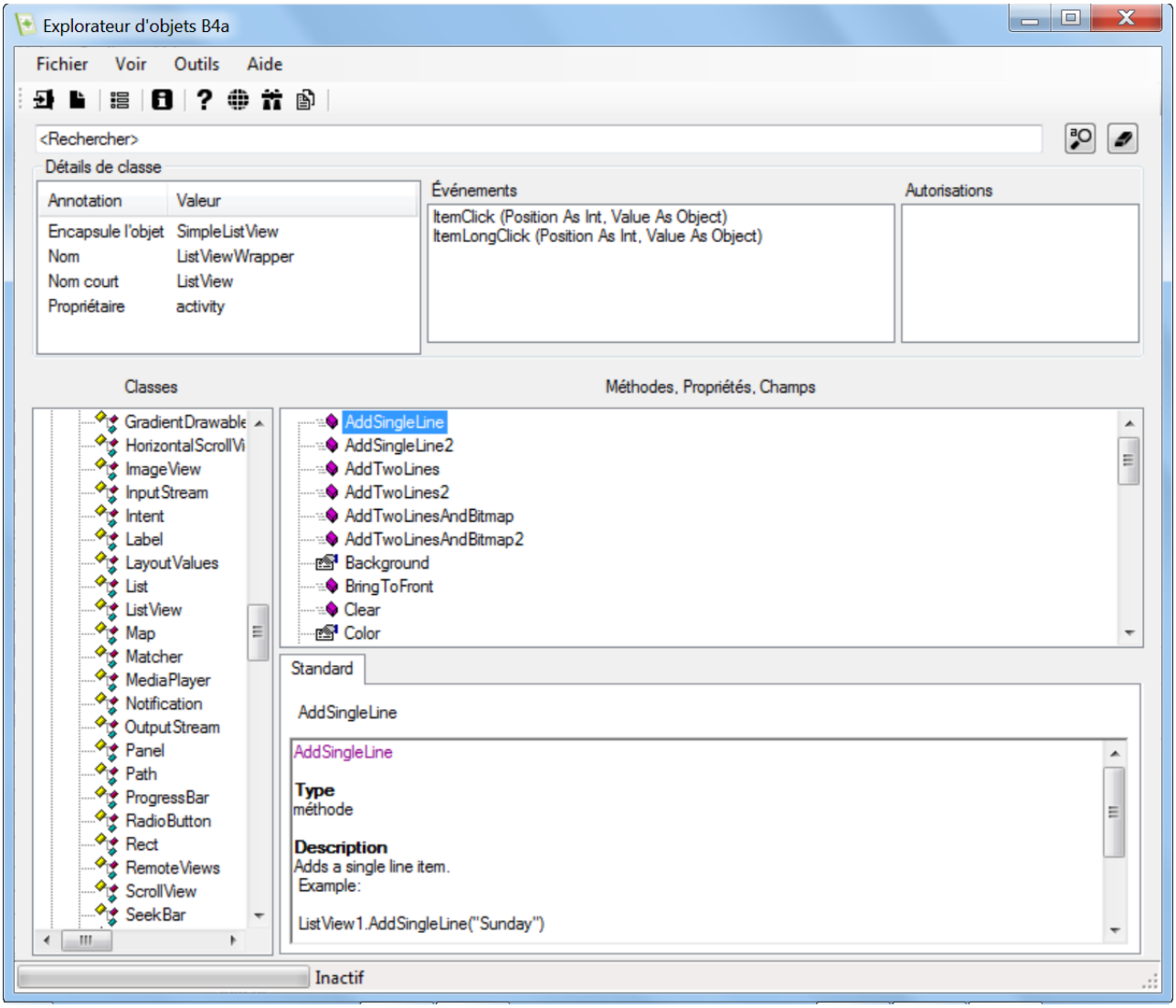
Une fois sélectionné, le nom du dossier est mémorisé pour les démarrages suivants du programme.

6.3 Help documentation - B4A Object Browser

Ceci est aussi un programme autonome Windows affichant les fichiers d'aide des bibliothèques.

Il a été écrit par Vader et peut être téléchargé ici.

Un mode d'emploi en pdf sur son utilisation fait partie du téléchargement.



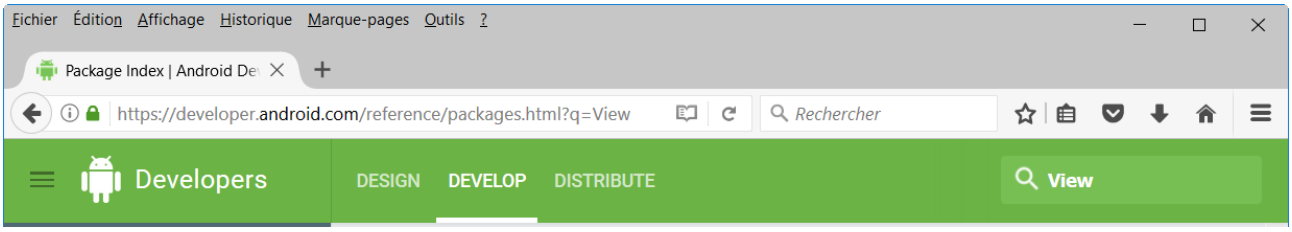
6.4 Liens utiles

6.4.1 B4A

Un lien utile, en anglais, pour des graphiques de mise en page.
Android cheat sheet for graphic designers

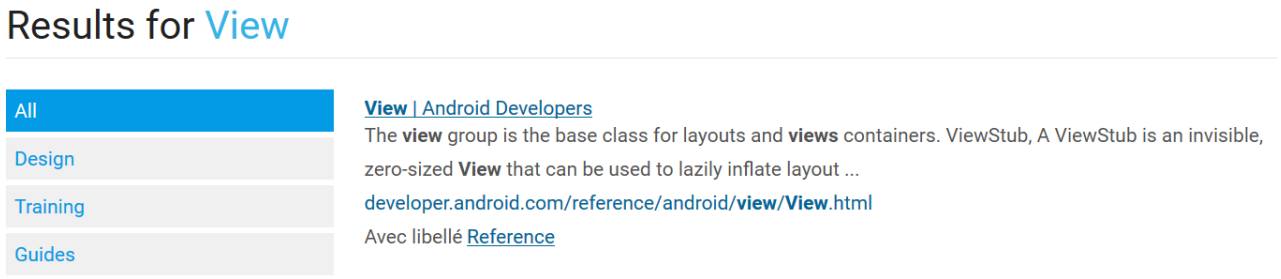
Android Developers. Design Develop Distribute

Android Developers recherche pour une requête.



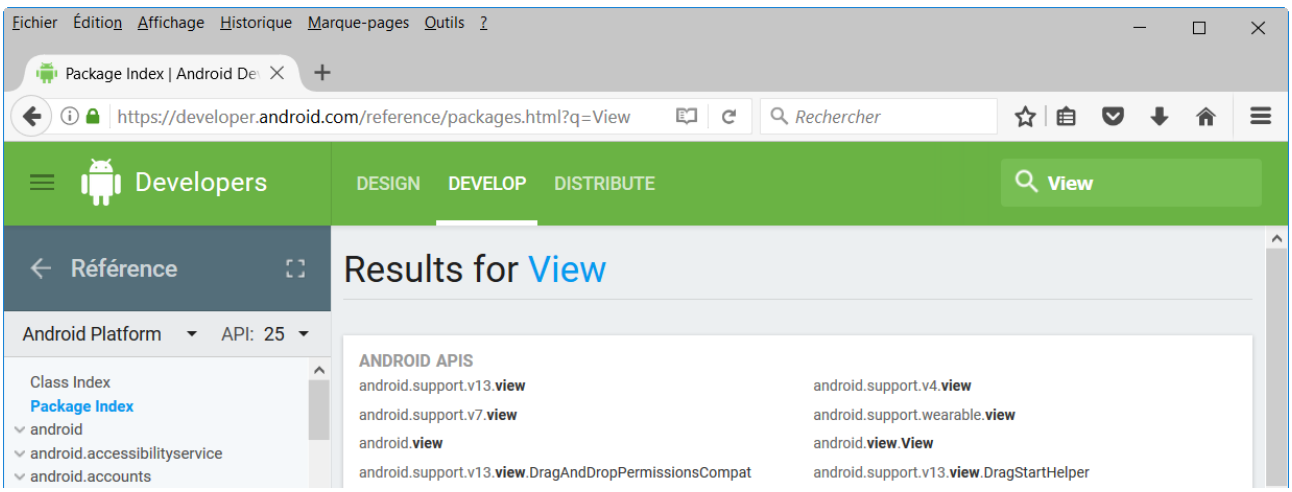
Dans le coin supérieur droit vous trouvez le champ de recherche.

Enter *View* in the field : 



Cliquez sur le lien View | Android Developers.

Et vous obtenez toutes les informations sur l'objet View.

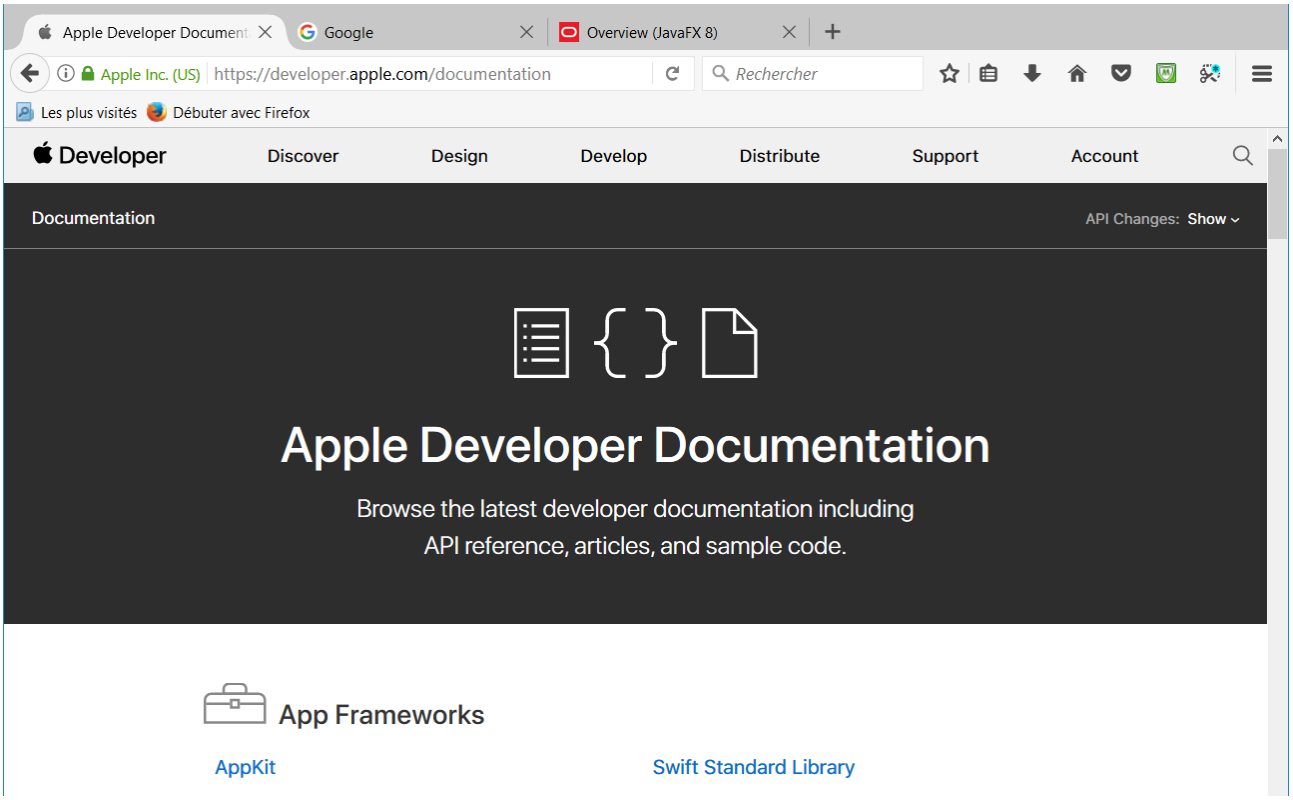


6.4.2 B4i

iOS developers :

https://developer.apple.com/ios/human-interface-guidelines/overview/themes/

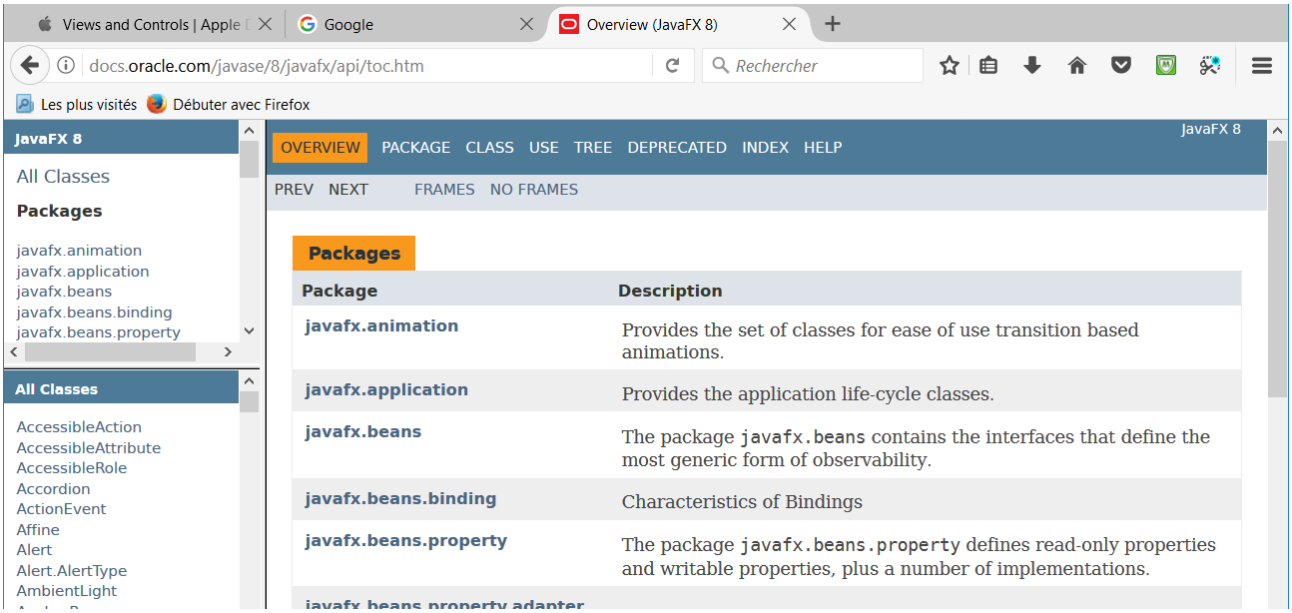
C'est le site de Apple avec toutes les informations sur iOS, en anglais.



6.4.3 B4J

Oracle Overview JavaFX API :

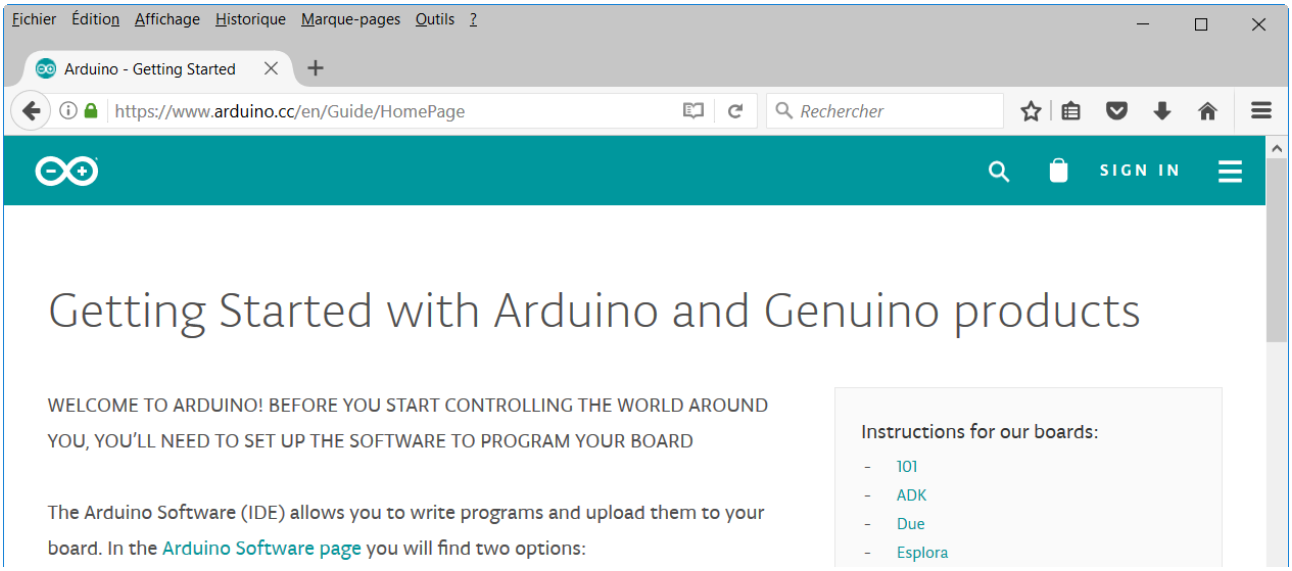
http://docs.oracle.com/javase/8/javafx/api/toc.htm



6.4.4 B4R

Page d'accueil Arduino :

https://www.arduino.cc/en/Guide/HomePage



6.5 Livres

Livre B4A (en anglais)

Écrit par Philip Brown sous le pseudo Wyken Seagrave.

http://pennypress.co.uk/b4a-book/

MagBook Build your own Android App (en anglais).

Écrit par Nigel Whitfield.

http://www.magbooks.com/product/build-your-own-android-app/

7 Dictionnaire

- **Activity** Dans B4A objet activité, équivalent à un écran.
- **EDI** **E**nvironnement de **D**éveloppement **I**ntégré, l'éditeur de B4x.
  IDE en anglais **I**ntegrated **D**evelopment **E**nvironment
- **EditText** Objet d'interface homme-machine. Permet à l'utilisateur d'éditer du texte.
- **Height** Propriété Hauteur, hauteur d'une view.
- **IHM** **I**nterface **H**omme – **M**achine
  UI en anglais **U**ser **I**nterface
- **Label** Objet d'interface homme-machine : étiquette. Visualise du texte.
- **LED** **L**ight **E**mitting **D**iod Diode Electo-Luminescente DEL.
- **Left** Propriété Gauche, position du bord gauche d'une view.
- **Node** Dans B4J, objet d'interface homme-machine générique.
- **Panel** Dans B4A et B4i, objet d'interface homme-machine: panneau.
- **Pane** Dans B4J, objet d'interface homme-machine : panneau.
- **PWM** **P**ulse **W**idth **M**odulation modulation de largeur d'impulsion.
- **Top** Propriété Haut, position du bord supérieur d'une view.
- **TextColor** Propriété couleur de texte.
- **TextSize** Propriété grandeur de texte.
- **TypeFace** Propriété police de caractère.
- **View** Dans B4A et B4i, objet d'interface homme-machine générique.
- **Width** Propriété Largeur, largeur d'une view.
- **WYSIWYG** **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et (ce que vous voyez est ce que vous obtenez)
  L'utilisateur voit directement à quoi ressemblera le résultat final.