📱 Tutoriel Complet : Transformer une Application Angular en Application Android

📋 Table des Matières

1. Prérequis
2. Création de l'Application Angular
3. Installation de Capacitor
4. Configuration Android
5. Préparation des Assets
6. Build et Génération APK/AAB
7. Signature de l'Application
8. Publication sur GitHub
9. Déploiement Web (Bonus)

🔧 Prérequis

Logiciels Nécessaires

1. **Node.js et npm**

```
# Télécharger depuis https://nodejs.org/
# Vérifier l'installation
node --version
npm --version
```

2. **Angular CLI**

```
npm install -g @angular/cli
ng version
```

3. **Android Studio**

   - Télécharger depuis https://developer.android.com/studio
   - Installer les composants :
     - Android SDK
     - Android SDK Platform
     - Android Virtual Device (pour émulateur)

4. **Java Development Kit (JDK)**

```
# JDK 11 ou supérieur
# Télécharger depuis https://adoptium.net/
java -version
```

5. **Variables d'Environnement Windows**

```
# Ajouter dans les variables système :
ANDROID_HOME = C:\Users\VotreNom\AppData\Local\Android\Sdk
JAVA_HOME = C:\Program Files\Eclipse Adoptium\jdk-11.0.xx

# Ajouter au PATH :
%ANDROID_HOME%\platform-tools
%ANDROID_HOME%\tools
%JAVA_HOME%\bin
```

🎨 Création de l'Application Angular

1. Initialiser le Projet Angular

```
# Créer un nouveau projet Angular
ng new flashanzan-app

# Options recommandées :
# - Would you like to add Angular routing? Yes
# - Which stylesheet format would you like to use? CSS

# Naviguer dans le dossier
cd flashanzan-app
```

2. Structure du Projet Angular

```
flashanzan-app/
├── src/
│   ├── app/
│   │   ├── game/                    # Composant principal du jeu
│   │   ├── services/                # Services (traduction, etc.)
│   │   ├── app.component.ts
│   │   └── app.routes.ts
│   ├── assets/                      # Images, sons, etc.
│   ├── index.html
│   └── main.ts
├── angular.json                     # Configuration Angular
├── package.json
└── tsconfig.json
```

3. Développer l'Application

```
# Lancer le serveur de développement
ng serve

# L'application sera disponible sur http://localhost:4200
```

4. Installer les Dépendances Nécessaires

```
# RxJS pour la programmation réactive (déjà inclus)
npm install rxjs

# Autres dépendances selon vos besoins
npm install @angular/animations
```

Installation de Capacitor

1. Installer Capacitor

```
# Installer Capacitor Core et CLI
npm install @capacitor/core
npm install --save-dev @capacitor/cli

# Initialiser Capacitor
npx cap init
```

Questions lors de l'initialisation :

```
? App name: FlashAnzan
? App Package ID: com.azmicro.app
? Web asset directory (default is 'www'): dist/flashanzan-app
```

2. Installer la Plateforme Android

```
# Installer le package Android de Capacitor
npm install @capacitor/android

# Ajouter la plateforme Android au projet
npx cap add android
```

3. Fichiers de Configuration Créés

`capacitor.config.json`

```
{
  "appId": "com.azmicro.app",
  "appName": "FlashAnzan",
  "webDir": "dist/flashanzan-app",
  "bundledWebRuntime": false,
  "server": {
    "androidScheme": "https"
  }
}
```

`capacitor.config.ts` (alternative TypeScript)

```
import { CapacitorConfig } from '@capacitor/cli';

const config: CapacitorConfig = {
  appId: 'com.azmicro.app',
  appName: 'FlashAnzan',
  webDir: 'dist/flashanzan-app',
  server: {
    androidScheme: 'https'
  }
};

export default config;
```

Configuration Android

1. Modifier le Nom de l'Application

Fichier : `android/app/src/main/res/values/strings.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">FlashAnzan</string>
    <string name="title_activity_main">FlashAnzan</string>
    <string name="package_name">com.azmicro.app</string>
    <string name="custom_url_scheme">com.azmicro.app</string>
</resources>
```

2. Configuration du Build Gradle

Fichier : `android/app/build.gradle`

```
android {
    namespace "com.azmicro.app"
    compileSdk 34

    defaultConfig {
        applicationId "com.azmicro.app"
        minSdk 22
        targetSdk 34
        versionCode 5
        versionName "1.1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'

            // Configuration pour la signature
            signingConfig signingConfigs.release
        }
    }
}

dependencies {
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'androidx.coordinatorlayout:coordinatorlayout:1.2.0'
    implementation 'androidx.core:core-splashscreen:1.0.1'
}
```

3. Configuration Gradle du Projet

Fichier : `android/build.gradle`

```
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:8.2.1'
        classpath 'com.google.gms:google-services:4.4.0'
    }
}

allprojects {
    repositories {
        google()
        mavenCentral()
    }
}
```

4. Permissions Android

Fichier : `android/app/src/main/AndroidManifest.xml`

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

    <!-- Permissions si nécessaires -->
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:launchMode="singleTask"
            android:theme="@style/AppTheme.NoActionBarLaunch">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

🎨 Préparation des Assets

1. Créer l'Icône de l'Application

Dimensions requises pour l'icône principale :

- **512x512 pixels** (pour le Play Store)
- Format : PNG avec fond transparent ou solide

2. Générer les Icônes Android

Utiliser **icon.kitchen** ou un outil similaire :

1. Aller sur https://icon.kitchen/
2. Uploader votre icône 512x512
3. Télécharger le package Android
4. Extraire dans `android/app/src/main/res/`

3. Script PowerShell pour Générer les Icônes

Fichier : `update-icons.ps1`

```powershell
# Script pour générer toutes les tailles d'icônes Android depuis une image 512x512

Add-Type -AssemblyName System.Drawing

$sourceIcon = "android\app\src\main\res\play_store_512.png"
$densities = @{
    "mdpi"    = 48
    "hdpi"    = 72
    "xhdpi"   = 96
    "xxhdpi"  = 144
    "xxxhdpi" = 192
}

foreach ($density in $densities.GetEnumerator()) {
    $folder = "android\app\src\main\res\mipmap-$($density.Key)"
    $size = $density.Value

    Write-Host "Generating icons for $($density.Key) ($size x $size)..."

    # Créer le dossier s'il n'existe pas
    if (-not (Test-Path $folder)) {
        New-Item -ItemType Directory -Path $folder -Force | Out-Null
    }

    # Charger l'image source
    $image = [System.Drawing.Image]::FromFile((Resolve-Path $sourceIcon))

    # Créer une nouvelle image redimensionnée
    $resized = New-Object System.Drawing.Bitmap($size, $size)
    $graphics = [System.Drawing.Graphics]::FromImage($resized)
    $graphics.InterpolationMode = [System.Drawing.Drawing2D.InterpolationMode]::HighQualityBicubic
    $graphics.DrawImage($image, 0, 0, $size, $size)

    # Sauvegarder les différentes versions
    $resized.Save("$folder\ic_launcher.png", [System.Drawing.Imaging.ImageFormat]::Png)
    $resized.Save("$folder\ic_launcher_round.png", [System.Drawing.Imaging.ImageFormat]::Png)
    $resized.Save("$folder\ic_launcher_foreground.png", [System.Drawing.Imaging.ImageFormat]::Png)

    $graphics.Dispose()
    $resized.Dispose()
    $image.Dispose()
}

Write-Host "☑ All icons generated successfully!" -ForegroundColor Green
```

Exécution :

```powershell
# Donner les permissions d'exécution
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

# Exécuter le script
.\update-icons.ps1
```

4. Structure des Icônes

```
android/app/src/main/res/
├── mipmap-mdpi/
│   ├── ic_launcher.png (48x48)
│   ├── ic_launcher_round.png (48x48)
│   └── ic_launcher_foreground.png (48x48)
├── mipmap-hdpi/
│   ├── ic_launcher.png (72x72)
│   ├── ic_launcher_round.png (72x72)
│   └── ic_launcher_foreground.png (72x72)
├── mipmap-xhdpi/
│   ├── ic_launcher.png (96x96)
│   ├── ic_launcher_round.png (96x96)
│   └── ic_launcher_foreground.png (96x96)
├── mipmap-xxhdpi/
│   ├── ic_launcher.png (144x144)
│   ├── ic_launcher_round.png (144x144)
│   └── ic_launcher_foreground.png (144x144)
├── mipmap-xxxhdpi/
│   ├── ic_launcher.png (192x192)
│   ├── ic_launcher_round.png (192x192)
│   └── ic_launcher_foreground.png (192x192)
└── play_store_512.png (512x512)
```

🏗 Build et Génération APK/AAB

1. Build de l'Application Angular

```bash
# Build de production
npm run build

# ou avec Angular CLI
ng build --configuration production
```

Configuration dans `angular.json` :

```
{
  "projects": {
    "flashanzan-app": {
      "architect": {
        "build": {
          "configurations": {
            "production": {
              "budgets": [
                {
                  "type": "initial",
                  "maximumWarning": "500kb",
                  "maximumError": "1mb"
                }
              ],
              "outputHashing": "all",
              "optimization": true,
              "sourceMap": false,
              "namedChunks": false,
              "aot": true,
              "extractLicenses": true,
              "vendorChunk": false,
              "buildOptimizer": true
            }
          }
        }
      }
    }
  }
}
```

2. Synchroniser avec Capacitor

```
# Copier les fichiers web vers Android
npx cap sync android

# ou séparément
npx cap copy android
npx cap update android
```

3. Ouvrir dans Android Studio

```
# Ouvrir le projet Android dans Android Studio
npx cap open android
```

4. Générer un APK de Debug

Via Gradle (PowerShell) :

```
cd android
.\gradlew assembleDebug
```

Fichier généré :

```
android/app/build/outputs/apk/debug/app-debug.apk
```

5. Tester l'APK sur un Appareil

```
# Connecter votre téléphone en USB (avec débogage USB activé)
# Vérifier la connexion
adb devices

# Installer l'APK
adb install android/app/build/outputs/apk/debug/app-debug.apk
```

Signature de l'Application

1. Créer un Keystore

```
# Naviguer vers le dossier android
cd android

# Créer le keystore
keytool -genkey -v -keystore flashanzan-release.keystore -alias flashanzan-key -keyalg RSA -keysize 2048 -validity 10000

# Répondre aux questions :
# - Mot de passe du keystore : [CRÉER UN MOT DE PASSE FORT]
# - Nom et prénom : Azmicro
# - Unité organisationnelle : Development
# - Organisation : Azmicro
# - Ville : Votre Ville
# - État : Votre État
# - Code pays : MA (par exemple)
```

⚠ IMPORTANT : Sauvegarder le Keystore !

```
# Copier le keystore dans plusieurs endroits sécurisés :
# - Disque dur externe
# - Cloud (Google Drive, Dropbox)
# - Clé USB

# Le mot de passe et le keystore sont IRREMPLAÇABLES !
```

2. Configurer Gradle pour la Signature

Créer : `android/key.properties`

```
storePassword=VotreMotDePasse
keyPassword=VotreMotDePasse
keyAlias=flashanzan-key
storeFile=flashanzan-release.keystore
```

⚠ Ajouter à `.gitignore` :

```
# Ne JAMAIS committer ces fichiers !
android/key.properties
android/*.keystore
android/*.jks
```

3. Modifier `android/app/build.gradle`

```
android {
    // ... autres configurations ...

    signingConfigs {
        release {
            // Charger les propriétés du keystore
            def keystorePropertiesFile = rootProject.file("key.properties")
            def keystoreProperties = new Properties()

            if (keystorePropertiesFile.exists()) {
                keystoreProperties.load(new FileInputStream(keystorePropertiesFile))

                keyAlias keystoreProperties['keyAlias']
                keyPassword keystoreProperties['keyPassword']
                storeFile file(keystoreProperties['storeFile'])
                storePassword keystoreProperties['storePassword']
            }
        }
    }

    buildTypes {
        release {
            signingConfig signingConfigs.release
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}
```

4. Générer un AAB Signé (Pour Google Play Store)

```
# Naviguer vers le dossier android
cd android

# Nettoyer le build précédent
.\gradlew clean

# Générer le AAB signé
.\gradlew bundleRelease

# Fichier généré :
# android/app/build/outputs/bundle/release/app-release.aab
```

5. Générer un APK Signé (Pour Distribution Directe)

```
# Générer l'APK signé
.\gradlew assembleRelease

# Fichier généré :
# android/app/build/outputs/apk/release/app-release.apk
```

6. Vérifier la Signature

```
# Vérifier la signature de l'APK
jarsigner -verify -verbose -certs android/app/build/outputs/apk/release/app-release.apk

# Voir les détails du certificat
keytool -list -v -keystore android/flashanzan-release.keystore -alias flashanzan-key
```

Publication sur GitHub

1. Initialiser Git

```
# Initialiser le dépôt Git
git init

# Vérifier le status
git status
```

2. Créer le Fichier `.gitignore`

Fichier : `.gitignore`

```
# Node modules
node_modules/
npm-debug.log
yarn-error.log

# Build outputs
dist/
www/
*.log

# Android
android/app/build/
android/app/release/
android/app/debug/
android/.gradle/
android/gradle/
android/local.properties
android/*.keystore
android/*.jks
android/key.properties

# IDE
.idea/
.vscode/
*.swp
*.swo
*.swn
.DS_Store

# Environment
.env
.env.local
.env.production

# Capacitor
android/.idea/
android/.gradle/
android/build/
ios/App/Pods/
ios/App/build/

# TypeScript
*.tsbuildinfo
```

3. Créer un Dépôt sur GitHub

1. Aller sur https://github.com
2. Cliquer sur "New repository"
3. Nom : `flashanzan`
4. Description : "Application de calcul mental Flash Anzan"
5. Public ou Private
6. Ne pas initialiser avec README (on le fait localement)

4. Configurer Git et Faire le Premier Commit

```
# Configurer Git (si pas déjà fait)
git config --global user.name "Votre Nom"
git config --global user.email "votre.email@example.com"

# Ajouter tous les fichiers
git add .

# Premier commit
git commit -m "🎉 Initial commit: FlashAnzan Angular + Android app"

# Ajouter le remote GitHub
git remote add origin https://github.com/votre-username/flashanzan.git

# Pousser vers GitHub
git branch -M main
git push -u origin main
```

5. Créer des Commits Organisés

```
# Commit pour une nouvelle fonctionnalité
git add .
git commit -m "✨ feat: Add floating bottom navigation bar"
git push

# Commit pour une correction
git add .
git commit -m "🔧 fix: Correct app name in strings.xml"
git push

# Commit pour une mise à jour de version
git add android/app/build.gradle
git commit -m "📦 release: Version 1.1.0 with new features"
git push
```

6. Créer un README.md

Fichier : `README.md`

```
# 🧮 FlashAnzan - Application de Calcul Mental

Application mobile de calcul mental basée sur la méthode japonaise Anzan.

## 📱 Fonctionnalités

- Calcul mental avec nombres qui défilent
- Mode personnalisable (vitesse, quantité, longueur)
- Système de scores et badges
- Support multilingue (Français, Anglais, Arabe)
- Interface moderne et responsive

## 🚀 Technologies

- **Frontend:** Angular 14.2.0
- **Mobile:** Capacitor 7.4.4
- **Platform:** Android (API 22+)
- **Build:** Gradle 8.2.1

## 🛠️ Installation

```bash
# Cloner le dépôt
git clone https://github.com/votre-username/flashanzan.git
cd flashanzan

# Installer les dépendances
npm install

# Lancer en développement
ng serve
```

📦 Build Android

```bash
# Build Angular
npm run build

# Sync Capacitor
npx cap sync android

# Build APK/AAB
cd android
.\gradlew bundleRelease
```

📄 Licence

MIT License

👤 Développeur

Azmicro - aissatahri81@gmail.com
```

```
### 7. Créer des Branches pour les Fonctionnalités

```powershell
# Créer une nouvelle branche pour une fonctionnalité
git checkout -b feature/new-game-mode
# ... faire des modifications ...
git add .
git commit -m "✨ feat: Add new game mode"
git push -u origin feature/new-game-mode

# Retourner sur main
git checkout main

# Merger la branche (après revue)
git merge feature/new-game-mode
git push
```

8. Créer des Tags pour les Versions

```
# Créer un tag pour une version
git tag -a v1.1.0 -m "Version 1.1.0 - Nouvelle interface"
git push origin v1.1.0

# Lister les tags
git tag

# Voir les détails d'un tag
git show v1.1.0
```

🌐 Déploiement Web (Bonus)

1. Déploiement sur Vercel

Installation :

```
npm install -g vercel
```

Configuration : `vercel.json`
```

```
{
  "version": 2,
  "buildCommand": "npm run build",
  "outputDirectory": "dist/flashanzan-app",
  "framework": "angular",
  "routes": [
    {
      "src": "/assets/(.*)",
      "dest": "/assets/$1"
    },
    {
      "src": "/(.*\\.(js|css|png|jpg|jpeg|gif|svg|ico|json))",
      "dest": "/$1"
    },
    {
      "src": "/(.*)",
      "dest": "/index.html"
    }
  ]
}
```

Déploiement :

```
# Se connecter à Vercel
vercel login

# Déployer
vercel

# Déployer en production
vercel --prod
```

2. Déploiement sur GitHub Pages

Installation :

```
npm install -g angular-cli-ghpages
```

Build et Déploiement :

```
# Build avec base-href
ng build --configuration production --base-href "/flashanzan/"

# Déployer sur GitHub Pages
npx angular-cli-ghpages --dir=dist/flashanzan-app
```

3. Page de Politique de Confidentialité

Créer : `privacy-policy.html`

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Politique de Confidentialité - FlashAnzan</title>
</head>
<body>
    <h1>Politique de Confidentialité</h1>
    <p>FlashAnzan ne collecte aucune donnée personnelle.</p>
    <p>L'application fonctionne entièrement hors ligne.</p>
    <p>Contact : aissatahri81@gmail.com</p>
</body>
</html>
```

Déployer sur GitHub Pages :

```
# Créer un dossier docs
mkdir docs
cp privacy-policy.html docs/index.html

# Commit et push
git add docs/
git commit -m "📄 docs: Add privacy policy"
git push

# Activer GitHub Pages dans les settings du dépôt
# Settings > Pages > Source: main branch, /docs folder
```

📋 Checklist Complète

☑ Phase 1 : Développement

- [ ] Créer le projet Angular
- [ ] Développer les composants et services
- [ ] Tester localement avec `ng serve`
- [ ] Optimiser les performances

☑ Phase 2 : Configuration Mobile

- [ ] Installer Capacitor
- [ ] Ajouter la plateforme Android
- [ ] Configurer `capacitor.config.json`
- [ ] Modifier `strings.xml` et `build.gradle`

☑ Phase 3 : Assets et Design

- [ ] Créer l'icône 512x512
- [ ] Générer toutes les densités d'icônes
- [ ] Créer les screenshots pour le Play Store
- [ ] Tester l'affichage sur différents appareils

☑ Phase 4 : Build et Signature

- [ ] Créer le keystore de release
- [ ] Sauvegarder le keystore en lieu sûr
- [ ] Configurer `key.properties`
- [ ] Générer l'AAB signé
- [ ] Tester l'APK sur un appareil réel

☑ Phase 5 : GitHub

- [ ] Créer le dépôt GitHub
- [ ] Configurer `.gitignore`
- [ ] Faire le commit initial
- [ ] Créer un README complet
- [ ] Pousser le code vers GitHub

☑ Phase 6 : Play Store (Optionnel)

- [ ] Créer un compte développeur Google Play
- [ ] Préparer la fiche du Play Store
- [ ] Uploader l'AAB
- [ ] Soumettre pour revue

🆘 Résolution des Problèmes Courants

Problème : "SDK location not found"

```
# Créer le fichier local.properties
echo "sdk.dir=C:\\Users\\VotreNom\\AppData\\Local\\Android\\Sdk" > android/local.properties
```

Problème : "Execution failed for task ':app:processDebugResources'"

```
# Nettoyer et rebuild
cd android
.\gradlew clean
.\gradlew build
```

Problème : Port 4200 déjà utilisé

```
# Trouver le processus
netstat -ano | findstr :4200

# Tuer le processus
taskkill /PID [PID_NUMBER] /F

# ou utiliser un autre port
ng serve --port 4300
```

Problème : Icônes manquantes

```
# Réexécuter le script de génération d'icônes
.\update-icons.ps1

# Puis synchroniser
npx cap sync android
```

Problème : Erreur de signature

```
# Vérifier que key.properties existe
ls android/key.properties

# Vérifier que le keystore existe
ls android/flashanzan-release.keystore

# Vérifier les permissions
icacls android/flashanzan-release.keystore
```

📚 Ressources Utiles

Documentation Officielle

- **Angular:** https://angular.io/docs
- **Capacitor:** https://capacitorjs.com/docs
- **Android Developers:** https://developer.android.com/docs

Outils en Ligne

- **icon.kitchen:** https://icon.kitchen/ (génération d'icônes)
- **App Icon Generator:** https://appicon.co/
- **Google Play Console:** https://play.google.com/console

Commandes Git Utiles

```
# Voir l'historique
git log --oneline --graph

# Annuler le dernier commit (garder les changements)
git reset --soft HEAD~1

# Voir les différences
git diff

# Créer un .gitignore global
git config --global core.excludesfile ~/.gitignore_global
```

🎯 Commandes Rapides de Référence

```
# BUILD COMPLET
npm run build && npx cap sync android && cd android && .\gradlew bundleRelease && cd ..

# COMMIT ET PUSH
git add . && git commit -m "message" && git push

# TESTER SUR APPAREIL
adb install android/app/build/outputs/apk/debug/app-debug.apk

# VOIR LES LOGS ANDROID
adb logcat | Select-String "Capacitor"

# NETTOYER TOUT
npm run clean && cd android && .\gradlew clean && cd ..
```

Support

Pour toute question ou problème :

- **Email:** aissatahri81@gmail.com
- **GitHub Issues:** https://github.com/votre-username/flashanzan/issues

---

Créé par Azmicro | Dernière mise à jour : Novembre 2025