Power Systems

Modernisation IBM i – Nouveautés 2014-2015

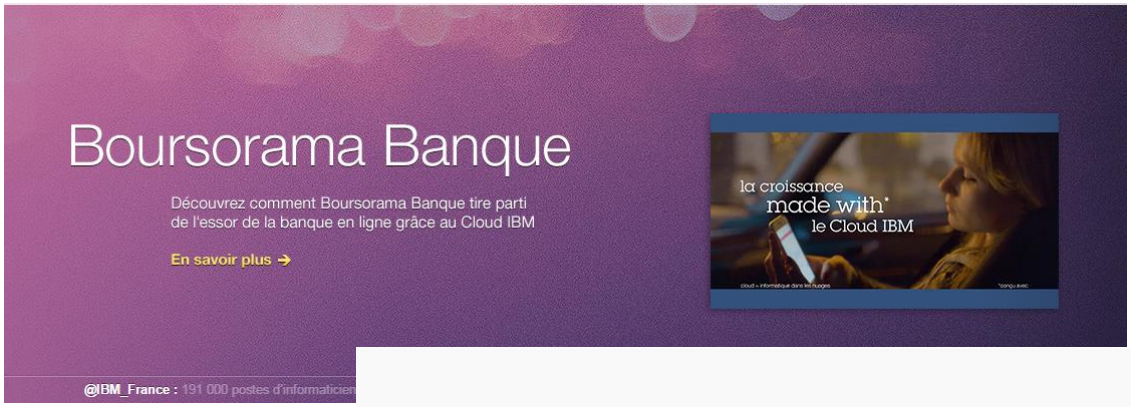
19 et 20 mai 2015 – IBM Client Center, Bois-Colombes

S2 – Développement d'applications mobiles Android pour IBM i

Mardi 19 mai – 14h00-15h30

Nathanaël BONNET – Gaia

Site web IBM France



Site web Microsoft France

À la maison

OFFICE 365
Achetez Office 365 et profitez d'1 To de stockage sur OneDrive.

HP STREAM 7
Tablette 7" avec un abonnement d'un an à Office 365 Personnel inclus pour seulement 99,00 €.

LUMIA 640
Rapide, efficace et proposé avec Office 365 Personnel.

XBOX ONE
Quatre jeux Halo et une console. À un prix optimal (dans la limite des stocks disponibles).

Au travail

VISUAL STUDIO COMMUNITY 2013
Créez gratuitement des applications performantes sur la majorité des plateformes

OFFICE 365 BUSINESS
Des applications complètes sur tous les appareils. Obtenez-les maintenant

ENTERPRISE MOBILITY SUITE
Gérez des utilisateurs, appareils, applications et données.

MICROSOFT CLOUD
Le Cloud qui rend les données passionnantes. En savoir plus.

Modernisation IBM i 19 & 20 Mai 2015

- « Mobile » ou « mobilité »
  - S2 – Développement d'applications mobiles Androïd pour IBM i
  - S6 – IBM iAccess Mobile : gérez votre IBM i à partir de votre téléphone ou votre tablette
  - S12 – DB2 Web Query - Nouveautés 2014-2015
  - S15 – SystemObjects, des solutions de mobilité pour l'IBM i
  - S19 – Offrez une interface Web et mobile à vos applications IBM i avec Profound UI et RPG Open Access
  - S24 – Modernisation d'applications – Un design du 21ème siècle pour nos applications IBM i
  - S28 – DB2 Web Query - Analytique et mobilité
  - S30 – Nouvelles interfaces utilisateurs pour IBM i : présentation de STRATEGI

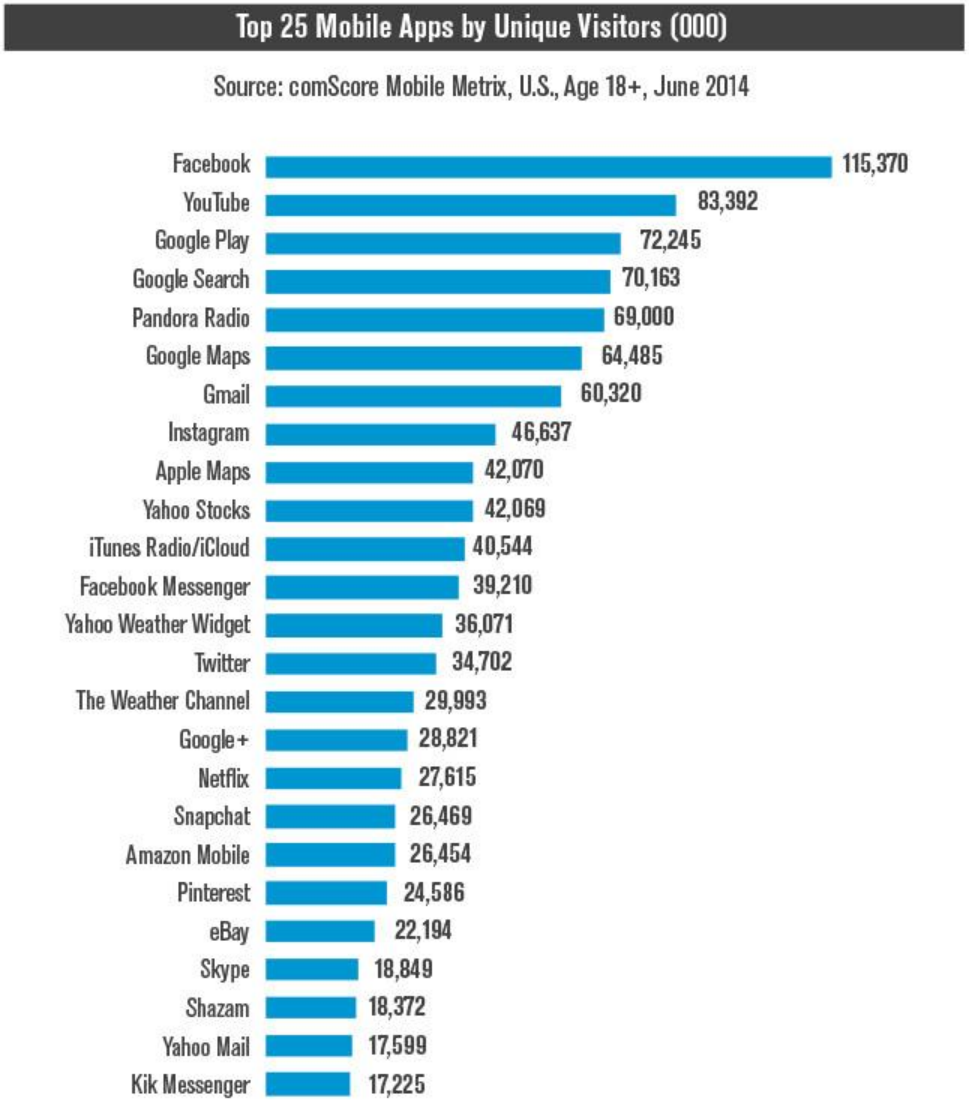
IBM (Alison Butterhill / Tim Row)

Power Systems

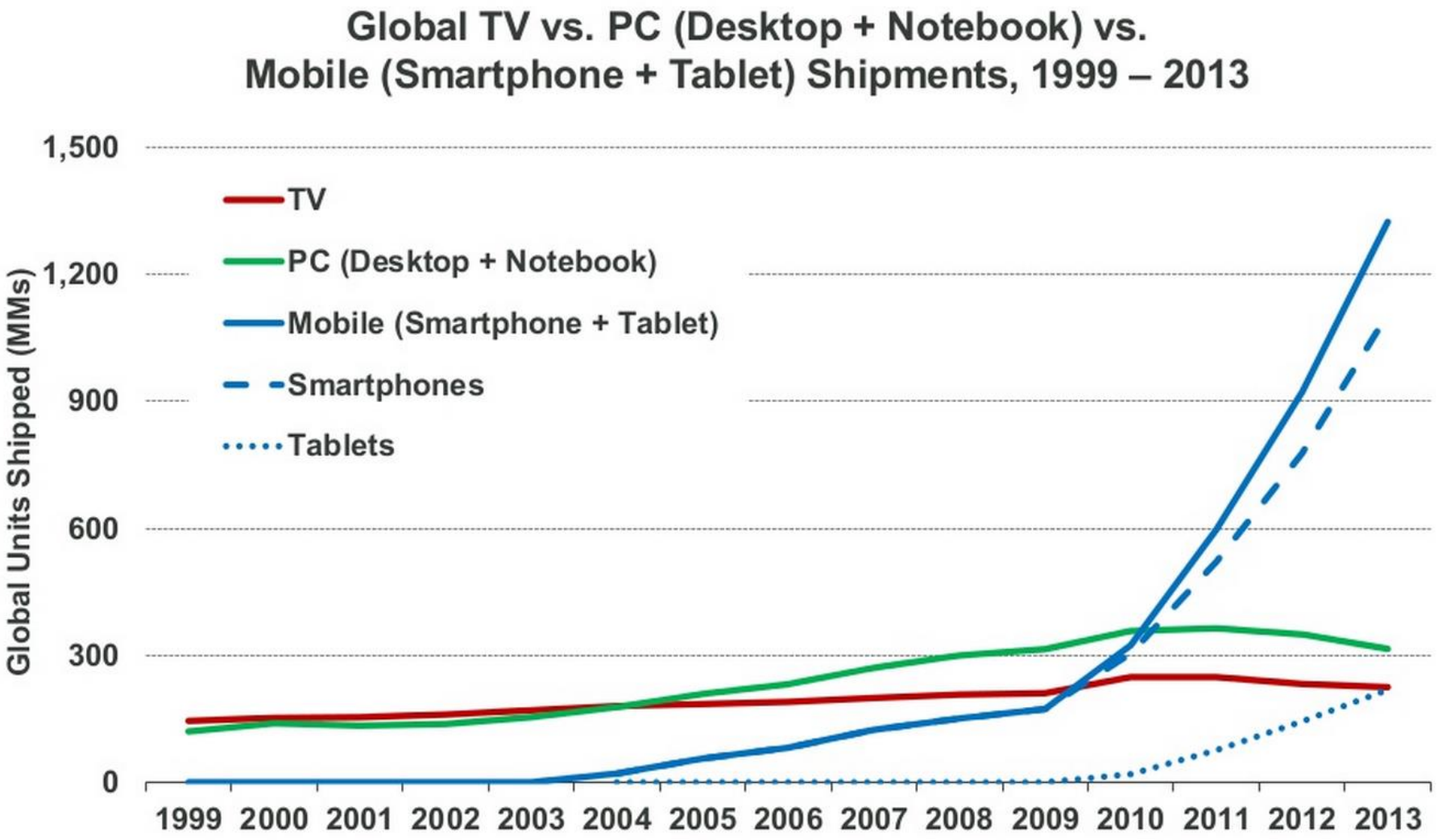
IBM

There are 7.1 billion people on the planet
6 billion of them have access to mobile phones,
only 3.5 billion of them use a toothbrush

Quel usage ?


Top 25 Mobile Apps by Unique Visitors (000)
Source: comScore Mobile Metrix, U.S., Age 18+, June 2014
Facebook 115,370
YouTube 83,392
Google Play 72,245
Google Search 70,163
Pandora Radio 69,000
Google Maps 64,485
Gmail 60,320
Instagram 46,637
Apple Maps 42,070
Yahoo Stocks 42,069
iTunes Radio/iCloud 40,544
Facebook Messenger 39,210
Yahoo Weather Widget 36,071
Twitter 34,702
The Weather Channel 29,993
Google+ 28,821
Netflix 27,615
Snapchat 26,469
Amazon Mobile 26,454
Pinterest 24,586
eBay 22,194
Skype 18,849
Shazam 18,372
Yahoo Mail 17,599
Kik Messenger 17,225

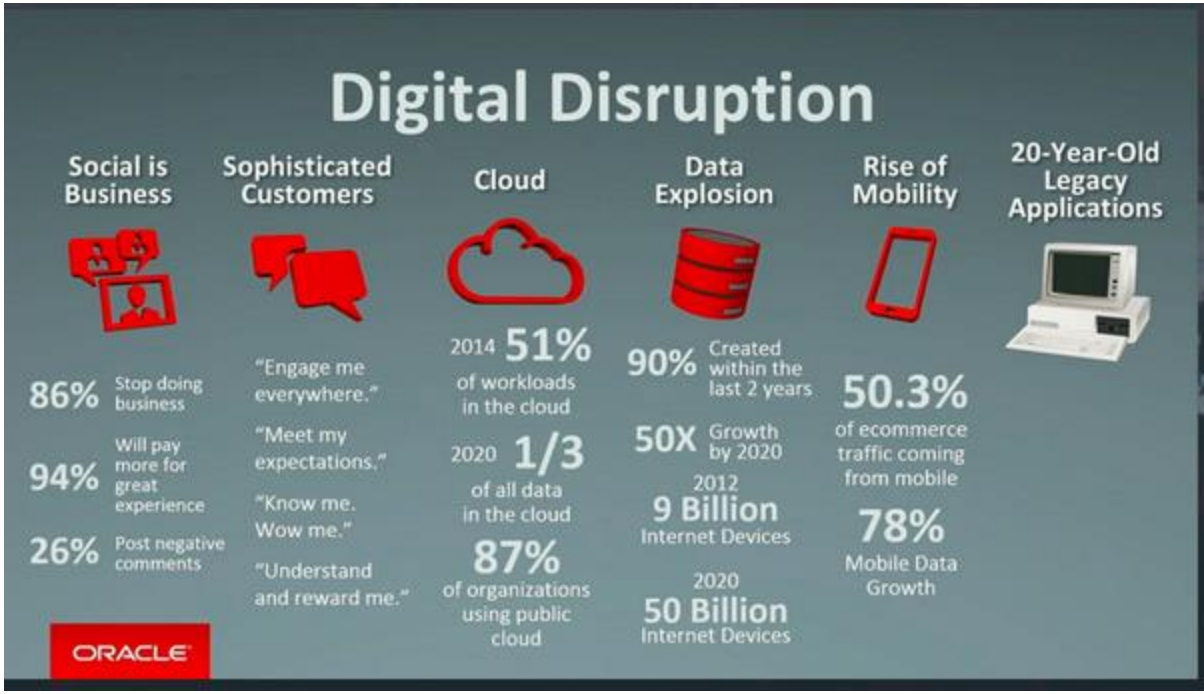
PC vs tablette/smartphone



Et donc ?

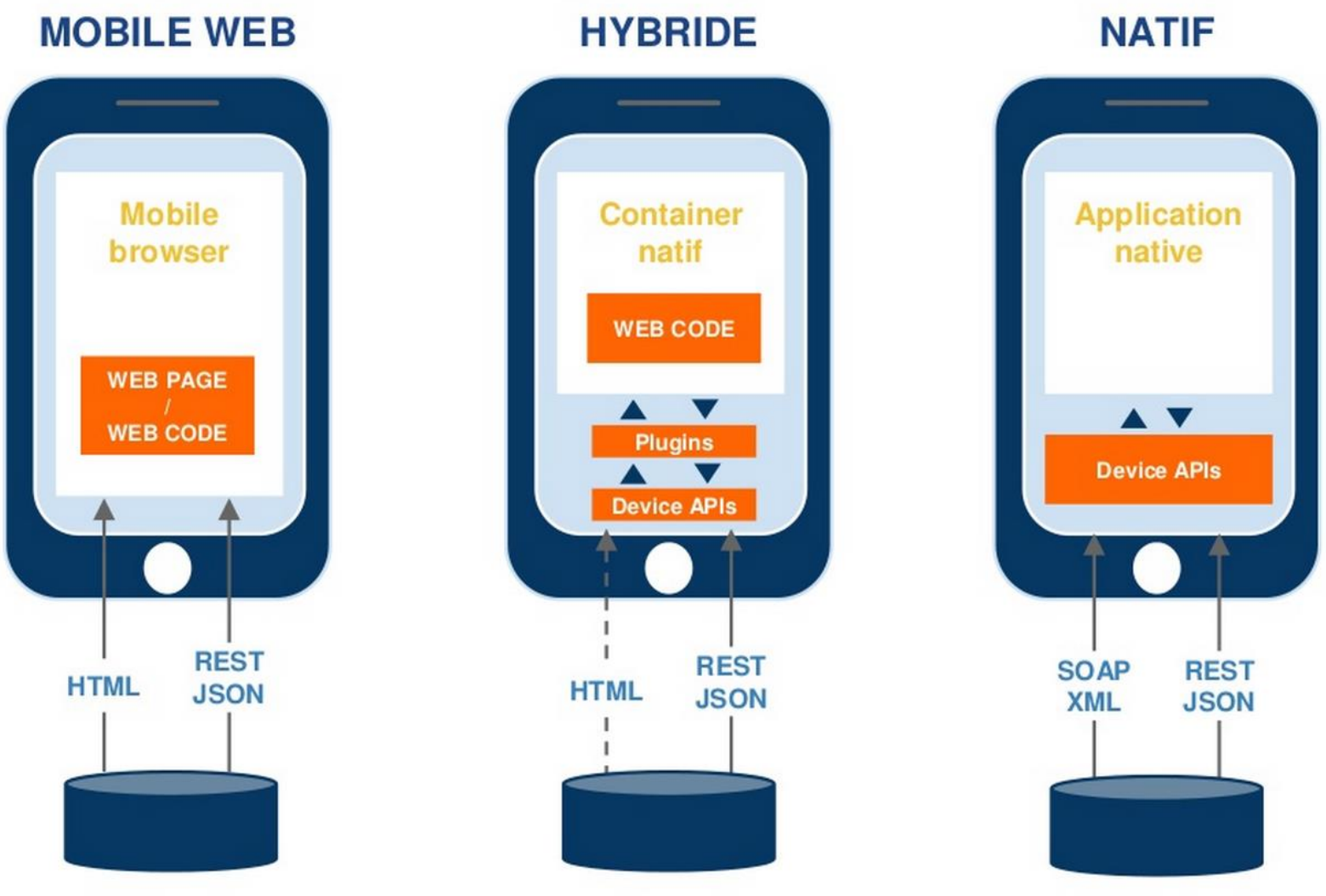
Disruption digitale

- Disruption
  - Remise en question des conventions culturelles dominantes
- Plus de frontière entre
  - les mondes physique et virtuel
  - l'entreprise et la maison, la matériel professionnel et personnel
- Demande une agilité sans précédent, de remettre en cause les modèles
  - voir les expressions « Uberiser », « Googler », « Liker » …



Exemple

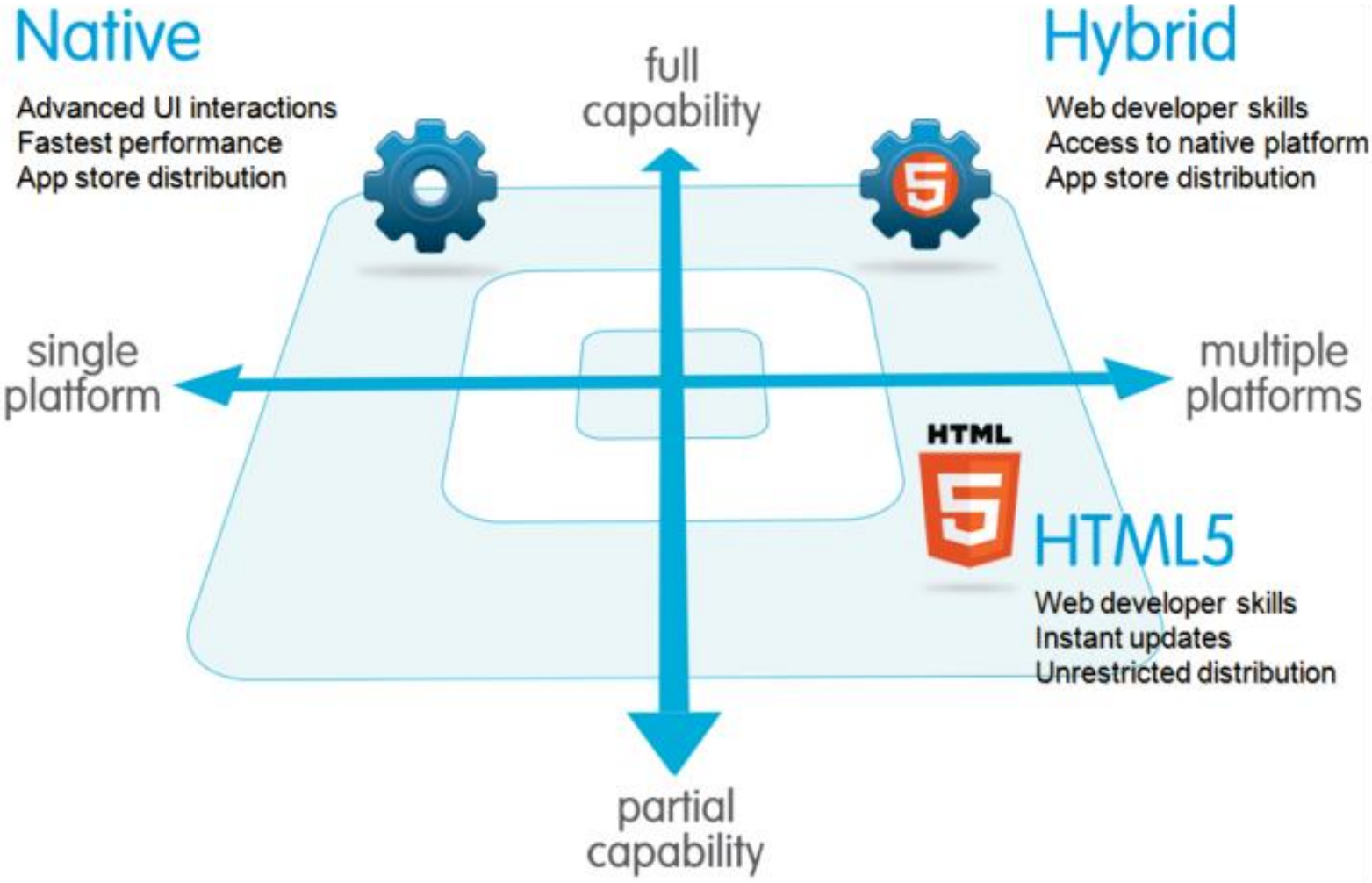
Les types d'applications mobiles



Les types d'applications mobiles

- Mobile web
  - Aucune application à installer
  - Site web « responsive »
  - Ne permet pas d'accéder à l'ensemble des ressources du périphérique (sécurité)
  - Interface utilisateur plus pauvre
  - Fonctionne uniquement en mode connecté
- Native
  - Application spécifique à un OS, à télécharger via un store
  - Permet l'accès à tous les périphériques, spécificités de chaque matériel …
  - Permet de réaliser la meilleure interface
  - Peut travailler hors connexion
- Hybride
  - C'est une application native
  - Qui va charger et afficher dynamiquement certaines informations au travers de pages HTML

Les types d'applications mobiles



Les outils standards pour l'IBM i

- Le seul outil proposé par IBM est le Toolbox Java
  - Distribué via sourceforge
    - http://sourceforge.net/projects/jt400/
  - Disponibles via PTF
    - V7R2: SI56274
    - V7R1: SI56273
    - V6R1: SI56271 SI56272

- Différents packages sont livrés
  - jt400.jar
  - jt400android.jar
  - jtopenlite.jar
- Le package jtopen.jar n'existe plus depuis que jt400.jar est diffusé via le projet jtopen

Les outils standards pour l'IBM i

- jt400.jar
  - Version complète du toolbox permettant d'accéder à l'ensemble des ressources IBM i : programmes, commandes, db2 SQL, RLA, IFS, data area, data queues, user index ...
- jt400android.jar
  - Version spécifique Android
    - Certaines classes nécessaires à l'exécution de jt400.jar n'existent pas dans le runtime Android
    - La migration d'un code qui utilise jt400 vers jt400android est très simple
- jtopenlite.jar
  - Version allégée du toolbox
    - Moins de fonctions, mais plus rapides, avec une empreinte mémoire réduite
    - N'est pas spécifique Android !
    - Migration vers/depuis jt400 complexe
  - Fonctions supportées : appel de programmes et commandes, JDBC, RLA, IFS

Les outils non spécifiques IBM i

- L'IBM i dispose de différents mécanismes permettant d'exposer des traitements existants
  - Web Services
    - SOAP ou REST
  - XMLSERVICE
  - SQL (à la fois standard et spécifique – db2)
  - Site web depuis l'IBM i, CGI
  - …
- Utiliser d'autres langages disponibles sur l'IBM i, supportés officiellement ou non par IBM
  - Node.js, Python
  - PHP
  - Ruby on rails
  - A noter : tous ceux-ci utilisent XMLSERVICE …

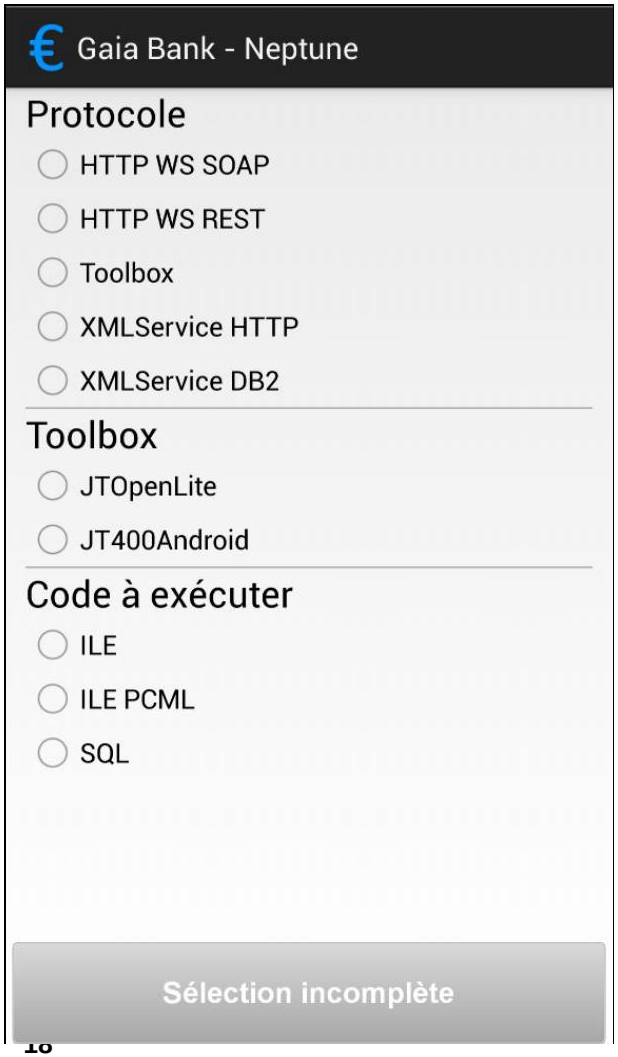
Notre cas d'étude

- Une application Android native
  - Utilisation des toolbox spécifiques
  - Utilisation des technologies « standards »
- Vocabulaire
  - Standard et spécifique s'entendent du point de vue de l'application mobile
- Technologies utilisées

Spécifiques IBM i	Standard
SQL via jtopenlite et jt400	Web services SOAP et REST
XMLSERVICE via db2	XMLSERVICE via http
Appel de programme via jtopenlite et jt400android	

L'application

- Simulation d'application bancaire



Architecture

Wifi IBM ou H+

SDSL 5Mo partagé

Power S814
7.2 TR1

Logique de présentation

Logique métier

BD

*SVPGM

- Client de services web REST et SOAP
- Client XMLSRVICE http et db2
- jtopenlite et jt400android

- Serveur de services web REST et SOAP
- XMLSERVICE http et DB2
- Service DB2

Mini démo

Technologies

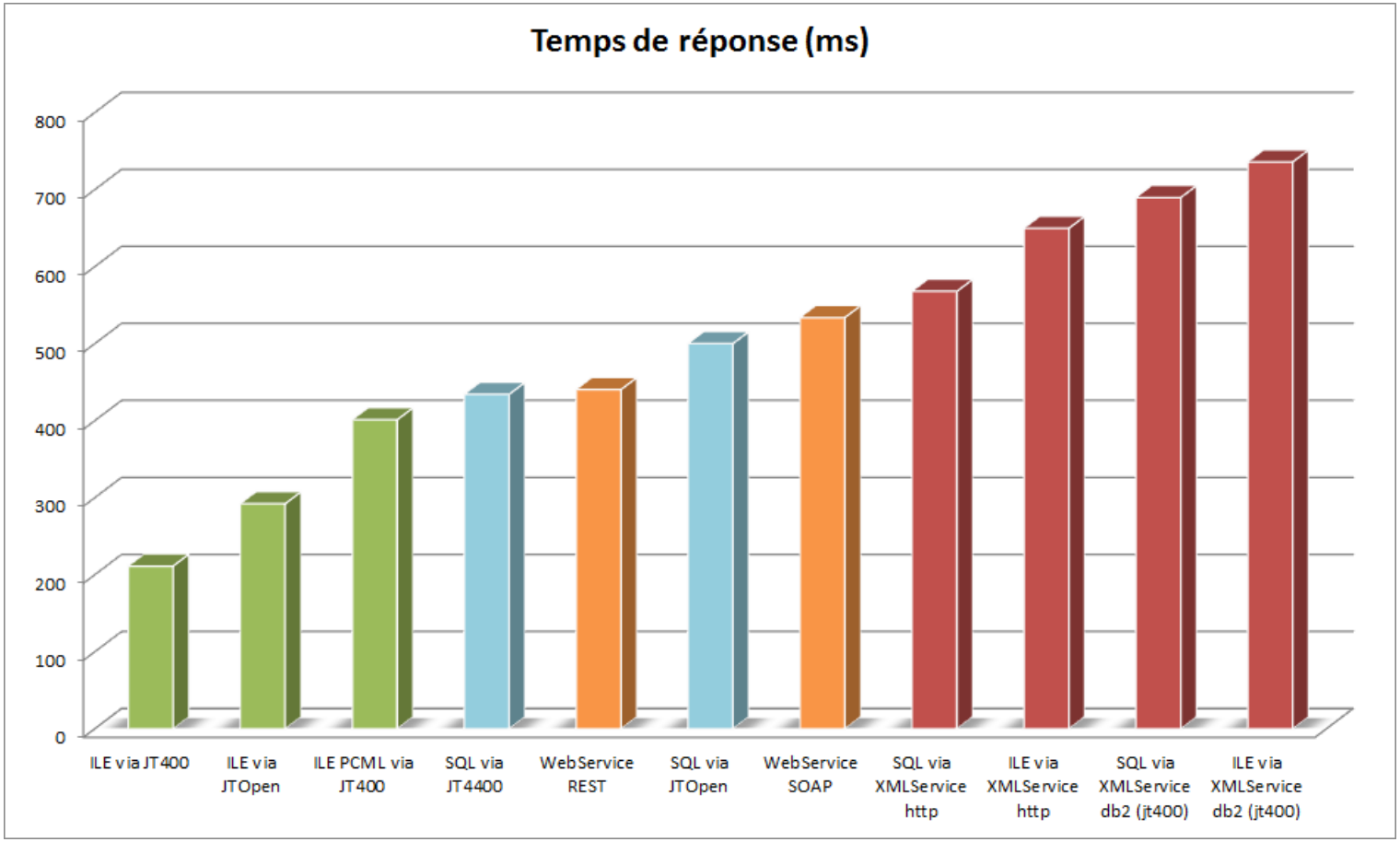
- A priori tout fonctionne !
- On ne va pas tout faire
  - Enfin, pas dans la vraie vie
- Quelques critères
  - Les performances
  - Les limites et détails techniques de chaque solution
  - Les compétences spécifiques nécessaires

Performances

- Remarque
  - Les performances sont mesurées par l'application Android
    - Temps entre la demande utilisateur et le décodage du résultat
    - Les temps de connexion ne sont pas représentés ici
  - Les chiffres présentés ici représentent des ordres de grandeur
    - Tests réalisés dans des conditions « non contrôlées »
      - Accès internet partagé (bureau et domicile)
      - Accès internet distant partagé entre plusieurs machines
    - Variabilité des résultats
  - L'application mobile n'est pas optimisée, pour aucune des technologies utilisées
  - Idem côté IBM i où aucun tunning n'a été réalisé
  - Les quantités de données transmises ici sont très faibles
    - Attention à l'extrapolation du comportement des différentes technologies avec des volumes plus élevés
      - Temps de réponse
      - Stabilité

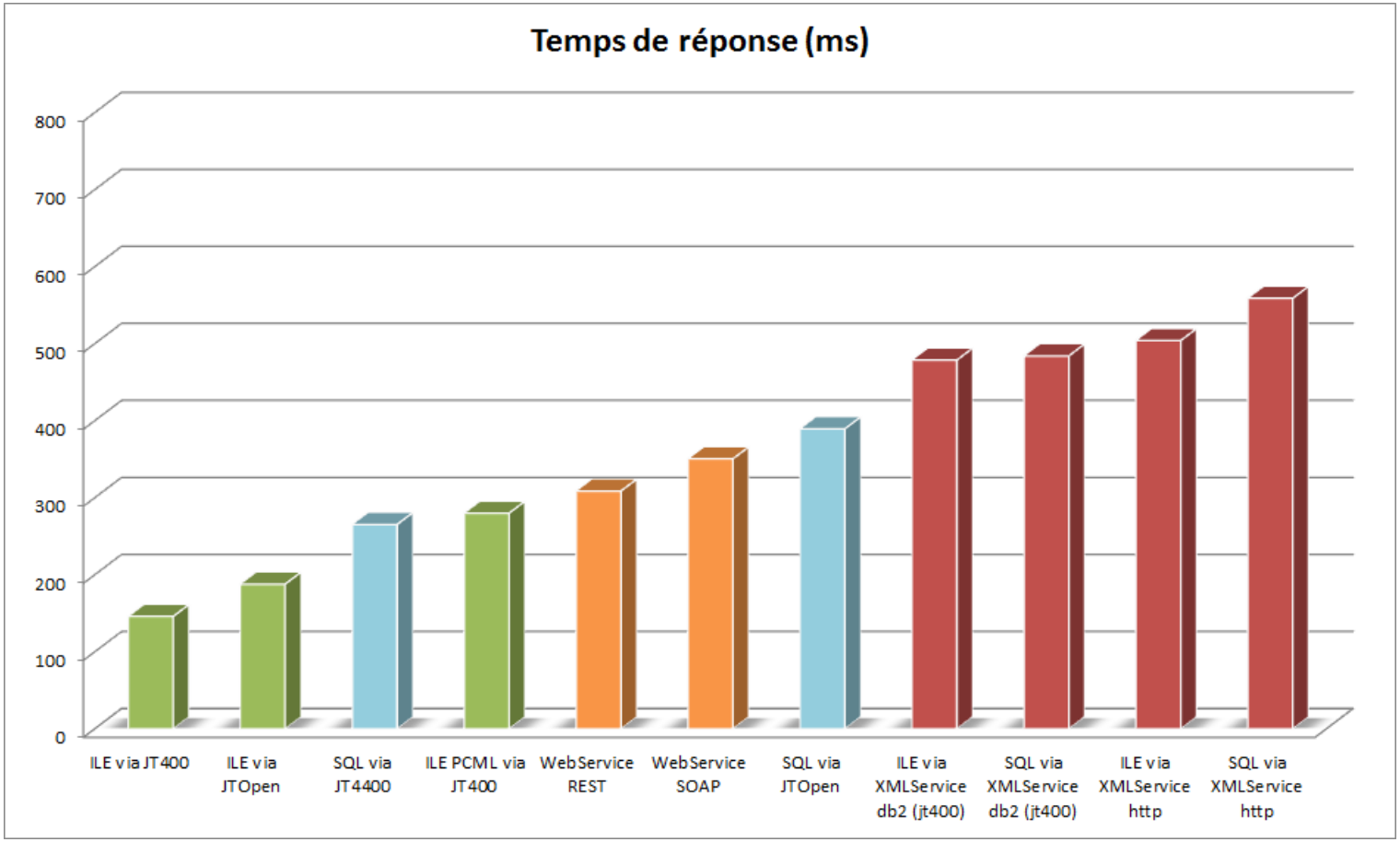
Performances

- Situation 1
  - IBM i 7.2 TR1 / Power 8 / Nantes / Accès Wifi box ADSL



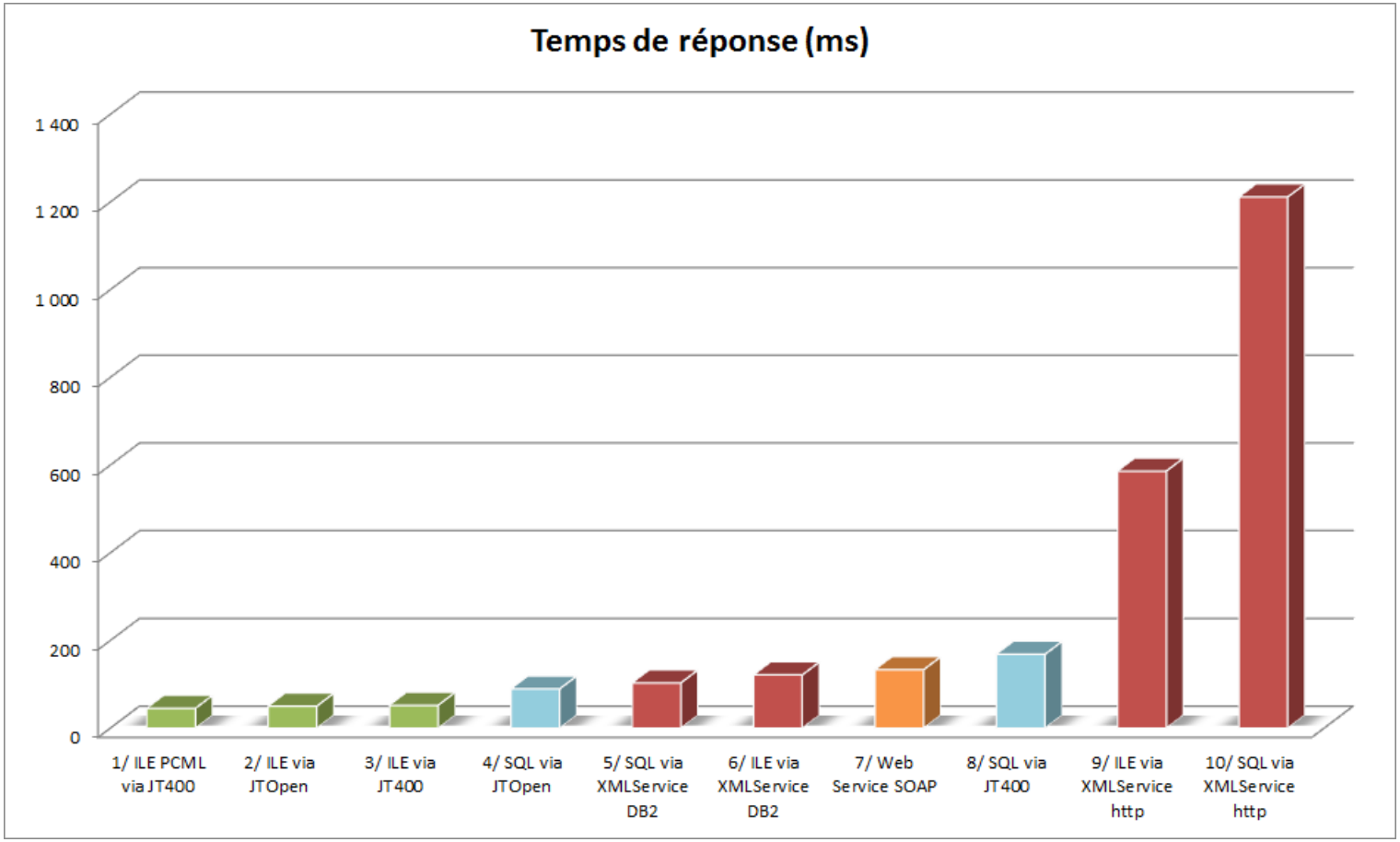
Performances

- Situation 2
  - IBM i 7.2 TR1 / Power 8 / Nantes / Accès SDSL 10 Mo



Performances

- Situation 3
  - IBM i 6.1 / Power 6+ (cache désactivé) / Lyon / Accès réseau local



Premiers constats

- Les temps de transport (réseaux) sont les plus importants
  - Différences de temps entre IBM i distant et local
- Les techniques les plus performantes, hors connexion
  - Appel de programme/procédure ILE via toolbox (jtopenlite ou jt400android)
  - SQL via toolbox, à égalité avec les services web REST et SOAP
    - REST est toujours plus rapide que SOAP, le côté client étant bien plus léger
  - XMLSERVICE à la traîne
    - Aussi bien l'accès par db2 que par http

Limites et détails techniques des solutions

- De façon générale
  - Seuls les programmes ILE ou les programmes de service sont utilisables par toutes les technologies
  - La gestion des paramètres avec les mots-clés ***omit** / ***nopass** n'est pas satisfaisante dans la plupart des cas
  - Bien sûr tous les mots-clés du type **RTNPARM**, **OPDESC**, **ALIGN** … sont à éviter sur les prototypes RPG

Limites et détails techniques des solutions

- Appel de programme/procédure via toolbox
  - jt400android
    - Toolbox le plus complet
      - Permet les appels de programmes, mais aussi JDBC, IFS, et la manipulation de nombreux types d'objets natifs
    - Dans notre application
      - Appel de procédure ILE construit en dynamique
      - Appel de procédure ILE via fichier PCML séralisé
  - jtopenlite
    - Moins complet mais avec une emprunte mémoire améliorée
    - Dans notre application
      - Appel de procédure ILE construit en dynamique
      - Ne gère pas le PCML
  - Limite pour les 2 toolbox
    - Pas plus de 7 paramètres pour une procédure (32 pour un programme)
      - Limite de l'API d'appel dynamique de procédure
        - Call Service Program Procedure (QZRUCLSP) API

PCML

- Fichier PCML
  - Généré par le compilateur RPG
  - Permet de décrire l'emplacement, les procédures, les paramètres d'un programme ou programme de service ILE
  - Peut être stocké dans l'objet lui-même ou généré dans l'IFS

- Limites du PCML
  - Pas de support des dates, heures, horodatages
    - Résolus en 7.1 avec la version 6 de PCML
  - Les valeurs de retour supportées sont uniquement binaires sur 4 octets
  - Pas de pointeur …

PCML - exemple

- Extrait
  - Pour procédure getClient

```
// - Retrouver les attributs du client
// ------------------------------------------------------
// Valeur de retour : ERROR / SUCCESS
// Code retour : RC_*
// Remarque : si RC_NOT_FOUND, alors SUCCESS
d getClient       pr             10i 0
d  id                                 like( T_Client.id )
d                                     const
d  client                             likeds( T_Client )
d  rc                                 like( T_ReturnCode )
d                                     options( *nopass )
```

```
<pcml version="4.0">

   <!-- 42 -->
   <struct name="T_CLIENT">
      <data name="ID" type="int" length="4" precision="31" usage="inherit" init="0" />
      <data name="NOM" type="char" length="25" usage="inherit" init="" />
      <data name="PRENOM" type="char" length="25" usage="inherit" init="" />
      <data name="MAJ" type="char" length="26" usage="inherit" init="0001-01-01-00.00.00.000000" />
   </struct>

   <program name="GETCLIENT" entrypoint="GETCLIENT" returnvalue="integer">
      <data name="ID" type="int" length="4" precision="31" usage="input" />
      <data name="CLIENT" type="struct" struct="T_CLIENT" usage="output" />
      <data name="RC" type="char" length="3" usage="output" />
   </program>

</pcml>
```

PCML - limites

- Dans les cas où le fichier PCML ne peut pas être généré par le compilateur
  - Il faut le créer ou modifier manuellement
    - Risque d'erreur
  - Ce que j'ai fait ici
    - Modification des zones horodatages en 26A
    - Bug : les zones numériques condensées étaient déclarées en « zoned » et non « packed »
- Avec la version 6 du PCML
  - Moins de cas à gérer

jt400android et PCML

- Il faut sérialiser le PCML

```
AS400 as400System;          // com.ibm.as400.access.AS400
ProgramCallDocument pcml;   // com.ibm.as400.data.ProgramCallDocument
System.setErr(System.out);

// Construct AS400 with parameters
as400System = new AS400("10.2.0.1", "NB", "[redacted]");

try
{
    pcml = new ProgramCallDocument(as400System, "pcml.serialize.gaia.pcml");
    pcml.serialize(new File("src/pcml/serialize/demo_gaia.neptune.pcml.ser"));
}
```

  - Les éléments de connexion sont codés dans le PCML sérialisé

jt400android et PCML

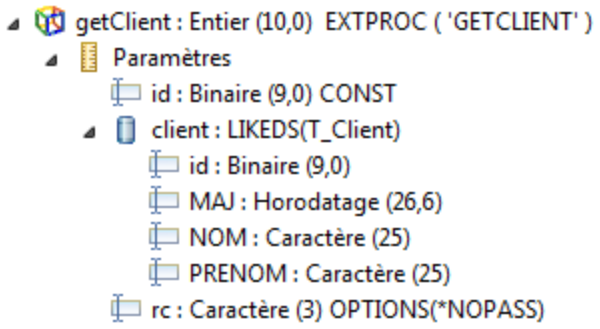
- Utilisation

```
// _____________________ Créer le document PCML _____________________

// Instancer le PCML Java depuis le PCML embarqué séralisé
ProgramCallDocument pcml = new ProgramCallDocument(as400, "fr.gaia.bankneptune.beans.loader.demo_gaia");
// Indiquer le path de l'executable
pcml.setPath("GETCLIENT", "/QSYS.LIB/DEMO_GAIA.LIB/DG_SRV.SRVPGM") ;
// Alimenter les paramètres en entrée
pcml.setValue("GETCLIENT.ID", this.client.id);

// Il n'est pas nécessaire d'alimenter les autres paramètres car output seul
// Il faut alimenter les inputoutput

// _____________________ Appel _____________________

// Appel de la procédure getClient
if (pcml.callProgram("GETCLIENT") != true) {
    // Retrouver les messages en cas d'erreur
    AS400Message[] messageList = pcml.getMessageList("GETCLIENT");
    for (int i = 0; i < messageList.length; ++i) {
        Log.d("MSG", messageList[i].getText());
    }
    // Sortie
    return;
}
```

jt400android sans PCML

- Plus complexe
  - Il faut indiquer tous les paramètres manuellement

```
// - Retrouver les attributs du client
// ------------------------------------------------------------
// Valeur de retour : ERROR / SUCCESS
// Code retour : RC_*
// Remarque : si RC_NOT_FOUND, alors SUCCESS
d getClient       pr             10i 0
d  id                                 like( T_Client.id )
d                                     const
d  client                             likeds( T_Client )
d  rc                                 like( T_ReturnCode )
d                                     options( *nopass )
```




```
// _____________________ Créer les paramètres _____________________

// Créer une liste de 3 paramètres :
ProgramParameter[] parmList = new ProgramParameter[3];

// Paramètre 1 : id client
AS400Bin4 bin4_cvt = new AS400Bin4();
parmList[0] = new ProgramParameter(ProgramParameter.PASS_BY_REFERENCE,
                                   bin4_cvt.toBytes(this.client.id));

// Paramètre 2 : structure client

// Créer une structure correspondante
AS400DataType[] getClient_client = {
        new AS400Bin4(),    // id client
        new AS400Text(25),  // nom
        new AS400Text(25),  // prenom
        new AS400Text(26),  // maj  => horodatage non supporté
};

// Créer un objet de conversion
AS400Structure clientConverter= new AS400Structure(getClient_client) ;

// Créer l'objet Java qui contient les données à transmettre à la procédure ILE
Object[] inz_data_client = { 0,     // id
                             "",    // nom
                             "",    // prenom
                             "" };  // maj
// Convertir les données Java en AS400
parmList[1] = new ProgramParameter(
        ProgramParameter.PASS_BY_REFERENCE,
        clientConverter.toBytes(inz_data_client),
        clientConverter.getByteLength());

// Pramaètre 3 : code retour
AS400Text char3_cvt = new AS400Text(3);
parmList[2] = new ProgramParameter(ProgramParameter.PASS_BY_REFERENCE,
                                   char3_cvt.toBytes(""),
                                   char3_cvt.getByteLength());
```

jt400android sans PCML

- Appel

```
// _____________________ Appel _____________________

// Instancier l'objet d'appel de programme de service
ServiceProgramCall srvpgm = new ServiceProgramCall(as400);
// Alimenter le nom du programme de service
srvpgm.setProgram( "/QSYS.LIB/DEMO_GAIA.LIB/DG_SRV.SRVPGM", parmList);
// Nom de la procédure
srvpgm.setProcedureName("GETCLIENT");
// Format du paramètre de retour
srvpgm.setReturnValueFormat(ServiceProgramCall.RETURN_INTEGER);

// Appel de la procédure
if (srvpgm.run() != true) {
    // Retrouver les messages en cas d'erreur
    AS400Message[] messageList = srvpgm.getMessageList();
    for (int i = 0; i < messageList.length; ++i) {
        Log.d("MSG", messageList[i].getText());
    }
    // Sortie
    return;
}
```

jtopenlite

- Limites
  - Ne supporte pas le PCML
  - Ne dispose pas de wrapper pour les types AS400
- Il faut écrire une classe qui décrit, encode et décode les paramètres
  - Fastidieux
  - Risqué : nécessité de travailler en buffer
    - Devient délicat avec les DS DIM(x)

jtopenlite

```
@Override
public int getParameterLength(int parmIndex) {
    switch (parmIndex)
    {
      case 0:
          return 4;
      case 1:
          return 80;
      case 2:
          return 3;
    }
    return 0;
}

@Override
public void setOutputData(int parmIndex, byte[] buffer, int offset) {
    switch (parmIndex)
    {
      case 0:
        break;
      case 1:
              client_id_ = Conv.byteArrayToInt(buffer, offset);
              client_nom_ = Conv.ebcdicByteArrayToString(buffer, offset + 4, 25).trim() ;
              client_prenom_ = Conv.ebcdicByteArrayToString(buffer, offset + 29, 25).trim() ;
              client_maj_ = Conv.ebcdicByteArrayToString(buffer, offset + 54, 26).trim() ;
          break;
      case 2:
              rc_ = Conv.ebcdicByteArrayToString(buffer, offset, 3);
        break;
    }

}
```

Accès JDBC/SQL via toolbox

- Les 2 solutions sont strictement équivalentes à l'usage
  - Pilotes JDBC de type 4

```
// Construction de la connection à l'IBM i
public JDBCConnection() throws Exception {
    // Mémoriser le mode en cours
    this.mode_toolbox = Data.getInstance().mode_toolbox ;
    // Construire la connexion pour le mode en cours
    Data.getInstance().logger.start();
    if ( Data.getInstance().mode_toolbox == Data.MODE_JTOPENLITE )
    {
        Class.forName("com.ibm.jtopenlite.database.jdbc.JDBCDriver");
        connection = DriverManager.getConnection(
                "jdbc:jtopenlite://" + Data.LAN_URL, Data.USER, Data.PASSWORD);
    }
    if ( Data.getInstance().mode_toolbox == Data.MODE_JT400ANDROID )
    {
        Class.forName("com.ibm.as400.access.AS400JDBCDriver");
        connection = DriverManager.getConnection(
                "jdbc:as400://" + Data.LAN_URL, Data.USER, Data.PASSWORD);
    }
    Data.getInstance().logger.end("Connexion JDBC :");
}
```

Accès JDBC/SQL via toolbox

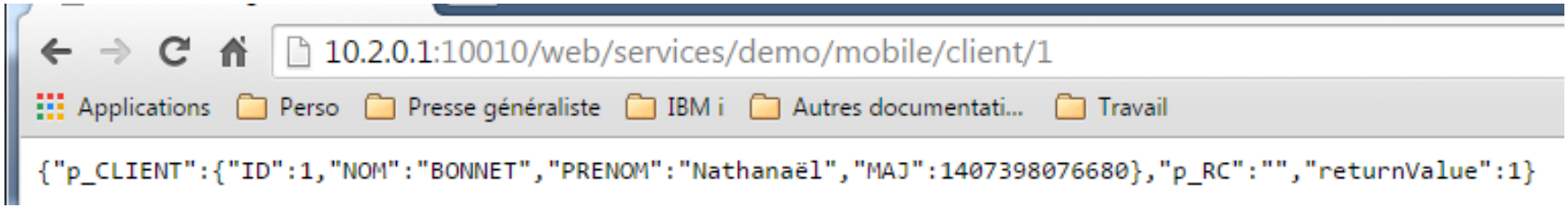
- Aucune particularité à l'usage

```
// Retrouver la connexion à la BD
Connection connection = JDBCConnection.getInstance().connection;
Statement statement = connection.createStatement();

Data.getInstance().logger.start();

// Charger les attributs du client
ResultSet rs = statement
        .executeQuery("SELECT * FROM DEMO_GAIA.DG_CLIENT WHERE ID = " + this.client.id );
if (rs.next()) {
    this.client.nom = rs.getString("nom").trim();
    this.client.prenom = rs.getString("prenom").trim();
}
rs.close();
```

Accès par service web

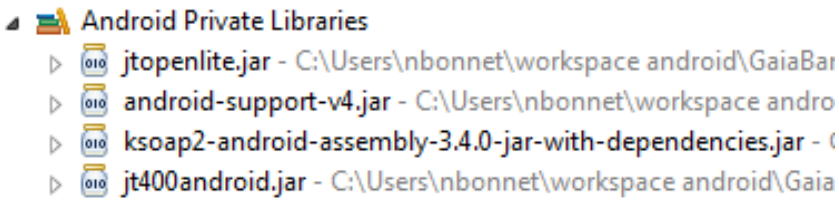
- Les services web REST et SOAP
  - Sont créés sur la même instance de serveur
  - Sur la base du même programme de service
- Client REST
  - Rien de particulier, il suffit d'invoquer une URL avec les méthodes GET, PUT, DELETE …

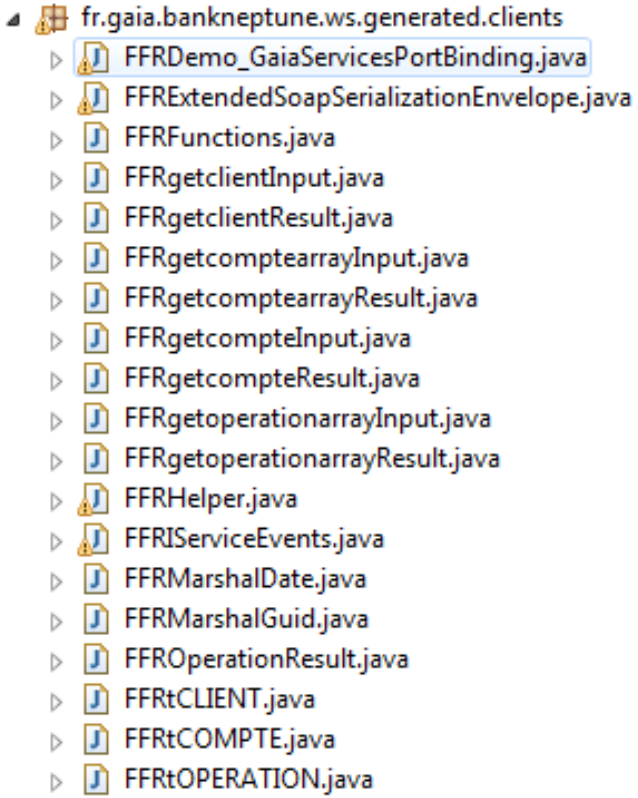
  `GET` http://10.2.0.1:10010/web/services/demo/mobile/client/1
  - Ensuite décoder le JSON



Accès par service web

- Client SOAP
  - Plus complexe
    - Il faut générer un message SOAP correspondant au WSDL du service à invoquer
    - Il existe des bibliothèques ou des outils
      - Sauf à coder en dur …
  - J'ai utilisé ici un générateur automatique
    - www.easywsdl.com
    - Génère les « stubs »
    - Nécessite également des bibliothèques
      - ksoap2-android-assembly





Utilisation de XMLSERVICE

- XMLSERVICE est une couche de communication avec l'IBM i permettant l'utilisation de ressources natives
  - SQL
  - Appel de programmes/procédures
  - QSH, Rexx …
- C'est un produit à installer
  - Cf http://yips.idevcloud.com/wiki/index.php/XMLService/XMLService
  - Totalement Open Source RPG
  - Utilisé par PHP, Python, Ruby on rails …
- Il est utilisable via 2 moyens
  - http au travers de programmes CGI
  - db2 au travers de procédures stockées

Utilisation de XMLSERVICE

- Principe de fonctionnement
  - On envoie une demande sous forme d'un flux XML
    - Décrit l'opération à effectuer
      - Appel de commande, de programme, SQL …
  - On reçoit en retour un flux XML contenant le retour
    - Même pour des commandes de type DSP*
  - Possibilité de réaliser des scripts
    - 1 opération de type appel de commande
      - CRTDUPOBJ ….
    - 1 appel de programme
      - CALL …
    - 1 appel de script shell
      - Qsh …

Utilisation de XMLSERVICE

- Dans les deux cas (http et db2)
  - La construction du script à exécuter est identique
  - db2 nécessite une connexion préalable par un toolbox
    - On utilisera uniquement jt400android, jtopenlite ne prenant pas en charge les CLOB nécessaires à l'appel des procédures stockées de XMLSERVICE
- Appel ILE ou SQL
  - La construction du script SQL est plus simple (plus court), mais aucun des 2 ne présente de difficulté
- Résultat
  - Le résultat est toujours un flux XML à parser

Utilisation de XMLSERVICE

- Exemple de scripts

```
<sql>
 <script>
   <options options="myoptions" naming="sql" sqllib="demo_gaia"/>
   <connect conn="maconn" options="myoptions"/>
   <query conn="maconn" stmt="user">select * from dg_client where id = 1</query>
   <fetch stmt="user" block="all" desc="on"/>
   <query conn="maconn" stmt="lst_cpt">select * from dg_compte where id_cli = 1</query>
   <fetch stmt="lst_cpt" block="all" desc="on"/>
   <free conn="maconn"/>
 </script>
</sql>
```

```
<script>
 <pgm name="DG_SRV" lib="DEMO_GAIA" func="GETCLIENT" mode="ile">
  <parm comment="id" by="ref" io="in">
   <data type="10i0">1</data>
  </parm>
  <parm comment="client" io="out" by="ref">
   <ds>
    <data comment="id" type="10i0"/>
    <data comment="nom" type="25a"/>
    <data comment="prenom" type="25a"/>
    <data comment="maj" type="26a"/>
   </ds>
  </parm>
  <parm io="out" by="ref">
   <data comment="rc" type="3A"/>
  </parm>
  <return>
   <data comment="return" type="10i0"/>
  </return>
 </pgm>
</script>
```

Synthèse

- Appel de programme/procédure via les toolbox
  - jt400android est bien plus simple
    - Surtout avec PCML
- SQL
  - Obligatoirement via les toolbox
  - Seule différence rencontrée : CLOB géré par jt400android et non par jtopenlite
- XMLSERVICE
  - Demande peu de code pour l'appel
    - Le retour est un flux XML, donc assez classique à parcourir
    - Globalement plus verbeux que SQL ou PCML
  - Demande une installation serveur
- Services web
  - REST
    - Peu de code, efficace en termes de développement
  - SOAP
    - Demande beaucoup de code pour gérer la mécanique SOAP

Modes de connexion

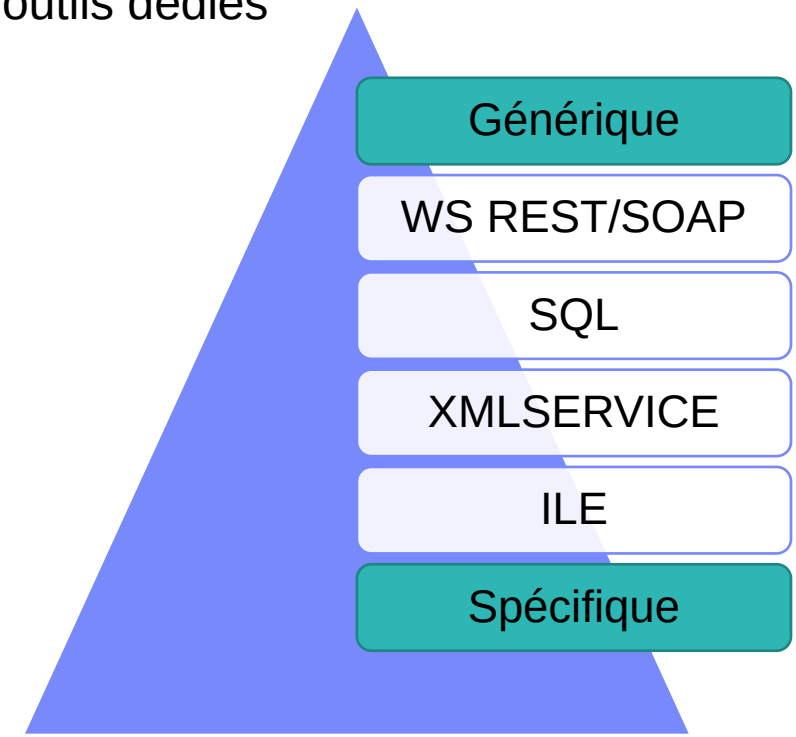
- En environnement mobile, un critère de choix important
  - Mode connecté
    - Maintient de session entre les appels
  - Mode déconnecté
    - Tous les appels sont autonomes
- Da façon générale
  - Tout ce qui est basé sur http est en mode déconnecté
    - Web Services SOAP et REST
    - XMLSERVICE via http
  - Tout ce qui est basé sur les toolbox est en mode connecté
    - ILE et SQL
    - XMLSERVICE via db2

Modes de connexion

- Mode connecté
  - Temps de connexion
  - Appels suivants plus rapides puisque le contexte d'exécution est déjà en place côté serveur
  - Nécessite une authentification forte
- Mode déconnecté
  - Les serveurs logiciels sur l'IBM i peuvent ou non maintenir des pools de connexions locales et réutilisables
- Réseaux mobiles
  - Fortement instables
    - Passage rue/métro/bus/tram … Wifi … 4G … zone blanche …
  - Les applications sont donc plus à l'aise avec le mode déconnecté
    - Le mode connecté nécessite une gestion fine des reconnexions et pertes de sessions, données de la session etc …

Compétences nécessaires

- Parmi toutes ces techniques
  - Certaines sont accessibles sans compétences particulières
    - Comprendre pour n'importe quel développeur Android (ce qui est déjà une compétence)
  - D'autres nécessitent la prise en main d'outils spécifiques à l'IBM i
    - Les toolbox et outils dédiés



Compétences nécessaires

- Intérêt d'investir sur des compétences spécifiques ?
  - Alors même que les technologies changent rapidement
    - Aussi bien côté serveur que client
  - Cela peut être un choix guidé par des contraintes techniques ou organisationnelles
- En général, deux équipes distinctes
  - Une équipe IBM i
    - Doit être en mesure de garder la maîtrise des opérations back office
    - Doit être capable d'ouvrir ses traitements à destination d'autres applications dont mobile
  - Une équipe mobile (et/ou autres technos)
    - Doit être capable de consommer ce qui est mis à disposition depuis l'IBM i
  - Les équipes mixtes sont rares pour des questions de double compétences inexistantes

Ce qu'il faut en conclure

- Ne pas se focaliser sur les performances
  - Hors usage spécifique
- Choisir une technologie adaptée au mode réel d'utilisation de l'application
  - On pourra choisir un mode connecté pour une application qui travaille en Wifi dans un entrepôt, alors qu'une application grand public devra être utilisable partiellement hors connexion
- Toutes ces solutions sont valables, c'est l'usage qui doit guider le choix
  - XMLSERVICE par exemple, qui souffre d'un déficit de performance dans notre test, a la capacité de pouvoir envoyer des scripts (enchaînement d'actions) en effectuant un unique appel
    - Et donc retrouver de la performance
  - Les toolbox sont les plus complets
    - Mais a-t-on vraiment besoin de manipuler des user spaces depuis une application mobile ? L'appel de programme/procédure, l'IFS et db2 ne suffisent-ils pas ?

Dans la vraie vie

- Je ne donnerai pas le feu vert à mon application pour passage en production !
  - C'est un démonstrateur, je n'ai pas tout codé avec délicatesse
    - Les exceptions, principalement les changements de mode de connexion
  - Certains choix ne sont pas admissibles en production
    - SQL
      - J'utilise directement les fichiers
      - On devrait utiliser des procédures stockées db2 qui font le travail équivalent
        - Garder la maîtrise côté IBM i des accès et de la logique d'accès et logique métier

Les clés pour réussir son projet mobile

- Satisfaire l'utilisateur
  - L'utilisateur final n'a que faire des débats techniques : il veut une application mobile fonctionnelle, réactive et intuitive !
    - Pas de manuel utilisateur sur une application mobile
  - Certains utilisateurs se verront imposé un terminal, choisi par la DSI sur certains critères. D'autres catégories d'utilisateurs (directions) choisiront eux-mêmes, et il vous faudra le gérer dans tous les cas !
  - Dans la plupart des cas, le terminal est affecté à un utilisateur unique (en dehors des terminaux de type « entrepôt/logistique »), et sera utilisé à titre personnel en dehors de l'entreprise. Ce sera donc un outil professionnel, mais auquel on attache une importance personnelle, véhiculant en plus un statut social/hiérarchie/image
- Comprendre les différences des mondes mobile et IBM i
  - La population : il est très difficile d'imposer un outil mobile auquel les utilisateurs n'adhèrent pas (en fonction de la population)
  - La durée de vie : les applis IBM i sont « immortelles ». Une application mobile est par définition temporaire/jetable ! Les plateformes/OS/technos évoluent très vite. Elles revêtent souvent un aspect marketing également qui justifie leur renouvellement.

Les clés pour réussir son projet mobile

- Faire un choix d'implémentation
  - IBM fournit des produits natifs :
    - Android uniquement pour le mobile : Android toolbox. Open source
  - Sur IBM i : IWS
  - D'autres solutions éditeurs existent, ce n'est pas le propos ici (PHP et autres)
- Concevoir en service pour réutiliser
  - Pour faciliter la réutilisation des programmes existants, que ce soit vers du mobile ou d'autres technos, il faut s'orienter vers une architecture service par le haut
    - Dans l'état de l'art : stateless
  - Dans l'idéal, chaque fonctionnalité n'existe qu'en un unique exemplaire dans le SI et est accessible aux autres briques logicielles
- Maîtriser les différences de conception entre mobile et IBM i
  - Différences techniques : On ne construit pas une appli mobile comme une appli 5250 : on va chercher à limiter les A/R sur le réseau, éventuellement maintenir de la donnée en local de façon asynchrone pour permettre l'utilisation non connectée de l'appli (il faut surtout gérer la resynchro), cinématique différente (entraîne des besoins en info différents) …

Les clés pour réussir son projet mobile

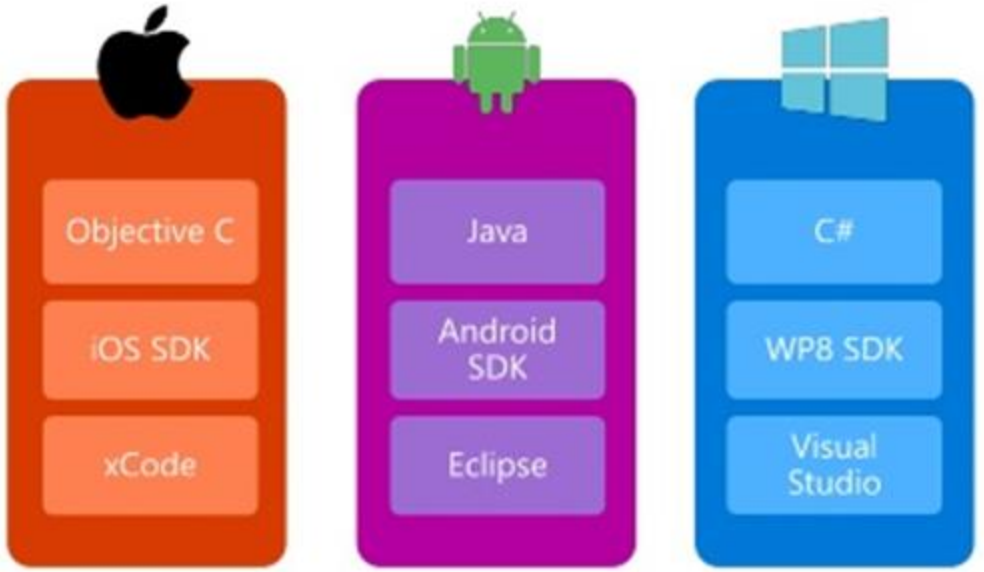
- Augmenter ses compétences
  - Partant du principe que vous aurez obligatoirement quelques compétences en développement mobile, il faudra compléter par certaines compétences spécifiques IBM i <-> mobile en fonction des choix techniques effectués :
    - Toolbox Java Android
    - Services Web IBM i
    - XMLService : IBM i et mobile
  - Il faut connaître les outils pour connaître leurs limites : limites du PCML, du toolbox android …
- Gérer les périphériques
  - Les périphériques de l'entreprise devront être gérés par la DSI, avec de nouvelles problématiques :
    - Gérer des périphériques dont une part de l'usage est personnelle
    - Assurer la sécurité des données, dont certaines sont stockées sur le terminal
    - Capacité à maîtriser le parc, gérer les maj systèmes, les maj logicielles …
  - Pour cela, les outils de MAM/MDM sont indispensables dès que le parc devient conséquent

Les applications natives

Les applications natives

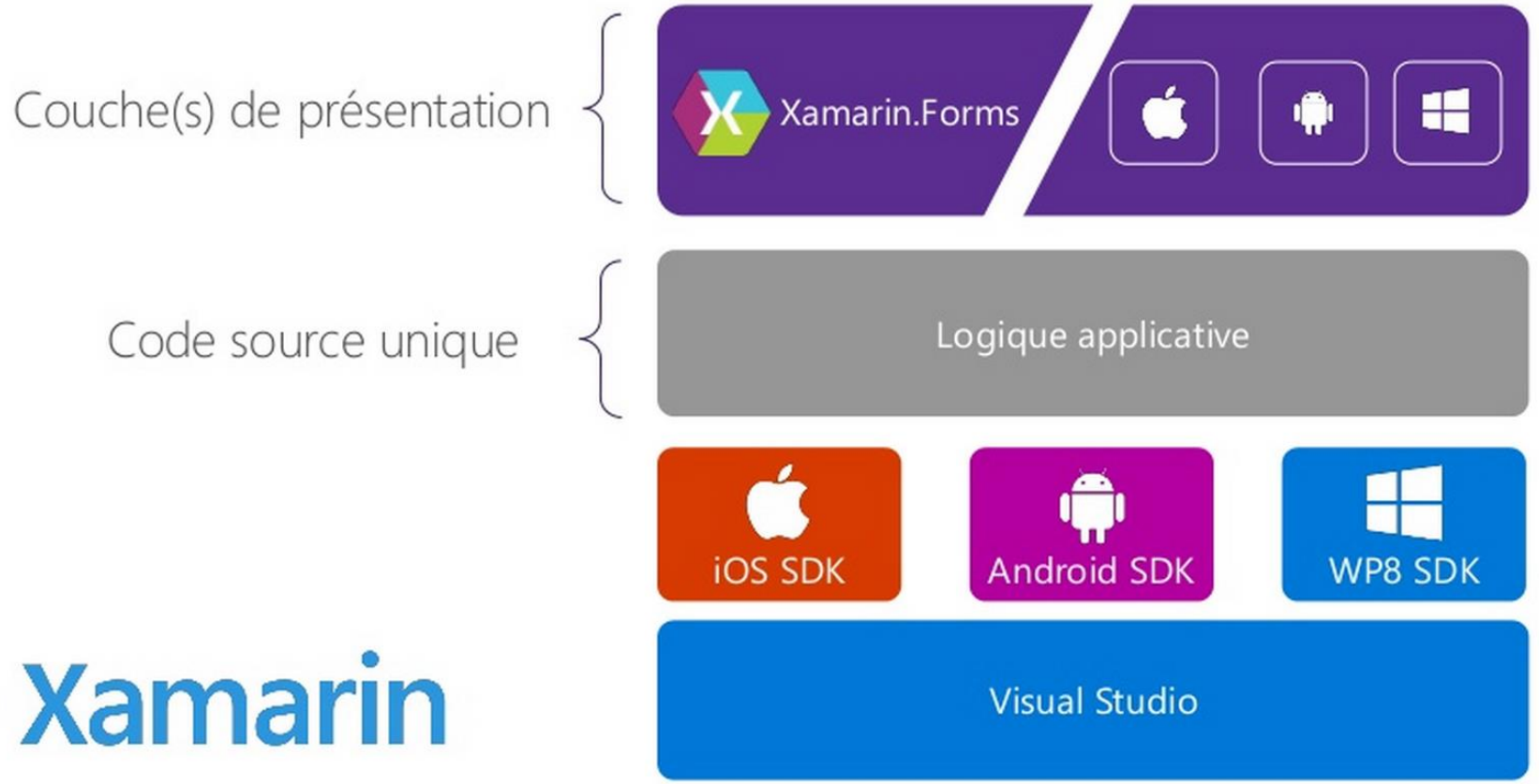
- Nécessité d'écrire autant d'applications que de plateformes
  - Dans des langages différents
    - Même s'il n'est pas très difficile de passer de l'un à l'autre, il est compliqué d'être efficace
  - Avec des outils différents
    - Des éditeurs et SDK spécifiques

Une application par plateforme



Compilation cross-plateform

- Plusieurs produits existent
  - 70% de code commun
  - 30% de code spécifique / plateforme



Xamarin

Positionnement IBM

- Accord IBM – Apple
  - http://www.ibm.com/mobilefirst/us/en/mobilefirst-for-ios/
- Accord IBM – Xamarin
  - http://xamarin.com/ibm
- Produits IBM
  - IBM Mobile first (worklight)
  - IBM Blue Mix Mobile Services (cloud)

En production

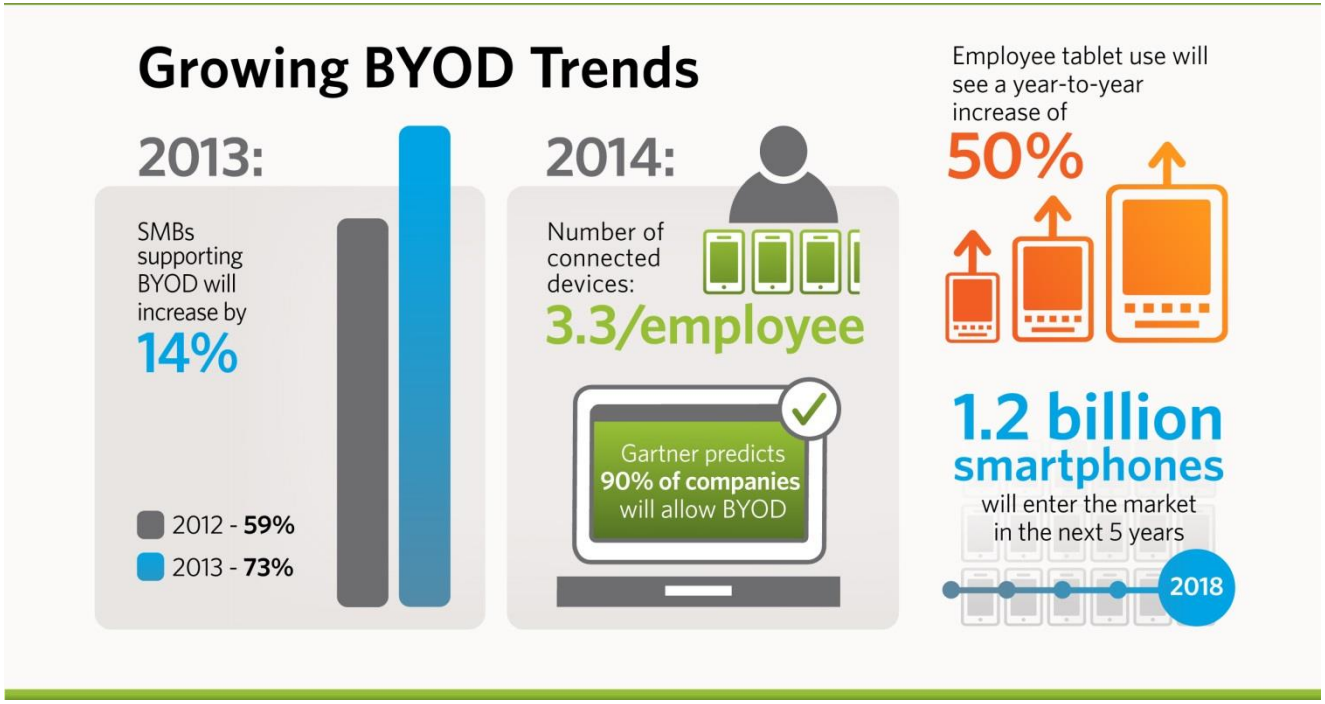
- On attaque rarement directement les machines back office
  - Sécurité
  - Disponibilité
  - Maîtrise de la charge
  - …

Solutions éditeurs

- Vous apportent des solutions clés en main aux problèmes courants
  - Données locales
    - Capacité de resynchroniser des données lorsque la connexion revient, gérer les deltas …
  - Sécurité
    - Identification des utilisateurs au plus tôt
  - Gestion de versions applicatives
    - Synchronisation avec les versions serveur
  - Gestion de flotte
    - Permet de gérer / forcer des versions
      - % version OS mobile et version serveur

Ouverture

- BYOD – Bring Your Own Device
  - Questions
    - Sécurité, responsabilité, sociales et juridiques
- Cloud, elearning, réalité virtuelle, réalité augmentée …
- Sujet libre



Power Systems

for Business

MERCI

IBM

Sources

- https://fr.pinterest.com/jeannehopkins/mobile-marketing-infographics/
- http://www.webdam.com/2014-mobile-marketing-infographic/
- https://fr.pinterest.com/infographmania/mobile-marketing-infographics/
- http://www.smartinsights.com/digital-marketing-strategy/internet-trends-2014-mary-meeker/
- http://techcrunch.com/2014/08/21/majority-of-digital-media-consumption-now-takes-place-in-mobile-apps/
- http://fr.slideshare.net/odemarez/applications-mobiles-quels-choix-43948480
- https://tomasid.wordpress.com/2013/10/08/applications-mobile-natives-vs-hybrides-quelle-est-la-difference/
- http://fr.slideshare.net/Developpeurs/par202
- http://www.ibm.com/developerworks/ibmi/library/i-using-jtopen/
- http://www.cnil.fr/documentation/fiches-pratiques/fiche/article/byod-quelles-sont-les-bonnes-pratiques/
- http://www.ravepubs.com/byod-part-1-enterprise-strategy-policy/
- http://www.thinkcomputers.org/infographic-why-we-use-mobile-media-more-than-our-tv/

www.gaia.fr