# Réflexion sur l'architecture générale de CC.NG

## Utilisation du logiciel

### Modes d'exploitation

Le logiciel peut être exploité de diverses manières :

1. Poste unique, isolé.
2. Plusieurs postes isolés, avec transmission de la comptabilité d'une machine à une autre.
   Support : clé USB ou synchronisation de type *DropBox*.
3. Plusieurs postes en réseau local (LAN).
4. Plusieurs postes connectés par Internet.

### Scénarios

Voici quelques scénarios qui nécessitent le partage des données entre plusieurs machines :

1. L'utilisateur comptabilise les factures émises au bureau mais tient sa comptabilité à domicile.
2. L'utilisateur travaille sur sa comptabilité au bureau mais transmet chaque mois la comptabilité à sa fiduciaire pour vérification, puis réintègre les modifications chez lui.
3. Plusieurs utilisateurs travaillent avec la comptabilité : une personne émet des factures, une autre personne paie les salaires, une troisième suit les encaissements bancaires et tient la comptabilité, un directeur consulte les résultats et prépare les budgets.
4. Deux succursales tiennent des comptabilités qui sont fusionnées dans une comptabilité « maître ».
5. Plusieurs entités tiennent des comptabilités qui sont consolidées dans une comptabilité « maître » (exemple de l'EERV : régions et canton).

## Accès aux données

L'accès aux données peut s'envisager de plusieurs manières :

1. Fichier sur disque, accédé directement par le logiciel.
   - Utilisé par Crésus Comptabilité 1…10.
   - Bien adapté pour un scénario où un seul utilisateur modifie les données.
   - Difficile à mettre en œuvre si on doit travailler à plusieurs (c'est possible, cf. OneNote).
2. Base de données relationnelle, accédée au travers de requêtes SQL (moteur intégré au logiciel ou tournant sur un serveur dédié).
   - Utilisé par CORE.
   - Bien adapté au multiutilisateur (transactions, atomicité, consistance).
   - Difficile d'avoir une solution efficace (par ex. calcul du solde d'un compte) sans implémenter un cache dans le logiciel, ce qui rend le travail en mode multiutilisateur complexe.
   - Dissonance entre entités et tables relationnelles.
3. Serveur de données, accédé au travers de requêtes de haut niveau (moteur intégré au logiciel ou tournant sur un serveur dédié).
   - Pas d'expérience chez nous.
   - Bien adapté à la manipulation de données de haut niveau, avec cache et multiutilisateur.
   - Report de la complexité « métier » du logiciel au serveur.

## Serveur de données

Pour moi, le concept de serveur de données mérite qu'on s'y attarde, dans la mesure où il semble offrir le modèle idéal pour une comptabilité.

Il peut être implémenté sous la forme d'une API (de style OData) qui permette à la fois un accès direct (poste seul, tout en mémoire dans l'application) et un accès distant par requêtes HTTP (en cas d'utilisation en réseau).

### Format des données

Les données sont stockées de manières très différentes en fonction de leur utilisation :

1. Dans le logiciel, elles sont manipulées sous la forme d'entités avec des types natifs (`decimal`, `bool`, `string`, `Date`, `FormattedText`, etc.), pour une productivité maximale (C#).
2. Entre le logiciel et le serveur, elles sont transférées dans un format texte (par ex. XML ou JSON), pour une interopérabilité maximale et un couplage restreint au maximum entre client et serveur.
3. Dans le serveur, elles sont stockées de manière compacte et optimisée pour les recherches (par ex. localité des données dans le cache du processeur), pour maximiser les performances.
4. Sur disque, les elles sont stockées dans un format texte (par ex. XML ou format maison), pour garantir une maintenance aisée des données.

### Corollaires

- Le logiciel client est totalement indépendant de la structure des données du serveur.
- Des logiciels tiers peuvent se connecter au serveur pour l'alimenter ou pour extraire des données.
- L'organisation des données en mémoire peut être optimisée en fonction du volume de données à traiter (une compta qui manipule des montants ne dépassant pas quelques millions, avec comme unité le centime, peut par ex. se contenter de nombres 32-bit), sans se bloquer en cas d'évolutions futures.
- Le stockage des données sur disque est totalement indépendant des contraintes imposées par les autres composants. Rien n'empêche de stocker des libellés de quelques MB, par exemple.

### Structure du programme

Le programme est découpé en (a priori) deux couches, ce qui partage clairement les responsabilités :

- Logiciel client :
  - Interaction avec l'utilisateur (commandes).
  - Saisie des données.
  - Mise en forme et présentation des données.
  - Impression.
- Logiciel serveur :
  - Extraction des données.
  - Recherches dans des extractions.
  - Stockage des données.

## Scénarios non résolus

Les scénarios déconnectés (qui sont aujourd'hui résolus par « FiduSync ») ne sont pas résolus par une structure client/serveur (à moins que le serveur ne soit accessible en permanence par le client distant, ce qui n'est souvent pas le cas). Ils doivent donc de toute manière être résolus par d'autres biais (import/export, par ex.), comme c'est déjà le cas maintenant.

PA, Yverdon-les-Bains, 18 juin 2012, rév.1