système privé d'information domestique

Avec Windows sur Raspberry Pi

Veikko Krypczyk (Allemagne)

Les réseaux et services d'information affichent la météo, les actualités, l'horaire des transports en commun, et plein d'autres données actuelles. Tout le monde peut en profiter chez soi, ou sur son lieu de travail, à condition de bien choisir le matériel et les logiciels pour que le système soit flexible.

Les réseaux et services d'information publique sont en vogue, et utilisables à des fins diverses dans le cercle privé ou professionnel. Un **s**ystème **p**rivé d'**i**nformation **d**omestique (disons SPID) pourrait par exemple fournir aux utilisateurs les informations suivantes :

- météo du jour, avec prévisions pour les jours à venir
- nouvelles, locales, nationales ou mondiales, éventuellement limitées à un domaine spécifique tel que sport ou culture
- diaporama
- programme TV ou cinéma
- horaire des transports en commun à proximité, avec mise à jour des retards éventuels
- localisation en temps réel des membres de la famille par leurs appareils mobiles
- prochains rendez-vous

Pour le secteur professionnel (SPIP), les applications possibles ne manquent pas selon l'utilisation prévue. Exemples :

- plans de travail et horaires, mis à jour en permanence selon le calendrier de l'équipe dans le nuage
- objectifs et données réelles d'une opération en cours
- actualité interne de l'entreprise
- informations pour les clients, les participants aux événements, etc.

La liste est ouverte et vous trouverez bien d'autres applications intéressantes. Dans ces exemples, la présentation de l'information est passive, mais des fonctions interactives pouvaient être intégrées, avec un écran tactile par exemple.

De tels systèmes d'information peuvent être achetés prêts à l'emploi ou produits par vos soins. Ce qui importe, c'est l'interaction appropriée entre matériel et logiciels. Si c'est vous qui construisez votre système, vous restez flexible ; vous pouvez l'adapter à vos besoins et ne dépendez pas du logiciel d'un fournisseur. Cet article présente l'approche pour un SPID ou *HomeInfoSystem* (HIS) personnalisable et facile à réaliser. Mon logiciel est mis à votre disposition gratuitement pour l'utilisation et un développement ultérieur [1]. J'illustre le principe par l'utilisation de services web (météo, actualités, photos), libre à vous d'étendre le SPID en fonction de vos besoins.

Architecture du matériel

Le système fonctionnera 24 h sur 24, 7 j sur 7. Sa consommation d'énergie doit être la plus faible possible, quel que soit l'affichage choisi (taille de l'écran). Il est donc conseillé d'utiliser un mini-ordinateur sous la forme d'un ordinateur monocarte, par exemple un Raspberry Pi, un Rock Pi 4, un Asus Tinkerboard S ou un Banana Pi M3. Ce qui plaide pour le Raspberry Pi, c'est son prix, ses performances (suffisantes), un bon support pour la tâche prévue et, surtout, la vaste expérience de ce mini-ordinateur. Le Raspberry Pi (**fig. 1**) est donc au cœur de notre système. Le modèle 3B convient parfaitement. On peut aussi utiliser le modèle 2B, puisque le système d'exploitation utilisé est Windows 10 IoT. Les performances du modèle 4 sont supérieures, mais il n'est pas officiellement pris en charge par W10 IoT.
Le Raspberry Pi est alimenté comme il se doit par une bonne alimentation stabilisée (5 V / 2 A réels !). La connexion au réseau passe par le câble ou sera sans fil (WLAN). Pour l'écran, le choix est vaste en matière de taille, résolution et interface tactile. La taille de l'écran dépend directement de l'utilisation prévue, et le choix va du (petit) écran sur table à un grand affichage lisible à distance. La résolution maximale du Raspberry Pi 2 ou 3 atteint 1 920 x 1 080 pixels (Full HD). Plus la résolution est élevée, plus les infos affichées sont nettes et plus le contenu pourra être dense. À vous de juger en fonction des goûts et des coûts, surtout s'il doit s'agir d'un écran tactile. Les critères pour ce dernier choix sont :

- l'écran est-il à la portée de l'utilisateur ?
- une interaction est-elle prévue ?
- l'utilisateur spectateur doit-il sélectionner quelque chose ?

La connexion entre Raspberry Pi et écran se fait toujours par un câble HDMI ; pour un écran tactile, il y aura une connexion supplémentaire. Sur le RPi ce sera un port USB, sur l'écran ça dépend, les modèles les plus récents utilisent USB-C.

Le moniteur doit également être alimenté. Pour mon SPID, j'utilise un écran tactile mobile Full HD de 13 pouces avec les connexions HDMI, USB-C pour le tactile, USB-C pour l'alimentation et une batterie intégrée. Une option consiste à monter le RPi sur un grand moniteur fixé au mur ou au plafond. Dans ce cas, le câble HDMI sera court.

Architecture du logiciel

Le système installé sur le Raspberry Pi est Windows 10 IoT. Voici les arguments qui justifient ce choix :

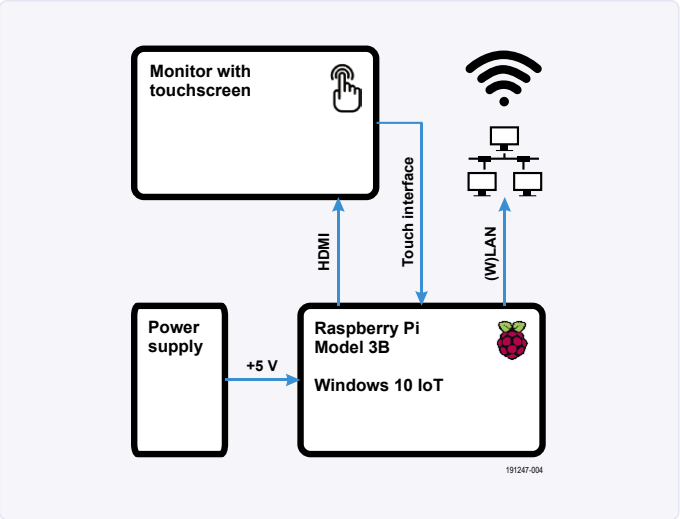

Figure 1. Structure du système ***HomeInfoSystem*** *(SPID).*

- mode *kiosque* actif et direct : exécution directe d'une application dans laquelle l'utilisateur a des droits d'accès limités
- options complètes pour concevoir l'interface utilisateur dans le langage de description XAML
- administration facile du RPi par l'interface graphique web
- conception, test et déploiement directement à partir de l'environnement de développement intégré Visual Studio
- mise à jour automatisée de l'application possible via le Store (intéressant pour une utilisation commerciale)

Sur le Raspberry Pi, vous devez d'abord installer Windows 10 IoT (cf. **encadré**). Pour la conception, il vous faut un PC (de bureau ou portable) avec une version à jour de Win 10, sur lequel l'application est créée puis testée. L'environnement de développement est Visual Studio.
Ne peuvent être exécutées sur Windows 10 IoT que les applications multi-dispositifs pour l'environnement d'exécution *Universal Windows Platform* (UWP), pas les applications Windows classiques. Basées sur .NET Core, les applications pour l'UWP sont créées en C# ou en *Visual Basic* .Net et avec l'aide de l'environnement de développement Visual Studio. L'édition *Community* gratuite [2] suffit. Dans l'installateur de *Visual Studio*, sélectionnez la charge de travail (*workload*) appropriée pour le développement d'applications pour l'UWP. Vous pourrez alors commencer à développer votre première application. Changez le type d'application, de X86 à ARM, car Raspberry Pi utilise un processeur ARM.
Les applications sont entièrement testées sur l'ordinateur de développement, et ce n'est que lorsqu'une version fonctionnera sans erreur que le «paquet d'applications» sera créé et installé sur le RPi via l'interface web. À ce stade, il est également possible d'exécuter l'application directement sur Raspberry Pi pour la tester et la déboguer. La seule exigence pour ce débogage à distance est que les deux ordinateurs soient sur le même réseau local. Comme les applications pour l'UWP sont basées sur le noyau .NET, on dispose de toute la puissance des API correspondantes. Pour de nombreux problèmes standards, vous pouvez utiliser des classes toutes faites. Comme langages de programmation, vous pouvez choisir C# ou Visual Basic .NET.
Sur le Raspberry Pi, une application fonctionne au premier plan et en mode plein écran (mode kiosque). L'interface utilisateur est conçue

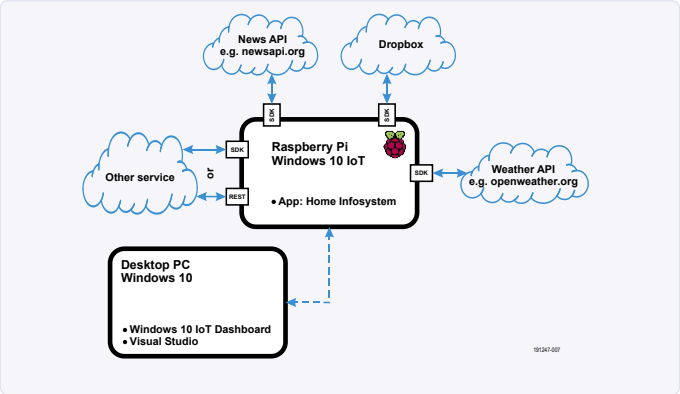

Figure 2. Le logiciel ***HomeInfoSystem*** *dans le système.*

de manière déclarative dans le langage de description XAML. À cette fin, des contrôles prédéfinis sont insérés dans des conteneurs de mise en page. De cette manière, on obtient une mise en page relative, qui s'adapte automatiquement aux différentes tailles et résolutions d'écran. L'éventail des contrôles disponibles comprend les éléments habituels tels que les boutons ou les champs de texte. Démarrez un nouveau projet, concevez une interface utilisateur minimale, exécutez l'application sur votre PC pour la tester, puis transférez le paquet d'application sur le Raspberry Pi pour boucler le cycle de développement.
L'élément central du système logiciel est l'app *HomeInfoSystem* (**fig. 2**), installée sur le RPi au moyen d'un paquet d'application et via l'interface web. Cette app est utilisée surtout pour présenter le contenu, c'est-à-dire les données récupérées sur les services externes, à l'exception des fonctions et contenus statiques tels que l'affichage de l'heure.
Un menu sur la gauche permet de sélectionner le contenu à afficher. L'application fait défiler automatiquement les contenus à un rythme déterminé. Dans le mode de présentation actif, le menu est caché ; l'icône du menu *hamburger* ☰ en haut en rappelle l'existence. Le changement automatique entre contenus est configurable individuellement pour chaque service. La durée de chaque photo (extraite de Dropbox) pourrait durer par exemple 45 s, alors que pour la météo 20 s devraient suffire. Contenu et services sont intégrables dans l'application de trois manières.

- *Interne :* n'affiche que des données internes comme l'heure. L'ensemble de la fonction, y compris les données, est complètement intégré statiquement dans le code source de l'application.
- *Externe (SDK) :* données fournies par des prestataires (météo, nouvelles, services de stockage dans le nuage (Dropbox) ou issues de l'intégration de l'agenda. Des kits de développement logiciel (SDK) sont disponibles pour un grand nombre de ces services. Un tel SDK est adapté à l'environnement de programmation spécifique, en l'occurrence .NET Core et C#. Comparée à une interface de programmation générique, basée p. ex. sur REST, l'avantage est ici qu'intégration et programmation sont simplifiées. Un tel SDK est généralement fourni par le fournisseur de services ou il est disponible sous forme de bibliothèque grâce à la communauté. Dans Visual Studio, un SDK est intégré via le gestionnaire de paquets NuGet.
- *Externe (REST) : s'il* n'y a pas de SDK, tout service peut être intégré en tant qu'interface REST via l'interface web générale. En utilisant la méthode habituelle du web (GET), vous envoyez une demande au service et recevez une réponse dans un format de données structuré (XML, JSON). Les données peuvent maintenant être analysées, interprétées, préparées graphiquement et affichées selon vos besoins. Les services en nuage, qui peuvent être utilisés par des applications tierces, offrent une telle interface et la documentation correspondante.

Lorsque vous utilisez des services externes – que ce soit via SDK ou REST – on vous demande généralement de vous inscrire. Une

Listage 1. Exemple de données météo de https://openweathermap.org.

```
{"coord": { "lon": 139,"lat": 35},
   "weather": [
  {
         "id": 800,
         "main": "Clear",
         "description": "clear sky",
         "icon": "01n"
  }
  ],
   "base": "stations",
   "main": {
      "temp": 281.52,
      "feels_like": 278.99,
      "temp_min": 280.15,
      "temp_max": 283.71,
      "pressure": 1016,
      "humidity": 93
   },
   "wind": {
      "speed": 0.47,
      "deg": 107.538
   },
   "clouds": {
      "all": 2
   },
   "dt": 1560350192,
   "sys": {
      "type": 3,
      "id": 2019346,
      "message": 0.0065,
      "country": "JP",
      "sunrise": 1560281377,
      "sunset": 1560333478
   },
   "timezone": 32400,
   "id": 1851632,
   "name": "Shuzenji",
   "cod": 200
}
...
```

INSTALLER WINDOWS 10 IOT SUR RASPBERRY PI

L'installation du système d'exploitation sur la carte SD du RPi se fait (ainsi que le développement du logiciel) à partir d'un PC sous Win10. Si nécessaire, mettez à jour la version de Win10 à l'avance. À l'heure d'écrire ces lignes, Windows 10 IoT Core est en version 1809 (Build 17763). Il vous faut donc au moins cette version. La carte SD est insérée dans un emplacement approprié (interne ou externe) du PC.

- **Tableau de bord :** Lorsque tout est prêt, téléchargez le fichier d'installation du tableau de bord de l'IdO Windows 10 [4], ce qui permettra de charger l'application complète.
- **Écrire sur la carte SD :** Dans le tableau de bord (**fig. A**), sélectionnez l'option de menu *Setup a new device* (= *Configurer un nouvel appareil)* et là, le type d'appareil correct (RPi 2 ou 3) et le lecteur cible, la carte SD. Il est utile de pouvoir écrire directement sur la carte SD le profil WLAN, le nom et le mot de passe du RPi. Dès que vous acceptez les termes de la licence, le téléchargement commence et le tableau de bord transfère le système d'exploitation sur la carte SD.
- **Démarrage du RP :** insérez la carte SD décrite ci-dessus dans le RPi et connectez le matériel (moniteur, écran tactile, alimentation) ainsi qu'une souris, un clavier et éventuellement un câble LAN. Le Raspberry Pi n'a pas d'interrupteur et démarre aussitôt qu'il est alimenté. L'installation initiale dure un peu. Après avoir choisi la langue, l'écran de démarrage de Win 10 IoT finit par apparaître. Aucune app personnalisée n'est encore en service.
- Le **Raspberry Pi est configuré à** partir d'un PC via le tableau de bord ou le *Windows Device Portal* (WDP) (**Fig. B**). Le modèle 3 de Raspberry Pi dispose d'un WLAN à bord. Windows 10 IoT reconnaît automatiquement le matériel, de sorte que l'installation se déroule sans problème. Pour être branché et configuré, le modèle 2 sans WLAN nécessite un *dongle* WLAN externe, ce qui est un peu plus compliqué que le modèle 3. La puissance de calcul plus élevée plaide également en faveur de la version 3.

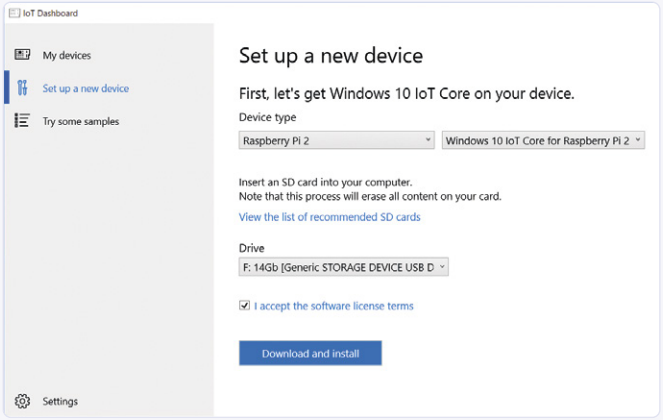

Figure A. Tableau de bord de Windows 10 IoT.

Le RPi avec Windows 10 IoT est accessible avec un navigateur. Le tableau de bord de configuration est accessible via l'adresse IP du RPi. Vous pouvez y installer, démarrer et quitter une application, consulter des informations sur l'état de la charge du système, gérer les connexions réseau ou arrêter et redémarrer le système. Vous pouvez également utiliser le *Power Shell* (ligne de commande). Ce type de configuration permet la plupart des options de configuration, mais sans confort. C'est la seule façon d'installer des pilotes de matériel supplémentaires. Ici nous faisons tout via l'interface web.

Utilisation et développement du code

Le logiciel du projet [1] peut être utilisé tel quel. Il contient : Page d'accueil (affichage d'un texte de bienvenue), horloge, spectacle photo avec intégration de Dropbox, affichage des données météo et des actualités. Une configuration minimum des services est nécessaire avant utilisation :

- *Dropbox* : dans votre propre compte Dropbox, passez à la zone des développeurs (console) et créez une application. Autoriser l'accès au répertoire de l'app et produisez la clé secrète à saisir plus tard dans les paramètres de l'app.
- *Nouvelles* : enregistrez-vous [5] et produisez une clé personnelle pour accéder gratuitement à la zone de nouvelles. La clé est nécessaire, car le nombre d'accès est limité. Vous enregistrez la clé directement dans le fichier *NewsControlViewModel.cs* (*ligne 95*).
- *Météo* : Produisez une clé d'accès gratuit et stockez-la dans le fichier *OpenWeatherMapProxyForecast.cs* (ligne 13).

La saisie des clés des services d'information et de la météo se fait directement dans Visual Studio. Ensuite, recompilez le projet et créez les paquets d'application à installer sur le Raspberry Pi.

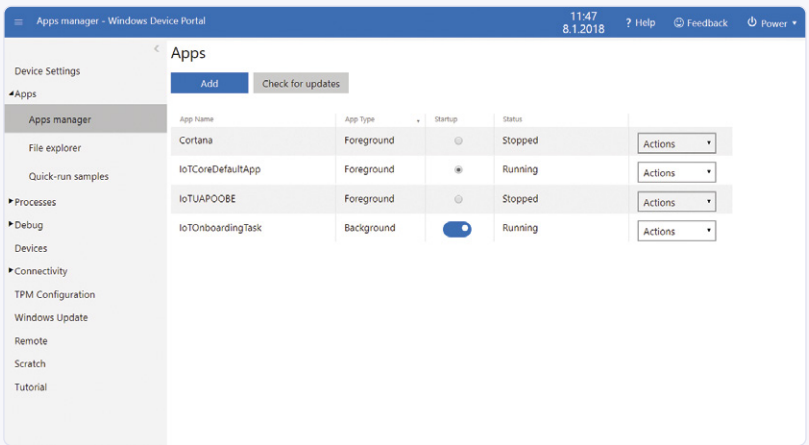

Figure B. Interface web pour la configuration du Raspberry Pi.

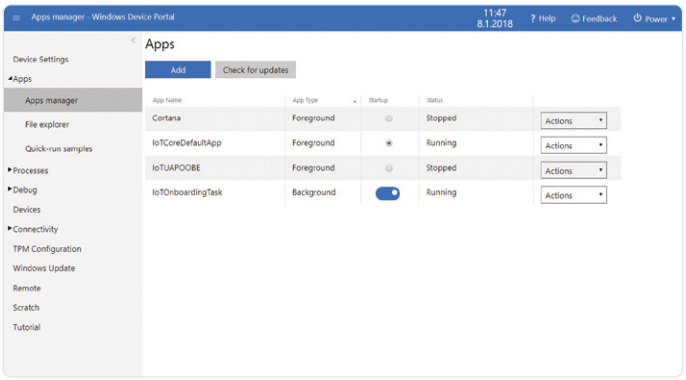

Figure 3. Options de l'API ***OpenWeather.***

identification unique est nécessaire. Le service vous fournit une clé à transmettre avec chaque demande. De nombreux services sont gratuits pour l'usage privé et commercial limité, même si le nombre de récupérations de données (appels/jour) est limité ou si les données sont plus rudimentaires. Un grand nombre de services sont satisfaisants, même en version gratuite.

Services

L'app *HomeInfoSystem* combine plusieurs options de contenu (services), extensibles et combinables à volonté. Ici nous décrivons la connexion et la mise en œuvre de certains services. Vous pourrez en déduire des possibilités d'extension :

- *Intégration du service météorologique* OpenWeather [3]. En plus des services payants, les services *Current weather* et *5 days/3 hour forecast* (*météo actuelle* et *prévisions 5 j/3 h)* sont gratuits (**fig. 3**). Après l'enregistrement, une clé API est produite avec laquelle les données météorologiques peuvent être récupérées. La convention d'un tel appel est par exemple : `api.openweathermap.org/data/2.5/weather?q=&appid=` Les paramètres `{city name}` et `{your api key}` devront être remplacés par les infos idoines. La réponse est un ensemble de données au format JSON (l**istage 1)**, assez facile à lire. Elle commence par les coordonnées du lieu, suivies d'informations de température, de pression atmosphérique, de couverture nuageuse, etc. L'ensemble est décrit en détail sur le site du service. Un SDK existe pour l'application, ce qui facilite l'utilisation du service. Le l**istage 2 d**onne le code source correspondant.
- *Intégration du stockage en nuage :* pour afficher des photos, il est préférable d'accéder à l'un des systèmes de stockage en nuage habituels tels que Dropbox ou OneDrive. Pour l'accès via des applications externes, ces services offrent leur propre API et également un SDK pour de nombreux langages de programmation. La version gratuite de Dropbox offre 5 Go d'espace de stockage, entre autres pour des photos. Sur le portail en ligne, vous pouvez accéder directement à la zone des développeurs. Installez une application sur Dropbox (**fig. 4**). Vous pouvez utiliser le portail pour configurer l'application, par exemple quel dossier de Dropbox doit être accessible. Il existe plusieurs façons d'accéder aux données (photos) de Dropbox depuis l'extérieur du portail, en utilisant un nom d'utilisateur et un mot de passe (authentification) ou sans interaction de l'utilisateur avec une clé unique (*Key*, *App Secret*). Cette dernière variante est optimale pour nous, car l'application s'identifie simplement sur Dropbox avec une telle clé. Nous utilisons le SDK pour .NET

Listage 2. SDK (extrait) pour l'utilisation du service openweathermap.org.

```
// Note : Le type de données RootObject contient une structure arborescente de données météo.

public async static Task<RootObject> GetWeather(string location)
  {
    try
    {
      var httpClient = new HttpClient();
      string uri = basisUri + endPoint + "?q=" + location + "&appid=" + apiKey + "&units=metric";
      var response = await httpClient.GetAsync(uri);
      var result = await response.Content.ReadAsStringAsync();
      var serializer = new DataContractJsonSerializer(typeof(RootObject));
      var ms = new MemoryStream(Encoding.UTF8.GetBytes(result));
      var data = (RootObject)serializer.ReadObject(ms);
      return data;
    }
    catch
    {
      return null;
    }
  }
```

Core, intégrable directement dans notre application. La lecture des photos de Dropbox devient un jeu d'enfant : le **listage 3** est la section de code correspondante. Les utilisateurs peuvent prendre des photos avec leur téléphone et les enregistrer directement dans Dropbox depuis leur appareil mobile.

- *Contenu statique :* Le contenu statique est implémenté directement dans l'application en utilisant la logique du programme, par exemple l'affichage de l'heure avec une représentation visuelle spéciale.

Cette liste présente des exemples de mise en œuvre de contenu pour l'app *HomeInfoSystem*. Pour les applications professionnelles (*EnterpriseInfoSystem*), les mêmes connexions doivent être créées pour certains services spécifiques.

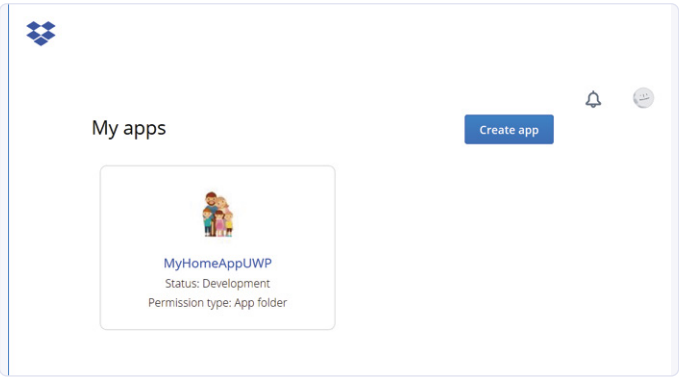

Figure 4. Sur le portail des développeurs de Dropbox, vous devez mettre en place une application.

Listage 3. SDK (extrait) pour accéder aux données dans Dropbox.

```
public static async Task<ObservableCollection<string>> GetFilesFromDropBoxAsync()
  {
    ObservableCollection<string> itemsList = new ObservableCollection<string>();
    // chemin d'accès pour l'enregistrement du fichier
    StorageFolder storageFolder = ApplicationData.Current.LocalFolder;
    try
    {
      using (var dbx = new DropboxClient(ProgrammSettings.DropBoxAppToken))
      {
        // liste de tous les fichiers dans le dossier
        var list = await dbx.Files.ListFolderAsync(string.Empty);
        // boucle sur tous les dossiers
        for (int i = 0; i < list.Entries.Count; i++)
        {
          string localFileName = storageFolder.Path + "/" + list.Entries[i].Name;
          if (File.Exists(localFileName) == false)
          {
            // télécharger le fichier
            var file = await dbx.Files.DownloadAsync(list.Entries[i].PathLower);
            // extraction du flux de données
            Stream stream = await file.GetContentAsStreamAsync();

            // écrire le flux de données dans un fichier local
            using (var fileStream = File.Create(localFileName))
            {
              (await file.GetContentAsStreamAsync()).CopyTo(fileStream);
            }
          }
          itemsList.Add(localFileName);
        }
      }
    }
    catch
    {
    }
    return itemsList;
  }
```

Extensions

- Les services décrits ci-dessus ne sont que des exemples et devraient vous encourager à lancer votre propre SPID. Vous trouverez rapidement les services que vous souhaitez utiliser. Voici des idées d'extensions possibles :
- Le *HomeInfoSystem* pourrait également afficher les valeurs de différents capteurs. Les valeurs des capteurs sont transmises à une infrastructure IoT et sont exploitées par l'application à certains intervalles. Les électroniciens peuvent construire des extensions matérielles complètes et les intégrer de manière flexible dans le système.
- *Afficher le contenu en fonction de critères temporels :* Certains contenus peuvent être fournis avec une référence temporelle. Il est p. ex. utile de donner les informations en temps réel relatives aux transports en commun en semaine entre 7 et 8 h, mais en fin de semaine, vous serez peut-être plus intéressé par des infos sur les programmes du ciné du quartier, que vous trouverez grâce à un programme de cinéma API.
- *Services mobiles :* Vous pourriez ajouter une application mobile au système. Les utilisateurs peuvent envoyer des données à un serveur (*backend*) où l'app récupère ces données. Diverses applications sont envisageables. Une autre application intéressante est la géolocalisation. Les utilisateurs mobiles pourraient, après approbation, transmettre automatiquement leur position géographique à un serveur où l'app récupérerait régulièrement cette position pour l'afficher sur une carte. De cette façon, on peut savoir à tout moment où se trouvent tous les membres d'une famille.
- *Services en ligne : D*'autres services intéressants peuvent être intégrés via une API publique : Intégration dans l'agenda de divers services tels que l'agenda de Microsoft ou Google…
- *Structure d'interface alternative :* au lieu de passer manuellement d'un contenu à l'autre ou en fonction du temps, vous pourriez également créer une vue d'ensemble, particulièrement intéressante sur les grands écrans : vous pourriez afficher la météo d'un côté et les nouvelles et photos de l'autre.

Les d'idées pour étendre *HomeInfoSystem* ne manquent donc pas. Sans parler de connecter des capteurs externes (IoT) par exemple pour pouvoir jeter un rapide coup d'œil sur votre résidence secondaire (caméra). Pour les entreprises aussi, les possibilités d'application sont innombrables.

Conclusion & perspectives

Tant à la maison que dans une entreprise, un tel système d'information est fascinant. À la maison, il peut fournir des informations toujours à jour (météo, actualités, dates) et dans l'entreprise, il peut présenter des données actuelles importantes (quantités, dates des réunions d'équipe). Le plus intéressant, ce sont les possibilités d'adapter tout cela à vos propres besoins sans difficulté. Après avoir choisi la configuration matérielle, en particulier l'écran, le logiciel décrit ici fournit d'emblée un ensemble de fonctions de base, qui peuvent ensuite être étendues et adaptées de manière flexible. Le code source [1] est ouvert. Dans Visual Studio 2019, vous pouvez l'étudier et le modifier selon vos besoins.

191247-03

LIENS

[1] **logiciel du projet** : http://www.elektormagazine.fr/191247-01
[2] **VisualStudio** : https://visualstudio.microsoft.com/downloads/
[3] **OpenWeather** : https://openweathermap.org/
[4] **Windows 10 IoT-Core** : https://docs.microsoft.com/en-us/windows/iot-core/downloads
[5] **News-API** : https://newsapi.org/