3/17/2022

* Xiaogang Li (API2, Site Web, documentation) : [xiaogang.li.pwc@gmail.com](mailto:xiaogang.li.pwc@gmail.com)
* Marc-André (S.O) : [stjma.code@gmail.com](mailto:stjma.code@gmail.com)
* Joseph Kasumba (API3, design, documentation) : [joseph\_kasumba@yahoo.ca](mailto:joseph_kasumba@yahoo.ca)
* Richard Tchicou (API1, documentation, coordination des rencontres): [richard@nonstandardlogic.com](mailto:richard@nonstandardlogic.com)

**Processus: IT-02PROCESSUS No 20-MOT-EA-HRS-98277-2Transports Canada**

Documentation de l’Évaluation TC

Contenu

[Architecture et Design 3](#_Toc98425609)

[Site web 3](#_Toc98425610)

[API 3](#_Toc98425611)

[Technologie 3](#_Toc98425612)

[Prérequis 4](#_Toc98425613)

[Exécution 4](#_Toc98425614)

[Téléchargement du package complet 4](#_Toc98425615)

[Exécution des projets 4](#_Toc98425616)

[Résultats 5](#_Toc98425617)

[URL utilisés 5](#_Toc98425618)

[Site Web 5](#_Toc98425619)

[Annexe Site Web 6](#_Toc98425620)

**Send source file to appropriate API (Call API-x Post) x4**

**Search function *GET* of the data** **(Call API-x GetALL) x4**

**Search function *GET* of the data** **(Call API-x GetByID) x4**

**Json Results**

**Site Web**

**Post function in API-1 to receive CSCompVehicleRecallStart.json**

Get function to return All data api1.json

Using this data, each team member will need iterate through each record value set to call the VRD and extract additional data elements described further below. **The final results (api1.json) should be stored and used to provide the input for the next API, along with using it for the get functionality.**

The final results **(api1.json)**: using it for the **get ALL** functionality.

Get function to return json data by key word

The final results **(api1.json)**: using it for the **Search record** functionality **by key word Recall Number**

**Post function in API-2 to receive api1.json**

Get function to return All data **api2.json**

Using this data, each team member will need iterate through each record value set to call the VRD and extract additional data elements described further below. **The final results (api2.json) should be stored and used to provide the input for the next API, along with using it for the get functionality.**

The final results **(api2.json)**: using it for the **get ALL** functionality.

Get function to return json data by key word

The final results **(api2.json)**: using it for the **Search record** functionality **by key word Category**

**Web Services (Example of API-1 et API-2)**

# Architecture et Design

Le diagramme à la page 1 présente le flow d’information du site web aux API. Il se base sur 2 API à titre indicatif seulement.

Chaque API dispose de méthodes suivantes :

* POST : Recevoir les données à partir du site web. Ces données sont soit en provenance du fichier source « **CSCompVehicleRecallStart.json** » fourni, soit en provenance du fichier généré par l’un des API.
* Get ALL : Obtenir toutes les données générées par chaque API en format json. Voici le nomenclature des fichiers générés par chaque API.
  + API1 🡺 api1.json
  + API2 🡺 api2.json
  + API3 🡺 api3.json
  + API4 🡺 api4.json
* Get ById : Permet rechercher les données dans le json généré avec identifiant. Ceci dépend de l’API tel qu’instruit dans l’énoncé.

**Il est important de prendre connaissance des prérequis pour l’exécution de du projet. Le Package de la solution est fournis sur GitHub.**

## Site web

Il permet :

* + de téléverser le fichier source « CSCompVehicleRecallStart.json »
    - Bouton « Upload »
  + d’appeler chaque API
    - Les listes déroulantes permettent de sélectionner la source et la destination de l’appel
  + d’afficher les résultats de chaque appel

**La section** [**Annexe Site Web**](#_Annexe_Site_Web) **est fournie avec plus de détails sur le site. Merci de le consulter.**

## API

* + Les API ne sont pas tous développé dans la même technologie,
  + Les noms des fonctions et classes ne suivent pas forcément les mêmes conventions, mais sont censés accomplir les mêmes résultats,

# Technologie

* Site Web : php, Ajax, html, css, WAMP
* API1 : .net core 6 C#
* API2 : php, Ajax, html, css, WAMP
* API3 : .net core 3.1 C#
* API4: S.O

# Prérequis

* Machine : Windows
* Site Web : nécessite l’installation d’un environnement WAMP. Consulter la section [**Annexe Site Web**](#_Annexe_Site_Web) consacré au site web.
* API1 : Veuillez installer .net Core 6 Hosting Bundle 🡪 [**Télécharger ici**](https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/runtime-aspnetcore-6.0.3-windows-hosting-bundle-installer)
* API2 : Veuillez installer WAMP 64 bit 🡪 Télécharger ici 🡪[**Téléchager ici**](https://www.wampserver.com/en/)
* API3 : Veuillez installer .net Core 3.1 Hosting Bundle 🡪[**Téléchager ici**](https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/runtime-aspnetcore-3.1.23-windows-hosting-bundle-installer)

IDE :

* MS Visual studio code (API1 et API2)
* MS Visual Studio (API3)

# Exécution

## Téléchargement du package complet

* Ouvrir le lien Github public suivant : <https://github.com/TransCa98277>
* Télécharger tous les dossiers visibles (SiteWeb, API-x, Doc, etc.)
  + Doc contient : le fichier de documentation complet
  + SiteWeb contient : le folder `Evalu98277` qu’on doit le copier au complet dans le répertoire `C:\wamp64\www\` du WAMP server
  + API-1 contient:
    - Le Release Exécutable :
    - Le code source
  + API-2 contient:
    - Le Release Exécutable : le folder `TransCa98277` qu’on doit le copier au complet dans le répertoire `C:\wamp64\www\` du WAMP server
    - Le code source
  + API-3 contient:
    - Le Release Executable : [API3.PackageRelease\_Final.zip](https://github.com/TransCa98277/API-3/blob/main/API3.PackageRelease_Final.zip)
    - Le code source
  + API-4 contient: S.O
* Dézipper tous les fichiers téléchargés sous un même répertoire

## Exécution des projets

* API1 : Rouler le fichier : **API3.Package\_Final\.exe**
* API2 : **Suivre les instructions …**
* API3: Rouler le fichier : **API3.Package\_Final\api.web.exe**
  + Il est configuré pour ouvrir avec le port 5004 par défaut : par exemple;

Get 🡪  <http://localhost:5004/api/VehicleRecall/TestApi>

🡪 Résultat: "Test Action Task"

* + Il crée 2 dossiers :

Données (dataSource.json et api3.json): **API3.Package\_Final\Api3Data**

Erreur logs : **API3.Package\_Final\Api3logs**

* API4 : S.O
* **Rouler le site Web : Suivre la section** [**Annexe Site Web**](#_Annexe_Site_Web)

# Résultats

## URL utilisés

Voici la liste des principaux url appelés par le site web

|  |  |  |  |
| --- | --- | --- | --- |
|  | Post | Get ALL | Get By ID |
| API1 | http://localhost:5000/api1/vehicles | http://localhost:5000/api1/vehicles | http://localhost:5000/api1/vehicles/2017182 |
| API2 | <http://localhost/TransCa98277/webservice.php?action=receivePost> | <http://localhost/TransCa98277/webservice.php?action=getResultAPI&api=2&recur=n> | [http://localhost/TransCa98277/webservice.php?action=getResultAPI&api=2&recur=n&filter=["recallNumber|2015441](http://localhost/TransCa98277/webservice.php?action=getResultAPI&api=2&recur=n&filter=%5b%22recallNumber|2015441)"] |
| API3 | <http://localhost:5004/api/VehicleRecall/AcceptInput> | <http://localhost:5004/api/VehicleRecall/GetAllData> | <http://localhost:5004/api/VehicleRecall/GetBySystemType/2017182> |
| API4 | S.O | S.O | S.O |

## Site Web

**Consulter la section** [**Annexe Site web**](#_Annexe_Site_Web)**.**

# Annexe Site Web

**Prérequis et setup du serveur web (WAMP server)**

1. Installer le « **Wampserver 3.2.6 64 bit x64** » <https://www.wampserver.com/en/> sur le PC.
2. Laisser toutes les options par défaut jusqu’à la fin d’installation
3. Le serveur web Apache démarré automatiquement après l’installation et on voit son icône en bas à gauche de l’écran.

Graphical user interface, text, application

Description automatically generated

Si le serveur WAMP n’a pas démarrage automatiquement, aller dans le menu de démarrage et cliquer sur l’icône de WAMP, il faut être administrateur du PC pour pourvoir lancer le serveur WAMP. Il faut aussi noter que si on redémarre le PC le WAMP Server doit être démarré manuellement avec cette façon.

Graphical user interface, text, application

Description automatically generated

1. Tester l’URL <http://localhost/> avec un browser (IE, Chrome) si on voit la page suivante, c.-à-d. que le serveur WAMP roule correctement.

Text

Description automatically generated with medium confidence

1. Copier les codes sources de la page web du groupe et de l’API 2 de Xiaogang LI dans le répertoire `www` du serveur WAMP.

* <https://github.com/TransCa98277/SiteWeb>
* <https://github.com/TransCa98277/API-2>

Graphical user interface

Description automatically generated with medium confidence

1. Pour tester l’ensemble d’APIs on utilise cet URL : <http://localhost/Evalu98277/> , la page d’accueil ressemble a suivant :

Graphical user interface, text, application, email

Description automatically generated

1. Après avoir uploadé le « Start File », l’interface web ressemble à suivant :

Text

Description automatically generated

1. Après avoir finir un POST, l’interface est comme suivant :

Text

Description automatically generated

1. Pour recommencer le test à zéro :

* Supprimer les JSON ramasses par la page web se trouvent dans le répertoire « C:\wamp64\www\Evalu98277\json\ ».
* Supprimer le résultat de traitement d’API2 qui se trouve dans le répertoire ` « C:\wamp64\www\TransCa98277\json »