Requêtes d'optimisation déportées : utilisation du solveur toulbar2 par services web

Nathalie Rousse, Simon De Givry

▶ **To cite this version:**

Nathalie Rousse, Simon De Givry. Requêtes d'optimisation déportées : utilisation du solveur toulbar2 par services web. INRAE, MIAT, Toulouse. 2023. hal-04326148

HAL Id: hal-04326148
https://hal.inrae.fr/hal-04326148

Submitted on 6 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Requêtes d'optimisation déportées : utilisation du solveur toulbar2 par services web

Nathalie Rousse — **Equipe Record**

Simon de Givry — **Equipe SaAB**

Unité MIAT

INRAE

Décembre 2023

Table des matières

A. Contexte....3
A.1. Objet....3
A.2. Les services web ws....3
A.3. Préparation pour toulbar2....4
A.4. Représentation....5
B. Usages....6
B.1. Usages en l'état....6
B.2. Usages moyennant des développements supplémentaires....8
B.2.1. Introduction....8
B.2.2. Développements côté client....8
B.2.3. Développements côté serveur....10
B.2.4. Autre exemple avec des développements des 2 côtés (client et serveur)....13
C. Conclusion....16
Annexes....17
Annexe 1. Containers pour (py)toulbar2....17
Annexe 2. Documentation pour (py)toulbar2....18
Annexe 3. Exposition depuis des pages du site de toulbar2....20
Annexe 4. Logiciels produits et/ou exploités....22
A4.1. Logiciel toulbar2 22
A4.2. Logiciel ws....22
A4.3. Logiciel sudoku.py....23
A4.4. Logiciel toulbar2_visual_sudoku_puzzle.py....23
A4.5. Logiciel examtt 23
A4.6. APK VisualSudoku....24
A4.7. Pages du site toulbar2....24
A4.8. Page launch_sudoku.html 24
A4.9. Commandes Curl....24

A. Contexte

A.1. Objet

Les équipes **SaAB**[1] et **Record**[2] de l'unité **MIAT**[3] ont mené une collaboration basée d'une part sur le logiciel **toulbar2**[4] de l'équipe **SaAB** et d'autre part sur la technologie des services web, thématique portée par Nathalie Rousse au sein de la plateforme **Record**. L'outil logiciel **toulbar2** est un solveur pour réseaux de fonctions de coût, qui est développé dans l'équipe **SaAB** avec pour intégrateur Simon de Givry, responsable de l'équipe **SaAB**. La collaboration prend appui sur le logiciel de services web **ws**[5] pré-existant qui a évolué à cette occasion, développé par Nathalie Rousse à titre exploratoire.

Le questionnement portait sur l'idée de disposer de services web pour **toulbar2** et l'objectif était d'illustrer le potentiel que cela représenterait. Pour cela, il a été pris comme fil rouge la **résolution de grilles de sudoku** avec **toulbar2**. A partir de ce fil rouge, différents cas d'usages ont été déclinés.

L'objet de ce document est de présenter ces cas d'exploitation de **toulbar2** sous forme de services web.

A.2. Les services web ws

Démonstrateur ws :

Le démonstrateur de services web **ws**, mis en ligne de manière exploratoire et temporaire, permet d'exécuter sous forme de services web des logiciels préalablement mis sous **containers**.

Ressources :

URL de mise en ligne (temporaire) du **prototype des services web ws :** http://147.100.179.250 .

Spécification de l'**API des services web ws :** http://147.100.179.250/ws/html/api .

Productions associées :

- dépôt du logiciel **ws** : https://forgemia.inra.fr/nathalie.rousse/ws .
- dépôt ws_deliv documentant[6] des logiciels hébergés par les services web **ws** : https://forgemia.inra.fr/nathalie.rousse/ws_deliv (privé).

Infrastructure :

L'exécution des logiciels (containerisés) hébergés par les services web **ws** a lieu :

- soit directement sur la Machine Virtuelle[7] hébergeant le logiciel **ws**. Dans ce cas le service d'exécution est 'gratuit'.
- soit sur d'autres ressources rendues accessibles à partir d'un point de montage. La seule ressource actuellement implémentée est le cluster Muse de Montpellier. Pour y avoir recours, le logiciel **ws** a besoin d'un compte Muse : c'est le compte **record** qui est utilisé systématiquement (ie sur lequel sont débitées les heures consommées quel que soit le logiciel containerisé exécuté) et de manière transparente pour les utilisateurs de **ws**.

1 Equipe de recherche SaAB (Statistiques et Algorithmique pour la Biologie)
2 Plateforme Record de modélisation et de simulation informatique des agro-écosystèmes
3 Unité MIAT (Mathématiques et Informatique Appliquées de Toulouse)
4 Voir « **A4.1** » en « Annexe 4 »
5 Voir « **A4.2** » en « Annexe 4 »
6 Procédure de livraison à **ws**, source de la documentation en ligne, exemples d'exécution par appels à **ws**
7 Machine Virtuelle qui est 'fournie' par MIAT

Gestion des droits d'accès :

► **Droits d'accès, authentification, accès public :**

- Au niveau du site des services web **ws**, tout (pages d'informations et requêtes) est en accès public sauf les requêtes d'exécution de logiciels : requêtes **POST software/run/{name}** et **POST software/muse/run/{name}**.
- Le logiciel **ws** gère des **comptes** 'utilisateurs' pouvant être associés à un individu ou un groupe de personnes[8] ou un projet etc. Pour être en mesure de poster des requêtes demandant l'exécution de logiciels, il faut posséder un **compte** 'utilisateur' qui permettra de demander un **token** (requête **POST jwt/obtain**) à joindre ensuite à ses requêtes (en tant que paramètre 'jwt'). Un token a une durée de vie limitée (configurable au niveau du logiciel **ws**).

► **Hiérarchie des droits d'accès :**

- Des risques de 'mauvaise' utilisation des ressources (taille des problèmes soumis…) ont été soulevés notamment en imaginant utiliser les services web dans le cas de travaux pratiques destinés à des étudiants. La première solution apportée a été d'ajouter comme fonctionnalité au logiciel **ws** une gestion de **niveaux de droits** des utilisateurs : ou bien un compte donné n'a pas accès à l'exécution d'un logiciel donné, ou bien il en possède un certain niveau de 'droits' d'exécution :
  - 1 (visitor) : has minimum rights, no access to some parameters.
  - 2 (user) : has access to most of parameters but maybe with limited values domain.
  - 3 (superuser) : has access to most of parameters generally without limitations.
  - 4 (admin) : has all rights.
- Dans les usages qui ont été déployés, ce sont d'autres solutions qui ont permis d'encadrer les utilisations des services web **ws**. Elles seront détaillées plus loin (voir « B.2. Usages moyennant des développements supplémentaires »).

A.3. Préparation pour toulbar2

Containerisations :

En préalable à son utilisation par services web **ws**, **toulbar2**[9] a été mis sous container ainsi que son API Python **pytoulbar2**. Les images Docker produites sont générées en Intégration Continue avec le service de production **GitHub Actions** (gratuit). Deux images Docker « **toulbar2** » et « **pytoulbar2** » ont été livrées sur le dépôt **GitHub** du projet **toulbar2** (voir ligne « package » du tableau descriptif). De plus en cours de travaux (voir « B.2.4 Autre exemple »), il s'est avéré utile de produire une 3ème image Docker « **pytoulbar2plus** » dans laquelle sont pré-installées en plus de **pytoulbar2** un certain nombre de librairies Python, afin de ne pas avoir à commencer par le faire à chaque début d'exécution d'un logiciel les requérant. Ceci permet de réduire les temps d'exécution aux dépends de la taille du container (voir ligne « taille » du tableau descriptif).

Tableau descriptif des containers : voir « Annexe 1. Containers pour (py)toulbar2 ».

Requêtes génériques ws :

Une fois ces containers produits et hébergés par les services web **ws**, les requêtes génériques des services web **ws** ont pu être utilisées pour demander l'exécution de **(py)toulbar2**.

Avec ***name=toulbar2*** ou ***pytoulbar2*** :

8 Exemple : compte '**recordteam'** dédié à l'équipe **Record**

9 Voir « **A4.1** » en « Annexe 4 »

- Requête **POST software/run/{name}** (exécution sur la Machine Virtuelle)[10] : Dans ce cas l'utilisateur reste en attente de la réponse retournée en fin d'exécution.
- Requête **POST software/muse/run/{name}** (exécution sur cluster Muse)[11] : Dans ce cas l'utilisateur reprend la main sans attendre la fin d'exécution, il gère une séquence de requêtes :

  (1) POST software/muse/run/{name} pour demander l'exécution ;
  (2) GET software/muse/state/{name}/{key} pour s'informer sur l'avancement du process ;
  (3) GET software/muse/run/{name}/{key} pour demander le résultat.

Dans les 2 cas il faut s'authentifier (token requis en paramètre 'jwt' de requête).

Documentation, exemples d'appels : voir « Annexe 2. Documentation pour (py)toulbar2 ».

A.4. Représentation

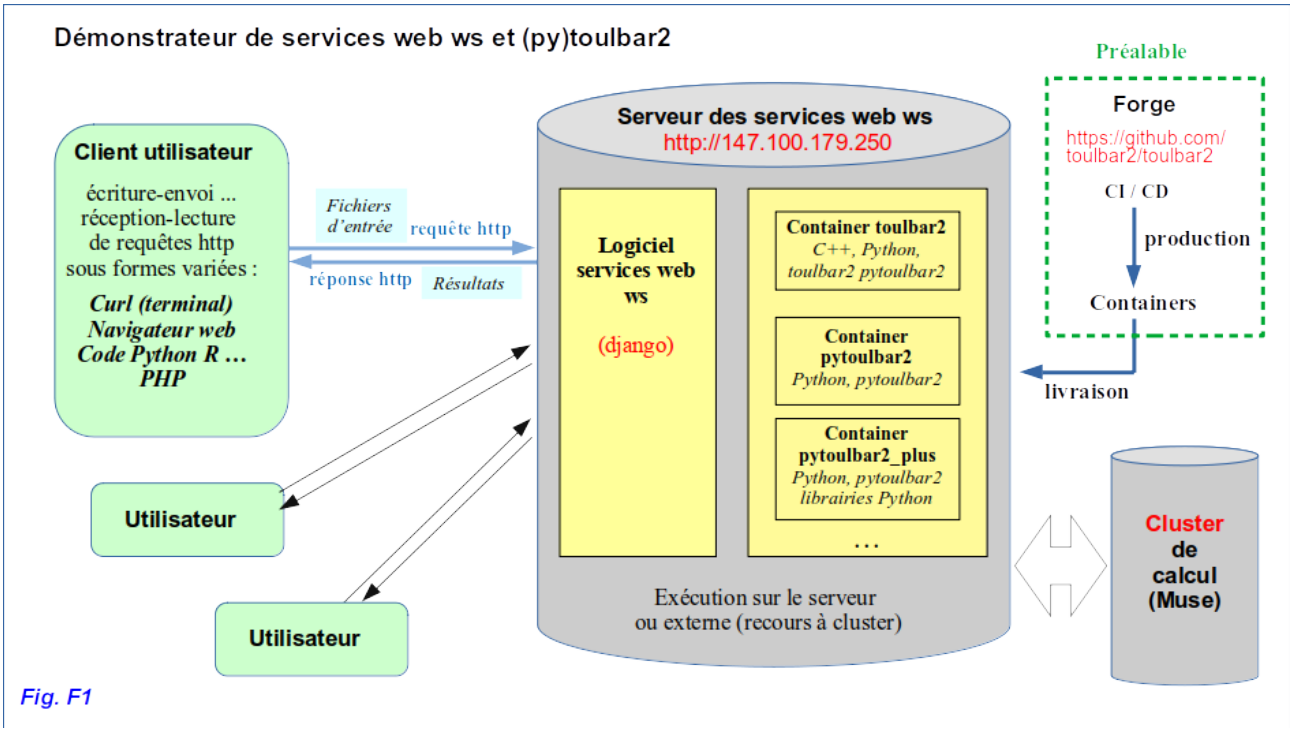


Fig. F1

10 Documentation : http://147.100.179.250/ws/html/api/post_software_run_name.html

11 Documentation : http://147.100.179.250/ws/html/api/post_software_muse_run_name.html

B. Usages

Dans l'idée de disposer de **toulbar2** sous forme de services web, l'aspect séduisant a priori était l'accessibilité de **toulbar2** sans avoir à l'installer (outil distant) ainsi que l'accès à des ressources de calcul (cluster Muse).

B.1. Usages en l'état

Utilisation directe des requêtes génériques :

Un premier niveau d'utilisation consiste en manipuler soi-même les requêtes génériques des services web **ws** (POST software/run/{name}…). Il s'agit de construire ses requêtes, côté client, sous la forme de son choix (programme informatique, commande en ligne…).

Cet usage s'adresse à des utilisateurs ayant des connaissances générales en terme de services web (savoir manipuler des requêtes HTTP) et connaissant également les spécifications de l'API **ws** (http://147.100.179.250/ws/html/api). De plus il leur faut posséder un compte 'utilisateur' puisque ces requêtes nécessitent une authentification.

Exemple Ex1 :

- exemple[12] présenté en **Fig. F2**.
- le code python **sudoku.py** utilise **pytoulbar2** pour résoudre une grille de sudoku lue dans un fichier d'entrée **valid.csv.xz**, et retourne en sortie les valeurs numériques de la grille solution.
- le script **ws.sh** contient la commande d'exécution du code python « **python3 sudoku.py** » (résolution du problème).
- l'utilisateur, ici *sur* ***terminal***, utilise l'outil **curl** pour envoyer sa requête (demande d'exécution de **ws.sh**).

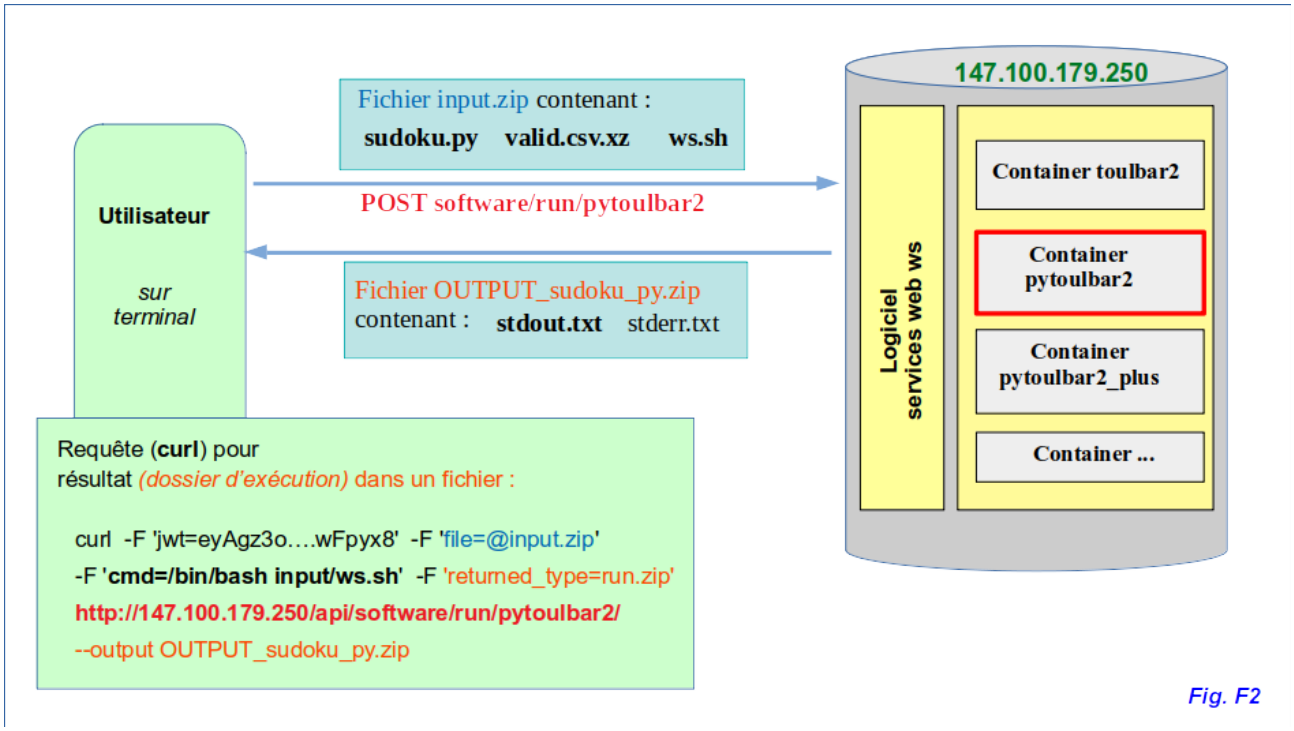


Fig. F2

12 Voir « **A4.3** » en « Annexe 4 »

Exemple Ex2 :

- exemple[13] présenté en **Fig. F3**.
- même cas que dans l'exemple **Ex1** sauf que cette fois, l'utilisateur est *un* ***programme python*** qui contient le code d'envoi de la requête (demande d'exécution de **ws.sh**) et le code de réception de la donnée résultat qui peut alors être **exploitée dans la suite de ce programme python**.

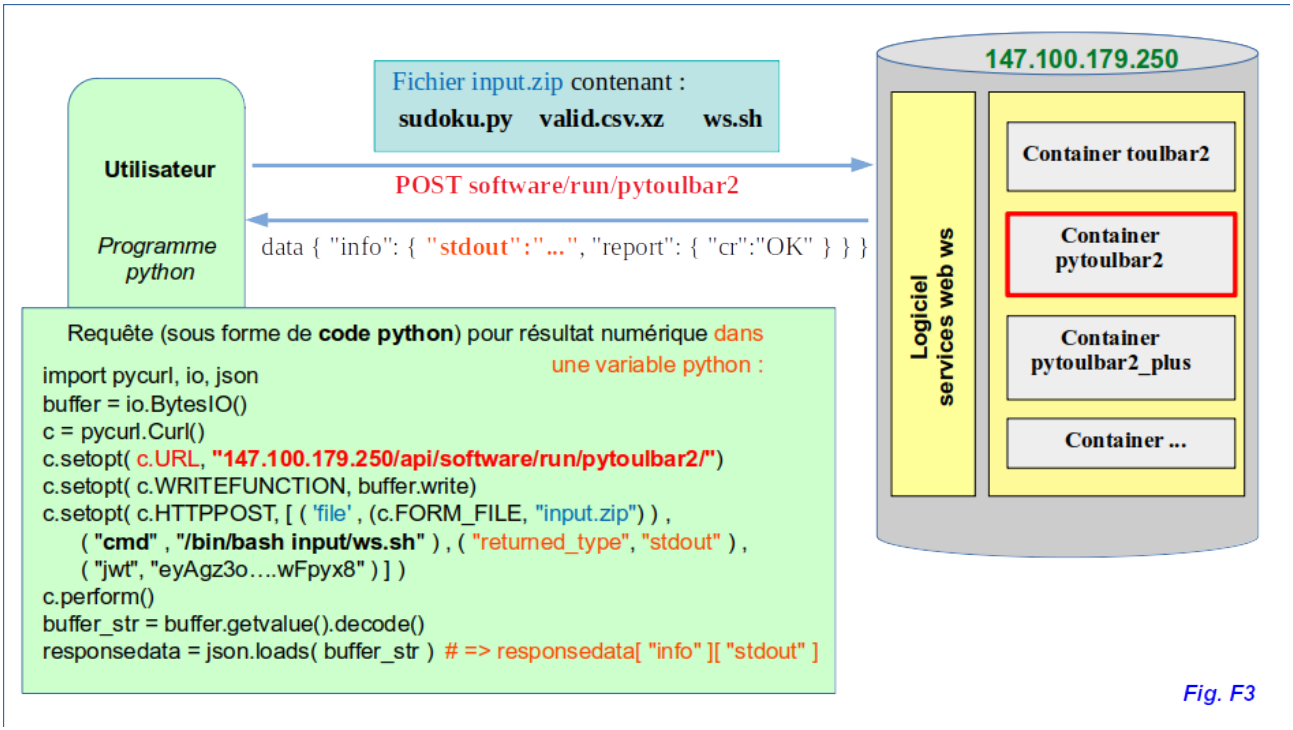


Fig. F3

Cas d'utilisations :

Sous cette forme, les services web **ws** sont mobilisables pour **des travaux à base de toulbar2** (résolution de problèmes, benchmarking…) dans la mesure où les capacités du cluster Muse[14] répondent aux besoins du problème traité, en terme de mémoire.

Ils ont été utilisés dans le cadre du développement d'un **problème de conception d'emploi du temps**[15] (construit sous **toulbar2** à partir d'un modèle existant sous **minizinc**[16]) : lancement d'exécutions (i) pour vérifier que la version **toulbar2** était bien conforme à la version **minizinc** (hébergé lui aussi par les services web **ws** pour l'occasion) puis (ii) pour des résolutions d'instances.

Dans un autre domaine, celui de l'**enseignement**, on pourrait imaginer mettre en place des **travaux pratiques** destinés à des étudiants. En supposant utiliser directement des requêtes génériques, les TPs pourraient être livrés sous une forme assez brute : fourniture de fichiers d'instances de problèmes et de la procédure de lancement de requête de résolution. Pour moins de manipulations, on pourrait aussi imaginer produire et faire héberger par les services web **ws** un **container dédié** à l'enseignement en question, rassemblant **(py)toulbar2** et le contenu du TP (fichiers d'instances de problèmes...).

13 Voir « **A4.3** » en « Annexe 4 »

14 Capacités de la plateforme Meso@LR : https://meso-lr.umontpellier.fr/plateformes-hpc

15 Voir « **A4.5** » et « **A4.9** » en « Annexe 4 »

16 minizinc : langage de modélisation de contraintes

B.2. Usages moyennant des développements supplémentaires

B.2.1. Introduction

On peut avoir envie/besoin de simplifier la tâche de l'utilisateur final, et pour cela de rendre transparent pour lui certains aspects de l'appel aux services web. Ce sera d'autant plus le cas que l'on s'adressera à des utilisateurs non spécialistes voire au grand public. Ceci passe par des **développements supplémentaires côté client (interfaces dédiées) ou du côté des services web (requêtes dédiées)**.

Illustration : **Développements supplémentaires**

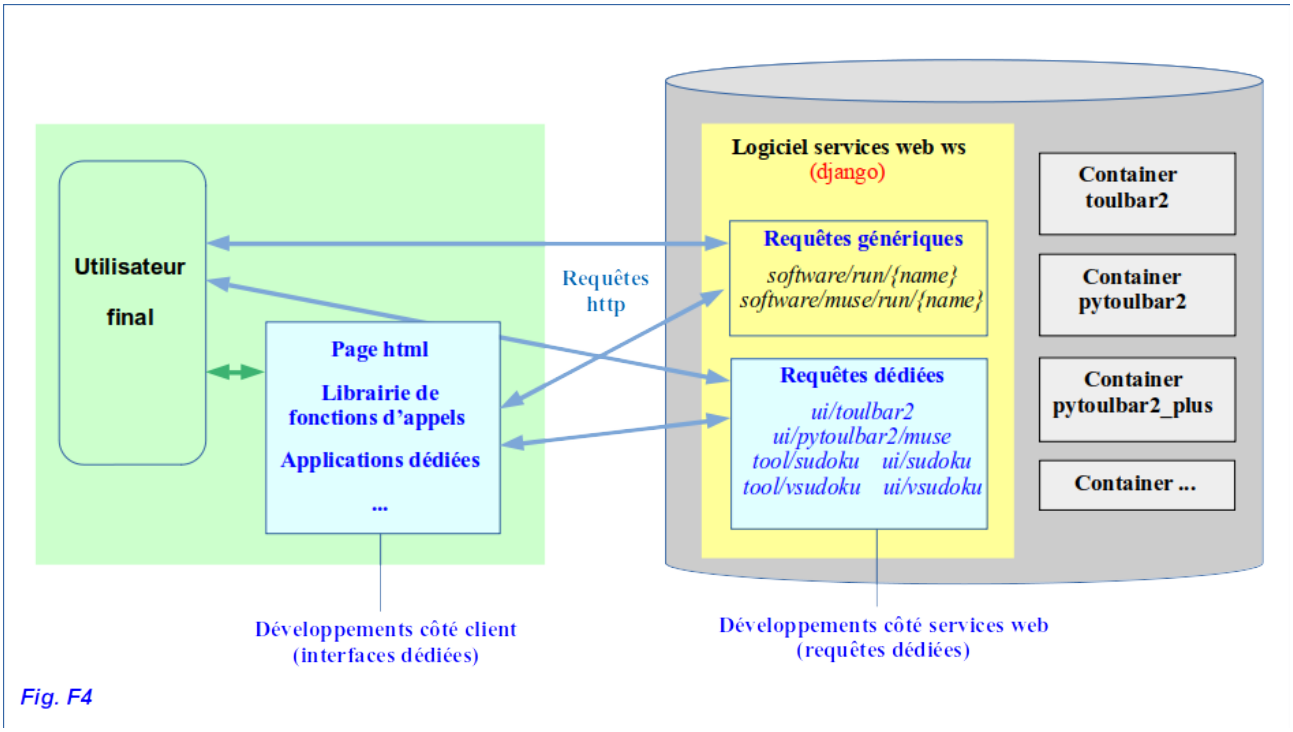


Fig. F4

B.2.2. Développements côté client

Les services web ont vocation à être appelés par leurs clients sous toutes sortes de formes plus ou moins sophistiquées. On a vu la forme la plus directe/brute consistant en manipuler directement les requêtes génériques des services web **ws**[17]. Les clients peuvent également fabriquer/développer de leur côté différentes interfaces logicielles facilitant les appels aux services web par rapport à leur propre cas d'utilisation : pages html, librairies de fonctions d'appels, applications logicielles… Par exemple, pour reprendre le cas d'enseignement (évoqué en « B.1. Usages en l'état »), un enseignant pourrait mettre à disposition de ses étudiants (sur internet, depuis son environnement Jupyter de prédilection) des travaux pratiques construits sous forme de Jupyter NoteBooks dans lesquels les requêtes d'appels aux services web seraient pré-écrites (en Python).

17 Voir « B.1. Usages en l'état »

Exemple Ex3 :

- exemple[18] présenté en **Fig. F5** et **F6**.
- cet exemple reprend le cas de l'exécution du code python **sudoku.py** (+ valid.csv.xz + script .sh) qui a déjà servi dans les exemples **Ex1** et **Ex2** où l'utilisateur manipulait directement la requête.
- ici une petite interface d'appel a été écrite : il s'agit d'*une page html* (**launch_sudoku.html** en **Fig. F5)** prémâchant la construction de la requête POST software/run/pytoulbar2plus et guidant l'utilisateur dans la sélection des paramètres à joindre à la requête (fichiers...). Cette page launch_sudoku.html est située côté client, l'utilisateur peut donc la modifier et l'adapter à ses besoins comme il l'entend.

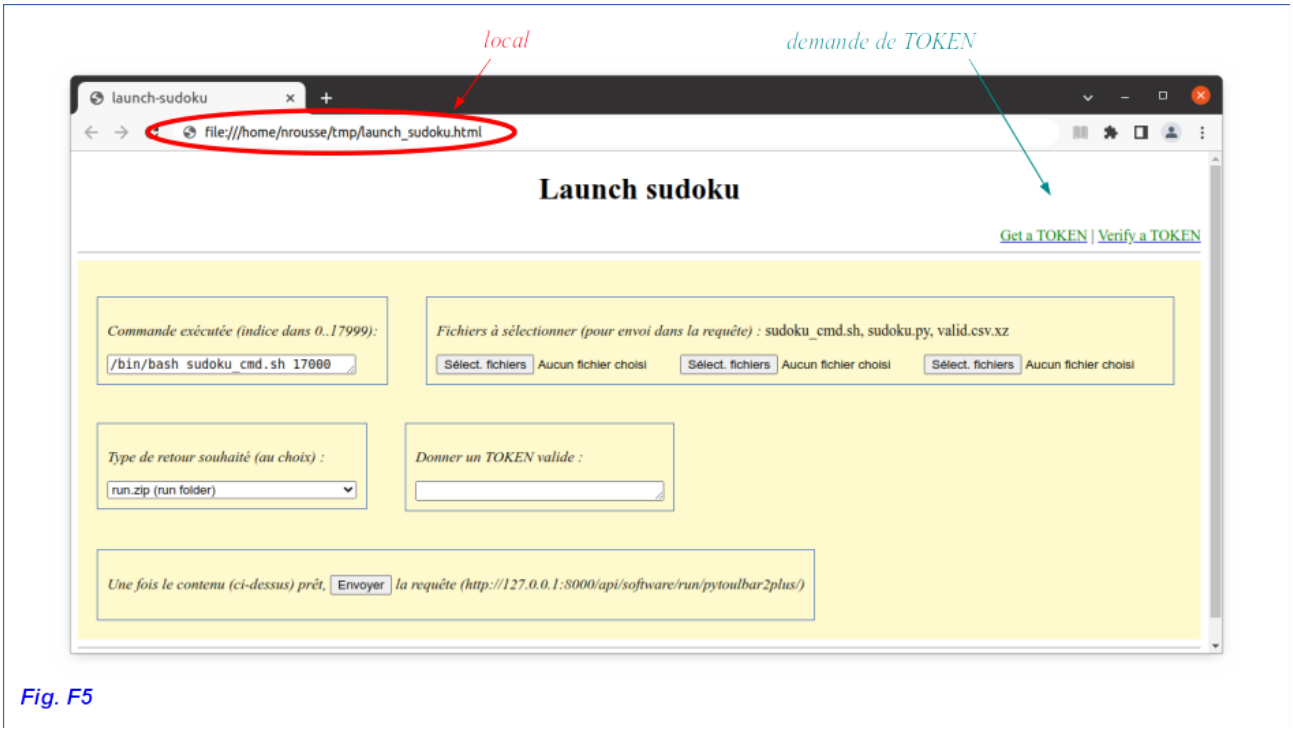


Fig. F5

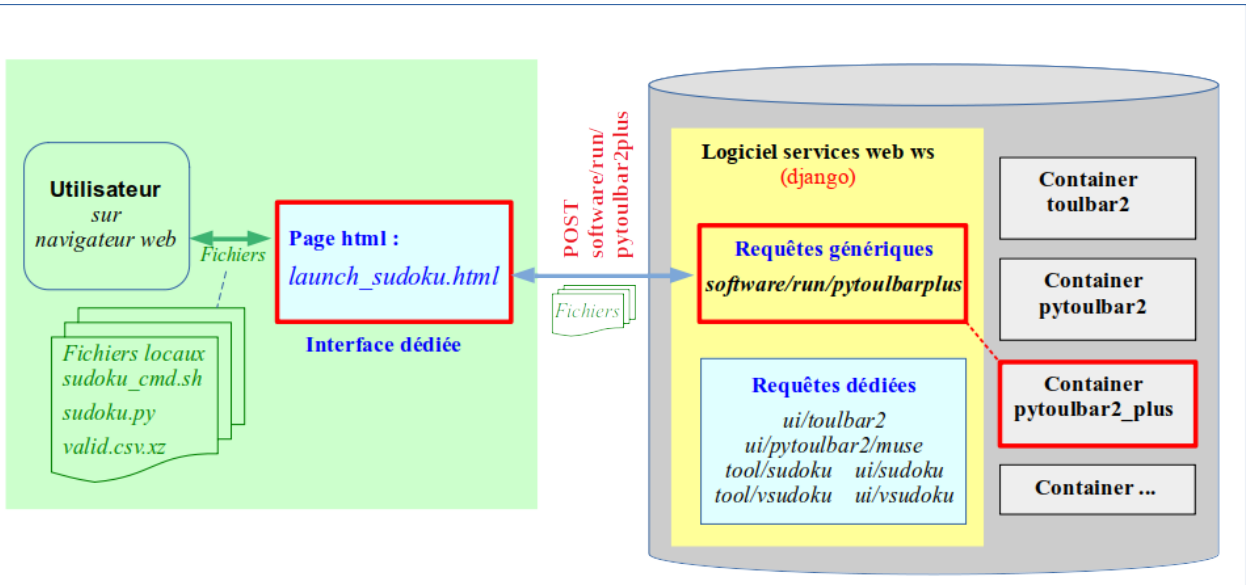


Fig. F6

18 Voir « A4.3 », « A4.8 » en « Annexe 4 »

Autres exemples : voir « B.2.4. Autre exemple avec des développements des 2 côtés (client et serveur) ».

B.2.3. Développements côté serveur

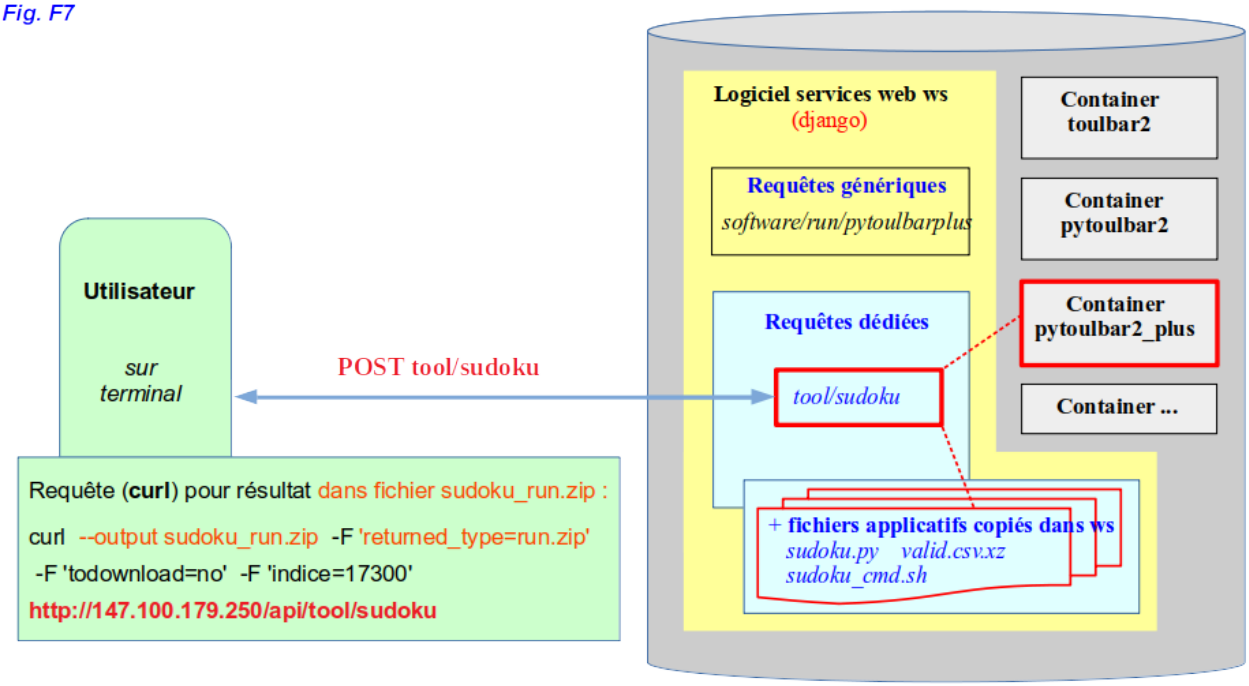
Tout ce qui a été présenté jusqu'alors repose sur les requêtes génériques des services web **ws**, qui sont valables pour tous les containers hébergés. Il est également possible de développer des requêtes **spécialisées** répondant à des besoins spécifiques, par exemple vis-à-vis de containers particuliers ou de certains cas d'utilisation. Ces services web spécialisés se situent côté serveur dans le code du logiciel **ws** (sur lequel l'utilisateur n'a pas la main), engendrant à ce niveau des coûts en maintenance et développement. De telles requêtes, dédiées à **toulbar2**, ont été développées.

Exemple Ex4 :

- exemple[19] présenté en **Fig. F7**, **F8**, **F9** (+ **FA3.1**).
- cet exemple reprend le cas de l'exécution du code python **sudoku.py** (+ valid.csv.xz + script .sh) qui a déjà servi dans les exemples **Ex1** et **Ex2** (où l'utilisateur manipulait directement la requête) et dans l'exemple **Ex3** (où l'utilisateur avait recours depuis son navigateur web à une page **launch_sudoku.html** située sur son poste). Dans ces 3 exemples, le programme **sudoku.py** était exécuté par appel d'une requête générique. Celle-ci requérant un token, il fallait : (i) posséder un compte 'utilisateur' (nécessaire pour obtenir un token), (ii) commencer par une requête POST jwt/obtain de demande d'un token (une seule requête ne suffisait pas).
- maintenant, on voudrait pouvoir exécuter le programme **sudoku.py** en une seule requête et sans authentification. L'objectif de ceci serait d'exposer cet exemple (résolution de sudoku par **toulbar2**) sur le site de **toulbar2** et de permettre aux visiteurs du site (grand public) de demander eux-même des résolutions de grilles « en un seul clic ». Pour cela, on va créer une nouvelle requête **tool/sudoku** qui sera en accès public (sans token) et sans pré- ni post- traitements (retour de résultat directement exploitable en l'état).
- La requête dédiée **tool/sudoku** est complètement spécifique de l'application sudoku.py dont elle **bride** fortement le paramétrage en contre partie d'en donner l'accès sans authentification. Cette requête utilise les **fichiers applicatifs** (sudoku.py, , valid.csv.xz, script .sh) qui ont été **copiés dans le logiciel ws (version figée côté serveur)**, alors que dans les exemples précédents (Ex1, Ex2, Ex3) c'était l'utilisateur qui les fournissait à la requête générique. Ici, l'utilisateur a juste à renseigner un paramètre 'indice' de la grille choisie dans valid.csv.xz, il peut également choisir le type de format souhaité pour la réponse (fichier texte, donnée json...). En réponse à sa requête, il reçoit comme données numériques la grille d'entrée du sudoku et la grille résolue.
- La **figure F7** montre un exemple d'appel de la requête **tool/sudoku** depuis *un terminal*.

19 Voir « **A4.3** », « **A4.2** » en « Annexe 4 »

Fig. F7



- Une seconde requête dédiée **ui/sudoku** a été créée conjointement à la requête dédiée **tool/sudoku**. Cette nouvelle requête **ui/sudoku** expose une page html (contenue dans le logiciel **ws**) facilitant l'appel à la requête **tool/sudoku**. La figure **F8** montre la page html à laquelle donne accès la requête **ui/sudoku**.
- Exposition sur le site de **toulbar2** : la figure **FA3.1** en annexe[20] montre la page https://toulbar2.github.io/toulbar2/examples/snum.html du site **toulbar2** (http://miat.inrae.fr/toulbar2) qui expose/appelle la requête **ui/sudoku**.

20 En « Annexe 3. Exposition depuis des pages du site de toulbar2 »

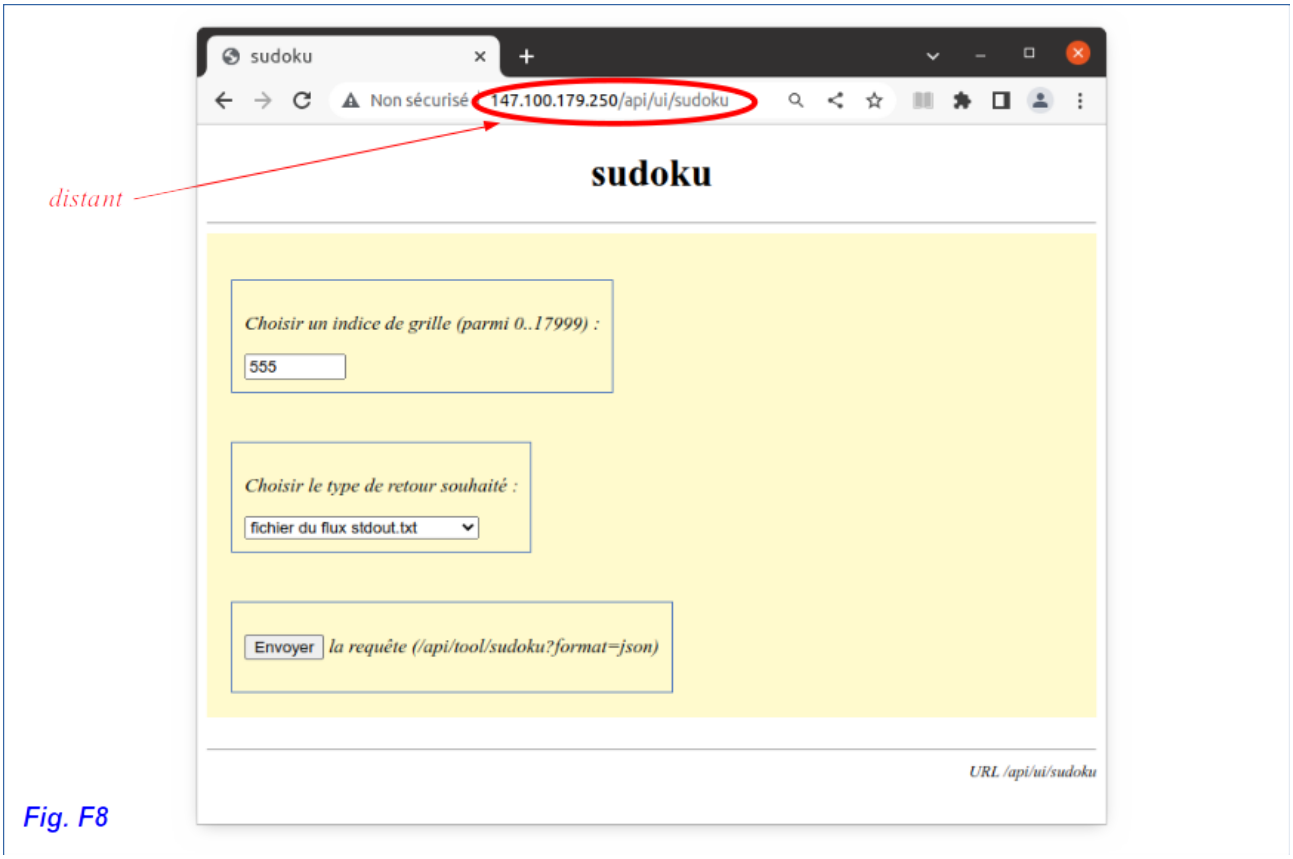
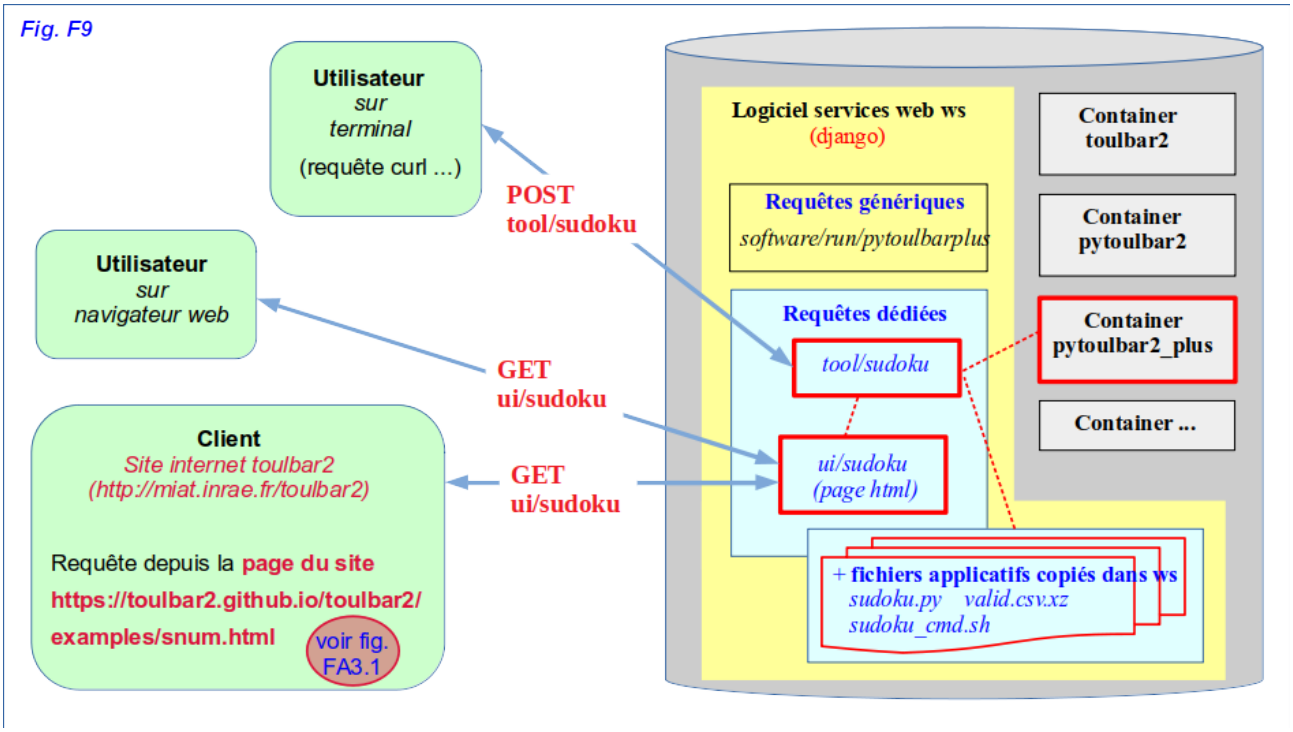


Fig. F8

Illustration bilan pour **Ex4** :



Autres exemples : voir « B.2.4. Autre exemple avec des développements des 2 côtés (client et serveur) ».

B.2.4. Autre exemple avec des développements des 2 côtés (client et serveur)

Exemple Ex5 :

- exemple[21] présenté en **Fig. F10**, **F11**, **F13** (+ **FA3.2**).
- dans cet exemple il est toujours question de résolution de sudoku avec **pytoulbar2**, mais cette fois au moyen d'un programme plus perfectionné dont l'entrée (grille à résoudre) et la sortie (grille solution) sont des fichiers image. L'objectif comme dans l'exemple **Ex4** est l'exécution « en un seul clic », en accès public (sans token).

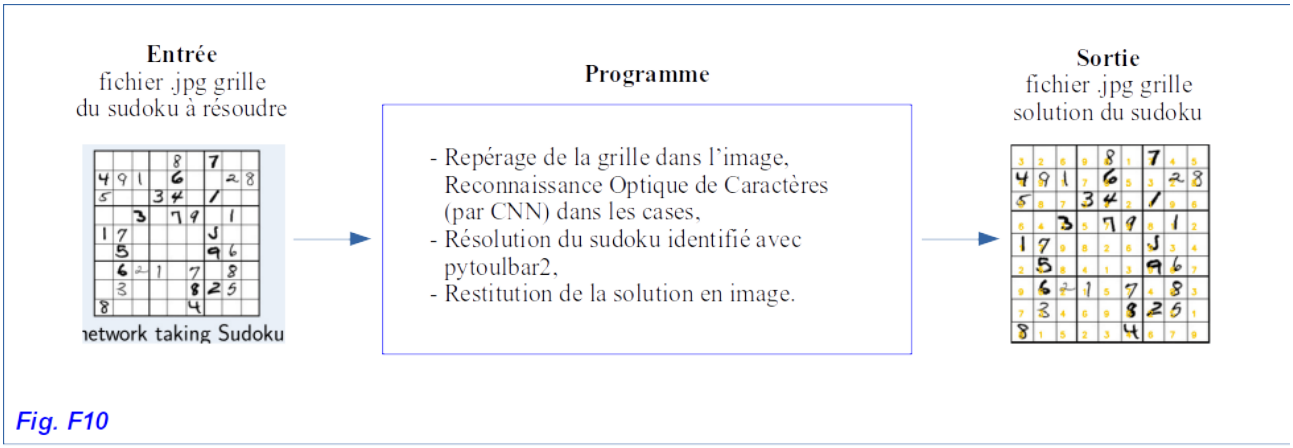


Fig. F10

21 Voir « **A4.4** », « **A4.2** » en « Annexe 4 »

Développement côté serveur :

- Il a été créé une requête dédiée **tool/vsudoku** qui est complètement spécifique du programme dont le code (**toulbar2_visual_sudoku_puzzle.py**...[22]) a été **copié dans le logiciel ws (version figée côté serveur)** : l'utilisateur donne dans sa requête un fichier image de grille de sudoku à résoudre (il peut également donner des valeurs de paramètres de réglage optionnels) et reçoit en réponse le fichier image de la grille résolue.
- Comme dans l'exemple **Ex4**, il a été créé conjointement une seconde requête dédiée **ui/vsudoku** qui expose une page html facilitant l'appel à la requête **tool/vsudoku**. La figure **F11** montre la page html à laquelle donne accès la requête **ui/vsudoku**.

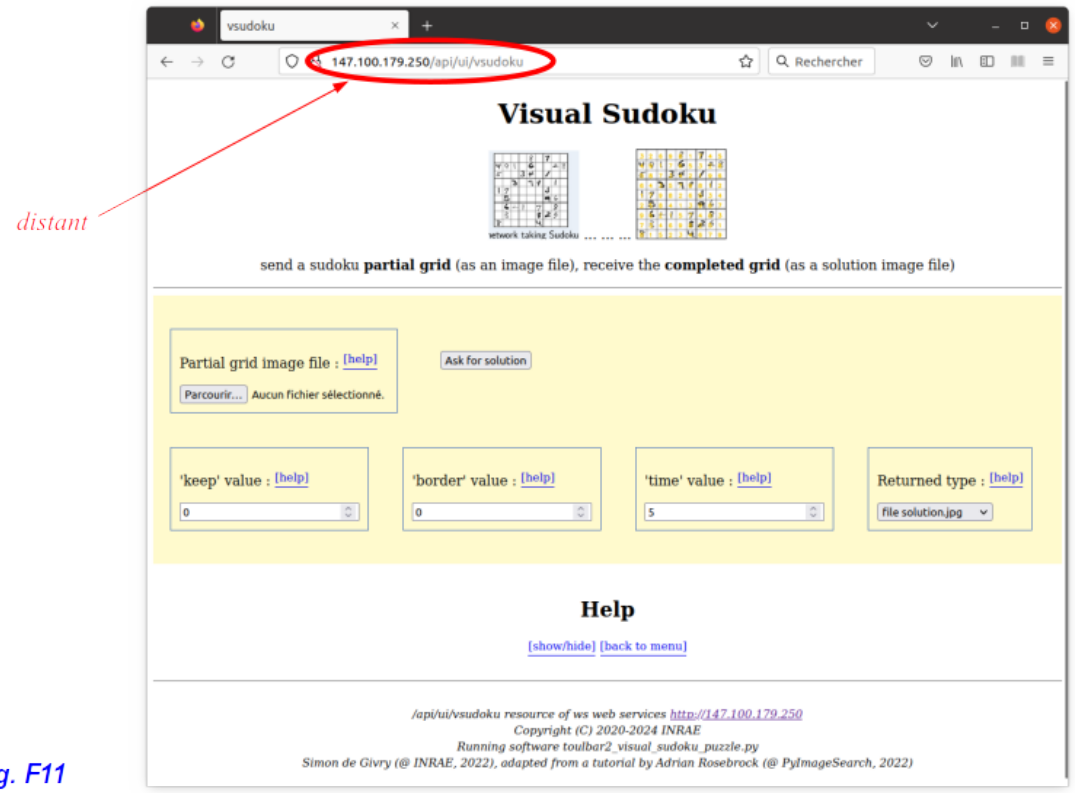


Fig. F11

Exemple Ex6 :

- exemple[23] présenté en **Fig. F12**, **F13** (+ **FA3.3, FA3.2**).
- dans cet exemple la requête dédiée **tool/vsudoku** existante, qui vient d'être présentée dans l'exemple **Ex5**, va servir de socle à la création d'une application logicielle **VisualSudoku** fonctionnant sur smartphone. Il s'agit du développement côté client d'une interface dédiée, sous une forme plus élaborée que la simple page html (nommée launch_sudoku.html) présentée dans l'exemple **Ex3** en « B.2.2. Développements côté client ».

22 Voir « **A4.4** » en « Annexe 4 »

23 Voir « **A4.6** » en « Annexe 4 »

Développement côté client :

Fig. F12 **VisualSudoku Application for Android**

L'Application « Visual Sudoku » permet de résoudre le problème de sudoku depuis un smartphone :
Elle permet de prendre en photo la grille de sudoku à résoudre,
ou de la sélectionner parmi les photos pré-existantes du smartphone ('DCIM').
Une fois la grille sélectionnée, elle résoud le sudoku de la grille choisie et affiche à l'écran la solution
(qu'il est possible de sauvegarder dans un fichier image .jpg).

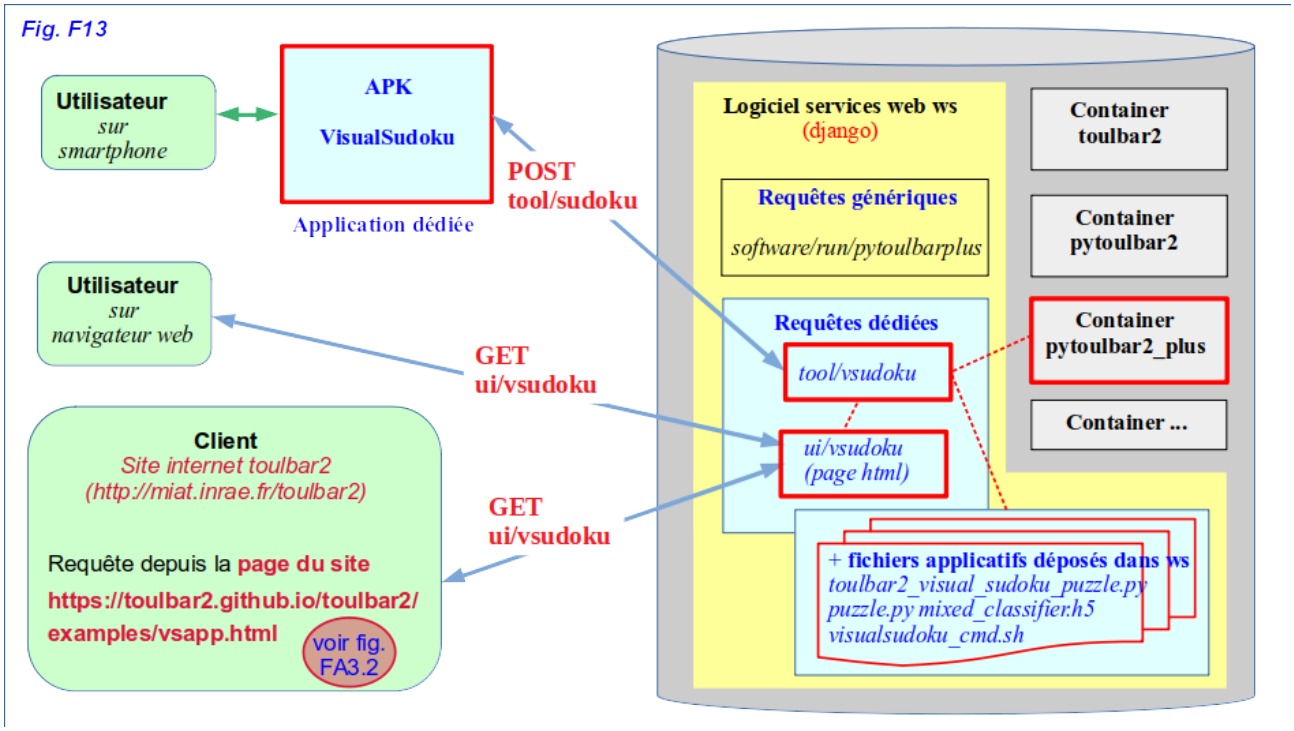
- L'essentiel du développement de l'application dédiée **VisualSudoku** a porté sur des aspects IHM, étant donné que la résolution de sudoku y est effectuée par simple appel de la requête dédiée **tool/vsudoku**,
- La figure **FA3.3** en annexe[24] montre la page de présentation de l'application **VisualSudoku** (description, installation…) https://toulbar2.github.io/toulbar2/examples/vsapp_apk.html du site **toulbar2** (http://miat.inrae.fr/toulbar2).

Exposition sur le site de **toulbar2** : la figure **FA3.2** en annexe[25] montre la page https://toulbar2.github.io/toulbar2/examples/vsapp.html du site **toulbar2** (http://miat.inrae.fr/toulbar2) :

- qui expose/appelle la requête dédiée **ui/vsudoku** (partie '**As a Web service'**[26])
- qui expose l'application dédiée qui a été développée côté client (partie '**As an APK'**[27]).

Illustration bilan pour **Ex5** et **Ex6** :

24 En « Annexe 3. Exposition depuis des pages du site de toulbar2 »
25 En « Annexe 3. Exposition depuis des pages du site de toulbar2 »
26 https://toulbar2.github.io/toulbar2/examples/vsapp.html#as-a-web-service
27 https://toulbar2.github.io/toulbar2/examples/vsapp.html#as-an-apk



C. Conclusion

Différents cas d'usages de **toulbar2** sous forme de services web ont été présentés. Leur mise en œuvre s'appuie sur le démonstrateur de services web **ws** pré-existant, qui permet d'exécuter sous forme de services web des logiciels qui ont préalablement été mis sous **containers** afin d'être hébergés par le démonstrateur.

Les **requêtes génériques** existantes du logiciel de services web **ws** présentent l'avantage d'être valables et disponibles pour tous les containers hébergés/accueillis. Elles sont destinées à des utilisateurs plutôt **avertis/spécialistes** car leur manipulation requiert un certain savoir faire (des connaissances générales en terme de services web et concernant les spécifications de l'API **ws**). Par ailleurs des utilisateurs finaux **moins spécialistes** - voire le grand public – peuvent être ciblés moyennant des **développements supplémentaires** leur simplifiant la tâche. Ces développements peuvent se situer côté client (un code appelant les services web) ou bien côté serveur (du code au sein même du logiciel **ws** fournisseur des services web) :

- ► Développements côté client (interfaces dédiées) : Chaque client de son côté peut développer autant de logiciels appelant les services web qu'il le souhaite, sous une forme plus ou moins sophistiquée (selon ses moyens, ses besoins) : il peut s'agir d'une page html très légère (Ex3), d'une application pour smartphone (Ex6) ou de tout autre outil logiciel, application web, Jupyter NoteBook, librairie de fonctions d'appels, etc. La maintenance et le développement de ces logiciels sont du ressort du client, celui-ci peut en modifier le code comme et quand il l'entend.
- ► Développements côté serveur (requêtes dédiées) : Il s'agit d'incorporer de nouveaux traitements au niveau du logiciel **ws** lui-même, induisant des coûts en maintenance et développement. Les traitements placés de ce côté (requêtes spécialisées prêtes à l'emploi) bénéficient à tous les clients qui, dans ce cas, n'ont pas la main sur le code (fréquence et contenu des modifications). De telles requêtes spécialisées ont été créées, dédiées à **toulbar2** (Ex4, Ex5) au bénéfice de certains cas d'utilisation qui requéraient notamment l'accès sans authentification (de sorte à cibler le grand public) et la simplicité d'appel (appel « unique » sans enchaînement d'opérations).

Annexes

Annexe 1. Containers pour (py)toulbar2

Container	toulbar2	pytoulbar2	pytoulbar2plus
Dépôt	https://github.com/toulbar2/toulbar2		https://github.com/nrousse/toulbar2 (fork)
Package	toulbar2/toulbar2	toulbar2/pytoulbar2	toulbar2/pytoulbar2plus
Deploy	.github/workflows/docker-publish.yml		.github/workflows/docker-publish-pytoulbar2plus.yml
Desc	Container for toulbar2 and its pytoulbar2 Python API	Container for pytoulbar2 Python API of toulbar2	Container for pytoulbar2 Python API of toulbar2 and some other Python libraries
Contains	C++, toulbar2, Python3, pytoulbar2	Python3, pytoulbar2	Python3, pytoulbar2, numpy, nbconvert, more-itertools, matplotlib, pandas, tensorflow-cpu, keras, sklearn, opencv-python, scikit-image ...
Install method	From sources with *cmake options -DPYTB2=ON and -DXML=ON*	by ‘pip install’ *python3 -m pip install pytoulbar2*	by ‘pip install’
Fichier source image Docker	docker/toulbar2/ Dockerfile__toulbar	docker/pytoulbar2/ Dockerfile__pytoulbar2	docker/pytoulbar2/ Dockerfile__pytoulbar2plus
Taille (.simg associée)	393 Mo	257 Mo	1 300 Mo
Exemples d’appel	docker/toulbar2/using	docker/pytoulbar2/using	docker/pytoulbar2/using

Annexe 2. Documentation pour (py)toulbar2

Au niveau des services web **ws**, une documentation en ligne est associée à chaque logiciel hébergé. Dans cette documentation on peut trouver pour certains logiciels des exemples prêts à être lancés [28] :

- **documentation toulbar2** : http://147.100.179.250/softwares/toulbar2/home.html
- **documentation pytoulbar2** : http://147.100.179.250/softwares/pytoulbar2/home.html
- Le dossier '**ws_TRY**' de **pytoulbar2** contient des **exemples** de requêtes lancées depuis un **terminal** (commandes « curl »), requêtes lancées depuis une **page html** (adaptable à son propre cas d'appel), requêtes lancées depuis un **code R** … Voir d'autres exemples dans la documentation http://147.100.179.250/softwares/pytoulbar2_v0/home.html d'une ancienne version de container pytoulbar2_v0 (exemples de requêtes lancées depuis un **code python …).**

Exemple de demande de résolution d'un problème présent dans le dossier « TUTORIALS » du container :

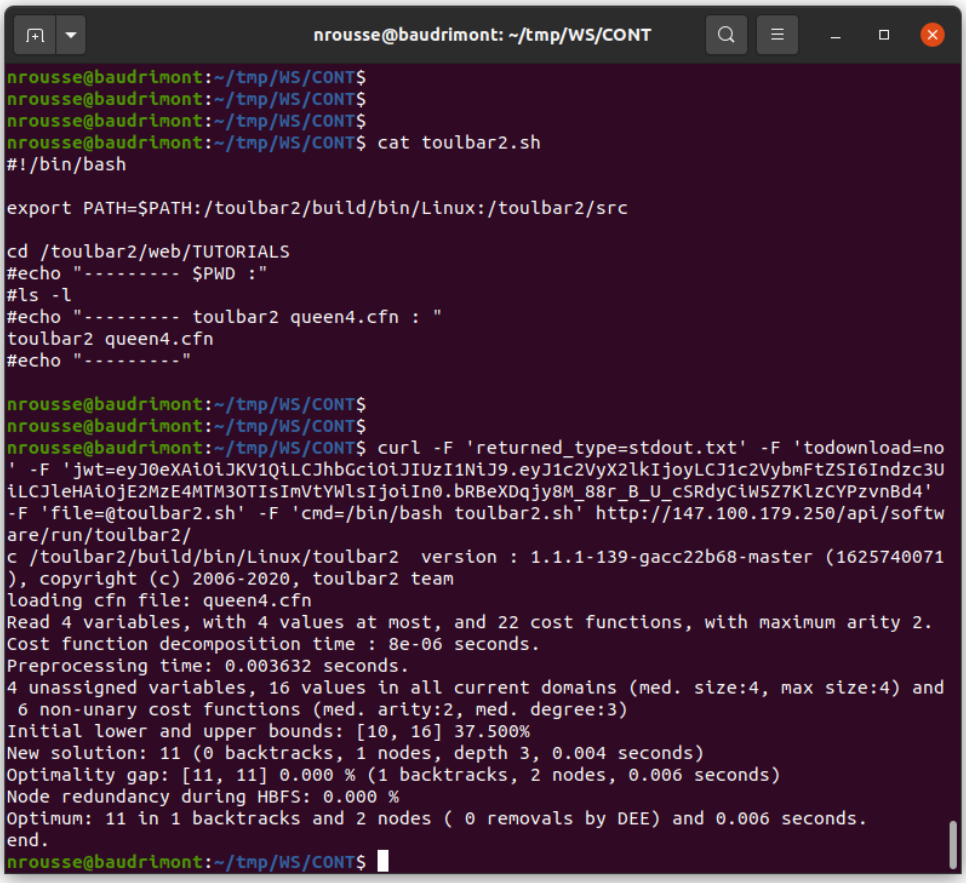


Figure FA2.1

28 Voir répertoire 'ws_TRY', fichier README

Exemple de demande de résolution d'un problème **example.wcsp** envoyé dans la requête (générique) :

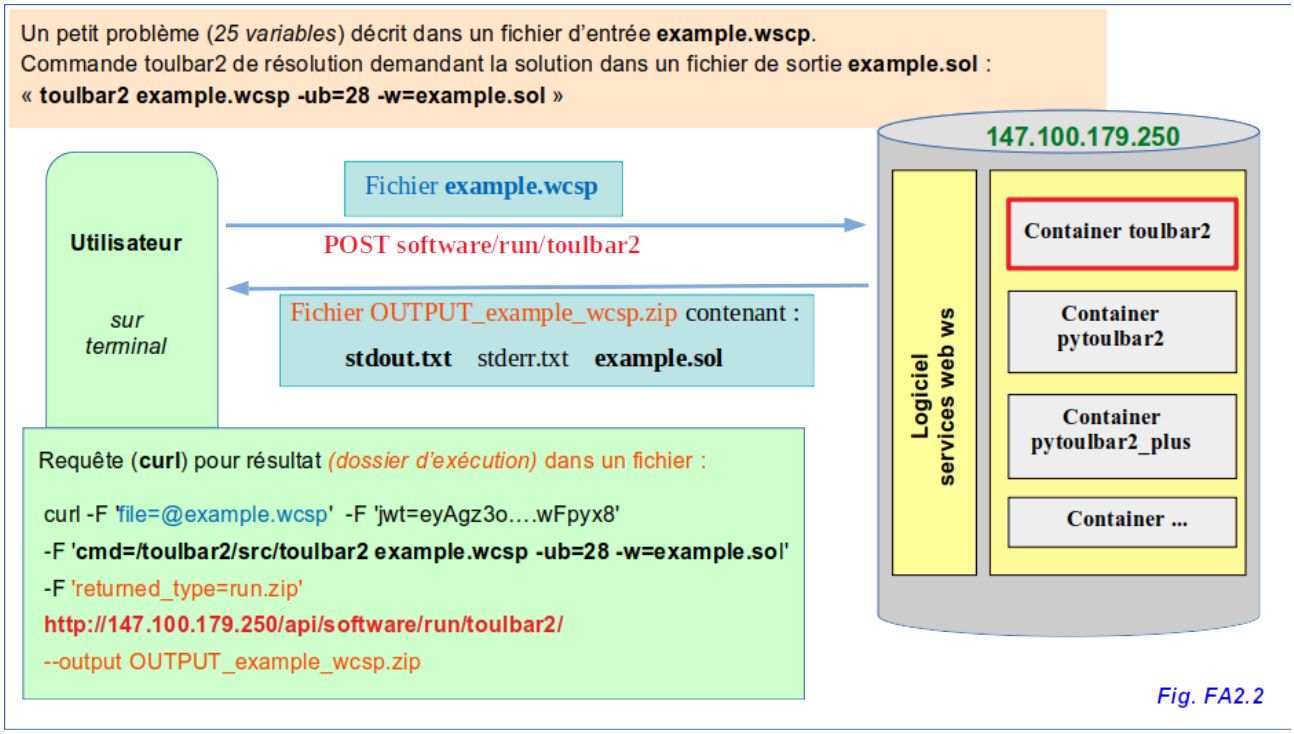


Fig. FA2.2

Exemple de requêtes dédiées **ui/toulbar2 ui/toulbar2/muse ui/pytoulbar2 ui/pytoulbar2/muse** : ces pages html interface d'appel de **(py)toulbar2** facilitent l'appel de **toulbar2** et **pytoulbar2**. Elles sont implémentées dans le logiciels **ws** et exposées sous forme de requêtes spécifiques des containers **toulbar2** et **pytoulbar2** :

http://147.100.179.250/api/ui/**toulbar2** http://147.100.179.250/api/ui/**toulbar2**/muse

http://147.100.179.250/api/ui/**pytoulbar2** http://147.100.179.250/api/ui/**pytoulbar2**/muse

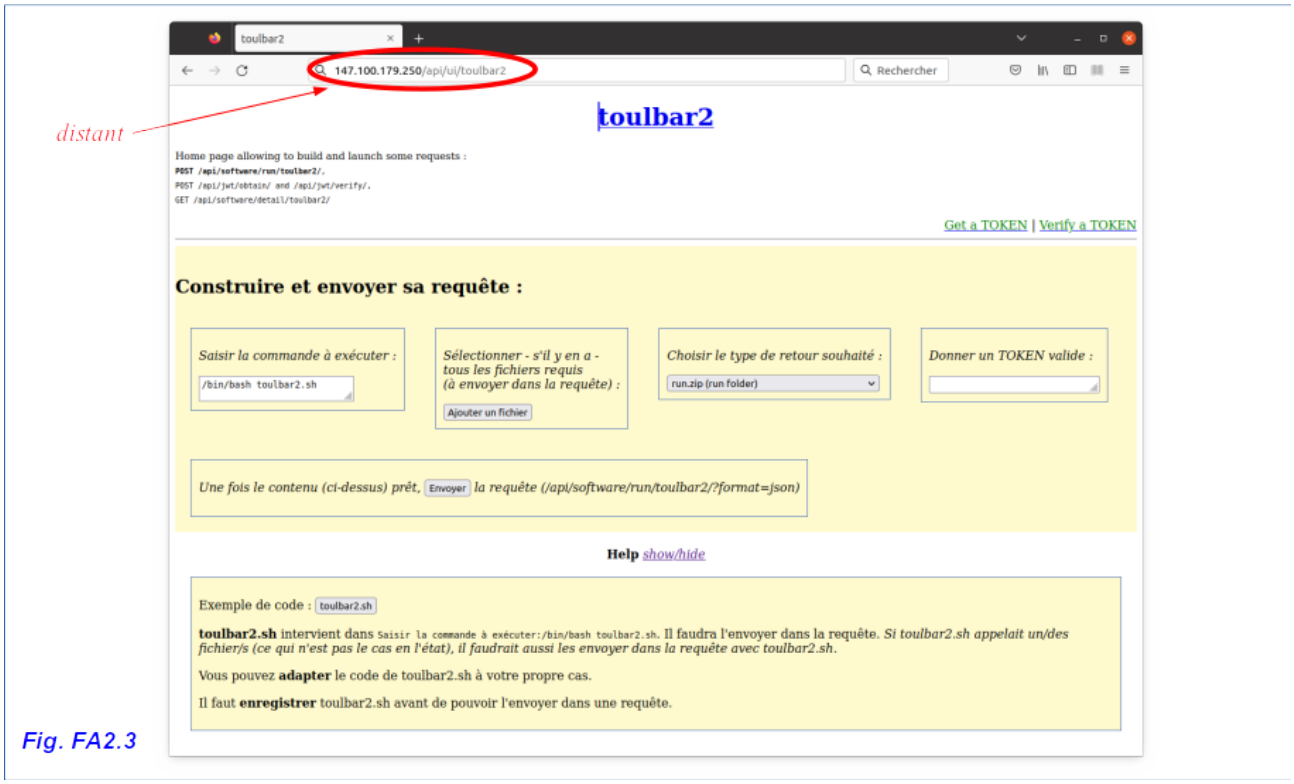


Fig. FA2.3

Annexe 3. Exposition depuis des pages du site de toulbar2

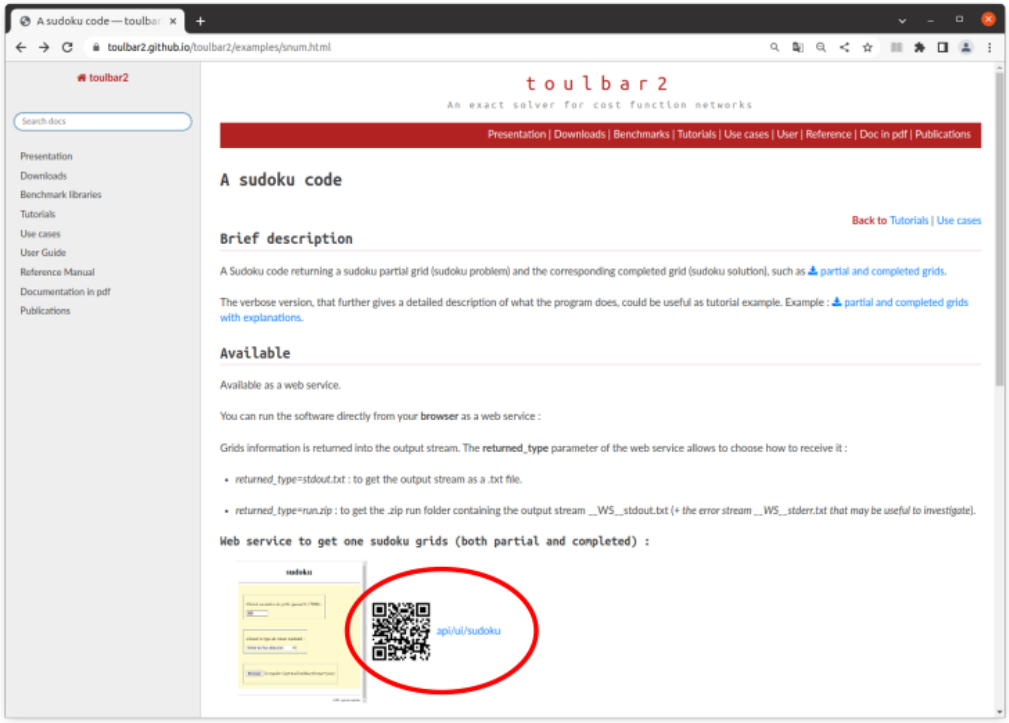


Fig. FA3.1

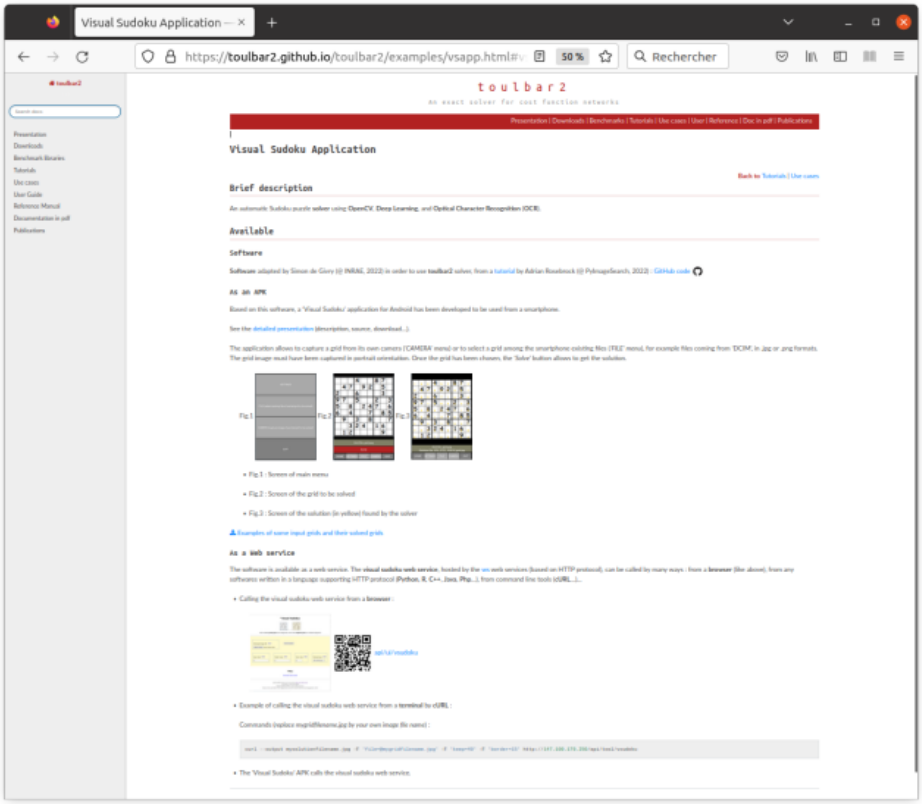


Fig. FA3.2

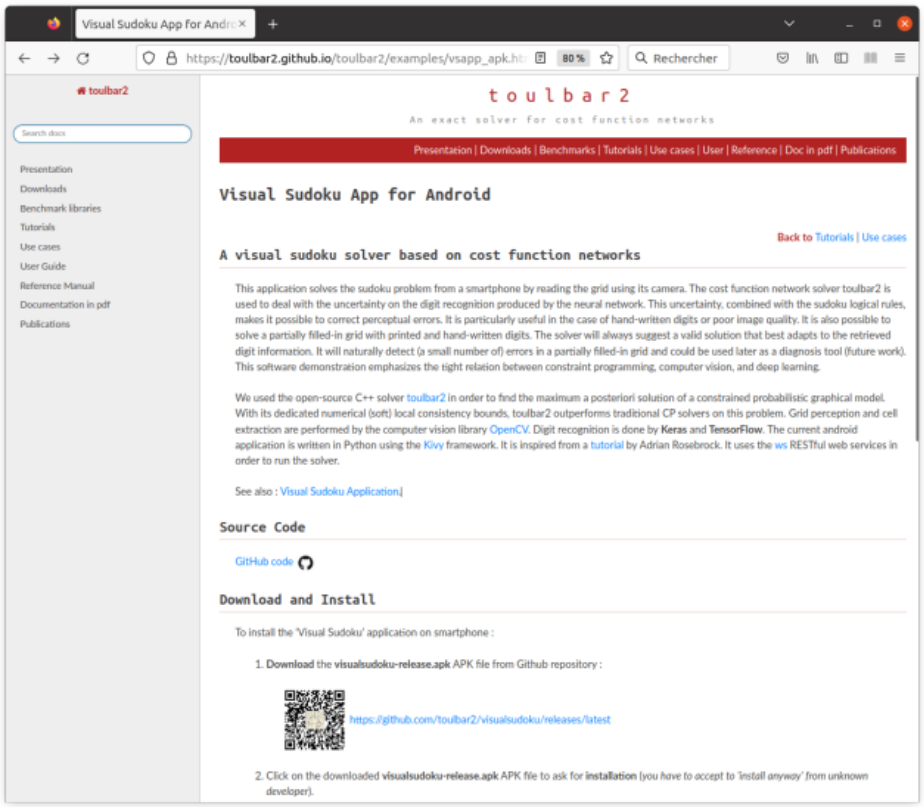

Visual Sudoku App for Andr
https://toulbar2.github.io/toulbar2/examples/vsapp_apk.ht
80 %
Rechercher
toulbar2
Search docs
Presentation
Downloads
Benchmark libraries
Tutorials
Use cases
User Guide
Reference Manual
Documentation in pdf
Publications
t o u l b a r 2
An exact solver for cost function networks
Presentation | Downloads | Benchmarks | Tutorials | Use cases | User | Reference | Doc in pdf | Publications
Visual Sudoku App for Android
Back to Tutorials | Use cases
A visual sudoku solver based on cost function networks
This application solves the sudoku problem from a smartphone by reading the grid using its camera. The cost function network solver toulbar2 is used to deal with the uncertainty on the digit recognition produced by the neural network. This uncertainty, combined with the sudoku logical rules, makes it possible to correct perceptual errors. It is particularly useful in the case of hand-written digits or poor image quality. It is also possible to solve a partially filled-in grid with printed and hand-written digits. The solver will always suggest a valid solution that best adapts to the retrieved digit information. It will naturally detect (a small number of) errors in a partially filled-in grid and could be used later as a diagnosis tool (future work). This software demonstration emphasizes the tight relation between constraint programming, computer vision, and deep learning.
We used the open-source C++ solver toulbar2 in order to find the maximum a posteriori solution of a constrained probabilistic graphical model. With its dedicated numerical (soft) local consistency bounds, toulbar2 outperforms traditional CP solvers on this problem. Grid perception and cell extraction are performed by the computer vision library OpenCV. Digit recognition is done by Keras and TensorFlow. The current android application is written in Python using the Kivy framework. It is inspired from a tutorial by Adrian Rosebrock. It uses the ws RESTful web services in order to run the solver.
See also : Visual Sudoku Application.
Source Code
GitHub code
Download and Install
To install the 'Visual Sudoku' application on smartphone :
1. Download the visualsudoku-release.apk APK file from Github repository :
https://github.com/toulbar2/visualsudoku/releases/latest
2. Click on the downloaded visualsudoku-release.apk APK file to ask for installation (you have to accept to 'install anyway' from unknown developer).

Fig. FA3.3

Annexe 4. Logiciels produits et/ou exploités

► Logiciels sur lesquels est basée la collaboration :

A4.1. Logiciel **toulbar2**

Desc Le logiciel **toulbar2** est un solveur de réseau de fonctions de coût, écrit en langage C++, possédant une API Python **pytoulbar2**.

Dev Logiciel existant. Développé par SaAB. Intégrateur : S. de Givry.
Développement annexe par N.Rousse : mise sous containers de **(py)toulbar2** et hébergement par les services web **ws**. Voir « **A.3** ».

Ref Site internet **toulbar2** : http://miat.inrae.fr/toulbar2 .
Dépôt **toulbar2** : https://github.com/toulbar2/toulbar2 .
Containers : voir « **Annexe 1** ».

A4.2. Logiciel **ws**

Desc Le logiciel **ws** est un démonstrateur de services web permettant d'exécuter sous forme de services web des logiciels préalablement mis sous containers. Voir « A.2 ».

Dev Logiciel existant (créé à titre exploratoire) développé par N.Rousse, proposant des requêtes génériques. Le logiciel a évolué dans le cadre de la collaboration :
- Ajout de fonctionnalités. Voir « A.2 ».
- Ajout de traitements de requêtes dédiées basées sur la résolution par **toulbar2** du problème de sudoku :
  - Un service web utilisant/encapsulant le logiciel **sudoku.py** (voir « **A4.3** ») : Avec la requête **tool/sudoku**, l'utilisateur choisit un indice de grille à résoudre (parmi 18000 prédéfinies). La requête **ui/sudoku** expose une page html facilitant l'appel à la requête **tool/sudoku**.
    URL requêtes : : http://147.100.179.250/api/tool/sudoku et http://147.100.179.250/api/ui/sudoku .
    Voir **Ex4**.
  - Un service web utilisant/encapsulant le logiciel **toulbar2_visual_sudoku_puzzle.py** (voir « **A4.4** ») : La requête **tool/vsudoku** reçoit une photo de grille à résoudre et retourne la photo de la grille résolue. La requête **ui/vsudoku** expose une page html facilitant l'appel à la requête **tool/vsudoku**.
    URL requêtes : http://147.100.179.250/api/tool/vsudoku et http://147.100.179.250/api/ui/vsudoku .
    Voir **Ex5**.

Ref Mise en ligne (temporaire) : http://147.100.179.250 .
Dépôt **ws** : https://forgemia.inra.fr/nathalie.rousse/ws .
Voir « **A.2** »

► **Codes de problèmes toulbar2 pour la résolution/l'exécution desquels les services web ont été mobilisés :**

A4.3. Logiciel **sudoku.py**

Desc Le logiciel **sudoku.py** (+ valid.csv.xz, script .sh) utilise **pytoulbar2** pour résoudre une grille de sudoku lue dans un fichier d'entrée **valid.csv.xz**, et retourne en sortie les valeurs numériques de la grille solution.
Voir **Ex1**, **Ex2**, **Ex3**, **Ex4**.

Ref Le code (de la version embarquée) est inclus dans le logiciel **ws**.

A4.4. Logiciel **toulbar2_visual_sudoku_puzzle.py**

Desc Le logiciel **toulbar2_visual_sudoku_puzzle.py** (+ puzzle.py, mixed_classifier.h5, visualsudoku_cmd.sh) utilise **pytoulbar2** pour résoudre une grille de sudoku. Les entrées/sorties étant des photos, le logiciel met en œuvre du traitement d'image et de la reconnaissance optique de caractères (par réseau de neurones).
Voir **Ex5**.

Dev Le logiciel repose sur un code préexistant[29] dont le traitement de résolution a été adapté à **toulbar2** par S. de Givry, avec amélioration des performances du réseau de neurones par Marianne Defresne (**SaAB**).

Ref Code : espace '**crafted/visualsudoku**' du dépôt https://forgemia.inra.fr/thomas.schiex/cost-function-library .

A4.5. Logiciel **examtt**

Desc Le logiciel **examtt** est un problème de conception d'emploi du temps construit avec **toulbar2** à partir d'un modèle existant sous **minizinc**[30].

Dev Logiciel existant développé avec **toulbar2** par N.Rousse à partir d'un problème pré-défini[31] et de sa version modélisée sous **minizinc**[32].

Ref Code : espace '**real/examtt**' du dépôt https://forgemia.inra.fr/thomas.schiex/cost-function-library .

29 Code d'Adrian Rosebrock de résolution d'une image grille de sudoku : https://pyimagesearch.com/2020/08/10/opencv-sudoku-solver-and-ocr

30 minizinc : langage de modélisation de contraintes (https://www.minizinc.org)

31 Une approche de recherche locale de ce problème de conception d'emploi du temps est présentée dans l'article « Local search and constraint programming for a real-world examination timetabling problem » de Michele Battistutta, Sara Ceschia, Fabio De Cesco, Luca Di Gaspero, Andrea Schaerf et Elena Topan.

32 https://bitbucket.org/satt/examtimetablinguniuddata/src/master

► **Des logiciels/codes 'utilisateurs' de toulbar2 par services web :**

A4.6. APK **VisualSudoku**

Desc L'APK (*Android Package Kit*) **VisualSudoku** est une application pour smartphone de résolution de sudoku, qui permet de prendre en photo une grille partielle de sudoku (ou de la sélectionner parmi des photos du smartphone) et qui affiche à l'écran l'image de la grille résolue (avec possibilité de sauvegarde). La grille est résolue par appel du service web **tool/vsudoku**.
Voir **Ex6**.

Dev Logiciel développé par N.Rousse.

Ref Dépôt **visualsudoku** (code source et téléchargement) :
https://github.com/toulbar2/visualsudoku.
L'APK **VisualSudoku** est présentée sur le site **toulbar2** :
https://toulbar2.github.io/toulbar2/examples/vsapp_apk.html .

A4.7. Pages du site **toulbar2**

Desc Au titre de cas d'utilisations, des pages du site **toulbar2** appellent des services web ws afin de permettre aux visiteurs du site de 'jouer' avec de petits problèmes **toulbar2** :
- La page https://toulbar2.github.io/toulbar2/examples/snum.html expose/appelle la requête **ui/sudoku**.
- La page https://toulbar2.github.io/toulbar2/examples/vsapp.html expose/appelle la requête dédiée **ui/vsudoku** (partie '**As a Web service'**[33]) et expose l'application dédiée **VisualSudoku** qui a été développée côté client (partie '**As an APK'**[34]).

A4.8. Page **launch_sudoku.html**

Desc La page html **launch_sudoku.html** est une petite interface d'appel d'un service web (requête **générique**) facilitant la construction (choix des paramètres…) et l'envoi de la requête dans un cas d'utilisation bien précis (résolution de sudoku). Cette page est détenue par l'utilisateur.
Voir **Ex3**.

A4.9. Commandes **Curl**

Desc Les services web **ws** ont été utilisés dans le cadre du projet logiciel **examtt** (voir « **A4.5** »). Après que le logiciel **minizinc** containerisé ait été hébergé par les services web **ws**, les requêtes **génériques** ont pu être appelées pour résolution du problème dans sa version **toulbar2** ainsi que dans sa version **minizinc**. Les services web **ws** ont alors été utilisés pour vérification de conformité de la version **toulbar2** avec la version **minizinc** puis pour des résolutions d'instances.
Voir « B.1 ».

33 https://toulbar2.github.io/toulbar2/examples/vsapp.html#as-a-web-service

34 https://toulbar2.github.io/toulbar2/examples/vsapp.html#as-an-apk