Plateforme de tests subjectifs

Manuel de l’utilisateur







Étudiants :

|  |  |  |
| --- | --- | --- |
|  | Alexis | BLOND |
|  | Grégoire | COULOMBEL |

Encadrants :

|  |  |  |
| --- | --- | --- |
|  | Claude | SIMON |
|  | Damien | LOLIVE |

**Table des matières**

[1. Mise en place de l’environnement 3](#_Toc475715821)

[2. Architecture de l’application 3](#_Toc475715822)

[3. Création d’une plateforme de tests 4](#_Toc475715823)

[4. Configuration 7](#_Toc475715824)

[5. Les modèles bottle 8](#_Toc475715825)

[5.1. Les variables 8](#_Toc475715826)

[5.2. Structures de contrôle 8](#_Toc475715827)

[5.3. Les champs cachés 9](#_Toc475715828)

[5.4. Les sliders 9](#_Toc475715829)

[5.5. Ajouter du JavaScript / CSS 10](#_Toc475715830)

[5.6. Les samples 10](#_Toc475715831)

[6. Base de données 11](#_Toc475715832)

[6.1. Structure 11](#_Toc475715833)

[6.2. Exportation des données 11](#_Toc475715834)

[Annexes 13](#_Toc475715835)

[Les templates 13](#_Toc475715836)

[Le fichier Json de configuration 19](#_Toc475715837)

# Mise en place de l’environnement

L’environnement de notre application se base sur Python (2.7), car c’est un environnement présent sur de nombreuses machines. Cependant afin de pouvoir mettre en place notre application et la faire fonctionner, il est nécessaire d’installer des modules python supplémentaires sur la machine qui accueillera le serveur.

Voici ci-dessous la liste des modules nécessaires au fonctionnement de notre application :

* argparse
* beaker.middleware
* bottle
* csv
* datetime
* itertools
* json
* operator
* os
* paste
* pprint
* random
* re
* shutil
* sqlite3
* string
* sys

Les modules à installer sont indiqués dans le fichier *requirements.txt*. Ils peuvent être installés grâce au gestionnaire d’extension de python (pip) par le biais de la commande suivante :

|  |
| --- |
| pip install -r requirements.txt |

# Architecture de l’application

L’application se compose principalement d’un générateur qui fournira diverses plateformes pour les divers tests. Chaque plateforme créée par le générateur permet de servir un site web qui permet à des utilisateurs de venir se connecter dessus pour y réaliser des tests subjectifs. Les plateformes fonctionnent selon le modèle MVC.

L’architecture de l’application suit le schéma ci-dessous, avec :

* le générateur, nommé *generator.py*
* les plateformes, créées dans le dossier *tests*
* les données d’entrées, placées dans le dossier *input*

Cette architecture est celle utilisée, mais peut être modifiée sans difficulté.

Les plateformes contiennent les fichiers suivants :

* plateform.py : fichier python utilisant le moteur Bottle afin de gérer les routes et que l’on peut caractériser comme contrôleur.
* model.py : fichier python gérant les accès aux données présentes dans le fichier de configuration et faisant ainsi office de modèle.
* config.py : fichier python contenant les données de configuration ainsi que les métadonnées dans des variables.
* data.db : la base de données comprenant les questions posées ainsi que les réponses fournies par les différents utilisateurs
* ainsi que tous les fichiers propres au rendu des pages (à savoir les templates, les CSS, le JavaScript et les médias)

input

generator.py

templates

CSV

JSON

test 2

script.sh

tests

test 1

media

media1

static

views

media2

media3

templates

css-js files

plateform.py

model.py

config.py

data.db

*Schéma de l’architecture de l’application*

# Création d’une plateforme de tests

La création de l’application peut se faire de différentes façons, par le biais d’un script ou en ligne de commande, il y a 2 étapes principales lors de la création de l’application :

**Génération de l’application** : cette étape consiste à lancer le script de génération en lui fournissant en entrée les ressources nécessaires qui sont le fichier de description au format JSON et le fichier de template. Pour cela, il suffit de lancer le script par le biais de cette commande :

|  |
| --- |
| python generator.py |

Il faut cependant ajouter les options nécessaires :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **court** | **long** | **paramètres** | **description** | **requis** |
| -j | --json | chemin | fichier JSON | oui |
| -t | --main-tpl | chemin | modèle principal | oui |
| -i | --index-tpl | chemin | modèle pour la page d'index | oui |
| -c | --completed-tpl | chemin | modèle pour la page de fin de test | oui |
| -s | --systems | liste de chemins | liste des fichiers CSV des systèmes | oui |
| -n | --name | aucun | active l’écriture du nom après le chemin du système (par défaut : désactivé) | non |
| -v | --verbose | aucun | mode d’affichage verbeux | non |
|  | --csv-delimiter | caractère | définit le délimiteur CSV utilisé (par défaut : ';') | non |

Exemple de commande complète:

|  |
| --- |
| python generator.py -j config.json -t template.tpl -c .completed.tpl -i index.tpl -e export.tpl -s sys0.csv system1 sys1.csv system2 -n |

On obtient par la suite un dossier avec toute son arborescence qui a été créé dans le dossier tests du générateur.

De plus à la fin de l’opération, un token vous est fourni il vous servira à accéder au contenu de la BDD de votre test (voir partie 5. Exportation des données). Il vous faut donc le garder soigneusement.

**Remarque** : Si vous l’avez perdu ou oublié, vous pouvez toujours accéder au fichier config.py présent sur le serveur hébergeant l’application. En effet, ce fichier contient le token.

**Copie des fichiers nécessaires au fonctionnement de l’application** : après l’étape de génération, l’application n’est pas encore fonctionnelle. Pour cela, il faut lui fournir les fichiers js et css nécessaires dans le dossier *static* présent au sein de l’application générée. Dans le cas de la création par le biais du script, il faut modifier le chemin dans lequel les fichiers doivent être copiés.

La ligne de copie est de la forme :

|  |
| --- |
| cp -rf static/\* tests/nom du test/static/ |

**Remarque**: Il est tout à fait possible de rajouter autant de fichiers JavaScript et CSS que possible. Dans ce cas, il est nécessaire de ne pas oublier de copier les fichiers mentionnés dans les templates.

# Configuration

La principale façon de configurer l’application se fait par le fichier Json, pour cela voici les champs requis et leur fonctionnement :

|  |  |
| --- | --- |
| Tag | Description |
| name | Nom du test |
| author | Auteur du test |
| nbSteps | Nombre d’étapes à effectuer par utilisateur |
| nbSystemDisplayed | Nombre de systèmes affichés par étape |
| prefix | Préfixe utilisé pour le serveur |
| description | Description du test |
| nbQuestions | Nombre de questions par étape |
| nbFixedPosition | Nombre de systèmes présentés à la même position, ces systèmes sont présents dans les premiers indices du tableau |
| nbIntroductionSteps | Nombre d’étapes d’introduction |
| nbSampleBySystem | Nombre d’échantillons par système (global à tous les systèmes) |
| headersCSV | Liste des en-têtes des fichiers CSV |
| useMedia | Liste des en-têtes référençant un média (audio, vidéo, image) |

Le nombre de systèmes présentés par étape du test n’est pas nécessairement le même que le nombre total de systèmes. Cela peut se configurer grâce à la valeur *nbSystemDisplayed*. Le choix se fera alors aléatoirement parmi tous les systèmes.

Afin de présenter un ou plusieurs systèmes de référence (c'est-à-dire qui ne seront pas évalués et dont la position sur la page sera fixe et déterminée), cela peut se faire grâce au champ *nbFixedPosition*. Ce champ indique le nombre de systèmes de référence, qui seront les premiers systèmes présentés au générateur.

Les étapes d’introductions (dont le nombre est fixé par *nbIntroductionSteps*) sont prises parmi les possibles questions posées. Elles ne nécessitent pas ni systèmes ni échantillons supplémentaires.

En plus de ces champs indispensables, il est également possible d’ajouter d’autres champs. Ces champs seront accessibles et utilisables depuis les templates.

1. Les modèles bottle
   1. Les variables

Lorsque vous souhaitez interpréter le contenu des variables comme du code HTML, il faut l’indiquer grâce au caractère spécial “point d’exclamation” : {{ **!** my\_var }} sinon il sera interprété comme du texte.

Différentes variables sont proposées dans modèles (ou template), en effet il existe :

* *Samples* qui est une variable contenant l’ensemble des échantillons. L’utilisation de cette variable est détaillée dans la partie 4.5.
* *Systems* qui contient l’ensemble des systèmes.
* *User* contenant l’email de l’utilisateur qui s’est connecté.
* Ainsi que toutes les variables données dans le fichier de configuration.

**Remarque:** Pour accéder à un élément dans un tableau, il faut suivre la syntaxe suivante :

|  |
| --- |
| {{ my\_tab[index] }} |

* 1. Structures de contrôle

Il est aussi possible de rajouter des structures de contrôle au sein du modèle, qui peuvent être :

* **Itératives**:

|  |
| --- |
| % **for** myvar **in** domain  …   # il est possible d’appeler myvar dans le code avec {{ myvar }} % end |

* **Conditionnelles** :

|  |
| --- |
| % **if** condition  ...  # le code qui sera exécuté si la condition est vraie % end |

* 1. Les champs cachés

Afin de simplifier la création de samples pour les concepteurs de test, nous avons également ajouté une variable au sein du template afin de gérer les champs du formulaire qui doivent être présents. Ils sont donc générés automatiquement et utilisés dans le template grâce à une variable utilisée de la façon suivante : {{ ! hidden\_fields }}

Si vous oubliez cette variable dans le template ou que vous la mettez en dehors des balises <form> du template, une erreur vous sera retournée lorsque quelqu’un accèdera à votre page de test.

* 1. Les sliders

Pour intégrer les sliders, il faut ajouter dans le template le code JavaScript suivant :

**Code JavaScript**:

|  |
| --- |
| <**script**>  $(**function**() {  $("#slider{{i}}").slider({  range: "min",  value:5,  min: 0,  max: 10,  step: 1,  slide: **function**(event, ui) {  $("#rate{{i}}").html(ui.value);  $("#question{{i}}").attr("value",ui.value);  }  });  }); </**script**> |

**Code HTML** :

|  |
| --- |
| <**div** class="answer">  <**div** id="slider{{i}}">  <**div** id="rate{{i}}" class="ui-slider-handle">5</**div**>  </**div**>  <**label** style="color: red; margin-top:5px; margin-bottom:5px;">0: impossible</**label**>  <**label** style="color: green; margin-top:5px; margin-bottom:5px; float: right;">10: perfectly possible</**label**> </**div**> <**input** type="hidden" id="question{{i}}" name="question{{i}}" value="5"> |

**Code CSS** : (optionnel)

|  |
| --- |
| <**style**>  #rate{{i}} {  width: 3em;  height: 1.6em;  top: 50%;  font-weight: bold;  margin-top: -.8em;  text-align: center;  line-height: 1.6em;  } </**style**> |

**Remarque :** Dans ce cas, le slider est créé sur une variable i définie dans le template. Il est tout à fait possible de remplacer ce i par une valeur numérique.

* 1. Ajouter du JavaScript / CSS

Il est possible d’ajouter d’autres fichiers JavaScript ou CSS. Pour cela il faut ajouter leurs références dans le template par le biais de la ligne suivante :

* CSS : <link href="{{APP\_PREFIX}}/static/css/mystylesheet.css" rel="stylesheet">
* JS : <script src="{{APP\_PREFIX}}/static/js/myjsscript.js"></**script**>

**Attention : Il ne faut pas oublier de les mettre dans le dossier contenant les fichiers static utilisés lors de la génération ou de les copier à la main pour pouvoir y avoir accès. Leur emplacement se situe dans le fichier** *static/css* **ou** *static/js* **du test une fois créé.**

* 1. Les samples

L’accès aux samples se fait de manière simple, comme s’il s’agissait d’une structure stockant les données sous forme de tableau. La variable samples permet d’accéder à tous les samples. Il est alors nécessaire de choisir le numéro du sample, puis enfin la colonne voulue (qui est elle définie dans le fichier de configuration). Par exemple {{ !samples[0][‘‘text’’] }} accédera au champ *‘‘text’’* du premier échantillon de l’étape courante.

Il est aussi important de savoir que si l’affichage des systèmes dans un ordre aléatoire est choisi (en définissant un système fixe), ce système se situera toujours dans le premier indice des tableaux *samples* et *systems*.

Pour insérer des samples de type audio, il faut mettre:

|  |
| --- |
| <**audio** id="player" controls>  <**source** src="{{samples[0]["path"]}}"> </**audio**> |

Dans le cas d’une image, le code sera le suivant :

|  |
| --- |
| <**img** src="path/to/file"/> |

Et dans le cas d’une vidéo, cela deviendrait :

|  |
| --- |
| <**video** width="320" height="240" controls>  <source src="{{samples[0]["pathMP4"]}}" type="video/mp4">  <source src="{{samples[0]["pathOGG"]}}" type="video/ogg">  Your browser does not support the video tag. </**video**> |

**Remarque** : Il est également possible d’appeler une vidéo présente sur internet, cela dépendra cependant de la plateforme source qui est utilisée.

1. Base de données
   1. Structure

Notre base de données est composée de 3 tables :

* Answer : contenant l’ensemble des données stockées suite
* Sample : contenant les données des échantillons présents pour le test
* System : permettant l’identification des systèmes

Voici la composition de la table *answer* :

* id : identification de la réponse
* user : identifiant de la personne qui a répondu au test
* date : date de la validation de la réponse
* content : valeur de la réponse
* content-target : System concerné par la réponse (si nécessaire)
* sample-index : Indice de l’échantillon au sein du système
* question-index : Indice de la question qui a été répondue
* systemX : identifiant du système présent à la place X dans le test pour cette étape

On peut ainsi voir que la valeur contenu dans le champ **content** diffère, de même pour le champ **content-target**, dont la valeur dépend du type de question effectué. En effet si la question est de type “note”, la valeur présente dans le champ **content** sera une valeur numérique (la note donnée par la personne ayant effectué le test) et dans le champ **content-target**, nous aurons l’identifiant du système qui a été noté.

Alors que dans le cas d’une comparaison entre deux échantillons, le champ **content** contiendra le nom du système représenté par l’échantillon et le champ **content-target** sera vide.

* 1. Exportation des données

Les résultats des tests sont disponibles à l’URL suivant :

|  |
| --- |
| http://server-addr:8080/export |

Vous aurez alors accès à deux types différents d’exportation :

* la base de données SQLite complète au format .db
* un fichier csv comportant uniquement les informations utiles à l’exploitation des données (c’est-à-dire le contenu de la table *answer*)

Afin de pouvoir accéder à ces fichiers, vous devrez renseigner le token qui vous a été fourni lors de la création de l’application.

# Annexes

## Les templates

*Template principal :*

|  |
| --- |
| <!DOCTYPE html> <**html** lang="en"> <**head**>  <**meta** charset="utf-8">  <**title**>Perceptual test platform - IRISA/Expression</**title**>  <**link** href="{{APP\_PREFIX}}/static/css/bootstrap.min.css" rel="stylesheet">  <**link** href="{{APP\_PREFIX}}/static/css/tests.css" rel="stylesheet">  <**link** href="{{APP\_PREFIX}}/static/css/jquery-ui.min.css" rel="stylesheet">  <**script** src="{{APP\_PREFIX}}/static/js/jquery.js"></**script**>  <**script** src="{{APP\_PREFIX}}/static/js/jquery-ui.min.js"></**script**>   <**style**>  .rate {  width: 3em;  height: 1.6em;  top: 50%;  font-weight: bold;  margin-top: -.8em;  text-align: center;  line-height: 1.6em;  }  .vcenter {  display: inline-block;  vertical-align: middle;  float: none;  }  </**style**> </**head**>  <**body**>  <**nav** class="navbar navbar-default" style="margin-bottom:0">  <**div** class="container-fluid">  <**ul** class="nav navbar-nav navbar-right">  <**li**><**a**>{{user}}</**a**></**li**>  <**li**><**a** href="{{APP\_PREFIX}}/logout">Logout</**a**></**li**>  </**ul**>  </**div**>  </**nav**>  <**form** role="form" action="{{APP\_PREFIX}}/test2" method="POST">  <**div** class="jumbotron text-center">  <**h2**><**b**>Context:</**b**> Imagine someone tells the text below during a spontaneous conversation.</**h2**>  <**h2**><**b**>Question:</**b**> How likely do you judge the spoken propositions?</**h2**>  <**br**>  <**h4**><**i**>Remark: Some propositions may be identical, some may differ only very slightly.</**i**></**h4**>  </**div**>  % if introduction :  <**div** class="alert alert-warning text-center"><**strong**>Warning!</**strong**> This is an introduction step!</**div**>  % end  {{! hidden\_fields}}  <**div** class="container">  <**h1** class="text-center">Text {{step}}/{{totalstep}}</**h1**>  </**div**>  <**div** class="container">  <**div** class="col-md-4 col-md-offset-4">  <**div** class="progress" style="height: 2px">  <**div** class="progress-bar" role="progressbar" aria-valuenow="{{progress}}" aria-valuemin="0" aria-valuemax="100" style="width:{{progress}}%">  </**div**>  </**div**>  </**div**>  </**div**>  <**div** class="container">  <**center**>  <**samp** class="lead text-center">  {{!samples[0]['text']}}  </**samp**>  </**center**>  </**div**><**br**>  % for i in range(nfixed,len(systems)):  <**div** class="container">  <**h2**>Proposition {{i}}</**h2**>  <**div** class="row">  <**div** class="col-xs-12 col-md-7 col-lg-7 vcenter">  <**blockquote**>  <**samp** class="text-justify">{{!samples[i]['text']}}</**samp**>  </**blockquote**>  </**div**>  <**div** class="col-xs-12 col-md-4 col-lg-4 vcenter">  <**div** class="answer">  <**div** id="slider{{i}}">  <**div** id="rate{{i}}" class="ui-slider-handle">5</**div**>  </**div**>  <**label** style="color: red; margin-top:5px; margin-bottom:5px;">0: impossible</**label**>  <**label** style="color: green; margin-top:5px; margin-bottom:5px; float: right;">10: perfectly possible</**label**>  </**div**>  <**input** type="hidden" id="question{{i}}" name="question{{i}}" value="5">  <**input** type="hidden" id="target\_question{{i}}" name="target\_question{{i}}" value="{{systems[i]}}">  </**div**>  </**div**>   </**div**>  <**style**>  #rate{{i}} {  width: 3em;  height: 1.6em;  top: 50%;  font-weight: bold;  margin-top: -.8em;  text-align: center;  line-height: 1.6em;  }  </**style**>  <**script**>  $(**function**() {  $("#slider{{i}}").slider({  range: "min",  value:5,  min: 0,  max: 10,  step: 1,  slide: **function**(event, ui) {  $("#rate{{i}}").html(ui.value);  $("#question{{i}}").attr("value",ui.value+";;{{systems[i]}}");  }  });  });  </**script**>  % end  <**div** class="container">  <**div** class="row">  <!-- answer part -->  <**div** class="col-md-6 col-md-offset-3">  <**input** id="next" type="submit" class="btn btn-lg btn-success btn-block pull-right" value="Next" style="margin-top: 20px;">  </**div**>  </**div**>  </**div**>  </**form**>  <**br**><**br**><**br**> </**body**> </**html**> |

*Template de login :*

|  |
| --- |
| <!DOCTYPE html> <**html** lang="en"> <**head**>  <**meta** charset="utf-8">  <**title**>Subjective tests platform - {{config["name"]}}</**title**>  <!-- Bootstrap Core CSS -->  <**link** href="{{APP\_PREFIX}}/static/css/bootstrap.min.css" rel="stylesheet">  <**link** href="{{APP\_PREFIX}}/static/css/tests.css" rel="stylesheet">  <**link** href="{{APP\_PREFIX}}/static/css/jquery-ui.min.css" rel="stylesheet">  <**script** src="{{APP\_PREFIX}}/static/js/jquery.js"></**script**>  <**script** src="{{APP\_PREFIX}}/static/js/jquery-ui.min.js"></**script**> </**head**> <**body**>  <**div** class="container">  <**div** class="row">  <**div** class="col-md-4 col-md-offset-4 text-center">  <**h1**>{{config["name"]}}</**h1**>  <**p** class="lead">{{config["description"]}}</**p**>  </**div**>  </**div**>  </**div**>  <**div** class="jumbotron">  <**div** class="container">  <**div** class="row">  <**div** class="col-md-4 col-md-offset-4">  <**h3**>Please provide an e-mail address to identify yourself:</**h3**>  <**form** role="form" action="{{APP\_PREFIX}}/login" method="POST">  <**fieldset**>  <**div** class="form-group">  <**input** type="text" class="form-control" placeholder="E-mail" name="email" autofocus required>  </**div**>  <!-- Change this to a button or input when using this as a form -->  <**input** type="submit" class="btn btn-lg btn-success btn-block" value="Start/Continue">  </**fieldset**>  </**form**>  <**br**>  %if defined('error') and error != "" :  <**div** class="alert alert-danger">  <**p**><**strong**>Error !</**strong**> {{error}}</**p**>  </**div**>  %end  </**div**>  </**div**>  </**div**>  </**div**>  <**div** class="container">  <**div** class="row">  <**div** class="col-md-4 col-md-offset-2">  <**img** src="{{APP\_PREFIX}}/static/img/logo\_irisa.png" class="img-responsive center-block" alt="IRISA lab">  </**div**>  <**div** class="col-md-4">  <**img** src="{{APP\_PREFIX}}/static/img/logo\_expression.png" class="img-responsive center-block" alt="Expression team">  </**div**>  </**div**>  </**div**>  </**body**> </**html**> |

*Template d’exportation* :

|  |
| --- |
| <!DOCTYPE html> <**html** lang="en"> <**head**>  <**meta** charset="utf-8">  <**title**>Subjective tests platform - {{config["name"]}}</**title**>   <!-- Bootstrap Core CSS -->  <**link** href="{{APP\_PREFIX}}/static/css/bootstrap.min.css" rel="stylesheet">  <**link** href="{{APP\_PREFIX}}/static/css/tests.css" rel="stylesheet">  <**link** href="{{APP\_PREFIX}}/static/css/jquery-ui.min.css" rel="stylesheet">  <**script** src="{{APP\_PREFIX}}/static/js/jquery.js"></**script**>  <**script** src="{{APP\_PREFIX}}/static/js/jquery-ui.min.js"></**script**> </**head**> <**body**>  <**div** class="container">  <**div** class="row">  <**div** class="col-md-4 col-md-offset-4 text-center">  <**h1**>{{config["name"]}}</**h1**>  <**p** class="lead">{{config["description"]}}</**p**>  </**div**>  </**div**>  </**div**>  <**div** class="jumbotron">  <**div** class="container">  <**div** class="row">  <**div** class="col-md-4 col-md-offset-4">  <**h3**>Please provide the token given to you at the creation of the test</**h3**>  <**form** role="form" action="{{APP\_PREFIX}}/export" method="POST">  <**fieldset**>  <**div** class="form-group">  <**input** type="text" class="form-control" placeholder="Token" name="token" autofocus required>  </**div**>  <!-- Change this to a button or input when using this as a form -->  <**input** type="submit" class="btn btn-lg btn-success btn-block" value="Submit">  </**fieldset**>  </**form**>  <**br**>  %if defined('error') and error != "" :  <**div** class="alert alert-danger">  <**p**><**strong**>Error !</**strong**> {{error}}</**p**>  </**div**>  %end  </**div**>  </**div**>  </**div**>  </**div**>  </**div**>   <**div** class="container">  <**div** class="row">  <**div** class="col-md-4 col-md-offset-2">  <**img** src="{{APP\_PREFIX}}/static/img/logo\_irisa.png" class="img-responsive center-block" alt="IRISA lab">  </**div**>  <**div** class="col-md-4">  <**img** src="{{APP\_PREFIX}}/static/img/logo\_expression.png" class="img-responsive center-block" alt="Expression team">  </**div**>  </**div**>  </**div**> </**body**> </**html**> |

*Template de fin* :

|  |
| --- |
| <!DOCTYPE html> <**html** lang="en"> <**head**>  <**meta** charset="utf-8">  <**title**>Subjective tests plateform - {{config["name"]}}</**title**>  <!-- Bootstrap Core CSS -->  <**link** href="{{APP\_PREFIX}}/static/css/bootstrap.min.css" rel="stylesheet">  <**link** href="{{APP\_PREFIX}}/static/css/tests.css" rel="stylesheet">  <**link** href="{{APP\_PREFIX}}/static/css/jquery-ui.min.css" rel="stylesheet">  <**script** src="{{APP\_PREFIX}}/static/js/jquery.js"></**script**>  <**script** src="{{APP\_PREFIX}}/static/js/jquery-ui.min.js"></**script**> </**head**> <**body**>  <**div** class="container">  <**div** class="row">  <**div** class="col-md-4 col-md-offset-4 text-center">  <**h1**>{{config["name"]}}</**h1**>  <**p** class="lead">{{config["description"]}}</**p**>  </**div**>  </**div**>  </**div**>  <**div** class="jumbotron">  <**div** class="container">  <**div** class="row">  <**div** class="col-md-6 col-md-offset-3">  <**h2** class="text-center">Test completed!</**h2**>  <**h2** class="text-center">Thank you for your time.</**h2**>  <**br**><**br**>  <**h4** class="text-center">You have been logged out. You can close the page.</**h4**>  <**br**>  </**div**>  </**div**>  </**div**>  </**div**>  <**div** class="container">  <**div** class="row">  <**div** class="col-md-4 col-md-offset-2">  <**img** src="{{APP\_PREFIX}}/static/img/logo\_irisa.png" class="img-responsive center-block" alt="IRISA lab">  </**div**>  <**div** class="col-md-4">  <**img** src="{{APP\_PREFIX}}/static/img/logo\_expression.png" class="img-responsive center-block" alt="Expression team">  </**div**>  </**div**>  </**div**> </**body**> </**html**> |

## Le fichier Json de configuration

Voici ci-dessous un fichier de configuration à titre d’exemple :

|  |
| --- |
| {  "name": "Nom du test",  "author": "Nom de l’auteur",  "nbSteps": 8,  "nbSystemDisplayed": 5,  "description": "Une brève description du test...",  "prefix": "prefixe-de-l-application-sur-le-serveur",  "nbQuestions": 4,  "nbFixedPosition": 1,  "nbIntroductionSteps": 1,  "nbSampleBySystem": 10,  "headersCSV": ["type1","type2","..."],  "useMedia": ["type2","..."],  "custom": "exemple de valeur définie par le concepteur du test" } |

Il est également possible de rajouter dans le fichier de configuration autant de champs que voulu. Ces champs seront accessibles depuis les templates. Dans l’exemple ci-dessus, c’est le cas du champ “custom”.