Cégep à distance

Tableau de bord

Guide du développeur

Version 1.0

Historique des révisions

Date	Version	Description	Auteur
2022-04-16	1.0	Première version du document	Guillaume Rompré, Jasmine Roy

Table des matières

Guide du développeur

1. Information générale

1.1 Introduction

Le tableau de bord du Cégep à distance est constitué de quatre applications. Les applications sont interdépendantes. Pour le bon fonctionnement du système, il est donc nécessaire que toutes les applications soient déployées simultanément.

1.2 Dépendances

Le tableau de bord est dépendant des applications suivantes pour la consommation de données:

- Base de données Coba du Cégep à distance
- Base de données Moodle du cégep à distance

Une panne de l'un de ces systèmes aura des conséquences sur le bon fonctionnement du tableau de bord.

2. Déploiement

2.1 Déploiement de l'application de prédiction

2.1.1 Description

L'application de prédiction est programmée en Python 3.9. Le serveur qui accueille cette application doit donc posséder un environnement capable d'exécuter du code Python 3.9. Pour lancer l'application, il suffit de faire la commande suivante dans une console: **python main.py**

2.1.2. Dépendances

Le tableau 1 présente les dépendances nécessaires à installer.

Tableau 1: Dépendances de l'application de prédiction

#	Dépendance	Version
1	sqlalchemy	1.3.24
2	dotenv	0.20.0
3	pandas	1.4.2
4	pymongo	4.1.1
5	numpy	1.22
6	apscheduler	3.9.0

2.1.3. Variables d'environnement

Le fichier pour les variables d'environnement doit se trouver à la racine de l'application et avoir le nom suivant: **.env**. Chaque ligne doit comporter une variable et sa valeur. Par exemple, pour une variable V donc la valeur est X, il faut inscrire: V="X". Le tableau suivant présente les variables d'environnement nécessaires, un exemple ainsi qu'une description.

Tableau 2: Variables d'environnement de l'application de prédiction

#	Variable	Example	Description
1	COBA_SERVER	"10.88.0.107"	Adresse IP pour le serveur de base de données Coba
2	COBA_DATABASE	"TableauBordCad"	Nom de la base de données Coba
3	COBA_TABLE_INSCRIPTION	"TableauBordCad.dbo.VUE_CADREC04"	Nom de la table contenant les inscriptions
4	COBA_TABLE_STUDENT	"TableauBordCad.dbo.VUE_CADREC03"	Nom de la table contenant les étudiants
5	COBA_USERNAME	"CobaTableauBordCad"	Nom d'utilisateur pour accéder à la base de données Coba
6	COBA_PASSWORD	"XXX"	Mot de passe pour accéder à la base de données Coba
7	MOODLE_SERVER	"10.88.0.115"	Adresse IP pour le serveur de base de données Moodle
8	MOODLE_DATABASE	"entrepot"	Nom de la base de données Moodle
9	MOODLE_TABLE	"entrepot.donnees"	Nom de la table contenant les actions Moodle
10	MOODLE_USERNAME	"tableau_de_bord"	Nom d'utilisateur pour accéder à la base de données Moodle
11	MOODLE_PASSWORD	"XXX"	Mot de passe pour accéder à la base de données Moodle
12	DASHBOARD_DB_CONNECTION_STRING	"mongodb+srv://cad-db:JfsfvKTvkxT@cluster0.cnu5v.localhost"	Adresse de connexion à la base de donnée du tableau de bord
13	DASHBOARD_DATABASE	"cad-db"	Nom de la base de donnée du tableau de bord
14	DASHBOARD_INSCRIPTIONS_COLLECTION_NAME	"prediction-inscriptions"	Nom de la collection dans la base de données du tableau de bord pour les inscriptions
15	DASHBOARD_ACTIONS_COLLECTION_NAME	"prediction-actions"	Nom de la collection dans la base de données du tableau de bord pour les actions
16	DASHBOARD_PREDICTIONS_COLLECTION_NAME	"predictions"	Nom de la collection dans la base de données du tableau de bord pour les prédictions
17	DASHBOARD_SYNC_METADATA_COLLECTION_NAME	"prediction-synchronization"	Nom de la collection dans la base de données du tableau de bord pour les métadata de synchronisation.

#	Variable	Example	Description
18	SYNCHRONIZATION_TIME_HOUR	"5"	Heure de la synchronisation. Si la valeur de la variable 18 est “5” et celle de la variable 19 est “0”, alors la synchronisation se fera chaque jour à 5h AM.
19	SYNCHRONIZATION_TIME_MINUTE	"0"	Minute de synchronisation.

2.2 Déploiement de l’application backend

2.2.1. Description

L’application backend est programmée en TypeScript. Le serveur qui accueille cette application doit donc posséder un environnement capable d’exécuter du code JavaScript. Pour lancer l’application, il faut tout d’abord installer NPM (https://www.npmjs.com/), puis faire la commande suivante dans la console: **npm start**

2.2.2. Dépendances

L’application possède plusieurs dépendances. La façon la plus efficace et rapide est d’installer NPM (https://www.npmjs.com/) puis dans la console: **npm install**

2.2.3. Variables d’environnement

Le fichier pour les variables d’environnement doit se trouver à la racine de l’application et avoir le nom suivant: **.env**. Chaque ligne doit comporter une variable et sa valeur. Par exemple, pour une variable V donc la valeur est X, il faut inscrire: V=X. Le tableau suivant présente les variables d’environnement nécessaires, un exemple ainsi qu’une description.

Tableau 3: Variables d’environnement de l’application backend

#	Variable	Example	Description
1	DB_URL	"mongodb+srv://cad-db:JfsfvKTvkxT@cluster0.cnu5v.localhost”	Adresse de connexion à la base de donnée du tableau de bord
2	MAIL_HOST	smtp-mail.outlook.com	Hôte pour le serveur de courriel
3	MAIL_PORT	587	Port pour le serveur de courriel
4	MAIL_ADDRESS	email@email.com	Adresse courriel pour l’envoi de courriels
5	MAIL_PASSWORD	p4ssw0rd	Mot de passe de l’adresse courriel pour l’envoi de courriels
6	CLIENT_URL	http://tableau-de-bord.cad.ca	URL de l’application frontend
7	NODE_ENV	‘developement’ ou ‘production’	Inscrire ‘developement’ ou ‘production’ dépendamment quel environnement est déployé
8	COBA_SERVER	"10.88.0.107"	Adresse IP pour le serveur de base de données Coba

#	Variable	Example	Description
9	COBA_DATABASE	"TableauBordCad"	Nom de la base de données Coba
10	COBA_USERNAME	"CobaTableauBordCad"	Nom d'utilisateur pour accéder à la base de données Coba
11	COBA_PASSWORD	"p4ssw0rd"	Mot de passe pour accéder à la base de données Coba
12	COBA_TABLE_INSCRIPTION	"TableauBordCad.dbo.VUE_CADREC04"	Nom de la table qui contient les inscriptions
13	COBA_TABLE_STUDENT	"TableauBordCad.dbo.VUE_CADREC03"	Nom de la table qui contient les étudiants
14	COBA_TABLE_IND_OCCUPATION	"TableauBordCad.dbo.VUE_COLELE01"	Nom de la table qui contient les indices d'occupation
15	COBA_TABLE_DATE	"TableauBordCad.dbo.VUE_CADINS01"	Nom de la date qui contient les dates de remise
16	COBA_TABLE_NAME	"TableauBordCad.dbo.VUE_COLMEQ01"	Nom de la table qui contient les noms des cours
17	FIRST_NOSEQINSCR	753975	Numéro de la première inscription à synchroniser. Par la suite, la synchronisation va regarder l'historique pour savoir quelles inscriptions ajoutées.
18	TOKEN_VALIDITY_IN_MINUTES	10	Temps de validité du jeton d'authentification
19	TOKEN_GRACE_PERIOD_IN_MINUTES	60	Période de grâce pour le jeton d'authentification
20	SYNCHRONIZATION_TIME_HOUR	5	Heure de la synchronisation
21	SYNCHRONIZATION_TIME_MINUTE	0	Minute de la synchronisation
22	SYNCHRONIZATION_TIME_SECOND	0	Seconde de la synchronisation
23	PASSWORD_RESET_TOKEN_EXPIRATION_IN_MINUTES	1440	Période de temps (en minutes) durant laquelle le lien pour modifier le mot de passe est valide
24	MAX_SEARCH_RESULTS	10000	Pour éviter de surcharger la mémoire du serveur lors d'une requête SQL massive, ce paramètre permet de limiter le nombre de résultats
25	PROGRESSION_FILE_ROOT	C:\app\tableau-de-bord\src\web-app\backend\src\data-files\	Lien vers les fichiers de progression

2.3 Déploiement de l'application frontend

2.3.1 Description

L'application frontend est programmée en TypeScript. Le serveur qui accueille cette application doit donc posséder un environnement capable d'exécuter du code JavaScript. Pour lancer l'application, il faut tout d'abord installer Angular CLI (https://angular.io/cli) , puis faire la commande suivante dans la console: **ng build**. Les fichiers à déployer seront dans le dossier **dist**. Ensuite, il suffit de les servir à partir d'un serveur comme nginx (https://www.nginx.com/)

2.3.2 Dépendances

L'application possède plusieurs dépendances. La façon la plus efficace et rapide est d'installer NPM (https://www.npmjs.com/) puis dans la console: npm install

2.3.3 Variables d'environnement

Le fichier pour les variables d'environnement doit se trouver dans frontend/src/environments et avoir comme nom: environment.prod.ts ou bien environment.ts. Le fichier environment.prod.ts est utilisé lors d'un build alors que environment.ts est utilisé lors du développement.

```
export const environment = {
    production: false,
    server: 'http://localhost:4000'
}
```

Figure 1. Exemple de fichier environment.ts

```
export const environment = {
    production: true,
    server: 'http://localhost:8080'
}
```

Figure 2. Exemple de fichier environment.prod.ts

2.4 Déploiement de la base de données

2.4.1. Description

La base de données est MongoDB. Le serveur qui accueille cette application doit donc installer MongoDB puis suivre les étapes pour créer une base données: https://www.mongodb.com/docs/manual/installation/.

3. Maintenance

La section qui suit présente les recommandations pour la maintenance et amélioration de l'application.

3.1 Texte et internationalisation

3.1.1 Description

L'entièreté du texte visible de l'application est internationalisée, de sorte que l'on retrouve l'entièreté des valeurs textuelles de l'application en un seul dossier. Le module *TranslateModule* de *ngx-translate* (https://github.com/ngx-translate/core) est utilisé à cette fin.

On trouve ce fichier dans le frontend, sous **src/app/assets/i18n**. Le dossier contient actuellement uniquement le fichier pour la langue française, nommé **fr.json**.

3.1.2 Ajout de langues au dossier de ressources

Un fichier json contenant toutes les clés possibles est nécessaire pour chaque langue. Ainsi, pour ajouter une langue à l'application, il faudra d'abord ajouter un fichier au nom correspondant à la langue désirée, par exemple **en.json** pour l'anglais. Il faudra ensuite peupler ce fichier avec toutes les clés existantes dans le fichier français, puis remplacer les valeurs par les traductions adéquates.

3.1.3 Modification du texte existant

Pour modifier le texte existant, il suffit de trouver la clé correspondante à celui-ci dans le fichier de ressources textuelles et d'en changer la valeur associée. Le nouveau texte sera visible dès le rafraichissement de la page; aucune nouvelle compilation n'est nécessaire.

3.1.4 Ajout de nouveau texte

Pour ajouter une nouvelle valeur utilisable de texte dans l'application, il faut ajouter aux fichiers de ressource la nouvelle clé et son texte associé.

Par convention, les clés sont en majuscules et tout espace est remplacé par "_". Également, les clés sont divisées par sections afin de les retrouver et choisir plus facilement.

Pour afficher du texte dans un fichier HTML, il suffit d'utiliser la clé et le pipeline de traduction, comme montré à la figure 1. Les niveaux enfants des clés imbriquées sont séparés par des points (".").

Puisque les importations nécessaires sont présentes dans le **SharedModule**, il est important d'ajouter celui-ci aux **imports** de tout nouveau module créé.

```
<div *ngIf="!showQuestionnaire">
  <p>{{ 'WHEEL.QUESTIONNAIRE_GOAL' | translate }}</p>
  <p>{{ 'WHEEL.QUESTIONNAIRE_COMPLETION' | translate }}</p>
  <p>{{ 'WHEEL.READY' | translate }}</p>
</div>
```

Figure 3. Utilisation des clés de texte internationalisé dans un fichier HTML

3.2 Modification des questionnaires

3.2.1 Modification du texte des questions

Se référer à la section 3.1.3 Modification du texte existant.

3.2.2 Modification du questionnaire de parcours

Il n'est pas conseillé de modifier le questionnaire de parcours, car, bien que les questions soient affichées dynamiquement, le processus entourant la sauvegarde du questionnaire est particulièrement dépendant de l'infrastructure actuelle des questions.

On peut retrouver les questions dans le frontend sous **src/app/modules/questionnaire/services/question/question.service.ts**.

3.2.3 Modification du questionnaire de la roue de la réussite

Le processus d'affichage et de sauvegarde dynamique des questions de la roue de la réussite permet la modification post-déploiement du questionnaire.

Pour éviter un possible conflit de sauvegarde, le numéro de chaque question ne doit jamais être modifié ni réutilisé. Si une question est retirée du questionnaire, son numéro doit rester inutilisé dans le futur, à moins d'avoir préalablement effacé de la base de données tous les questionnaires déjà sauvegardés. Dans le même ordre d'idée, toute nouvelle question introduite se doit d'avoir un nouveau numéro. Les questions existantes sont présentement

numérotées de 1 à 98. Une nouvelle question prendra donc le numéro 99, peu importe son placement dans les sept aspects de la roue.

On peut retrouver les questions dans le frontend sous **src/app/modules/questionnaire/services/question/question.service.ts**. La première moitié du fichier décrit les questions du questionnaire de la réussite, il faut donc trouver la fonction **getWheelQuestions** qui marque le début des questions de la roue. Les deux premières questions sont les questions de consentement.

Pour modifier l'ordre d'apparition des aspects, il suffit de modifier les valeurs de **order** que l'on peut voir à la ligne 272 de la figure 2. Les questions de consentement doivent garder les positions 0 et 1, mais l'ordre des aspects (de 2 à 8) peut être modifié à souhait. L'aspect correspondant à la liste de question est indiqué par la valeur de dimension, visible à la ligne 275 de la figure 2. Les valeurs possibles de cette variable sont dictées par l'**enum WHEEL_DIMENSIONS** de la figure 3.

```
249         new WheelQuestion({
250           key: '1',
251           label: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.TEXT',
252           options: [
253             {key: '1', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.1'},
254             {key: '2', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.2'},
255             {key: '3', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.3'},
256             {key: '4', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.4'},
257             {key: '5', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.5'},
258             {key: '6', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.6'},
259             {key: '7', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.7'},
260             {key: '8', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.8'},
261             {key: '9', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.9'},
262             {key: '10', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.10'},
263             {key: '11', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.11'},
264             {key: '12', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.12'},
265             {key: '13', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.13'},
266             {key: '14', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.14'},
267             {key: '15', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.15'},
268             {key: '16', value: 'WHEEL_QUESTIONNAIRE.QUESTIONS.DISTANCE_STUDY.16'},
269
270           ],
271           numberControls: 16,
272           order: 2,
273         },
274         {
275           dimension: WHEEL_DIMENSIONS.distanceStudy,
276           invertedValues: [5, 7, 11]
277         }
```

Figure 4. Questions d'un aspect de la roue de la réussite

Pour ajouter une question dans un aspect, il suffit d'ajouter une ligne dans l'aspect souhaité. Pour ajouter une question à l'aspect *Études à distance* (**distanceStudy**) dans la figure 2, il suffit d'ajouter à la ligne 269 une nouvelle entrée, avec le numéro de la question (ex. 99) comme valeur à **key** et la clé du texte internationalisé dans **value**. Il faut également ajouter la clé et sa valeur dans les fichiers ressources, comme indiqué dans la section 3.1.4. Si la question ajoutée fait partie de la catégorie des questions inversées, il faut également ajouter son numéro dans **invertedValues**. Le calcul des résultats de la roue utilise cette liste pour savoir quel pointage donner aux choix de l'utilisateur. Pour chaque numéro de question ajouté, il faut également incrémenter la constante **NUMBER_WHEEL_QUESTIONS** dans le **export.service**, sous **src/services/search/export/export.service.ts**, du backend, sans quoi la question ne sera pas exportée par l'outil administrateur.

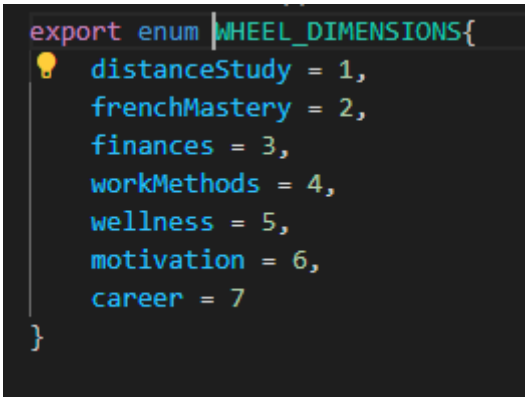

Figure 5. Valeurs possibles pour l'**enum WHEEL_DIMENSIONS**

Pour retirer une question de la roue de la réussite, il suffit de retirer la ligne correspondante dans le fichier de question, sous le bon aspect. La clé et son texte peuvent également être supprimés du fichier de ressources si désiré. La question n'apparaîtra plus, mais des valeurs préexistantes seront toujours disponibles dans l'exportation. Il ne faut pas décrémenter la constante, sans quoi des questions seront manquantes.

3.3 Migration en SQL

L'application utilise une base de données NoSQL. Le Cégep à distance utilise une base de données SQL. Pour augmenter les performances du système lors d'une recherche, il est suggéré de migrer la base de données de l'application en SQL. Cela requiert du travail de développement dans les classes *repository* du backend.

3.4 Sécurité

Afin d'augmenter la sécurité de l'application est éviter une attaque de type *brute force*, il est recommandé de changer la clé de cryptage du backend à intervalle régulier.