B5 - Application Development

B-DEV-500

Dashboard

Toutes vos informations en un clin d'oeil

1.1

{EPITECH.}

Dashboard

repository name: DEV_dashboard_$ACADEMICYEAR
repository rights: ramassage-tek
language: Java, .NET, node.js
compilation: docker-compose build && docker-compose up (cf. *Construction du Projet*)

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).

Considérations générales

Dans le cadre de ce projet, vous endosserez le rôle d'un Architecte Software.

Votre principal objectif n'est ni de réinventer la roue ni d'écrire beaucoup de lignes de code. Bien au contraire, votre action principale est de comprendre, de sélectionner puis d'intégrer une large palette de bibliothèques existantes.

Le code que vous écrirez implémentera seulement la logique dite *métier*. Autrement dit, votre travail principal consistera à rédiger la *glue* entre les composants logiciels séléctionnés afin de réaliser le projet demandé.

Avant de vous lancer dans la réalisation d'un tel projet, nous vous proposons de prendre le temps d'analyser et de comprendre le fonctionnement de chaque brique logicielle. Dans l'idée, nous parlerons d'**état de l'art** et de **POC** :

- **État de l'art** : Étudier les différentes solutions possibles et choisir le bon composant en fonction des besoins.
- **POC** (*Proof Of Concept*) : Faire un rapide programme de démonstration qui prouve le bon fonctionnement d'un composant ou d'un algorithme.

Votre participation active à la préparation des *Workshops* ainsi que votre attention pendant ces activités vous seront **indispensables** pour la réussite de ce projet.

Le projet

Le but de ce projet est de vous familiariser avec la plateforme logicielle que vous avez choisie (Java, .NET, node.js) à travers la réalisation d'un dashboard.

Pour ce faire, vous devez implémenter une application web dont le fonctionnement ressemble à celui de Netvibes.

Fonctionnalités

L'application proposera les fonctionnalités suivantes :

- L'utilisateur s'enregistre sur l'application afin d'obtenir un compte (cf. *Gestion des utilisateurs*)
- L'utilisateur enregistré s'authentifie sur l'application avant de pouvoir l'utiliser (cf. *Authentification / Identification*)
- L'application propose à l'utilisateur authentifié de s'abonner à des **Services** (cf. *Services*)
- Chaque **Service** propose des **Widgets** (cf. *Widgets*)
- L'utilisateur authentifié compose son **Dashboard** en y insérant des **instances de widget** préalablement configurées (cf. *Dashboard & Instance de widget*)
- Un **Timer** permet de rafraichir les informations affichées par les différentes **instances de widget** présentes sur le **Dashboard** (cf. *Timer*)

Groupe de travail

Le projet est à réaliser en groupe. La validation du module associé prendra en compte non seulement la qualité du travail effectué mais aussi la quantité des fonctionnalités disponibles.

Voici la configuration minimale attendue pour un groupe constitué de X étudiants :

- Soit NBS le nombre de **Services** supportés par votre **application web**
- Soit NBW le nombre total de **Widgets** supportés par l'ensemble des **Services** disponibles

Il faudra respecter les conditions suivantes :

- NBS >= 1 + X
- NBW >= 3 * X

Gestion des utilisateurs

L'application étant centrée sur les informations visualisées par les utilisateurs, elle se doit donc de proposer une gestion de ces derniers.

Pour ce faire, vous devez réaliser un module de gestion des utilisateurs.

L'application web propose aux utilisateurs non authentifiés de s'enregistrer via un formulaire. Inspirez-vous de ce que vous connaissez déjà pour cette étape (ex. renseigner une adresse mail, inscription via un service tiers comme Yammer, Facebook, Twitter, etc.)

Une partie administration serait un plus pour gérer les utilisateurs du site.

Authentification / Identification

L'utilisation de l'application nécessite de connaitre l'utilisateur en question.

Pour ce faire, il faut implémenter au choix :

- Une méthode d'authentification des utilisateurs via un formulaire username/password.
- Une méthode d'identification des utilisateurs via OAuth2 (ex. Yammer/Twitter/Facebook/etc.)

Concernant la méthode d'identification, pensez bien à relier le compte tiers à un utilisateur du système.

Services

Le but de l'application étant d'afficher des informations prevenant de différents **Services** (Intra Epitech, Outlook 365, Yammer, OneDrive, etc.), il faut dans un premier temps proposer à l'utilisateur authentifié de sélectionner les **Services** pour lesquels il a un compte.

Dans cette partie, il faudra donc proposer aux utilisateurs de s'abonner à ces **Services** (ex. à partir de sa page de profil, l'utilisateur renseigner ses identifiants Intra Epitech, l'utilisateur relie son compte Twitter/-Google/etc. via une authentification OAuth2).

Les **Services** disponibles proposés à l'utilisateur correspond à la liste des **Services** gérées par votre application web, c'est-à-dire la liste des **Services** proposant des **Widgets**.

Certains **Services** ne nécessitant pas d'identification, par exemple un **Service Météo**, seront disponibles par défaut pour tout utilisateur authentifié. Les autres devront être cachées jusqu'à ce que l'utilisateur s'y connecte via l'application.

WIDGET

Chaque **Service** proposera des **Widgets**. Des **instances de widgets** pourront être ajoutées au **Dashboard** après avoir été configurées.

Quelques exemples :

- **Service** Météo
  - Affichage de la température pour une ville V
- **Service** Bourse
  - Affichage du taux de change d'un couple de monnaie M1 / M2
  - Affichage de l'évolution du cours d'une action A
- **Service** Cinéma
  - Affichage de la liste des scéances du jour pour le cinéma C
- **Service** RSS
  - Affichage de la liste des N derniers articles pour le flux F
- **Service** Steam
  - Affichage du nombre de joueur pour le jeu J
- **Service** Google Map
  - Affichage du trajet entre le lieu actuel et une adresse A en utilisant le moyen de transport MT
- **Service** YouTube
  - Affichage du nombre de subscribers pour la chaine C
  - Affichage du nombre de vues pour la vidéo V
  - Affichage des N derniers commentaires pour la vidéo V
- **Service** Reddit
  - Affichage des N derniers posts pour le subreddit S

- Mais qu'elle est la différence entre un bon et un mauvais widget ?
- Nan mais bon heu… le mauvais widget, t'arrives, tu l'prends, tu l'mets sur le dashboard !
- Et le bon widget ?
- Nan mais le bon widget, t'arrives, tu l'prends, tu l'mets sur le dashboard, mais heu… C'est un bon widget !

Un **Widget** valide est un **Widget** proposant une configuration lors de son insertion sur le **Dashboard**. Le **Widget** "Affichage des tendances YouTube" ne proposant pas de configuration n'est donc pas **Widget** valide et ne sera donc par comptabilisé.

Dashboard & Instance de widget

La page principale de l'application représente le **Dashboard**. À partir de ce dernier, l'utilisateur authentifié pourra :

- Ajouter une nouvelle **instance de Widget** en proposant à l'utilisateur de :
  - Sélectionner un **Widget**
  - Configurer le **Widget** préalablement sélectionné
  - Renseigner la fréquence de rafraichissement du **Widget** préalablement configuré
  - Ajouter le **Widget** sur le **Dashboard**
- Reconfigurer une **instance de Widget** présentes sur le **Dashboard**
- Déplacer une **instance de Widget** présentes sur le **Dashboard**
- Supprimer une **instance de Widget** présentes sur le **Dashboard**

Un **Widget** est valide s'il permet de proposer à un instant T, deux **instances** de celui proposant des informations différentes.

Timer

Cet élément essentiel de l'application a pour objectif de rafraichir les informations présentées par les **instances de Widget** présentes sur le **Dashboard**.

Pour ce faire, pour chaque **instances de Widget** présentes sur le **Dashboard** va communiquer au **Timer** sa fréquence de rafraichissement afin que celui-ci déclanche les mises-à-jours nécessaires au bon moment.

Construction du Projet

Une grande liberté étant donnée dans le cadre de ce projet, vous serez donc libre d'utiliser les technologies de votre choix, tout en respectant les consignes suivantes:

- pour Java: utilisation de `gradle` pour la gestion des dépendances et la compilation
- pour .NET: utilisation de `nuget` et `dotnet` pour les dépendances et la compilation
- pour NodeJS: utilisation de `npm` pour les dépandances

Afin de pouvoir homogénéiser la construction d'un tel projet vous devrez également vous baser sur l'utilisation de Docker Compose

Merci de lire attentivement les points suivants

+ DOCKER-COMPOSE BUILD

Vous devrez rendre, à la racine de votre projet, un fichier docker-compose.yml qui se chargera de décrire les différents services docker utilisés.

Ce fichier devra comporter au minimum le service Docker **server** servant à lancer l'application sur le port 8080.

+ DOCKER-COMPOSE UP

La validation de l'intégrité de votre application s'effectura lors du lancement de la commande *docker-compose up*.

Les points suivants devront être respectés :

- Le service **server** s'executera en exposant le port 8080
- Le service **server** répondra à la requête http://localhost:8080/about.json (Cf. *Fichier about.json*)

Fichier about.json

Le **serveur** devra répondre à l'appel http://localhost:8080/about.json.

```
{
    "client": {
        "host": "10.101.53.35"
    },
    "server": {
        "current_time": 1531680780,
        "services": [{
            "name": "weather",
            "widgets": [{
                "name": "city_temperature",
                "description": "Affichage de la temperature pour une ville",
                "params": [{
                    "name": "city",
                    "type": "string"
                }]
            }]
        }, {
            "name": "rss",
            "widgets": [{
                "name": "article_list",
                "description": "Affichage de la liste des derniers articles",
                "params": [{
                    "name": "link",
                    "type": "string"
                }{
                    "name": "number",
                    "type": "integer"
                }]
            }]
        }]
    }
}
```

Les propriétés suivantes sont obligatoires :

- **client.host** indique l'adresse IP du client effectuant la requête HTTP
- **server.current_time** indique l'heure du serveur dans le format Epoch Unix Time Stamp
- **server.services** indique la liste des **Services** supportés par le serveur
- **server.services[].name** indique le nom du **Service**
- **server.services[].widgets** indique la liste des **Widgets** supportées par ce **Service**
- **server.services[].widgets[].name** indique l'identifiant de ce **Widget**
- **server.services[].widgets[].params** indique la la liste des paramètres pour configurer ce **Widget**
- **server.services[].widgets[].params[].name** indique l'identifiant de ce paramètre
- **server.services[].widgets[].params[].type** indique le type de ce paramètre. Les types supportés sont: integer, string

Documentation

Il est important de prendre le temps de définir une architecture simple, sans duplication de code et qui soit évolutive.

Il vous est demandé de nous fournir une documentation de conception claire et simple de votre projet. Vous pouvez intégrer quelques schémas de conception : diagramme de classe, diagramme de séquence …

Le but est d'avoir un document qui nous sert de support de travail pour comprendre facilement le projet afin de faciliter la communication dans les équipes de travail et de faciliter la montée en compétence des nouveaux développeurs.

Ainsi, il n'y a pas besoin de faire les diagrammes de classe ou de séquence de l'intégralité du projet, mais plutôt de bien choisir les parties importantes à comprendre qui nécessitent d'être documentées.

Vous devrez impérativement fournir un fichier README.md décrivant la liste des **Services** et des **Widgets** disponibles sur votre application. Ce fichier devra être écrit en utilisant le format markdown.

Bonus

Les widgets exploitant les outils de l'école (Intra Epitech, Yammer, Office 365, OneDrive, etc.) seront mieux valorisées pour ce projet.

En ce qui concerne la partie déploiement, une attention particulière sera portée sur l'utilisation de services de cloud computing. Voici une liste, non hexaustive, de sites à considérer pour le déploiement :

- Heroku
- Microsoft Azure
- Amazon AWS
- Google Cloud Platform

Have Fun!