Programme D-CLIC
Module : Node JS
Projet Fil Rouge - JARVIS partie 3

Durée de l’exercice : Tout au long du module

Du 06/07/2022 au 29/07/2022
Heure limite du dépôt : 29/07/2022 à minuit

I - Introduction

Vous continuez votre progression dans le vaste et complexe univers du backend et des bases de données. Vous allez découvrir la technologie back NodeJS, ainsi que son utilisation et l'interaction avec une base de données et une application existante. Cette technologie permet de développer des applications en utilisant JavaScript.

Ce n'est pas un outil qui vous permettra de mettre en place une application web rapidement avec peu de code. C'est un outil plus bas niveau qui vous permettra de communiquer avec le système à travers différentes librairies C++ et avec un langage familier. C'est donc un outil que l'on va sélectionner si on a besoin de gérer un grand nombre de demandes sur un seul thread en évitant les lenteurs dû à la nature synchrone d'autres langages.

Pour rappel, le projet JARVIS est un projet "fil rouge" que vous avez développé tout au long de votre parcours en Backend. Il est décomposé en plusieurs parties : UML/Merise, SQL/MySQL, Node JS. Le but de ce projet sera de créer un assistant pour la gestion domotique des appareils connectés d'une maison.

Il existe des étapes à respecter pour réaliser correctement la conception de ce type de projet.

II - Objectif

Le projet JARVIS partie 3 a pour objectif de créer et manipuler l'API qui permettra de connecter la base de données à l'application, validé en amont. Pour ce faire, voici les grandes étapes à réaliser :

- Création de l'API
    - Récupérer la base de donnée SQL
    - Récupérer la doc de l'API
    - Créer toutes les routes de l'API
    - Hébergement de l'API sur un serveur
- Connecter l'API à l'application mobile

On se rapproche donc de la fin du projet Jarvis, en rendant celle-ci fonctionnelle avec une API et application mobile. Toutefois, elle pourra être améliorée pour prendre en compte davantage de fonctionnalités et d'options qui ne seront pas développées, mais bel et bien présentes dans le cahier des charges initial du projet.

III - Énoncé

III.1 - Avant-propos :

Internet a évolué ces dernières décennies d'un réseau de calculateurs à un réseau d'ordinateurs personnels, puis vers un réseau qui intègre tout dispositif communiquant : les réseaux de capteurs, les actionneurs, les réseaux véhiculaires, etc. Cette évolution a donné naissance à l'Internet of Things (IoT).

Aujourd'hui, cette technologie est omniprésente dans notre environnement, que ce soit au travail, dans les espaces publics ou à la maison. En effet, les réseaux locaux, les interphones, les systèmes d'alarme et Internet sont devenus une solution incontournable pour améliorer la qualité et le confort de notre quotidien.

Nous pouvons dire que le grand nombre d'appareils électroniques sur le marché, avec leur prix décroissant et l'avancement dans la technologie de télécommunication, a rendu accessible l'utilisation de cette technologie. Nous la retrouvons partout et notamment dans nos maisons, d'où la naissance des systèmes domotique.

La domotique définit l'ensemble des techniques qui permettent de centraliser le contrôle de différents appareils de la maison ou d'un bâtiment. Elle vient donc s'installer sur nos ordinateurs, téléphones et/ou tablettes, pour piloter tous nos objets connectés, avec des solutions appropriées et peu coûteuses.

III.2 - Mission JARVIS et Cahier des charges

Vous êtes un développeur / une développeuse freelance et vous êtes sollicités pour développer ce projet ambitieux nommé JARVIS.

Le client souhaite créer une application mobile qui automatise la gestion des objets connectés ainsi que les autres actions relatives aux domaines de la domotique (sécurité, confort et gestion d'énergie). De plus, une assistance vocale spécialisée, conçue pour faciliter et améliorer le cadre de vie d'un foyer (couple/famille), sera présente dans l'application. Pour finir, toute la famille pourra connecter ses agendas et autres informations personnelles pour cohabiter plus facilement et organiser les activités de la famille. Elle pourra également échanger via une messagerie chiffrée sur les événements familiaux.

Votre mission consistera principalement à créer l'API, qui sera en lien avec l'application mobile, pour gérer les données d'utilisation des objets connectés de la maison.

L’application mobile sera quant à elle développée par une équipe de développeurs, en parallèle de votre travail. Une fois l’API terminée, vous devrez réaliser la connexion API-Application.

Ce projet JARVIS va se réaliser en trois phases bien distinctes :

1. UML/MERISE - Interprétation des diagrammes | **OK**
    a. Diagramme de cas d’utilisation
    b. Modèle Conceptuel de Données (MCD)
    c. Diagramme de classe
2. SQL/MySQL | **OK**
    a. Diagramme EER
    b. Les opérations SQL
3. Node JS | **En cours**
    a. Création d’une API
    b. Hébergement de l’API sur un serveur
    c. Connexion de l’API à l’application mobile

Ce projet tend à s’étendre de manière globale afin de faciliter la vie quotidienne à chacun, tout en jouant avec les nouveaux appareils qui apparaissent chaque jour, comme les Philips, Hue, Google Home ou bien Alexa. La solution doit prendre en compte uniquement les objets connectés au Wi-Fi, principalement les lumières connectées pour commencer.

L’utilisateur devra être équipé des appareils suivants :

- Ordinateur, Tablette ou Smartphone : permettant la création et gestion de son compte et de ses appareils sur le site web.
- Smartphone ou Tablette : permettant la gestion du comportement des appareils, partage de données avec les utilisateurs du même compte sur l’application mobile.
- Matériels domotiques : Google Home, objets connectés de la marque Philips, Samsung ou Konyks.
- Compte sur l’application : Pour utiliser l’application, n’importe quel utilisateur devra créer un compte au préalable sur l’application, quel que soit son rôle. Il est donc logique que l’utilisateur télécharge en amont l’application.

La totalité de vos fichiers source, sauf tous les fichiers inutiles (binary, temp files, obj fichiers,...), doivent être inclus dans votre livraison, dépôt Github, en respectant impérativement la nomenclature suivante :

SAYNA-NODEJS-JARVIS3-072022

⚠ Si la nomenclature n'est pas respectée, le projet ne sera pas pris en compte lors de la correction et l'évaluation ⚠

⚠ Pensez à mettre votre dépôt en "Public". Le projet ne sera pas corrigé si le dépôt se trouve en "Privé" ⚠

Consignes pour le rendu du projet :

- ➢ Nommez votre projet en respectant la nomenclature.
- ➢ Déposez votre projet sur Github dès le début et actualisez-le au fur et à mesure de votre avancée.
- ➢ Lorsque les rendus ne sont pas du code, il est tout à fait possible de mettre des documents sous format PDF, PNG, JPEG, etc.
- ➢ Vous scinderez bien les différentes parties du projet.

Notes pour les apprenants :

- Lisez l'intégralité de l'énoncé dès le début pour bien commencer.
- Prenez le temps de coder en commentant votre code dès qu'il est nécessaire. Lorsqu'il n'y a pas de code, vous pouvez tout de même commenter votre travail en créant un "Readme" dans le dépôt. Ce fichier "Readme", proposé par Github lors de la création d'un dépôt, peut reprendre les éléments du projet ainsi que vos observations et vos remarques liées à la réalisation des tâches à effectuer (prévue dans l'énoncé)
  Il faut que le correcteur puisse comprendre ce que vous avez fait !

IV - Ressources utiles

Doc de l'API :
Si besoin, voici un lien pour télécharger le logiciel Xmind. Sinon, référez-vous aux images capturées dans les assets de l'énoncé.

- Structure Database.xmind
- Tableau API.xmind

Outils nécessaires :

- Nom du serverless : à trouver lors de la veille.
- Logiciels et autres outils nécessaires (à valider avec l'intervenant) :
    - Github ou FTP
    - VS Code & ses extensions pour NodeJS (à voir avec l'intervenant)

V - Tâches à réaliser

Pour vous guider dans la réalisation de ce projet partie 3, nous allons vous détailler les différentes tâches à réaliser. Suivez correctement les étapes.

V.1 - Création de l'API :

Une API, pour Application Programming Interface, est un programme permettant à deux applications distinctes de communiquer entre elles et d'échanger des données. Cela évite notamment de recréer et redévelopper entièrement une application pour y ajouter ses informations. Par exemple, elle est là pour faire le lien entre des données déjà existantes et un programme indépendant.

C'est le cas pour l'application mobile JARVIS et la base de données développées en SQL. Elles doivent se connecter l'une à l'autre et être capable de se comprendre pour interagir ensemble.

DESCRIPTION des tâches à réaliser :

1. Récupérer la doc de l'API
2. Coder l'API sur VS Code au vu du cahier des charges et des conseils de l'intervenant webinar.
3. Il vous faudra également créer la page index.html qui permettra d'afficher le tableau de l'API, lorsque quelqu'un se connectera à celle-ci. Cette page affichera les informations suivantes :
    - Quel est l'objectif de l'API
    - Quelles sont les routes de l'API

4. Publier l’API sur Github ou autre outil en fonction du serverless choisi (cf plus bas). Pour réaliser cette étape entièrement, vous avez besoin de réaliser la partie “Configuration de l’API” en amont. Pensez à finir cette étape une fois que le serverless est choisi et configuré.

Consignes supplémentaires à intégrer :
En fonction du serverless que vous allez utiliser, vérifier s'il accepte les deux méthodes suivantes de connexion API à une Application Mobile :

- Connexion via Github pour disposer du code en ligne
- Transfert de code via FTP (File Transfer Protocole) vers le serverless

V.2 - Configuration du serverless :

L'informatique serverless est un modèle de développement cloud-native qui permet aux développeurs de créer et d'exécuter des applications sans avoir à gérer des serveurs.

Ce modèle nécessite quand même des serveurs, mais leur gestion est dissociée du développement des applications. Un fournisseur de cloud se charge du travail de routine : il approvisionne l'infrastructure de serveurs, assure son bon fonctionnement et la met à l'échelle. Les développeurs n'ont alors plus qu'à mettre en paquet leur code dans des conteneurs pour déployer les applications.

Une fois que les applications sans serveur sont déployées, elles répondent à la demande et se mettent à l'échelle automatiquement en cas de besoin. En général, les offres sans serveur des fournisseurs de cloud public sont facturées à la demande, sur la base d'un modèle d'exécution orienté événements. Par conséquent, lorsqu'une fonction sans serveur est inactive, elle ne coûte rien.

DESCRIPTION des tâches à réaliser :

1. Choisir le serverless que l’on souhaite utiliser, en réalisant une veille technologique.
2. Relancer le npm instal.
3. Vérifier que le fichier HTML est bien intégré dans le serverless.

V.3 - Connexion de l’API à l’application mobile :

l’API est l’intermédiaire pour les logiciels et les applications. Il s’agit d’une interface de programmation d’applications qui aide les développeurs à connecter le frontend avec son backend. Il s’agit d’un élément crucial de toute application mobile.

C’est parce que la programmation d’une application sans API prendra beaucoup de temps et de codage. D’autre part, lorsque l’API est utilisée pour une fonctionnalité, le temps de développement est économisé. Dans ce cas, les développeurs ne doivent travailler sur l’interface utilisateur que s’ils utilisent un backend géré ou un backend mobile en tant que service. Pour aller plus loin.

DESCRIPTION des tâches à réaliser :

1. Hébergement de l’API sur le serverless
2. Liaison du serverless avec l’application mobile, en précisant l’URL dans l’application mobile (application installée sur votre appareil mobile)