TP Internet of Things

LOGEMENT ECO-RESPONSABLE

Ces dernières années, l'éco-responsabilité a pris des proportions importantes au niveau des pouvoirs publiques (norme HQE, crédits d'impôts, recyclage etc.). Mais l'éco-responsabilité, chacun peut la vivre chez soi au quotidien en triant ses déchets, en économisant l'eau, en achetant des produits éco-responsables etc.
Le but de ce projet consiste à proposer à des particuliers une application internet clé en main permettant d'améliorer son impact sur l'environnement. Les données de cette application proviendront de l'utilisateur et d'un ensemble de capteurs. Elle permettra de visualiser sa consommation (électrique, d'eau etc.), de voir les économies réalisées, de visualiser l'état des différents capteurs etc.
On peut envisager plusieurs type de capteurs (température, luminosité, solaire, consommation électrique, niveau d'eau, poids, etc.) et d'actionneurs (volets, prises électriques, lumières, etc.), il n'y a aucune limitation à votre imagination.
Le projet se déroulera en 4 séances de TP :

1. Construction de la base de données
2. Conception de l'application client/serveur permettant le traitement des données
3. Intégration d'APIs REST pour obtenir des données externes (météo, factures etc.)
4. Conception du site web correspondant

A l'issue de chaque séance de TP, vous soumettrez **OBLIGATOIREMENT** votre code sur Moodle. Vous aurez la possibilité d'y retravailler (et de le soumettre à nouveau) jusqu'à la séance suivante.
Le travail étant personnel, il y aura un rendu par personne, et tout code identique avec d'autres sera sévèrement sanctionné.

1 Base de données

Nous allons intégrer toutes les données de l'application dans une base de données relationnelle. Le logiciel de gestion de base de données (SGBD) utilisé sera **sqlite3** qui s'utilise avec des requêtes **SQL** et peut être intégré directement dans un programme C.
En ligne de commande **sqlite3** s'utilise avec la commande `$ sqlite3 database.db`. Cela ouvre un *shell* **sqlite3** dans lequel on peut passer des commandes SQL sur la base de données `database.db`.
Pour chaque programme rendu, le critère d'évaluation sera la conformité avec ce qui est demandé.
Vous trouverez de nombreuses ressources sur **SQL** sur le site w3schools.

1.1 Spécifications de la base de données

Le modèle relationnel de la base de donnée doit respecter les spécifications suivantes :

- un logement est identifié par une adresse, un numéro de téléphone, une adresse IP et une date d'insertion dans la base,
- un logement peut avoir 1 à plusieurs pièces que l'on peut nommer et localiser par les coordonnées d'une matrice à 3 dimensions,
- chaque pièce peut avoir 0 à plusieurs capteurs/actionneurs,
- chaque capteur/actionneur a un type (température, électricité etc.), une référence commerciale, une référence à la pièce où il se situe, un port de communication avec le serveur et une date d'insertion dans la base,
- on pourra prévoir une table dédiée pour le type de capteur/actionneur, ce qui permettrait de lui associer une unité de mesure, une plage de précision ou toute autre information utile commune à tous les capteurs/actionneurs de ce type,
- chaque mesure est associée à un capteur/actionneur, a une valeur et une date d'insertion dans la base,
- des factures alimentent régulièrement la base, elles ont un type (eau, électricité, déchets, etc.), une date, un montant, une valeur consommée et sont rattachées à un logement.

Pour la date d'insertion dans la base, utiliser la construction
`TIMESTAMP DEFAULT CURRENT_TIMESTAMP.`

Question 1
Avant d'entreprendre tout codage informatique, dessiner le modèle relationnel de votre base de données et faites le valider par votre encadrant. Vous avez le droit d'ajouter toute table ou champ qui vous paraîtrait utile pour le fonctionnement de votre système.

Question 2
Dans un fichier nommé `logement.sql`, ajouter tous les ordres **SQL** permettant de détruire toutes les tables éventuellement présentes dans votre base. Il est fortement conseillé d'insérer des commentaires dans ce fichier. Ceux-ci commencent avec la séquence `--`. Par exemple :
`-- Ceci est un commentaire`
La commande **sqlite3** pour lire un fichier est `$ .read fichier.sql`

Question 3
Dans le même fichier `logement.sql`, ajouter tous les ordres **SQL** permettant de créer toutes les tables de votre base. Dans les commentaires associés, bien préciser le rôle de chaque table et champ.

Question 4
Dans le même fichier `logement.sql`, ajouter tous les ordres **SQL** permettant de créer un logement avec 4 pièces.

Question 5
Dans le même fichier `logement.sql`, ajouter tous les ordres **SQL** permettant de créer au moins 4 types de capteurs/actionneurs.

Question 6
Dans le même fichier `logement.sql`, ajouter tous les ordres **SQL** permettant de créer au moins 2 capteurs/actionneurs.

Question 7
Dans le même fichier `logement.sql`, ajouter tous les ordres **SQL** permettant de créer au moins 2 mesures par capteur/actionneur.

Question 8

Dans le même fichier `logement.sql`, ajouter tous les ordres **SQL** permettant de créer au moins 4 factures.

1.2 Remplissage de la base de données

Nous allons maintenant écrire un programme en Python permettant de compléter les données de la base par des requêtes **SQL**. Modifier le fichier `remplissage.py` disponible sur Moodle afin de rajouter plusieurs mesures et factures dans la base de données. Vous vous assurerez de la cohérence des dates et valeurs que vous générerez.

2 Serveur RESTful

Le but de ce TP est d'écrire le code correspondant au serveur RESTful. Pour chaque exercice, il faudra rendre le jeu de données permettant d'alimenter vos programmes. Le critère d'évaluation sera la conformité du code rendu avec ce qui est demandé.

2.1 Exercice 1 : remplissage de la base de données

Reprendre la partie 1.2 en utilisant désormais des requêtes GET et POST pour consulter/remplir la base de données et écrire le serveur REST correspondant en Python.

2.2 Exercice 2 : serveur web

Modifier votre serveur pour que celui-ci puisse répondre à une requête GET permettant de créer a à la volée une page HTML affichant un camembert à partir des valeurs des factures à l'aide de Google Charts.

2.3 Exercice 3 : météo

Certaines mesures éco-responsables peuvent dépendre de la météo (arroser le jardin, allumer le panneau solaire etc.). Intégrer à votre serveur, une requête à une API REST de météo et afficher les prévisions à 5 jours.

2.4 Exercice 4 : intégration

Utiliser maintenant vos capteurs et les ESP associés pour alimenter la base de données et faire des actions : par exemple allumer une led si la température extérieure (obtenue par l'exercice précédent) est supérieure à une température donnée.

3 HTML/CSS/Javascript

Ce TP a pour but de concevoir l'application internet (site web) permettant de gérer l'ensemble des fonctionnalités du logement éco-responsable. Ce site devra être totalement *responsive* et utiliser des fonctionnalités de HTML5 et JavaScript. L'utilisation de Bootstrap facilite beaucoup le développement de site web *responsive*. Enormément de ressources sont à disposition sur le site w3schools.
Le site sera composé d'au moins une page d'accueil et de 4 autres pages présentant :

1. la consommation (électricité, eau, déchets etc.) du logement sous forme de graphiques selon différentes échelles de temps,
2. l'état des différents capteurs/actionneurs,

3. les économies réalisées sous forme de comparatifs (graphiques) selon différentes échelles de temps,
4. la configuration : paramètres utilisateur, ajout de capteurs/actionneurs, etc.

La navigation dans et entre les différentes pages devra être particulièrement soignée.
Il faudra prévoir au moins 5 capteurs/actionneurs, leur état, configuration et les informations correspondantes en terme de consommation et économies réalisées.
Les critères d'évaluation seront :

— La conformité du site (pages, capteurs),
— L'interaction avec le serveur et la base de donnée,
— L'apparence du site,
— La navigabilité,
— Le côté *responsive*,
— L'utilisation de fonctionnalités HTML5,
— L'utilisation de JavaScript,
— L'originalité.