SAE-32 - Développer une application communicante

Spécificités

Une application client/serveur de suivi de livraison de colis.

On veut pouvoir localiser les colis à tout instant (position GPS: latitude,longitude) et suivre l'historique de leurs déplacements. A chaque étape de la livraison d'un colis, on va donc enregistrer la date et la nouvelle position du colis.

L'application client/serveur sera écrite en python fonctionnera en TCP sur le port 3456. En option, on pourra chiffrer les échanges de données (TLS).

- Fonctionnalités obligatoires:
    - identification à partir des utilisateurs d'une base de données.
    - notions de droits pour les utilisateurs (voir ci-dessous).
    - stockage de tous les évènements (authentification, commandes, erreurs, ...) dans un fichier de traces horodaté.
- Réaliser enfin une interface graphique et/ou WEB et/ou Mobile de suivi de colis ainsi que la partie gestion

▶

1. Plus spécifique
    - La partie web devant permettre à :
        - un expéditeur de "créer" le colis dans l'application en indiquant le transporteur et le destinataire.
        - un transporteur :
            - de préparer une livraison avec plusieurs colis dans un véhicule donné (liste ordonnée).
            - de modifier la position GPS d'un/plusieurs colis (par exemple: tous ceux restant dans un véhicule de transport).
            - d'indiquer la livraison d'un colis à un destinataire.
        - un destinataire de valider la réception d'un colis.

    Chaque colis est identifié et possède un poids, 3 dimensions (hauteur,largeur,longueur) et un état:

    - EMBALLE: Date d'emballage (expéditeur)
    - ARRIVE: Date de arrivée au dépot (transporteur)
    - DEPART: Date de départ du dépot + identifiant livraison (transporteur)
    - LIVRE: Date de livraison (transporteur)
    - RECU: Date de reception (destinataire)

    De plus on connaît pour chaque colis : `l'expéditeur`, `le transporteur` et `le destinataire` du colis (`nom`,`adresse`,`codePostal`,`ville`,`mail`,`tel`).

Un expéditeur n'aura pas les mêmes "commandes" qu'un transporteur ou qu'un destinataire :

- Expéditeur:
  - listDest, showDest id, createDest, modifDest[^1] id, deleteDest[^2] id
  - listTransp, showTransp id, createTransp, modifTransp[^1] id, deleteTransp[^2] id
  - listColis, showColis id, createColis, modifColis[^1] id, deleteColis[^2] id, gpsColis[^1] id
- Transporteur:
  - listVehic, showVehic id, createVehic, modifVehic[^1] id, deleteVehic[^2] id, gpsVehic[^1] id
  - listLivr, showLivr id, createLivr, deleteLivr[^2] id
  - listColis, showColis id, livreColis[^4] id
    - addColisLivr[^1] id, livrId, insertColisLivr[^1] id, rank, livrId, delColisLivr[^1] id, livrId
- Destinataire:
  - showColis[^3] id, recuColis[^3] id
- Administrateur:
  - possède tous les droits
  - listExp, showExp id, createExp, modifExp[^1] id, deleteExp[^2] id

[^1]: si il en est le créateur, [^2]: si il n'est pas utilisé, [^3]: si il en est le destinataire, [^4]: si il en est le transporteur

---

Exemple de valeurs

```
Aliexpress - c512,1.2 Kg,h8,l12,L21 - John DOE

ManoMano - c514,0.7 Kg,h4,l21,L30 - Bob MARTIN

Amazon - c513,1.7 Kg,h18,l20,L40 - John DOE

Ebay - c515,1.1 Kg,h12,l20,L35 - Alice SMITH

       c517,0.2 Kg,h1,l21,L30 - Clark BLACK

cdiscount - c516,0.8 Kg,h6,l12,L15 - Dilan WHITE

            c518,0.9 Kg,h7,l12,L15 - Elian GRAY

DHL Vehic4-Renault Trafic ; Livr18:c514,c517,c516

UPS Vehic5-Peugeot Boxer ; Livr19:c512,c513,c515

FEDEX Vehic6-Citroen Jumper

17-04-2023 09:32:15,c512,43.591476,3.892096

18-04-2023 10:44:18,c515,43.591587,3.891715

18-04-2023 10:52:04,c513,43.591587,3.891715

18-04-2023 11:32:04,Vehic5,43.591587,3.891715
```

```
18-04-2023 11:33:04,Vehic5,43.591154,3.883240

18-04-2023 11:34:04,Vehic5,43.597277,3.875687

18-04-2023 11:35:04,Vehic5,43.604238,3.867705

18-04-2023 11:35:48,Vehic5,43.607345,3.868199+c512-LIVRE+c513-LIVRE
```

Comparaisons des technologie

Pour la base de données

Pour la base de données, nous pouvons utiliser différentes sortes de Base de donnée (BDD):

- BDD Relationelle:

  Elles sont plus communément appelées SQL-like puisque elles adoptent le schéma relationelle et le plus communément connu par SQL.

  On retrouve les BDD suivantes:

  - MySQL
  - SQLite
  - MariaDB
  - PostgreSQL

  > **Note:**
  > Ces BDD ont bien souvent un shell SQL ou une implémentation de le leur langage en format SQL-like.

  Pour les modèles relationnels, nous avons leur force dans les cas où nous travaillons avec un système de données qui est centralisé sur différentes `classes` où nous avons un échange de clé et d'informations permettant de faire le lien d'un élément à un autre.

- BDD Non Relationelle

  Elles sont plus communément appellées no-SQL puisque elles ont choisi de ne pas implémenter de modèle relationnel.
  Elles sont donc le contraire des SQL-like.

  On retrouve les BDD suivantes:

  - MongoDB
  - InfluxDB

  Leur force est que nous avons des BDD capables de gérer des opérations particulières qui serait très complexe voir impossible d'implémenter via des modèles relationnels.

InfluxDB est pensé pour l'IoT et implémente l'horodatage automatique et prend une position sur le temps et s'est spécialisé dans la gestion de petites informations mais en grande quantité.

MongoDB est pensé pour gérer des transactions sous format de documents, notamment en JSON, ce qui simplifie le stockage et la manipulation des données.

Pour le Backend / serveur

Pour le Backend, nous devons utiliser du Python, ainsi donc nous n'avons pas beaucoup d'autres options de choix technologique vu qu'il nous est imposé.

Cependant, il faut noter qu'aussi facile qu'il puisse paraître Python sera incapable de gérer une charge intensive de `compute` ainsi que d'être l'élément centrale d'un système critique nécéssitant une puissance importante de calcul en peu de temps.

Pour le site Web

- HTML

  > **Note:**
  >
  > - Force: facile d'utilisation, pas compliqué sur le code.
  > - Faiblesse : Langage de balisage et non pas un langage de programation.

- ReactJS

  > **Note**
  >
  > - Force: Utilisation dans les applications mobilesou monopages
  > - Faiblesse: React ne s'occupe que de la gestion de l'état et du rendu de cet état dans le DOM (Document Object Model), de sorte que la création d'applications React nécessite généralement l'utilisation de bibliothèques supplémentaires pour le routage, ainsi que certaines fonctionnalités côté client.

- NextJS

  > **Note**
  >
  > - Force: Utilisationde ReactJS en implémentant une parite serveur.
  > - Faiblesse: Même désavantage que ReactJS.

- PHP

  > **Note**
  >
  > - Force: Permet deréaliser des pages webs dynamiques, le logiciel est putôt vieu ce qui des solutions faciles en cas de porbleme.
  > - Faiblesse: Je n'ai pas assez d'expérience sur PHP pour pouvoir réaliser des pages web avec. C'est un logiciel vieu qui apporte beaucoup d'erreurs et peut être bloquant sur les solutions que l'on peut aporter.

- Django

> **Note**
>
> - Force: Facilite le développement d'applications web et est basé sur la réutilisation de code. Il aide dans les petits problèmes de sécurité
> - Faiblesse:

Copyright © 2023 Alexis Opolka & Mathys Domergue - All Rights Reserved