

SAE 3.02

Développer des Applications Communicantes

Note d'introduction

Le participant d'une SAé informatique est amené à installer, adapter et développer des outils logiciels permettant l'échange d'informations au sein d'un environnement technologique en liaison avec le sujet. Il met en place une solution adaptée qui respecte les besoins et les contraintes techniques imposées par le sujet. Il utilise dans ce but différents protocoles de communication, différents moyens d'authentification et de sauvegarde des données sécurisées. Il commente le code produit et génère la documentation technique. Il doit aussi présenter sa solution en argumentant les choix opérés avant la validation.

Description des tâches

L'étudiant ou le groupe d'étudiants présente un cahier des charges à l'issue de l'analyse du sujet choisi. L'étudiant R&T développe une application communicante permettant l'échange et la sauvegarde de données. La réalisation de ce projet contient les étapes suivantes :

- Présenter un cahier des charges en mentionnant les technologies à utiliser et éventuellement leur impact environnemental et économique ;
 - Présentation & contexte ;
 - Description du projet (objectif, publique visé) ;
 - Spécifications fonctionnelles et techniques ;
 - Livrables (planning, annexe)
 - Résumé & difficultés.
- Développer une application client-serveur ;
- Authentifier les utilisateurs ;
- Sauvegarder les données échangées dans une DB au travers une API ;
- Concevoir une interface graphique, une application mobile adaptée ;
- Présenter les résultats lors d'une soutenance (rapport écrit et code source des programmes) ;
- L'objectif de ce genre d'exercice est de mettre le(s) participant(s) dans un environnement proche du réel.
- Les résultats du travail doivent être documentés au fur et à mesure de l'avancement sur le projet avec les commandes et la preuve de l'exécution en ScreenShots (captures d'écran).

Environnement de travail

Une machine virtuelle est disponible en Panama au téléchargement avec un SGBD **MariaDB-Server** préinstallé sur une machine virtuelle sous LINUX-DEBIAN.

Copier la machine virtuelle du répertoire :

```
cp /baru/local/profs/ivmad/src/debian-R309.tar.gz ~/vm/.
```

vers un répertoire de choix dans l'espace utilisateur (ex. vm).

La version alternative est de copier le fichier **debian-mysql-qcow2.tar.gz** plus petit en volume. Il faut prévoir alors la décompression de l'archive dans le répertoire dédié à la MV.

Pour lancer la MV avec KVM dans le mode bridge avec accès USB, il faut changer l'adresse MAC de la carte réseau virtuelle et assigner le port USB. Le mode bridge autorise l'accès réseau à la MV depuis la machine physique par **ssh** et aussi de copier des fichiers dans les deux sens avec **scp**. L'adresse IP de la MV est habituellement **10.10.10.40** ou **41**, à vérifier avec **ip a**. Par ailleurs, **ssh** peut servir de terminal distant pour toutes les opérations sur une BD.

Le mode de fonctionnement pour les projets SAE est de développer un client du côté machine physique qui se connecte à la machine virtuelle pour le traitement des données vers ou depuis la base de données. Ce modèle concerne tous les présentés plus bas. Cela implique aussi un accès à la salle Panama pour les heures consacrées sur ADE au sujet SAE3.02. Une liste d'émargement sera disponible au secrétariat.

Lancer la MV : (commande sur une seule ligne)

```
kvm -m 5G -smp 8 -hda vm/debian-mysql-R308.qcow2 -k fr -net  
nic,macaddr=42:01:02:03:04:05 -net vde,sock=/var/run/vde2/kvmtap0.ctl
```

La connexion à la MV :

Utilisateur : IUTRT

Mot de passe : Promo2022

Avant de procéder, vérifier la liste des dépendances avec la commande :

```
>pip list
```

Si absents : Pour les besoins des sujets l'installation de certains outils et logiciels sera nécessaire.

- Installer pip:
 >sudo apt install python3-pip
 >pip list
 >python3 -m pip install pip --upgrade
- Installer kivy
 >python3 -m pip install kivy
- Installer Plyer
 >python3 -m pip install plyer
- Installer WebSockets & WebSocket-Client
 >python3 -m pip install websockets
 >python3 -m pip install websocket-client
- Installer Buildozer
 >sudo apt update
 - Installer les dépendances (*requirements*)
 >sudo apt install -y git zip unzip openjdk-17-jdk python3-pip autoconf libtool pkg-
 config zlib1g-dev libncurses5-dev libncursesw5-dev libtinfo5 cmake libffi-dev libssl-dev
 >pip3 install --upgrade Cython==0.29.33
 >pip3 install --upgrade buildozer
 - Création du fichier buildozer.spec :
 >buildozer init
 - Lancer le processus de construction. Un fichier main.py obligatoirement présent dans le répertoire.
 >buildozer -v android debug
- Installer mysql.connector :
 >pip3 install mysql-connector-python==8.0.29
- Si message :
 Remote connection to MySQL database hosted on Plesk fails with the error: ERROR 2003
 (HY000): Can't connect to MySQL server on '<Plesk server Ip Address>'
 Solution :
 Connect to the server via SSH. Find the config file where bind-address is set to the 127.0.0.1
 >grep -R 'bind-address' /etc/mysql/* /etc/my.cnf
 >/etc/mysql/mariadb.conf.d/example.cnf:bind-address = 127.0.0.1

Comment or remove the line at the configuration file where the bind-address parameter is set to the localhost address @ (or remove this line completely), it should look like below:

```
>grep bind-address /etc/mysql/mariadb.conf.d/example.cnf
>bind-address = 127.0.0.1
>sudo service mysql restart
```

Sujet 1 - Gestion simplifiée d'un hôtel

Pour la gestion de l'hôtel **R&T**** vous aurez besoin des classes Python "Hotel", "Chambre", "Client", "reservation" avec les éléments :

- Une méthode qui réalise le formulaire de la saisie des données du client (consultez le screenshot). Utilisation du script Kv conseiller. (Kv a la fonction "écran multiple", à étudier) ;
 - Nom ;
 - Prénom ;
 - Adresse Mail ;
 - Confirmez l'adresse Mail ;
 - Une méthode Python vérifie l'exactitude des deux adresses mails au traitement de l'événement associé au champ de saisie. (Exception utilisateur par exemple)
 - Téléphone de contact ;
 - Date d'arrivée : jj/mm/aaaa ;
 - Date de départ : jj/mm/aaaa (champ de saisie) ;
 - Une méthode Python vérifie que la date de départ soit postérieure à la date d'arrivée.
 - Nombre de nuits : (Libellé texte dont la valeur est calculée par une méthode Python qui prend en compte les deux dates et affiche leur différence en nombre de jours) ;
 - Type de chambre : (liste déroulante, choix unique), exemple, single, double, ..., appart, suite) ;
 - Nombre de personnes ;
 - Une méthode Python vérifie la conformité du nombre de personne déclarée par rapport au type de chambre.
- Exemple : chambre single peut avoir 1 personne ; chambre double 2 personnes max, etc.
- Nombre de petit déjeuner ;
 - Confort : TV, minibar, Balcon vue mer, place de parking, etc. (cases à cocher, choix multiple) ;
 - Associer le bouton "Envoyer" à une méthode qui communique avec un serveur de BD pour enregistrer les données de la réservation.



- Pour la gestion de l'hôtel R&T** vous aurez besoin du schéma relationnel de la BD. Elle est constituée de trois tables en relation avec les données de la réservation.
- **Description des tables :**

Chambres : Descriptif des chambres de l'hôtel ;

Clients : Descriptif des clients de l'hôtel ;

Reservations : La réservation des chambres par clients ;

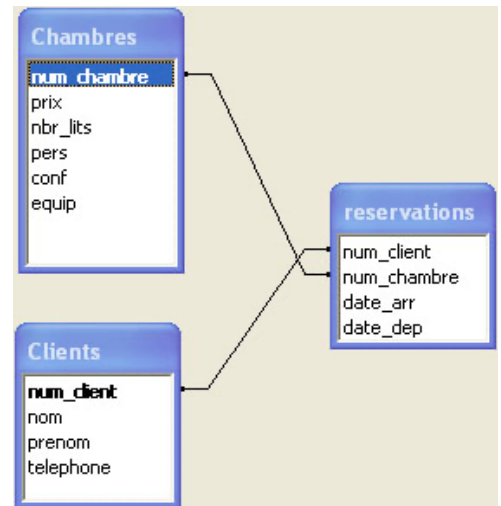
Les modèles entité-association et physique des données :

Chambres(Num_Chambre, Prix, lits, pers, conf, equip)

Clients(Num_Client, Nom, Prenom, Telephone, eMail)

Reservations(Num_Client, Num_Chambre, date_Arr, date_Dep)

Écrire la méthode Python qui crée la BD "Hotel_R&T", et les trois tables avec les attributs et types appropriés.



- L'accusée de réception complète le formulaire avec le prix à payer pour la réservation demandée par l'utilisateur. Ajouter une ligne en calculant le prix à payer et majorer du TVA tourisme. Le prix de la chambre est calculé par une fonction Python à partir d'un prix de base majoré avec un coefficient de type de chambre en ajoutant le nombre des petits déjeuners et le prix de base des options majorés par le coefficient correspondant à chaque option inclus dans la réservation. Exemple :

$\text{Prix} = \text{Nombre_nuits} * (\text{prix} + (\text{prix} * 0.2)) + \text{nb_pd} * 10 + (\text{ops} + (\text{ops} * 0.2));$

La base est choisie en avance et peut être une constante ~80€ par exemple. Le coefficient de majoration peut être 0.1 pour une chambre simple, 0.2 pour une double, etc. Même schéma pour les options.

Mettre le prix calculé dans un "Label" avec un bouton pour obtenir l'accord du client. Après accord, le client est enregistré avec son nom et adresse email dans une quatrième table de la BD. Cette table lui servira d'identification s'il souhaite revoir sa réservation. L'identification d'un client se fait par son nom et son mot de passe demandé au moment de la finalisation de la réservation.

- Les techniques à utiliser sont : Langage Python, interface graphique Kivy avec les scripts Kv (ici, il y a la possibilité d'étudier le script "kivymd", par définition plus adapté. C'est au binôme de prendre la décision); CGI-Python pour le traitement côté serveur; un SGBD MySQL ou MariaDB selon la version installée.
- Une version mobile est souhaitable.

Sujet 2 - Modèle d'enseignement : Gestion simplifiée du dossier étudiant

- Un étudiant évolue dans une formation. Elle est composée de matières à suivre. Son parcours est évalué par des notes à la suite des différentes épreuves. La tâche principale du projet est le développement d'une application Python communicante avec un serveur Web et une BD pour la gestion la situation courante d'un étudiant évoluant au sein d'un groupe d'étudiants.
- Les techniques à utiliser sont : Langage Python, interface graphique Kivy avec les scripts Kv (ici, il y a la possibilité d'étudier le script "kivymd", par définition plus adapté. C'est au binôme de prendre la décision); CGI-Python pour le traitement côté serveur; un SGBD MySQL ou MariaDB selon la version installée.
- L'objectif est de créer et mettre à jour une base de données qui représente le modèle des enseignements proposé au département R&T. Pour se renseigner, il faut réaliser plusieurs requêtes d'interrogation mettant en jeu les opérateurs relationnels : Projection, Sélection, Jointure. Certaines fonctions de calculs verticaux et orientaux seront évoquées.

- Les données pour les étudiants (classes Personne & Etudiant) sont acquises depuis un formulaire dont l'accès est contrôlé par un menu.
- Un étudiant est enregistré dans une BD avec son Prénom, Nom, Année, Adresse Mail, Matière, Moyenne, Photo.
- La BD contient les tables : "etudiants", "matières" (min 12 étudiants, 5 matières)
- Les photos sont stockées dans la table "etudiants" dans un champ de type blob.
- Les requêtes vers la BD sont limitées par une liste déroulante avec les noms des étudiants déjà inscrits dans BD.
- Le schéma relationnel normalisé de la base :

Etudiant (Num, Prenom, Nom, Annee, Email, Groupe)

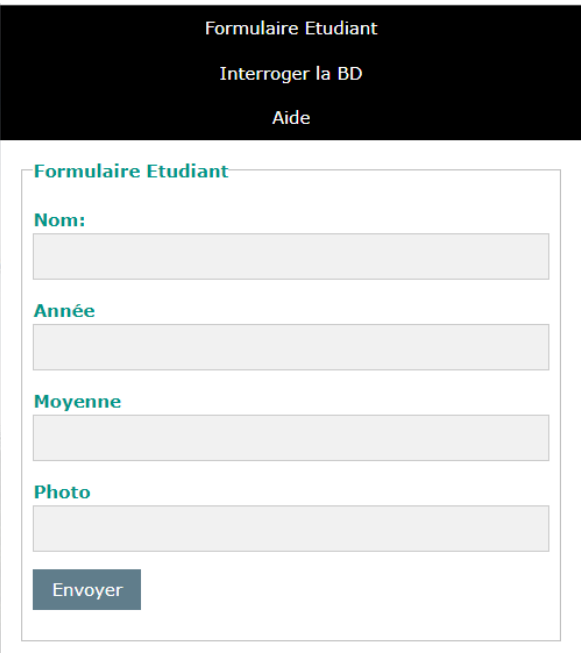
Prof (Num, Nom, Bureau, MatSpec)

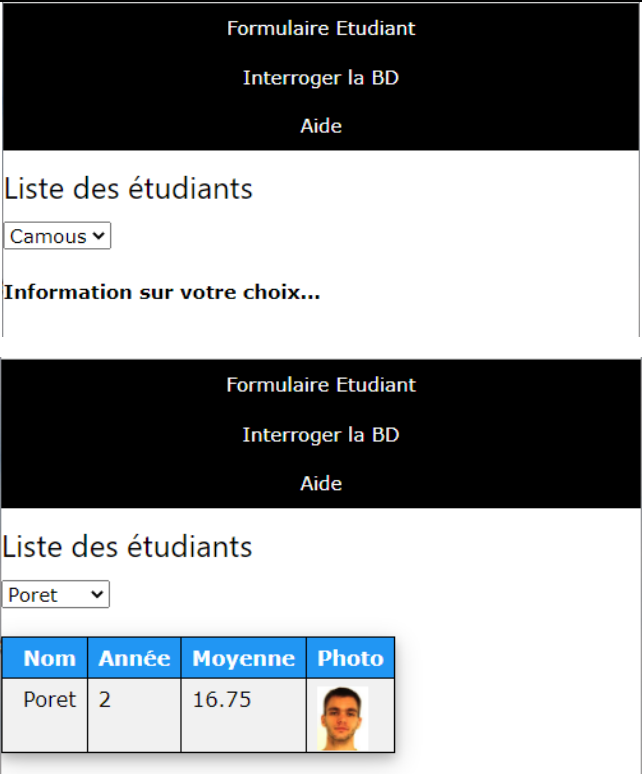
Matiere (Code, Libelle, Discipline, Coeff, ProfResp)


Enseignement (CodeMat, NumProf, NumEt)

Notation (NumEtud, CodeMat, NoteCC)

- Une idée sur la présentation non exhaustive.





Nom	Année	Moyenne	Photo
Poret	2	16.75	

Objectifs du projet :

- Un étudiant a l'obligation d'avoir un compte pour pouvoir accéder aux informations le concernant ;
- Un étudiant doit pouvoir consulter ses notes sur une matière et obtenir la moyenne en accord avec les coefficients accordés à cette matière ;
- Un professeur doit pouvoir consulter la liste des étudiants pour la matière qu'il enseigne, avec la possibilité d'ajouter des notes, des absences pour chacun de ses élèves.
- Une partie administrateur est souhaitable pour activer ou désactiver des comptes d'étudiant, ajouter et supprimer des matières ou des professeurs ainsi qu'associer un professeur aux différentes matières qu'il enseigne.
- Une version mobile est souhaitable.

Sujet 3 - La géolocalisation indoor par QR Code



Il est souvent difficile de se localiser à l'intérieur des bâtiments, le Wi-Fi n'étant pas toujours disponible ou le GPS non capté. Pour les mêmes raisons, il est difficile de faire de la collecte de données sur le patrimoine en intra-bâtiment (équipements...). Le QR Code permet de parer à cela et de se géolocaliser facilement et précisément. Ainsi la position d'un objet est déterminée avec une précision de l'ordre du dixième de mètre, voire du centimètre. Cela permet ainsi d'avoir une information plus précise du patrimoine d'un bâtiment et facilite sa gestion et sa maintenance, tout en permettant de suivre précisément la localisation d'un objet ou d'une personne en mouvement.

Les avantages du QR Code pourraient être particulièrement exploitables en géomarketing. En repérant des personnes utilisant un smartphone et son endroit dans un centre commercial ou dans un magasins, l'utilisateur et consommateur potentiel pourrait recevoir des notifications commerciales en rapport avec le rayon où il se situe. Par ailleurs, cela permet également de faciliter le déplacement du public dans les grands centres commerciaux et dans les infrastructures de transport. Avec cette technologie, l'emplacement de la personne est très précis et l'utilisateur peut donc se localiser à l'intérieur des bâtiments avec son smartphone à l'instar d'un GPS en extérieur. (OutDoor)

Pour définir l'objectif d'un sujet de projet, nous nous proposons de modéliser le cas de géolocalisation à l'intérieur des bâtiments (InDoor) à l'aide de QR Code en opposition des technologies Wi-Fi et Bluetooth.

Le mobile (Smartphone, Tablette) favorise la localisation et la navigation personnelle dans les zones urbaines ou extra-urbaines. Les espaces fermés représentent les zones les plus exigeantes pour la navigation personnelle. Exemple :

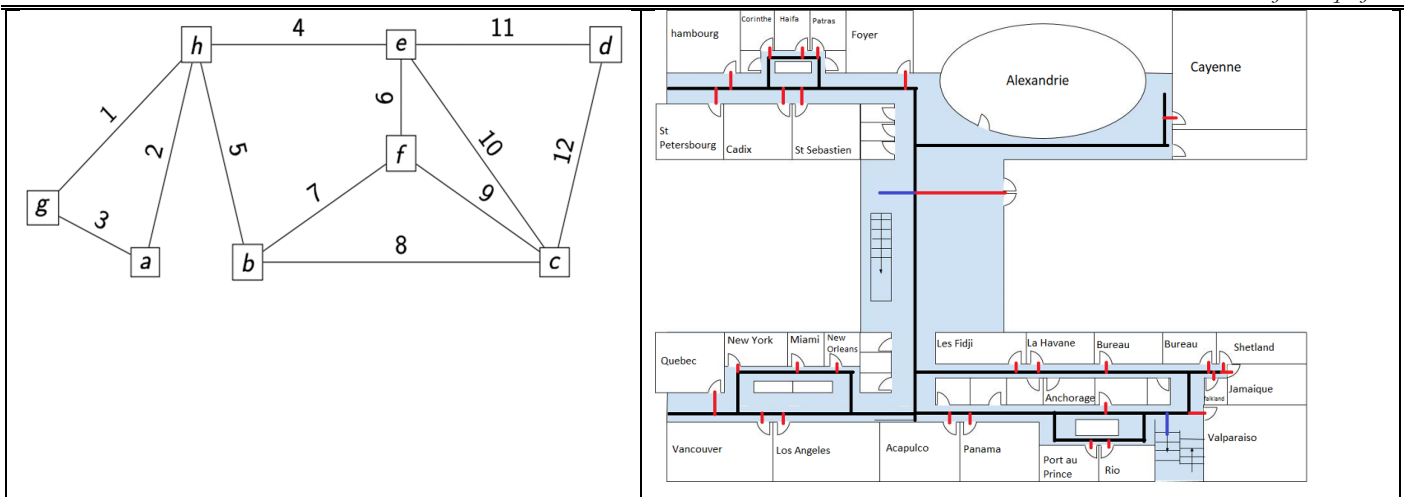
- Comment trouver une salle dans le bâtiment du département R&T sachant qu'elles portent des noms de ports maritimes et non des numéros.

Des problèmes comme la réception des signaux satellitaires rendent le positionnement impossible à l'intérieur d'un bâtiment. Parallèlement, les infrastructures de télécommunications sont en développement croissant, toutefois le positionnement basé sur ces systèmes sans fils (Wi-Fi, GPRS, Bluetooth, NFC), assurent une précision moyenne et n'offrent pas une couverture complète.

On peut pallier ce problème par :

- L'utilisation de tags (QR Code) et assurer ainsi la localisation de la personne sur plan par la lecture d'un code.
- Un service (API), disponible sur serveur, se chargera du calcul de l'itinéraire à suivre en fonction de la localisation courante.
- Pour calculer l'itinéraire, le service se servira d'un graphe étiqueté par les directions pour atteindre le prochain tag (nœud du graphe).
- On procède par une description sémantique de graphe disponible à télécharger depuis un serveur de bases de données dans le format JSON.
- En cas d'incendie ou panne d'ascenseur cette fonctionnalité sera très appréciée à la recherche d'une sortie de secours.

La notion de graphe par l'une de ses représentations classiques : il est composé de points (sommets du graphe) et de courbes reliant certains de ces points (arêtes du graphe). Guider une personne ou un objet connecté dans un bâtiment peut être ramené au "parcours d'un graphe". Au sommet l'application traite les données des capteurs, les connections, la lecture de tag, etc., pour donner le chemin (arête) à prendre. Plusieurs contraintes seront à prendre en compte en temps réel : travaux, encombrement, feu, etc.



L'objectif principal dans ce projet est de proposer une solution pour smartphone dans un cadre de connexion client serveur qui puisse guider un utilisateur depuis l'entrée du bâtiment R&T jusqu'à une salle ou un bureau demandé à son entrée dans le hall. (Voir le schéma).

Dans la théorie des graphes, un chemin dans un graphe est une séquence finie ou infinie d'arêtes qui relie une séquence de sommets. Deux approches algorithmiques sont proposées pour explorer le chemin entre deux sommets d'un graphe :

- Soit le parcours en largeur BFS (Breadth First Search) qui commence par explorer un nœud source, puis ses successeurs et ainsi de suite pour calculer les distances de tous les nœuds depuis un nœud source ;
- Soit le parcours en profondeur DFS (Depth First Search) qui cherche de manière récursive à déterminer s'il existe un chemin d'un sommet à un autre.

Pour tracer tous les chemins possibles dans le bâtiment et ainsi récupérer les données pour les stocker dans des tables d'une BD l'algorithme de Dijkstra peut être appliqué. Pour développer le problème, on peut se fier à un algorithme qui trouve le chemin le plus court entre un sommet source et tous les autres sommets du graphe.

A partir du pseudocode de l'algorithme Dijkstra on peut développer la méthode intégrée dans une application communicante pour le langage Python.

```

function dijkstra(G, S)
  for each vertex V in G
    distance[V] <- infinite
    previous[V] <- NULL
    If V != S, add V to Priority Queue Q
  distance[S] <- 0

  while Q IS NOT EMPTY
    U <- Extract MIN from Q
    for each unvisited neighbour V of U
      tempDistance <- distance[U] +
        edge_weight(U, V)
      if tempDistance < distance[V]
        distance[V] <- tempDistance
        previous[V] <- U
  return distance[], previous[]

```

Dans le code, la distance à parcourir est exprimée dans un tableau de taille V. La valeur V définit le nombre de sommets dans le chemin entre un début A et une destination X.

Les techniques à utiliser sont : Langage Python, interface graphique Kivy avec les scripts Kv (ici, il y a la possibilité d'étudier le script "kivymd", par définition plus adapté. C'est au binôme de prendre la décision); CGI-Python pour le traitement côté serveur; un SGBD MySQL ou MariaDB selon la version installée.

NB: La version mobile est obligatoire.

Sujet 4 - Commande directe de produits avec paiement en ligne.

Un ensemble de produits de communication (Smartphone) est proposé, avec photos et prix de chacun. L'utilisateur s'identifie par son adresse électronique et choisit ceux qu'il souhaite acheter. Une fois qu'il a validé son choix, il voit s'afficher une page qui lui rappelle les termes de sa commande et qui lui indique quelle sera sa facture globale. Il a alors la possibilité de valider ce choix ou d'abandonner. S'il valide, cela envoie automatiquement un message à une adresse fixe (IP du magasin) pour confirmer la commande. Cela lui envoie simultanément personnellement un message de rappel de sa commande au magasin (produits et total de la facture TTC) : Il faut réaliser un formulaire qui sera rempli par l'utilisateur, il contiendra :

- Les choix des produits à acheter, quantité, avec leurs prix associés.
- Un champ de texte pour que l'utilisateur donne son adresse électronique.
- Un champ texte pour son adresse de livraison.
- Un champ de texte où il donnera son nom.
- Un bouton de validation du formulaire.

Créer une base de données pour gérer les commandes des clients, avec les tables de la base de données :

- Table client avec identifiant, nom, adresse mail, mot de passe, adresse de livraison.
- Table articles avec identifiant, nom de l'article, description de l'article, nom fichier de la photo de l'article, quantité, prix unitaire HT.
- Table commande avec des informations sur la date, le client, l'article, la quantité, le prix TTC à facturer.

Prévoir un formulaire d'identification du client. Si le client est déjà dans la BD on lui propose de choisir ses articles, sinon il passe par le formulaire qui récupérera la totalité des informations pour le client.

Note: Kivy et le script Kv autorise l'utilisation de plusieurs écrans : problème à étudier.

Les techniques à utiliser sont : Langage Python, interface graphique Kivy avec les scripts Kv (ici, il y a la possibilité d'étudier le script "kivymd", par définition plus adapté. C'est au binôme de prendre la décision); CGI-Python pour le traitement côté serveur; un SGBD MySQL ou MariaDB selon la version installée.

Pour ses achats le client est muni d'un compte bancaire :

- Créer la classe Python **CompteBancaire** qui représente un compte bancaire, ayant pour attributs :
 - **numeroCompte** ,
 - **nom** (nom du propriétaire du compte) ;
 - **solde** (la somme en dépôt).
- Sécuriser les attributs de la classe **CompteBancaire** en les rendant privés.
- Ajouter les **accesseurs** et les **mutateurs** pour assurer le fonctionnement sécurisé du compte bancaire.
- Créer une méthode **Versement()** qui gère les versements.
- Créer une méthode **Retrait()** qui gère les retraits.
- Créer une méthode **afficher()** permettant d'afficher les détails sur le compte
- Créer l'interface graphique avec les composants graphique dans un conteneur pour la saisie des paramètres : *numeroCompte*, *nom*, *solde*.

Dans son fonctionnement normal, un compte bancaire est censé être en positif. Il arrive toutefois qu'il passe en position débitrice, qu'il soit "à découvert". Les banques peuvent autoriser ce découvert, mais en contrepartie, elles facturent à leurs clients des intérêts débiteurs, appelés **agios**.

- Créer une méthode **Agios()** permettant d'appliquer les agios à un pourcentage de 5 % du solde.
- A chaque retrait (achat du magasin) vérifier la solvabilité du compte bancaire.
- Utiliser les accesseurs (*get_solde()*) et les mutateurs (*set_solde()*) pour la gestion de la solvabilité du compte.
- Créer l'interface graphique qui intègre par héritage la classe **CompteBancaire**