Projet de session

Historique

Date	Description
23 septembre	Version initiale

Informations générale

L'objectif du projet de session est de développer et déployer une application Web responsable du paiement de commandes Internet.

Le projet est divisée en deux étapes pour chacune des remises.

Objectifs

- Se familiariser avec le développement Web
- Développer une API REST
- Utiliser des services Web distants
- S'assurer de la résilience et de la performance d'une application Web
- Déployer une application Web

Équipe

Le travail se fait individuellement ou en équipe de 2.

Échéances

Date	Description	Pondération
4 novembre 21h	Première remise	15%
16 décembre 21h	Remise finale	15%

Le projet de session représente 30% de la note globale du cours.

Évaluation

L'évaluation se fera à distance sans la présence d'un ou des membres de l'équipe.

Le code de l'application doit être hébergé sur Github. Les informations sur l'utilisation de Github sera fournie ultérieurement.

Aux deux dates de la remise, à 21h, le repo sera cloné et c'est ce qui sera utilisé pour l'évaluation.

Langage de programmation

Le langage de programmation recommandé pour le projet de session est Python. La version minimale qui sera utilisé est 3.6. Vous pourrez utiliser la version 3.7 si vous le désirer.

Le cadriciel de développement Web recommandé est Flask 1.1.

Le démonstrateur de laboratoire sera en mesure de vous apporter du support et du soutien technique pour ces deux technologies.

Vous êtes libres d'utiliser un autre langage de programmation et un autre cadriciel de développement Web. **Vous devez obtenir mon authorisation préalable avant d'utiliser un autre langage de programmation ou un autre cadriciel. Aucun support de ma part ou du démonstrateur ne vous sera offert et vous devez fournir les instructions claires pour lancer l'application.**

Le reste des exigences (formats de sérialisation, API) devra être respecté selon les énoncés de problèmes.

Le projet

Le projet consiste à développer une application Web responsable de prendre des commandes Internet. Cette application devra répondre à une API REST, mais devra également être utilisé à travers des pages HTMLs.

Première remise

La première partie se concentre sur le développement et l'utilisation d'une API REST.

L'application web doit répondre aux requêtes Web suivantes :

```
GET /
Content-Type: application/json

200 OK HTTP/1.1
```

```
{
   "products" : [
      {
         "name" : "Hapiness Bottle",
         "id" : 123,
         "in_stock" : true,
         "description" : "An incredible bottle filled with
         ↪  happiness and joy. One of a kind!",
         "price" : 5999,
         "image": "http://caissy.dev/shops/images/image.png"
      },
      {
         "description" : "A limited edition special Flying
         ↪  Squirrel. Hurry up, it won't last long!",
         "image": "http://caissy.dev/shops/images/image.png",
         "in_stock" : false,
         "id" : 456,
         "name" : "Flying Squirrel Limited Edition",
         "price" : 3149
      }
   ]
}
```

La page d'index doit afficher une liste des produits disponible pour passer une commande. Ces produits sont récupérés selon les spécifications de la section **Récupération des produits**.

```
POST /product/<int:product_id>/buy
Content-Type: application/json

302 Found HTTP/1.1
Location: /order/<int:order_id>
```

Cet URL permet d'initialiser le processus d'achat. Si l'item représenté par l'identifiant unique `product_id` est en inventaire (`in_stock == True`), un processus d'achat commence. Si le produit n'est pas en inventaire, l'API doit retourner un erreur avec le code HTTP 422 et le message d'erreur suivant :

```
POST /product/<int:product_id>/buy
Content-Type: application/json

422 Unprocessable Entity HTTP/1.1
```

```
{
   "errors" : {
       "product": {
           "code": "out-of-inventory",
           "name": "Le produit demandé n'est pas en inventaire"
        }
   }
}
```

Une fois le processus d'achat initialisé, on peut récupérer la commande complète à tout moment avec cette requête GET.

```
GET /order/<int:order_id>
Content-Type: application/json

200 OK HTTP/1.1
```

```
{
   "order" : {
      "id" : 6543,
      "total_price" : 9148,
      "email" : null,
      "credit_card": {},
      "shipping_information" : {},
      "paid": false,
      "transaction": {},
      "product" : {
         "id" : 123,
         "quantity" : 1
      },
      "shipping_price" : 1000
   }
}
```

Le champ `shipping_price` représente le prix total pour expédier la commande. Ce champ doit être calculé automatiquement en fonction du poids de l'item :

- Jusqu'à 500 grammes : 5$
- De 500 grammes à 2kg : 10$
- À partir de 2kg (2kg et plus) : 25$

Une commande qui n'existe pas doit retourner un code d'erreur 404.

Par défaut, une commande ne contient aucune information sur le client. On doit fournir le courriel et l'adresse d'expédition du client.

```
POST /order/<int:order_id>
Content-Type: application/json
```

```
{
   "order" : {
      "email" : "caissy.jean-philippe@uqam.ca",
      "shipping_information" : {
         "country" : "Canada",
         "address" : "201, rue Président-Kennedy",
         "postal_code" : "H2X 3Y7",
         "city" : "Montréal",
         "province" : "QC"
      },
   }
}
```

```
200 OK
Content-Type: application/json
```

```
{
   "order" : {
      "shipping_information" : {
         "country" : "Canada",
         "address" : "201, rue Président-Kennedy",
         "postal_code" : "H2X 3Y7",
         "city" : "Montréal",
         "province" : "QC"
      },
      "email" : "caissy.jean-philippe@uqam.ca",
      "total_price" : 9148,
      "paid": false,
      "product" : {
         "id" : 123,
         "quantity" : 1
      },
      "shipping_price" : 1000,
      "id" : 6543
   }
}
```

Tous les champs sont obligatoire (courriel, pays, adresse, code postal, ville et province). S'il manque un champ, l'API doit retourner un erreur en conséquence.

```
POST /order/<int:order_id>
Content-Type: application/json
```

```
{
   "order" : {
      "shipping_information" : {
         "country" : "Canada",
         "province" : "QC"
      },
   }
}
```

```
422 Unprocessable Entity HTTP/1.1
Content-Type: application/json
```

```
{
   "errors" : {
       "order": {
           "code": "missing-fields",
           "name": "Il manque un ou plusieurs champs qui sont
           ↪  obligatoire",
       }
   }
}
```

Lorsque la commande contient toutes les informations, on peut la payer avec une carte de crédit. Cette fonction doit transférer les informations de carte de crédit vers le service de paiement distant disponible à `https://caissy.dev/shops/pay`.

L'API REST de paiement distant est décrite dans la section **service de paiement distant**.

```
POST /order/<int:order_id>/pay
Content-Type: application/json
```

```
{
   "credit_card" : {
      "name" : "John Doe",
      "number" : "4242 4242 4242 4242",
      "expiration_year" : 2024,
      "cvv" : "123",
      "expiration_month" : 9
   }
}
```

```
200 OK
Content-Type: application/json
```

```
{
   "order" : {
      "shipping_information" : {
         "country" : "Canada",
         "address" : "201, rue Président-Kennedy",
         "postal_code" : "H2X 3Y7",
         "city" : "Montréal",
         "province" : "QC"
      },
      "email" : "caissy.jean-philippe@uqam.ca",
      "total_price" : 9148,
      "paid": true, # <- une commande complète est considéré
      ↪  comme payée
      "product" : {
         "id" : 123,
         "quantity" : 1
      },
      "credit_card" : {
         "name" : "John Doe",
         "first_digits" : "4242",
         "last_digits": "4242",
         "expiration_year" : 2024,
         "expiration_month" : 9
      },
```

```
        "transaction": {
            "id": "wgEQ4zAUdYqpr21rt8A10dDrKbfcLmqi",
            "success": true,
            "amount_charged": 10148
        },
        "shipping_price" : 1000,
        "id" : 6543
    }
}
```

Service de paiement distant

Lorsque l'API `/order/<int:order_id>/pay` est appelé, il doit communiquer avec l'API distant de paiement. Cette API est disponible à l'adresse `https://caissy.dev/shops/pay`.

```
POST /shops/pay
Host: caissy.dev
Content-Type: application/json
```

```
{
   "credit_card" : {
      "name" : "John Doe",
      "number" : "4242 4242 4242 4242",
      "expiration_year" : 2024,
      "cvv" : "123",
      "expiration_month" : 9
   },
   "amount_charged": 10148 # <- montant total incluant les fais
   ↪  d'expédition
}
```

```
200 OK HTTP/1.1
```

```
{
   "credit_card" : {
      "name" : "John Doe",
      "first_digits" : "4242",
      "last_digits": "4242",
      "expiration_year" : 2024,
      "expiration_month" : 9
   },
   "transaction": {
       "id": "wgEQ4zAUdYqpr21rt8A10dDrKbfcLmqi",
       "success": true,
       "amount_charged": 10148
   }
}
```

Les informations de la transaction doivent être stockés sur la commande.

En cas d'erreur, l'API distant va retourner un erreur de type :

```
POST /shops/pay
Host: caissy.dev
Content-Type: application/json
```

```
{
   "credit_card" : {
      "name" : "John Doe",
      "number" : "4000 0000 0000 0002",
      "expiration_year" : 2024,
      "cvv" : "123",
      "expiration_month" : 9
   },
   "amount_charged": 10148 # <- montant total incluant les fais
   ↪  d'expédition
}
```

```
422 Unprocessable Entity HTTP/1.1
```

```
{
   "errors" : {
       "credit_card": {
           "code": "card-declined",
           "name": "La carte de crédit a été déclinée"
        }
   }
}
```

Seul les 2 cartes de crédits de tests vont être acceptés :

- `4242 4242 4242 4242` : carte de crédit valide
- `4000 0000 0000 0002` : carte de crédit déclinée

Tous les autres cartes de crédits vont retourner un de code `incorrect-number`

Les champs `expiration_year` et `expiration_month` doivent être des entiers représentant une expiration valide. L'API va retourner un erreur si la carte est expirée.

Le champ `cvv` doit obligatoirement être un string contenant 3 chiffres.

Lorsqu'une commande est payé, l'API doit retourné toutes les informations de la commande :

```
GET /order/<int:order_id>
Content-Type: application/json

200 OK HTTP/1.1
```


```
{
   "order" : {
      "shipping_information" : {
         "country" : "Canada",
         "address" : "201, rue Président-Kennedy",
         "postal_code" : "H2X 3Y7",
         "city" : "Montréal",
         "province" : "QC"
      },
      "email" : "caissy.jean-philippe@uqam.ca",
      "total_price" : 9148,
      "paid": true,
      "product" : {
         "id" : 123,
         "quantity" : 1
      },
      "credit_card" : {
         "name" : "John Doe",
         "first_digits" : "4242",
         "last_digits": "4242",
         "expiration_year" : 2024,
         "expiration_month" : 9
      },
      "transaction": {
          "id": "wgEQ4zAUdYqpr21rt8A10dDrKbfcLmqi",
          "success": true,
          "amount_charged": 10148
      },
      "shipping_price" : 1000,
      "id" : 6543
   }
}
```

Récupération des produits

Lors du lancement de l'application, celle-ci va devoir se connecter à ce service, et récupérer la liste complète des produits et des informations applicables. Les informations du produits devront être persistés localement, c'est-à-dire que l'application Web ne doit pas récupérer les informations à chaque requête. Seulement une fois par lancement.

Les produits sont accessible à l'adresse suivante : `https://caissy.dev/shops/products`

La réponse sera une liste de produits répondant aux caractéristiques suivantes :

```
GET /shops/products
Content-Type: application/json
```

```
{
   "products" : [
      {
         "name" : "Hapiness Bottle",
         "id" : 123,
         "in_stock" : true,
         "description" : "An incredible bottle filled with
         ↪  happiness and joy. One of a kind!",
         "price" : 5999,
         "weight": 500,
         "image": "http://caissy.dev/shops/images/image.png"
      },
      {
         "description" : "A limited edition special Flying
         ↪  Squirrel. Hurry up, it won't last long!",
         "image": "http://caissy.dev/shops/images/image.png",
         "in_stock" : false,
         "weight": 6400,
         "id" : 456,
         "name" : "Flying Squirrel Limited Edition",
         "price" : 3149
      }
   ]
}
```

Nom du champ	Définition
`id`	Un identifiant numérique unique représentant la resource du produit
`name`	Le nom du produit
`description`	La déscription du produit
`price`	Le prix du produit en cents
`in_stock`	Si le produit est en stock ou pas
`image`	Une image représentant le produit

Exigences techniques

1. Le projet devra rouler sous Python 3.6+ et Flask 1.11+
2. Seul les packets `flask`, `pytest`, `pytest-flask`, et `peewee` sont permis. Vous avez droit d'utiliser tous les modules de la librairie standard de Python
3. L'application Web doit avoir une couverture de test de 100%
4. La base de donnée utilisé sera un fichier local `sqlite3`
5. La base de donnée doit être initialisé avec

   ```
   FLASK_DEBUG=True FLASK_APP=inf5190 flask init-db
   ```

6. À partir de la racine de votre projet, l'application doit pouvoir rouler avec la commande suivante :

   ```
   FLASK_DEBUG=True FLASK_APP=inf5190 flask run
   ```

Critères d'évaluations

Pointage	Description
20%	Respect des **exigences techniques**
10%	Récupération des produits à partir du service REST distant
15%	Utilisation de l'API de paiement distant
10%	Gestion d'erreurs (commande inexistante, erreur de validation, etc)

Pointage	Route	Description
5%	`GET /`	Affichage de la page d'accueil
10%	`POST /product/<int:product_id>/buy`	Création d'une commande
–	–	–
10%	`GET /order/<int:order_id>`	Récupération d'une commande
10%	`POST /order/<int:order_id>`	Rajout des informations à une commande
10%	`POST /order/<int:order_id>/pay`	Finalisation d'une commande

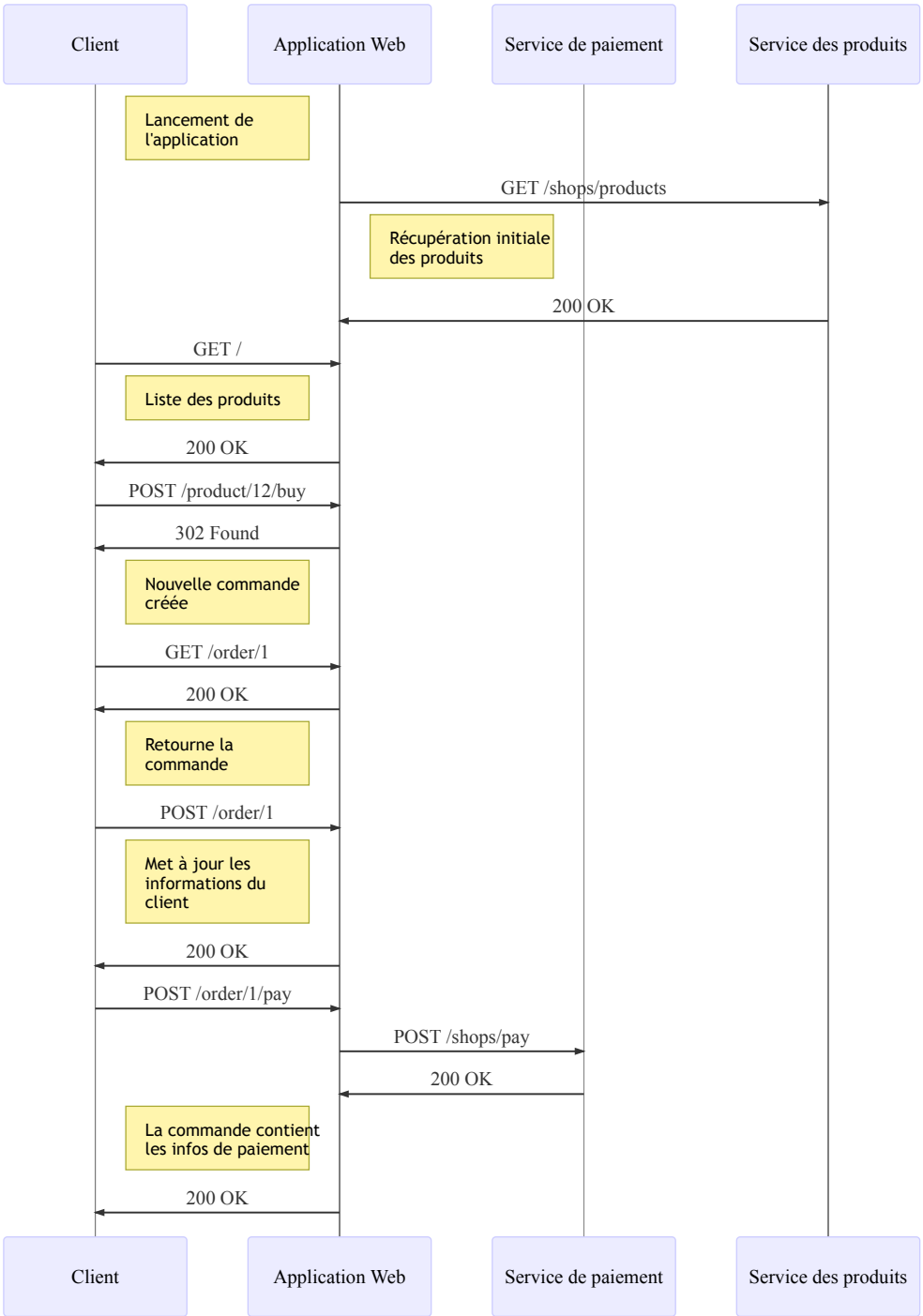

Figure 1: Diagramme de séquence