AMÉLIORATION D’UNE APPLICATION EXISTANTE.

PROJET 11 D.A PYTHON

ÉCHANGES PAR MAIL AVEC LE CLIENT (1/3)

Mail n°1 :

Bonjour,
Nous avons bien pris note de vos remarques concernant le programme, ce premier mail vous parvient afin de récapituler vos attentes :

- *Ajout d'une fonction d'export PDF pour l'utilisateur afin de retrouver sur papier ou smartphone ces nouveaux modes de consommation.*
- *Permettre à l'utilisateur de supprimer les produits enregistrés de son espace.*
- *Ajouter une fonction mettant à jour la BDD des produits enregistrés.*

Nous vous recontacterons rapidement afin de vous informer de l'avancée de ces fonctionnalités

l'équipe technique
Bien cordialement
ESTIVAL Thomas

DÉVELOPPEMENT DES NOUVELLES FONCTIONNALITÉS (EXPORT PDF)

```python
class ExportPdf:
    """New program to export substitutes in PDF format"""
    def export():
        """recovery of the number of products with count and registration of substitutes in the PDF"""
        with connection.cursor() as cursor:
            sql = "SELECT COUNT(*) FROM SUBSTITUTS"
            cursor.execute(sql, ())
            count = cursor.fetchone()['COUNT(*)']
            connection.commit()

        name = input("Quelle est votre nom ? \n>>> ")
        pdf = canvas.Canvas("substituts-{}.pdf".format(name))
        pdf.drawString(3*cm, 28*cm, u'Bienvenue {} vous avez enregistré {} produit{}'.format(name, count, test_plural(vartest=count)))
        pdf.line(7.5*cm, 23*cm, 7.5*cm, 0*cm)
        pdf.line(14.5*cm, 23*cm, 14.5*cm, 0*cm)
        #Create the column

        pdf.drawString(2*cm, 23.5*cm, u'Mes habitudes')
        pdf.drawString(9.5*cm, 23.5*cm, u'Mes substituts')
        pdf.drawString(17*cm, 23.5*cm, u'Magasins')
        #Create the lines
        nb_line, x_position, y_position = 21, 21, 21

        #Get the input_product from table SUBSTITUTS from Database
        with connection:
            cur = connection.cursor()
            cur.execute("SELECT INPUT_PRODUCT, PRODUIT_ID, STORE FROM SUBSTITUTS")
            data_sub = cur.fetchall()

        position = 20.4
        for s in data_sub:

            with connection:
                cur = connection.cursor()
                cur.execute("SELECT NOM FROM PRODUITS WHERE ID=%s" % (int(s["PRODUIT_ID"])))
                data_sub2 = cur.fetchall()
                pdf.drawString(9*cm, position*cm, str(data_sub2[0]["NOM"]))

            pdf.drawString(2.5*cm, position*cm, s["INPUT_PRODUCT"])
            pdf.drawString(15.5*cm, position*cm, s["STORE"])
            pdf.line(0*cm, x_position*cm, 21*cm, y_position*cm)
            nb_line -= 1
            position -= 1
            x_position -= 1
            y_position -= 1

        #Get the substituts from table SUBSTITUS from Database

        pdf.save()
        print("\nVotre PDF a bien été enregistré sous le nom : substituts-{}.pdf".format(name))
```

- Module reportalb
- Requêter les tables SQL
- Demander le nom de l'utilisateur
- Dessiner la mise en page du document
- Insérer les données dans le document
- Exporter le PDF au nom de l'utilisateur

DÉVELOPPEMENT DES NOUVELLES FONCTIONNALITÉS (INTERACTIONS SUPPLÉMENTAIRES AVEC LA BDD)

```python
class CleaningDB():

    """Class to clean the database with sql requests Delete and alter"""
    def cleaning_all_products():
        """Deleting data with a python loop interacting with SQL"""
        with connection.cursor() as cursor:
            for delete in TABLES:
                sql = "DELETE FROM %s;" %(delete)
                cursor.execute(sql, ())
                connection.commit()
            """resets the counters with auto_increment"""
            for reset in TABLES:
                sql = "ALTER TABLE %s AUTO_INCREMENT=0;" % (reset)
                cursor.execute(sql, ())
                connection.commit()

    def cleaning_only_product():
        Consult.consult_compare()
        choice_id = input("Tapez l'ID du produit que vous souhaitez supprimer\n>>> ")
        with connection.cursor() as cursor:
            sql = "DELETE FROM `SUBSTITUTS` WHERE ID=%s" % choice_id
            cursor.execute(sql, ())
            connection.commit()
```

- Interroger l'utilisateur (**A** Tous les produits - **B** Produit unique)
- **A) Boucle for itérant sur l'ensemble des table pour supprimer les produits; Réinitialisation de l'auto incrémentation**
- **B) Récupérer l'ID du produit à supprimer, Exécuter une requête SQL DELETE sur la table substitut**

ÉCHANGES PAR MAIL AVEC LE CLIENT (2-3)

Mail n°2 :

Bonjour,
En date du 30 juillet 2019, nous avons procédé aux modifications de votre programme récapitulées dans l'email précédent. Nous vous ferons parvenir sous 48h l'illustration des modifications effectuées.

l'équipe technique
Bien cordialement
ESTIVAL Thomas

DÉVELOPPEMENT DES NOUVELLES FONCTIONNALITÉS (AMÉLIORATION DU CODE)

- Approche Orientée objet
- Factorisation (Fonction, déclaration de fonctions “allégées”, organisation plus poussée du programme, noms de variables davantage explicites.
- Observation et remise en question du P5 précédemment évalué par un mentor évaluateur pour améliorer l’application

TEST UNITAIRES DES FONCTIONNALITÉS (PDF)

```python
import os
import unittest
import glob
import re
from P11_01_codesource import ExportPdf

class WidgetTestCase(unittest.TestCase):
    def setUp(self):
        print("\n\n---------- TEST DE L'EXPORT PDF ----------\n\n")
        self.generate = ExportPdf.export()
        self.search_pdf = glob.glob("./*pdf")

    def test_exist(self):
        if not self.search_pdf:
            search = False
            print("Test échoué la liste est vide")
        else:
            search = True
            print("Test réussi, la liste contient un fichier pdf en sortie")
        self.assertEqual(search, True)

if __name__ == '__main__':
    unittest.main()
```

TEST UNITAIRES DES FONCTIONNALITÉS (BDD)

test_P11_03_update.py × | test_P11_05_bdd.py ×

```python
import unittest #Test tools
import pymysql #mysql connection utility
import pymysql.cursors

from P11_01_codesource import CleaningDB #Function concerned by the test
from P11_02_constantes import LOGIN_CONNECT, TABLES

try:
    connection = pymysql.connect(host=LOGIN_CONNECT["HOST"],
                                 user=LOGIN_CONNECT["USER"],
                                 password=LOGIN_CONNECT["PASSWORD"],
                                 db=LOGIN_CONNECT["DB"],
                                 charset='utf8mb4',
                                 port=LOGIN_CONNECT["PORT"],
                                 cursorclass=pymysql.cursors.DictCursor)

except:
    print("Erreur de connexion, veuillez vérifier les paramètres dans le fichier constants.py")


class WidgetTestCase(unittest.TestCase):
    """Test the database cleaning"""
    def setUp(self):
        """Start of cleaning"""
        CleaningDB.cleaning_all_products()
        print("\n\n---------- TEST DU NETTOYAGE DE LA BASE DE DONNEES ----------\n\n")

    def test_clean(self):
        """SQL query for comparison"""
        for t in TABLES:
            with connection.cursor() as cursor:
                sql = "SELECT * FROM %s;" %(t)
                cursor.execute(sql, ())
                clean_test = cursor.fetchall()
                print(clean_test)

                if clean_test == ():
                    print(f"{t} est bien vide, test réussi !")
                else:
                    print(f"{t} n'est pas vide, test échoué !")

                connection.commit()

            if self.assertEqual(clean_test, ()) :
                print("\n\n---------- TEST DU NETTOYAGE DE LA BASE DE DONNEES OK ----------\n\n")


if __name__ == '__main__':
    unittest.main()

```

TEST UNITAIRES DES FONCTIONNALITÉS (UPDATE)

test_P11_03_update.py | test_P11_05_bdd.py

```python
import unittest #Test tools
import datetime #For get the date
from P11_01_codesource import update #My project

date_test = datetime.datetime.now()

"""standardize the date for the test"""

if date_test.day < 10 and date_test.month < 10:
    date_test = f"{date_test.year}-0{date_test.month}-0{date_test.day}"
    print(date_test)

elif date_test.month < 10:
    date_test = f"{date_test.year}-0{date_test.month}-{date_test.day}"
    print(date_test)

elif date_test.day < 10:
    date_test = f"{date_test.year}-{date_test.month}-0{date_test.day}"
    print(date_test)

else:
    date_test = f"{date_test.year}-{date_test.month}-{date_test.day}"
    print(date_test)

DATE_TEST = str(date_test)


class WidgetTestCase(unittest.TestCase):

    """Are the updates done? test by comparing the dates of the datetime module
     and the date recorded in DB"""

    def setUp(self):
        """Starting the update, and retrieving the date"""
        print("\n\n---------- TEST DE LA MISE À JOUR BDD ----------\n\n")
        self.date_control = update()

    def test_date(self):
        """Compare the SQL update date and today's date"""
        self.assertEqual(self.date_control, DATE_TEST)
        print("\n\n---------- TEST DE LA MISE À JOUR BDD OK ----------\n\n")



if __name__ == '__main__':
    unittest.main()
```

ÉCHANGES PAR MAIL AVEC LE CLIENT (3-3)

Mail n°3 :

Bonjour,
Voici deux illustrations ci-jointes présentant les fonctionnalités ajoutées à votre programme.
En vous en souhaitant bonne réception veuillez agréer Mme l'expression de mes salutations distinguées.

l'équipe technique
Bien cordialement
ESTIVAL Thomas