

## **Them 2: Projet technique – Application de gestion locale de l’UPF**

### **Introduction et objectifs**

La gestion des achats et des fournisseurs au sein de l’Université Privée de Fès (UPF) nécessite de nombreuses tâches administratives chronophages. Pour améliorer l’efficacité et la traçabilité de ces processus, une application locale de gestion des approvisionnements et du stock a été développée, conformément au cahier des charges. Ce projet couvre les fonctionnalités attendues par l’UPF : gestion des fournisseurs, d’un catalogue de produits/services, création et suivi des commandes d’achat, enregistrement des réceptions et génération de rapports et exports (PDF/Excel) analytiques. L’objectif général est d’automatiser ces processus administratifs, de centraliser les informations (fournisseurs, produits, commandes, stocks, dépenses, etc.) et de sécuriser l’accès (authentification par rôle) pour faciliter le travail du personnel administratif de l’UPF.

La solution intègre également un système de sécurité avec deux rôles (Administrateur et Utilisateur) afin de différencier les droits d’accès selon le profil de l’utilisateur.

### **Environnement technique**

L’application est développée en Python (version 3.10 ou supérieure) pour profiter de ses dernières fonctionnalités et de son large écosystème. Le micro-framework Flask est utilisé pour la partie web, car il est léger et rapide. La base de données choisie est SQLite, un moteur SQL embarqué sans serveur externe et « zéro-configuration » : l’accès aux

données se fait directement sur un fichier local, sans qu'il soit nécessaire de configurer un serveur de base de données distinct.

Pour la génération de documents, on utilise les bibliothèques Python ReportLab et FPDF pour créer les rapports au format PDF. Les données tabulaires sont manipulées avec pandas, ce qui simplifie l'export en Excel. Pour les visualisations graphiques (facultatives), on peut intégrer Matplotlib (bibliothèque de tracé) et Seaborn (surcouche pour des graphiques statistiques plus esthétiques).

L'application intègre un système d'authentification local avec gestion de rôles (Administrateur et Utilisateur) pour sécuriser l'accès. Le contrôle d'accès basé sur le rôle (RBAC) permet de définir les autorisations de chaque profil : l'Administrateur dispose de droits étendus (notamment la création de comptes), tandis que l'Utilisateur a un accès restreint aux fonctionnalités. Enfin, l'application est empaquetée sous Windows en un exécutable (.exe) via un outil comme PyInstaller, ce qui facilite son installation sur les postes de l'UPF sans nécessiter l'installation manuelle de Python ou de ses dépendances.

### **Architecture de l'application**

**Fournisseurs** : Module CRUD complet pour les fournisseurs (ajout, modification, suppression, consultation). Chaque fournisseur dispose d'un nom, d'un email, d'un téléphone, d'une adresse, d'un type de service/produit et d'une date d'ajout. Un tableau historique affiche la liste des fournisseurs avec barre de recherche et filtres par nom, type ou date.

**Produits / Services** : Gestion d'un catalogue local d'articles (produits et services). Chaque élément est associé à un fournisseur et classé par catégorie (produit ou service). Un tableau consultable (avec recherche et filtres) permet de retrouver facilement les articles par désignation, par fournisseur ou par type.

**Commandes d'Achat** : Création et suivi des commandes. L'utilisateur sélectionne un fournisseur, puis ajoute plusieurs lignes de commande (désignation du produit/service, quantité, prix unitaire). Chaque commande évolue selon des statuts (Brouillon, Validée, Livrée) et peut être exportée en PDF. Un historique des commandes (avec recherche et filtres par fournisseur, statut ou date) est disponible pour le suivi.

**Réceptions** : Enregistrement des livraisons partielles ou totales associées aux commandes d'achat. La réception des marchandises met automatiquement à jour le stock disponible. Un tableau récapitulatif affiche l'historique des réceptions avec filtres (par date ou par commande).

**Rapports et Export** : Suivi global des commandes, réceptions et dépenses. Les informations clés (statut des commandes, état des stocks, dépenses par fournisseur ou par produit/service) y sont présentées et peuvent être exportées au format PDF ou Excel (tableaux détaillés avec désignations, quantités, prix TTC, montants totaux, etc.). Des visualisations graphiques (courbes, histogrammes via matplotlib/seaborn) et des filtres dynamiques sur l'historique global offrent un aperçu analytique des dépenses et des stocks.

**Connexion / Sécurité** : Interface de connexion protégée (login/mot de passe) avec attribution de rôles utilisateur. Deux profils sont définis :

l'Administrateur (accès total à l'application et création de comptes) et l'Utilisateur (droits restreints). Ce contrôle d'accès par rôle respecte les principes du RBAC, assurant que chaque utilisateur n'accède qu'aux fonctionnalités autorisées.

### **Mise en œuvre du projet**

Le développement s'est déroulé en plusieurs étapes :

**Analyse et planification** : Étude du cahier des charges et recueil des besoins auprès du tuteur de stage. Définition des spécifications fonctionnelles et modélisation de la base de données (schémas UML, maquettes des pages).

**Conception de la solution** : Mise en place de l'architecture logicielle (structure Flask, modèles de données). Conception détaillée des tables SQLite et des interfaces utilisateur.

**Développement itératif** : Implémentation successive de chaque module fonctionnel (fournisseurs, produits/services, commandes, réceptions, rapports, authentification). Pour chaque module, les opérations CRUD ont été codées en Python/Flask, avec les logiques associées (calculs de totaux, mise à jour du stock, génération de PDF, etc.). Des tests unitaires intermédiaires ont été réalisés à chaque étape.

**Tests et validation** : Réalisation de tests fonctionnels pour vérifier la cohérence des opérations (ajout, modification, suppression, export de documents, mise à jour des stocks, etc.). Les retours du superviseur ont été pris en compte pour corriger les anomalies et améliorer l'ergonomie (ajustement des filtres de recherche, correction de bugs mineurs). Ce cycle

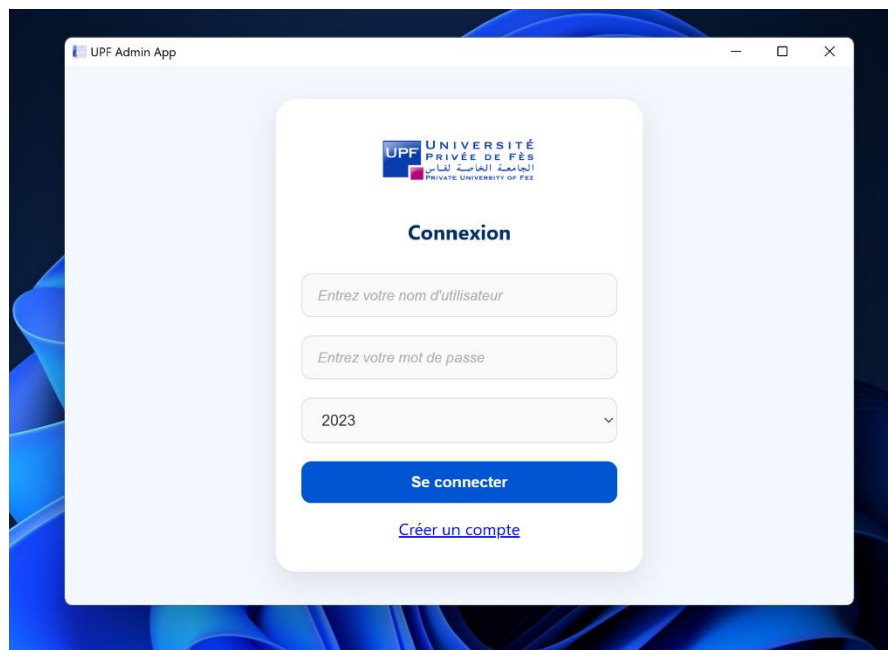
itératif de tests et corrections a permis d'assurer la stabilité du logiciel développé.

**Documentation et déploiement :** Rédaction de la documentation utilisateur décrivant l'installation et l'utilisation de l'application. Packaging de l'application en un exécutable Windows (.exe) pour une installation facile sur les postes de l'UPF, sans nécessiter d'installation manuelle des dépendances Python.

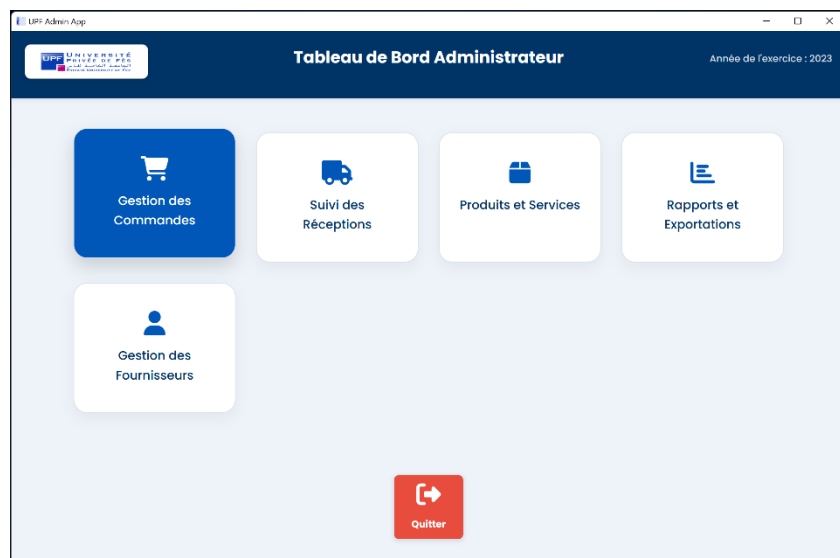
### Tests et résultats

Après le développement, l'interface a été testée avec des cas d'utilisation représentatifs : création et modification de fournisseurs, émission et validation de commandes, enregistrement de réceptions, génération de rapports, etc. Les tests ont confirmé la conformité aux attentes. Par exemple, l'export PDF d'une commande d'achat produit un document complet avec le récapitulatif des lignes de commande et les totaux TTC. Pour illustrer ces résultats, plusieurs captures d'écran seront insérées dans le rapport :

La page de connexion (login) où l'utilisateur saisit son identifiant et son mot de passe.



Le tableau de bord principal (après authentification), présentant les liens vers les différents modules de l'application.



L'interface du module **Fournisseurs** affichant la liste des fournisseurs dans un tableau (avec barre de recherche et filtres) et les actions d'ajout, modification ou suppression.

Nom du Fournisseur: fournisseur test 2

Email:

Téléphone: 06xxxxx

Adresse:

Ville:

Responsable: test

Contact:

Enregistrer

### Liste des Fournisseurs


Rechercher: Nom, Ville, Responsable...

Ville: Toutes les villes

Filtrer

Nom	Email	Téléphone	Adresse	Ville	Responsable	Statut	Actions
fournisseur test 1	test@gmail.com		test	fes		Actif	Modifier Supprimer

Un exemple de document PDF généré (par exemple un bon de commande et Rapport des operations), montrant le récapitulatif des lignes de commande et les totaux.



UNIVERSITÉ

PRIVÉE DE FÈS

الجامعة الخاصة لفes

PRIVATE UNIVERSITY OF FES

Date : 2025-05-21

Page : 1

**Bon de Commande**

Fournisseur : test

Commandée par : admin

Type de Commande : Standard

N° : 2/2025


Désignation	Qté	P.U. TTC	Montant	Affectation
test	12	17.00	204.00	test

Total HT : 170.00 MAD

Total TTC : 204.00 MAD

Adresse: Lot Quraouiyyine Route Ain Chkef Fes | Mail: benatiya@upf.ac.ma

Tel: 0535610320 | Fax: 0535601873 | ICE: 000226086000082



UNIVERSITÉ

PRIVÉE DE FÈS

الجامعة الخاصة لفes

PRIVATE UNIVERSITY OF FES

Date : 2025-05-22

Page : 1

**Rapport des Opérations**

Désignation	Fournisseur	Qté	P.U. TTC	Montant	Affectation	Statut	Date
test	test	12	17.00	204.00	test	brouillon	2025-05-21

Adresse: Lot Quraouiyyine Route Ain Chkef Fes | Mail: benatiya@upf.ac.ma

Tel: 0535610320 | Fax: 0535601873 | ICE: 000226086000082

## **Conclusion et perspectives**

L'application de gestion locale développée répond à l'ensemble des besoins initiaux de l'UPF en matière d'approvisionnement. Tous les modules prévus ont été implémentés et validés : gestion des fournisseurs, du catalogue, des commandes, des réceptions, des stocks, des rapports et des accès sécurisés. Ce projet m'a permis de renforcer mes compétences en développement Python/Flask, en conception de bases de données relationnelles (SQLite) et en gestion de projet (analyse des besoins, itérations, tests, déploiement). J'ai également consolidé mes savoir-faire en production de documents automatisés (PDF/Excel) et en design d'interfaces utilisateur.

Plusieurs pistes d'amélioration sont envisageables : déployer l'application en mode web (Flask sous Linux ou conteneur Docker) la rendrait accessible depuis n'importe quel poste du réseau. Enrichir l'interface (par exemple avec Bootstrap ou React) et ajouter des tableaux de bord interactifs (graphiques en temps réel, filtres dynamiques) améliorerait l'ergonomie et l'analyse des données. On pourrait également étendre l'application en proposant une interface multilingue, ajouter des notifications automatiques ou migrer vers une base de données plus robuste (ex. PostgreSQL) pour préparer une montée en charge. Ces évolutions prolongeraient la durée de vie du projet et ouvriraient de nouvelles possibilités pour l'UPF.