BTS SIO 2ème année – Option SLAM

Étudiant : Grégory Marcelin

BREVET DE TECHNICIEN SUPERIEUR SERVICE INFORMATIQUE AUX ORGANISATION OPTION B : SOLUTIONS LOGICIEL APPLICATION METIER

Épreuve E6

Dossier de Projet 2

Générateur d'attestation de naissance en PDF

SESSION 2025

BTS SERVICES INFORMATIQUES AUX ORGANISATIONS **SESSION 2024**

ANNEXE 9-1-B : Fiche descriptive de réalisation professionnelle (recto)

Épreuve E5 - Conception et développement d'applications (option SLAM)

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE		**N° réalisation :**
Nom, prénom : Marcelin Gregory		**N° candidat :**
Épreuve ponctuelle X	**Contrôle en cours de formation**	**Date :** ..... / ..... /.....
Organisation support de la réalisation professionnelle		
Intitulé de la réalisation professionnelle		
Période de réalisation : .....Lieu : ..... **Modalité : X Seul(e)** ☐ **En équipe**		
Compétences travaillées ☐ Concevoir et développer une solution applicative ☐ Assurer la maintenance corrective ou évolutive d'une solution applicative ☐ Gérer les données		
Conditions de réalisation[5] (ressources fournies, résultats attendus)		
Description des ressources documentaires, matérielles et logicielles utilisées[6]		
Modalités d'accès aux productions[7] et à leur documentation[8]		

BTS SERVICES INFORMATIQUES AUX ORGANISATIONSSESSION 2024

ANNEXE 9-1-B : Fiche descriptive de réalisation professionnelle
(verso, éventuellement pages suivantes)

Épreuve E5 - Conception et développement d'applications (option SLAM)

Descriptif de la réalisation professionnelle, y compris les productions réalisées et schémas explicatifs

SOMMAIRE

2. Contexte du projet....4
2. Objectifs du projet....5
3. Fonctionnalités réalisées....5
3.1 Génération de PDF....5
3.2 Interface graphique....6
3.3 Gestion des données (version évoluée)....6
4. Technologies utilisées....6
6. Réalisation technique....8
7. Difficultés rencontrées....9
Centrage de la fenêtre principale (PySide6)....9
Mise en page dynamique du PDF....9
Validation et gestion des erreurs de saisie....9
8. Compétences mises en oeuvre....10
9. Pistes d'amélioration / évolutions....10
10. Conclusion....11
Annexe – Schéma de la base de données SQLite....14

2. Contexte du projet

Dans certains pays ou régions défavorisées, l'obtention d'un certificat de naissance officiel peut prendre plusieurs semaines, voire plusieurs mois. Ce délai complique fortement certaines démarches administratives importantes pour la famille, comme l'obtention d'une couverture santé, la déclaration auprès de la mairie, ou encore l'accès à certains droits sociaux.

Inspiré par ces problématiques, j'ai imaginé un projet technique visant à simplifier la création d'attestations provisoires de naissance. L'idée était de proposer un outil numérique simple et rapide permettant de générer des documents PDF clairs, sécurisés et facilement partageables, utilisables par des associations, des familles ou des structures locales ne disposant pas d'un système officiel automatisé.

Face à ce constat, j'ai choisi de développer une application Python qui permet de créer des attestations de naissance précises et bien formatées sous forme de PDF, à l'aide d'une interface simple utilisable par tout agent, même sans compétence technique.

3. Objectifs du projet

L'objectif principal de ce projet est de concevoir une **solution simple et accessible** permettant de générer des **attestations de naissance personnalisées**. Le logiciel devait pouvoir être utilisé aussi bien par des personnes peu familières avec l'informatique que par des agents de structures locales ou associatives.

Pour répondre à cette exigence, j'ai choisi d'intégrer une **double interface** : une version en ligne de commande (terminal) pour les environnements minimalistes ou techniques, et une **interface graphique moderne** basée sur PySide6, offrant une meilleure ergonomie et facilitant la saisie des informations.

Le projet visait également à produire un **fichier PDF clair et bien structuré**, mettant en forme les données de manière professionnelle, avec des sections lisibles et un encadré visuel. Enfin, j'ai prévu la possibilité d'**ajouter une signature numérique optionnelle** au document, sous forme d'image, pour permettre une validation manuelle ou symbolique du document.

4. Fonctionnalités réalisées

L’application développée propose une série de fonctionnalités destinées à simplifier et automatiser la génération d’attestations de naissance au format PDF, tout en assurant une traçabilité des documents et une facilité d’utilisation.

4.1 Génération de PDF

L’utilisateur est guidé à travers une **interface intuitive** pour renseigner les informations essentielles concernant la naissance : nom, prénom, sexe de l’enfant, date et heure de naissance, lieu, ainsi que les coordonnées du déclarant. Une **validation en temps réel** est appliquée aux champs saisis afin de garantir la qualité et la cohérence des données.

Une fois les informations renseignées, l’application génère un **document PDF professionnel**, structuré selon un modèle administratif clair, comprenant des titres, un encadré principal, des sections distinctes, un pied de page ainsi qu’une **ligne de signature**. Une image de signature numérique peut également être ajoutée de manière optionnelle. Le fichier est automatiquement sauvegardé dans le **dossier Téléchargements** de l’utilisateur pour faciliter son accès.

4.2 Interface graphique

L’interface a été conçue avec **PySide6**, ce qui permet une ergonomie fluide, moderne et multiplateforme. Elle intègre différents widgets adaptés :

- **QLineEdit** pour les saisies textuelles,
- **QDateEdit** et **QTimeEdit** pour les sélections de date et d’heure,
- **QComboBox** pour les choix (ex : sexe),
- **QFileDialog** pour l’import d’une image de signature.

L’application propose une expérience utilisateur claire, avec des **messages d’erreur explicites**, des **boutons d’action visibles**, et une **navigation fluide**, rendant l’outil utilisable par tout agent, même sans compétences techniques.

4.3 Gestion des données (version évoluée)

Bien que facultative dans la version de base, une **base SQLite** peut être intégrée pour :

- sauvegarder l'historique des attestations générées,
- faciliter la régénération d'un document à partir des données stockées,
- permettre une consultation centralisée via une interface tabulaire.

5. Technologies utilisées

Le projet repose sur une stack technique légère, portable et robuste, 100 % en Python.

- **Python 3** : langage principal du développement, orienté objets, avec gestion d'exceptions, validations et modularité.
- **fpdf2** : bibliothèque de génération PDF, permettant une **mise en page personnalisée**, l'ajout d'images, de lignes de signature, de filigranes, et une gestion précise du format A4.
- **PySide6** (Qt pour Python) : pour la **création d'interfaces graphiques modernes**, multi-plateformes, avec gestion de widgets, événements et validations.
- **prompt_toolkit** : utilisé dans la version terminale du programme pour offrir une **saisie assistée et validée**, utile dans les contextes sans interface graphique.
- **Modules standards** : `os`, `sys`, `pathlib`, `datetime`, pour la **gestion des chemins système, de la date, de la compatibilité OS et de la génération de noms de fichiers dynamiques`.

L'ensemble des composants a été intégré de manière modulaire, facilitant la **maintenance**, la **portabilité** et les **évolutions futures** (ajout de base de données, version web, etc.).

6. Architecture de l'application

L'application est organisée de manière modulaire pour faciliter la maintenance, la lecture du code et les évolutions futures. Chaque fichier a un rôle spécifique, clairement défini :

- **`app.py`**
  Fichier principal de l'application. Il lance l'interface graphique avec **PySide6**. Ce fichier initialise l'application, affiche le formulaire, gère les événements (bouton de génération, sélection de signature) et récupère les données saisies par l'utilisateur pour les transmettre à la fonction de génération PDF.
- **`declaration_naissance.py`**
  Contient la fonction `creer_pdf()` responsable de la **génération du fichier PDF**. Elle utilise la bibliothèque `fpdf2` pour construire un document professionnel structuré avec en-tête, sections, signature numérique, pied de page et mention automatique.
- **`database.py`**
  Gère la **connexion à la base de données SQLite** (`birth_certificates.db`). Ce module contient les fonctions nécessaires à l'insertion, la lecture et la gestion des attestations enregistrées (dans les versions évoluées du projet).
- **`birth_certificates.db`**
  Fichier de base de données SQLite contenant les attestations enregistrées. Il stocke les informations saisies pour chaque génération de PDF, permettant ainsi un historique et une éventuelle régénération.
- **`requirements.txt`**
  Liste des bibliothèques nécessaires à l'exécution du projet (`fpdf2`, `PySide6`, etc.). Ce fichier permet d'installer toutes les dépendances rapidement avec la commande `pip install -r requirements.txt`.
- **`README.md`**
  Fichier de documentation du projet. Il explique le fonctionnement de l'application, les étapes d'installation, d'exécution et les dépendances requises. Il peut aussi servir pour GitHub ou une remise de projet.

7. Réalisation technique

La réalisation technique du projet s'appuie sur une architecture modulaire et claire, permettant de séparer l'interface utilisateur, la logique métier et la génération du document.

La génération du fichier PDF repose sur la bibliothèque **`fpdf2`**, qui permet de construire un document structuré en définissant précisément la mise en page. Le PDF final comporte :

- des **marges personnalisées** pour un rendu professionnel sur format A4,
- un **encadré principal** pour délimiter visuellement le document,
- des **blocs de données structurés** (informations de l'enfant, du déclarant, date de l'attestation),
- une **mention finale** précisant l'origine du document,
- une **ligne de signature**, et l'intégration possible d'une **signature numérique en image**.

Côté interface graphique, le projet utilise **`PySide6`**, le binding Qt pour Python. L'interface est composée de **widgets adaptés** à chaque type d'information :

- `QLineEdit` pour les saisies textuelles (nom, lieu, déclarant),
- `QComboBox` pour le choix du sexe,
- `QDateEdit` pour la sélection des dates,
- `QTimeEdit` pour l'heure de naissance,
- `QFileDialog` pour la sélection d'une signature optionnelle.

Un système de **validation des champs en temps réel** permet de bloquer la génération tant que toutes les informations obligatoires ne sont pas correctement remplies.

Le **bouton de génération** appelle une fonction centrale `creer_pdf()` définie dans un fichier externe (`declaration_naissance.py`). Cette fonction reçoit les données du formulaire, les traite, applique les styles nécessaires (polices, couleurs, espacements) et enregistre le fichier PDF généré dans le dossier Téléchargements de l'utilisateur.

L'ensemble du code suit une **structure orientée objet**, permettant une organisation claire entre la partie interface, les contrôles, et la logique de génération. Ce découpage facilite également les évolutions futures du projet (ajout d'une base de données, export, prévisualisation...).

8. Difficultés rencontrées

Le développement de cette application a présenté plusieurs défis techniques, qui ont nécessité des recherches, des essais et une adaptation constante pour garantir un résultat à la fois fonctionnel, fiable et professionnel.

Centrage de la fenêtre principale (PySide6)

- L'un des premiers défis a été d'assurer le **centrage automatique de la fenêtre graphique** sur l'écran de l'utilisateur, quelle que soit la taille de celui-ci ou le système d'exploitation utilisé. Bien que cela puisse sembler simple, **PySide6 ne propose pas de méthode directe** pour ce positionnement. Il a donc fallu manipuler les géométries des widgets, récupérer la résolution de l'écran actif et ajuster manuellement la position de la fenêtre à l'aide des coordonnées calculées.

Mise en page dynamique du PDF

- Le **positionnement précis des blocs de texte dans le PDF** a exigé de nombreux ajustements. Contrairement à un traitement de texte classique, la bibliothèque `fpdf2` repose sur un système de coordonnées absolues. Il a donc fallu anticiper la **hauteur des lignes, l'alignement horizontal**, et réserver de l'espace pour une éventuelle **signature numérique**. Toute modification d'un champ pouvait provoquer un décalage visuel, ce qui a nécessité l'écriture de fonctions de recalcul automatique des hauteurs et des sauts de ligne.

Validation et gestion des erreurs de saisie

Pour éviter tout comportement imprévu ou la génération de documents incomplets, l'application intègre des **vérifications de conformité sur les champs obligatoires**. Par exemple :

- les dates doivent être valides,
- les champs texte ne doivent pas être vides,
- les formats d'heure doivent respecter `HH:MM`.

9. Compétences mises en oeuvre

Ce projet m’a permis de mobiliser plusieurs compétences clés du développeur.

J’ai d’abord utilisé la **programmation orientée objet** pour organiser mon code de manière claire et modulaire, notamment pour la gestion de l’interface utilisateur avec PySide6. J’ai également manipulé des **bibliothèques tierces**, comme `fpdf` pour la génération de fichiers PDF et `PySide6` pour la création d’une interface graphique intuitive et responsive.

Le projet m’a amené à concevoir une **interface utilisateur complète**, comprenant des champs de saisie, des sélecteurs de date et des éléments d’interaction visuelle. Une attention particulière a été portée à la **validation des données saisies**, afin d’éviter les erreurs de format (date, heure, texte vide).

Enfin, j’ai appris à structurer l’information dans un document PDF, en assurant un **export propre, clair et réutilisable** qui respecte les standards d’un document administratif.

10. Pistes d'amélioration / évolutions

Plusieurs pistes d’amélioration ont été envisagées pour faire évoluer ce projet.

Il serait tout d’abord possible d’ajouter un **système de base de données SQLite** permettant d’enregistrer automatiquement toutes les attestations générées, afin de faciliter leur gestion, leur consultation ou leur réédition.

Une autre évolution consisterait à permettre la **génération par lot**, c’est-à-dire la création de plusieurs attestations à la suite, à partir d’un fichier source (CSV, JSON ou formulaire multipage).

Une **fonction de prévisualisation** du document avant export PDF pourrait également être intégrée à l’interface graphique, pour permettre à l’utilisateur de vérifier visuellement le contenu avant validation.Enfin, une version **mobile ou web**, développée avec des technologies comme **Flask** pour le web, permettrait de rendre l’outil accessible à distance, via un navigateur ou une application mobile connectée.

11. Conclusion

Ce projet m'a permis de m'exercer à la **conception complète d'une application**, en abordant à la fois la **logique métier**, la **gestion des données**, la **génération de documents structurés**, et la création d'une **interface utilisateur ergonomique**. J'ai pu mettre en pratique mes compétences en Python tout en découvrant des bibliothèques comme `fpdf` pour la création de PDF et `PySide6` pour le développement d'interfaces graphiques modernes.

Ce travail démontre ma capacité à **analyser un besoin concret**, à le traduire en solution fonctionnelle et à produire un outil **simple, portable, efficace et réutilisable**. Le générateur d'attestation de naissance que j'ai développé pourrait tout à fait être adapté à d'autres contextes professionnels ou humanitaires. Par exemple, il serait possible de le décliner pour générer des **certificats de présence**, des **attestations d'employeur**, des **fiches administratives**, ou encore des **documents internes normalisés** dans une petite structure.

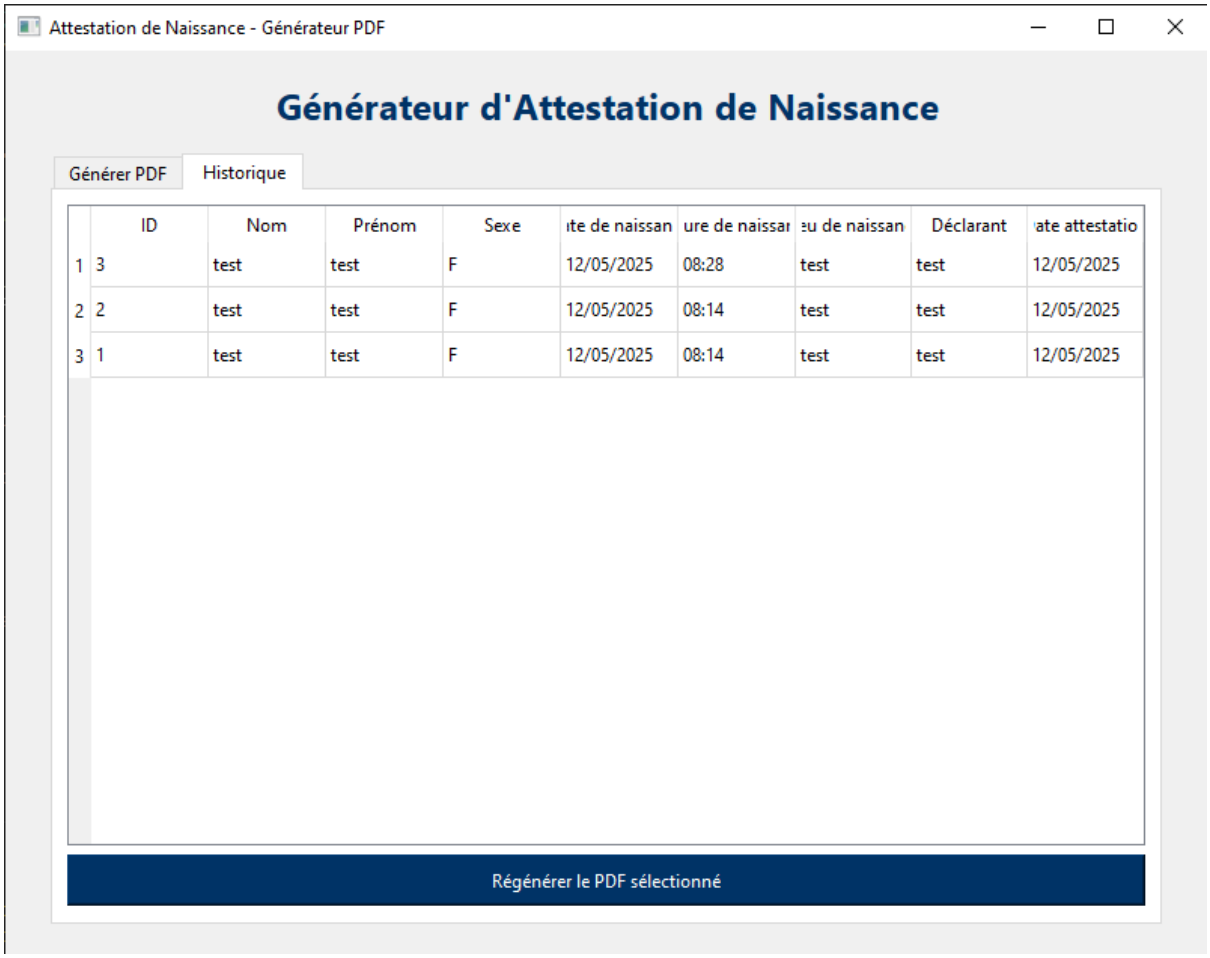
Ce projet m'a ainsi permis de comprendre l'importance de créer des outils numériques **accessibles**, notamment dans les environnements où l'automatisation est encore peu présente.

Annexes

Annexe – Interface graphique de génération d'attestation

Cette capture d'écran illustre la fenêtre principale de l'application desktop, développée avec PySide6. L'interface se compose de plusieurs sections organisées de manière intuitive : Informations sur l'enfant : champs de saisie pour le nom, le prénom, le sexe, la date et l'heure de naissance, ainsi que le lieu. Déclarant : zone dédiée au nom du déclarant et à la date de l'attestation. Signature (optionnelle) : champ permettant d'ajouter une image de signature via le bouton "Parcourir". Un bouton principal "Générer l'attestation PDF" déclenche la génération automatique du document après validation des champs. L'ensemble de l'interface est responsive et multilingue dans sa conception, avec une disposition claire pour garantir une utilisabilité optimale, même pour un utilisateur non technicien.

Annexe – Interface de gestion de l’historique



Cette capture représente l’onglet “Historique” de l’application. Il s’agit d’un tableau affichant la liste des attestations précédemment générées, stockées dans une base de données SQLite. Chaque ligne correspond à un enregistrement contenant les informations principales de l’attestation : identifiant, nom, prénom, sexe, date et heure de naissance, lieu, nom du déclarant, et date de l’attestation. Cette interface permet à l’utilisateur de retrouver rapidement un ancien document et de le régénérer au format PDF en un clic, grâce au bouton situé en bas de l’écran :
“Régénérer le PDF sélectionné”. L’ajout de cette fonctionnalité améliore considérablement la traçabilité des documents, tout en facilitant leur consultation et réédition sans avoir à ressaisir les données.

Annexe – Schéma de la base de données SQLite

Schéma de la table SQLite : certificates

Nom du champ	Type	Description
id	INTEGER	Clé primaire auto-incrémentée
nom	TEXT	Nom de famille de l'enfant
prenom	TEXT	Prénom(s) de l'enfant
sexe	TEXT	Sexe de l'enfant (M ou F)
date_naissance	TEXT	Date de naissance (JJ/MM/AAAA)
heure_naissance	TEXT	Heure de naissance (HH:MM)
lieu_naissance	TEXT	Lieu de naissance
declarant	TEXT	Nom du déclarant
date_attestation	TEXT	Date de l'attestation
date_creation	TEXT	Horodatage automatique à la génération

Ce schéma représente la structure de la table certificates utilisée dans la base de données birth_certificates.db.

Cette table centralise toutes les informations saisies lors de la génération d'une attestation de naissance. Elle stocke à la fois les données déclaratives (nom, prénom, lieu, date de naissance…) et des métadonnées (date de création, horodatage).

Chaque enregistrement possède un identifiant unique (id) auto-incrémenté, ce qui permet de retrouver facilement une attestation générée et, le cas échéant, de la régénérer automatiquement via l'application.

La base de données est locale, légère et exploitable directement via les fonctionnalités SQLite intégrées à Python.