École nationale supérieure d'ingénieurs sud Alsace

Rapport TP Programmation Orientée Objet : **Projet MeetingPro**

GRIMM-KEMPF Matthieu
DAGON Lucas

1er juin 2025

Résumé

Ce rapport présente le développement du projet MeetingPro, réalisé dans le cadre du cours de Programmation Orientée Objet. Nous y détaillons la démarche de conception, les choix techniques, la répartition des tâches, les difficultés rencontrées ainsi que les résultats obtenus.

Table des matières

1 Introduction

Ce projet vise à développer une application de gestion de salles de réunion en utilisant les principes de la programmation orientée objet. L'objectif est de fournir une solution robuste et évolutive pour la gestion des réservations de salles.

2 Méthodologie

Le développement de l'application s'est appuyé sur une approche structurée inspirée du modèle MVC (Modèle-Vue-Contrôleur). Notre priorité a été d'assurer une séparation claire entre les différents composants pour favoriser maintenabilité, évolutivité et testabilité du code.

2.1 Répartition du travail

La répartition du travail a été définie dès les premières étapes du projet :

— **GRIMM-KEMPF Matthieu** s'est chargé de la partie *Contrôleur*, du *Modèle*, du script d'installation du package, des scripts de tests, ainsi que de l'intégralité de la documentation (README et rapport).

— **DAGON Lucas** a développé l'*interface graphique* en utilisant la bibliothèque Tkinter.

2.2 Communication et gestion de version

La coordination entre les membres s'est effectuée principalement via **Discord**, avec des points réguliers pour suivre l'avancée des travaux. Pour la gestion de version, nous avons utilisé **Git** avec un dépôt hébergé sur **GitHub**, conformément aux recommandations de nos enseignants.

2.3 Choix techniques et architecture

L'application est écrite en Python et utilise :

— **Tkinter** pour l'interface graphique, qui, bien que rudimentaire, offre suffisamment de fonctionnalités pour réaliser une interface fonctionnelle.

— **Hmac** pour la génération d'ID pour l'utilisateur. La génération d'ID ne passe pas par UUID mais par SHA256 pour réutiliser du code déjà écris pour un projet personnel. Cette méthode ne garanti certe pas un ID unique malgré qu'une chance inférieure à $1.0 \cdot 10^{-6}\%$ mais le code l'entourant protège contre ce risque.

— **JSON** pour la persistance des données,

— **HMAC avec SHA-256** pour la génération d'identifiants utilisateurs. Contrairement à l'utilisation de UUID, nous avons opté pour une méthode basée sur HMAC pour réutiliser du code existant d'un projet personnel. Bien que cette méthode ne garantisse pas une unicité absolue des identifiants, la probabilité de collision est extrêmement faible (inférieure à $1.0 \cdot 10^{-6}\%$). De plus, des mécanismes de protection ont été mis en place pour gérer ce risque résiduel.

— une architecture **MVC** bien structurée.

Les entités principales (utilisateurs, salles) sont modélisées sous forme de classes, avec sérialisation personnalisée dans des fichiers JSON séparés pour plus de clarté et modularité.

2.4 Cycle de développement

Notre démarche fut itérative, avec les étapes suivantes :

1. Spécification des fonctionnalités principales,
2. Modélisation UML des entités,
3. Développement incrémental avec tests intégrés,
4. Intégration des différentes parties,
5. Rédaction de la documentation.

3 Fonctionnement du système

Le projet repose sur une architecture conforme au modèle MVC. Le diagramme suivant illustre les relations entre les composants :

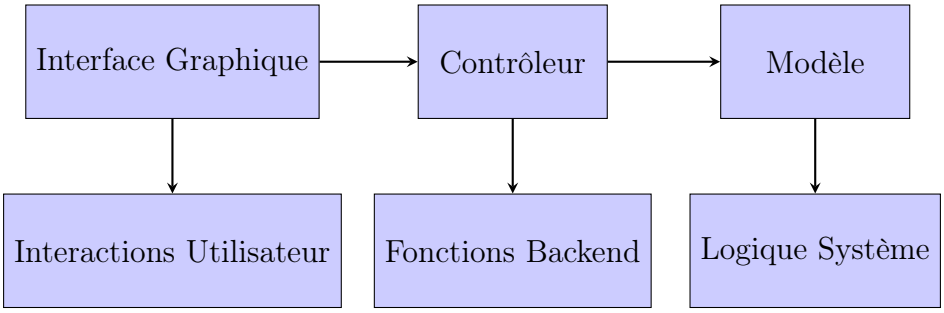


FIGURE 1 – Diagramme des relations entre les composants MVC

Pour le Modèle, deux groupes de classes ont été définis.

3.1 Gestion des utilisateurs

La classe **Person** gère les informations des clients ainsi que leurs réservations. Elle permet l'ajout, la suppression et la consultation de réservations, ainsi que la sauvegarde dans un fichier JSON.

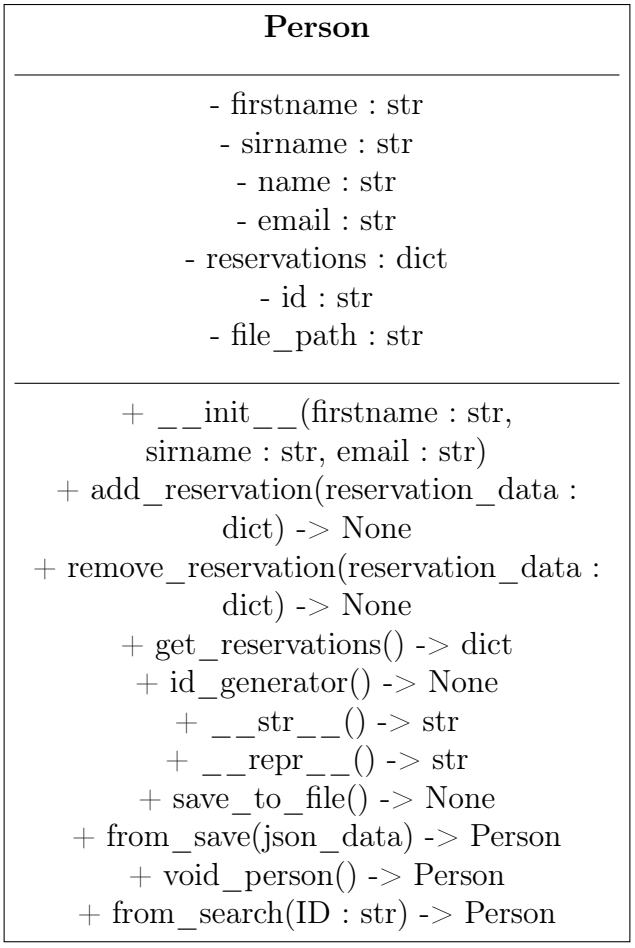


FIGURE 2 – Diagramme de gestion des utilisateurs

3.2 Gestion des salles

La classe **Standard** définit les fonctionnalités communes à toutes les salles.
Deux classes héritant de **Standard** spécialisent son comportement pour l'adapter aux types de salles également disponibles à la location : **Conference** vérifie que la salle n'est pas surdimensionnée pour l'événement, et **ComputerRoom** liste les équipements informa- tiques disponibles dans ce type de salle.
La représentation de ces classes et de leurs relations est la suivante :

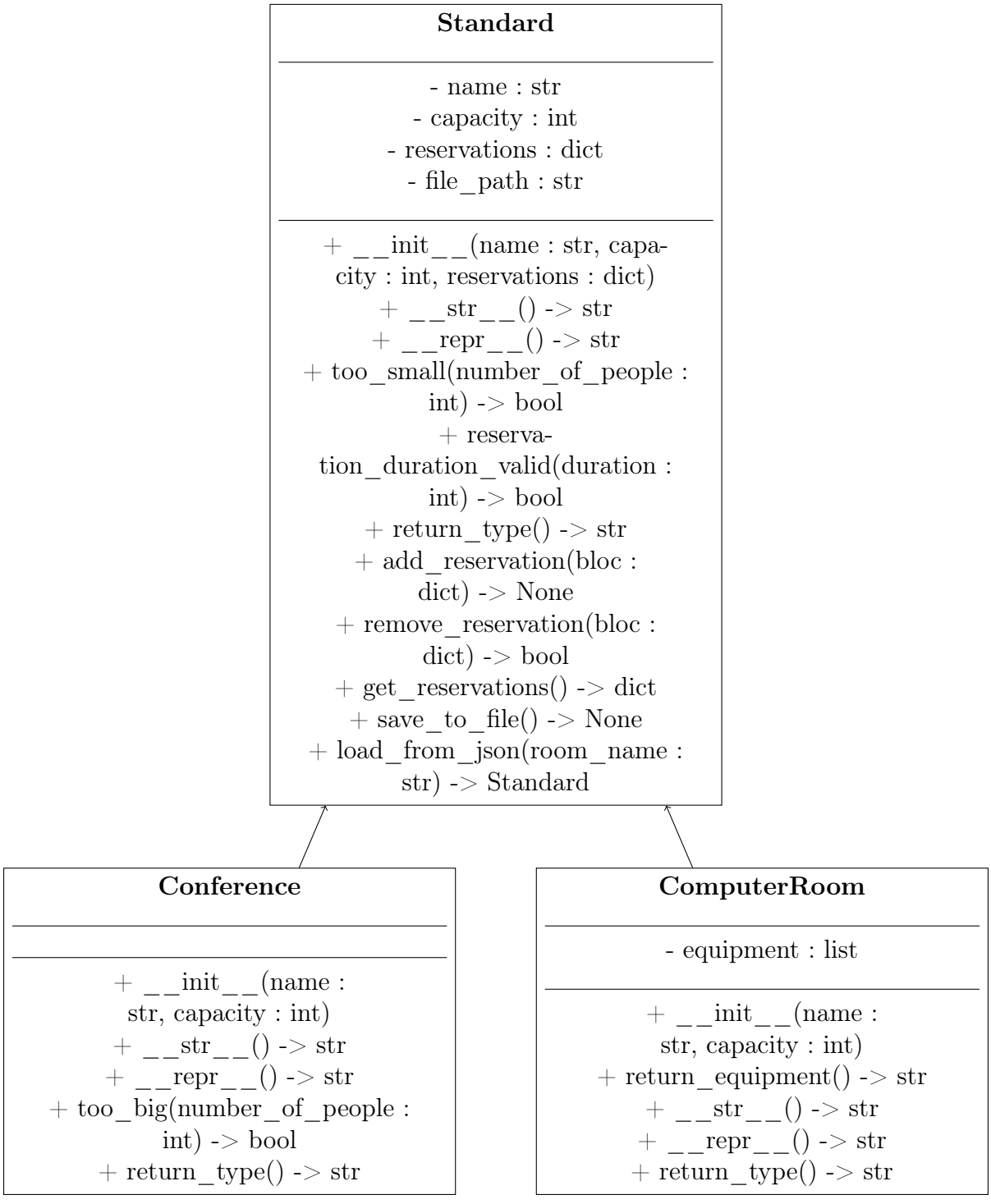
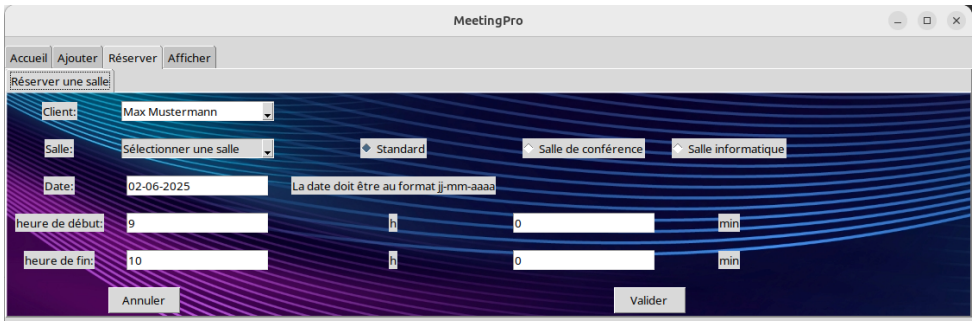


FIGURE 3 – Diagramme de gestion des salles

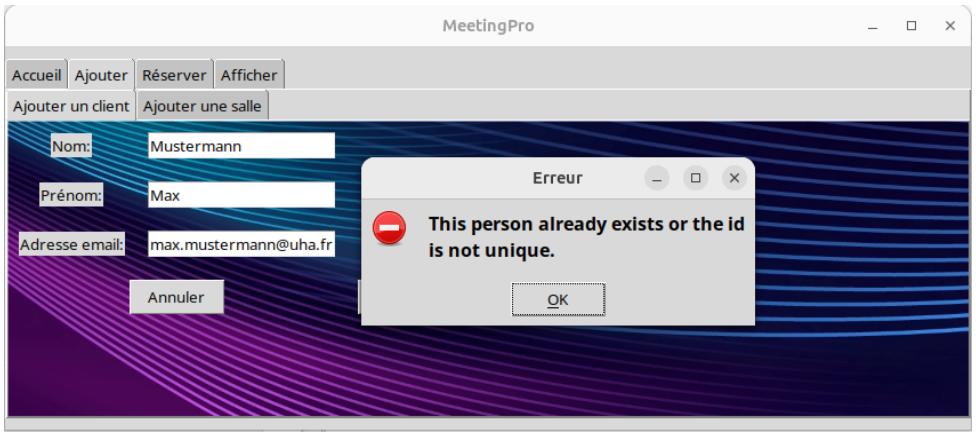
4 Résultats obtenus

L'application MeetingPro permet les fonctionnalités suivantes :
— Création, modification et suppression d'utilisateurs,
— Ajout et gestion de réservations de salles,
— Affichage dynamique des informations (utilisateurs, salles, réservations),
— Sauvegarde automatique dans des fichiers JSON distincts,
— Interface graphique interactive respectant les exigences fonctionnelles.

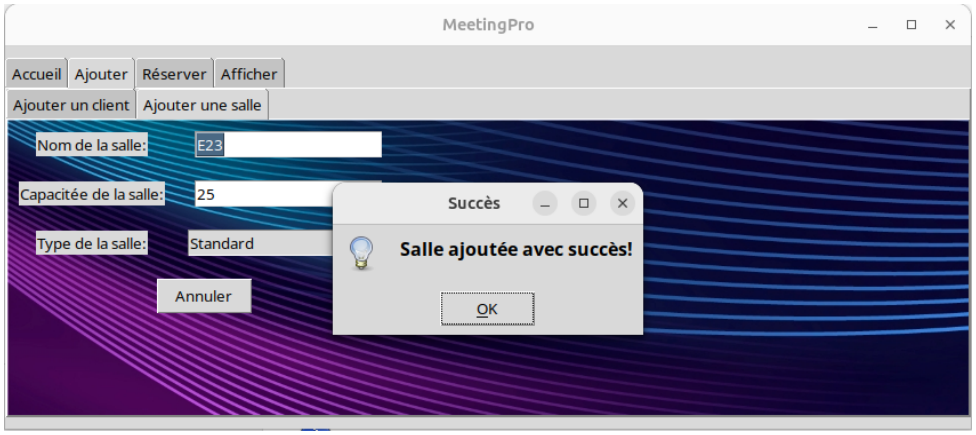
Aperçu de l'application (extraits d'écran) :



(a) Écran de réservation



(b) Erreur si l'on tente d'ajouter à nouveau un même client



(c) Information de réussite d'ajout de salle

FIGURE 4 – Captures d'écran de l'application MeetingPro

5 Difficultés rencontrées

Le projet a soulevé plusieurs défis :

— **Apprentissage de Tkinter et JSON** : la documentation étant parfois limitée, la courbe d'apprentissage a été marquée, notamment pour lier efficacement l'interface aux données.

— **Interface dynamique** : adapter les widgets Tkinter à des modifications en temps réel fut complexe.

— **Changement d'UI en cours de projet** : une nouvelle directive imposée par nos enseignants nous a contraints à revoir intégralement l'approche de l'interface, entraînant un travail de refonte important.

— **Adaptation du contrôleur** : l'interfaçage avec l'UI a nécessité des ajustements non anticipés malgré une modélisation initiale rigoureuse.

6 Perspectives

Nous avions prévu d'ajouter un **script de compilation** pour générer des exécutables utilisables sous Linux et Windows. Faute de temps, cette fonctionnalité n'a pas pu être intégrée, mais pourrait constituer une évolution future du projet.

Références

Références

[1] Philipp Acsany. « Working With JSON Data in Python ». In : *Real Python* (déc. 2024). URL : https://realpython.com/python-json.

[2] L'Équipe Alphorm et L'Équipe Alphorm. « Maı̂ triser la manipulation JSON en Python ». In : *Blog Alphorm* (jan. 2025). URL : https://blog.alphorm.com/maitriser-la-manipulation-json-en-python.

[3] *Graphical User Interfaces with Tk*. [Online ; accessed 19. May 2025]. Mai 2025. URL : https://docs.python.org/3/library/tk.html.

[4] *json — JSON encoder and decoder*. [Online ; accessed 19. May 2025]. Mai 2025. URL : https://docs.python.org/3/library/json.html.

[5] *SHA-256 - Hash SHA256 Mot de Passe - Décoder, Encoder en Ligne*. [Online ; accessed 19. May 2025]. Mai 2025. URL : https://www.dcode.fr/hash-sha256.

[6] *Stack Overflow*. [Online ; accessed 19. May 2025]. Mai 2025. URL : https://stackoverflow.com/questions/43562401/using-hashlib-sha256-to-create-a-unique-id-is-this-guaranteed-to-be-unique.

[7] *Stack Overflow*. [Online ; accessed 19. May 2025]. Mai 2025. URL : https://stackoverflow.com/questions/16129652/accessing-json-elements.

[8] *Stack Overflow*. [Online ; accessed 19. May 2025]. Mai 2025. URL : https://stackoverflow.com/questions/62527331/what-does-hexdigest-do-in-python.

[9] *W3Schools.com*. [Online ; accessed 19. May 2025]. Mai 2025. URL : https://www.w3schools.com/python/python_json.asp.