ECOLE SUPERIEURE
D'INGENIEURS DE RENNES

Université de Rennes

Rapport d’architecture

Equipe-projet :

Noam GEFFROY
Roland KOFFI
Moïra PERROT
Guillaume PINAULT

Mai 2024

Sommaire

1. Principe de mise en oeuvre de la solution

La mise en œuvre de la solution passe par plusieurs étapes. Notamment l'analyse du projet, la conception de la solution, le choix des technologies, l'implémentation et le déploiement.

Les fonctionnalités qui seront mises en oeuvre dans notre logiciel pour la création de trombinoscopes adaptés aux besoins des utilisateurs sont décrites dans le diagramme de cas d'utilisation ci-dessous :

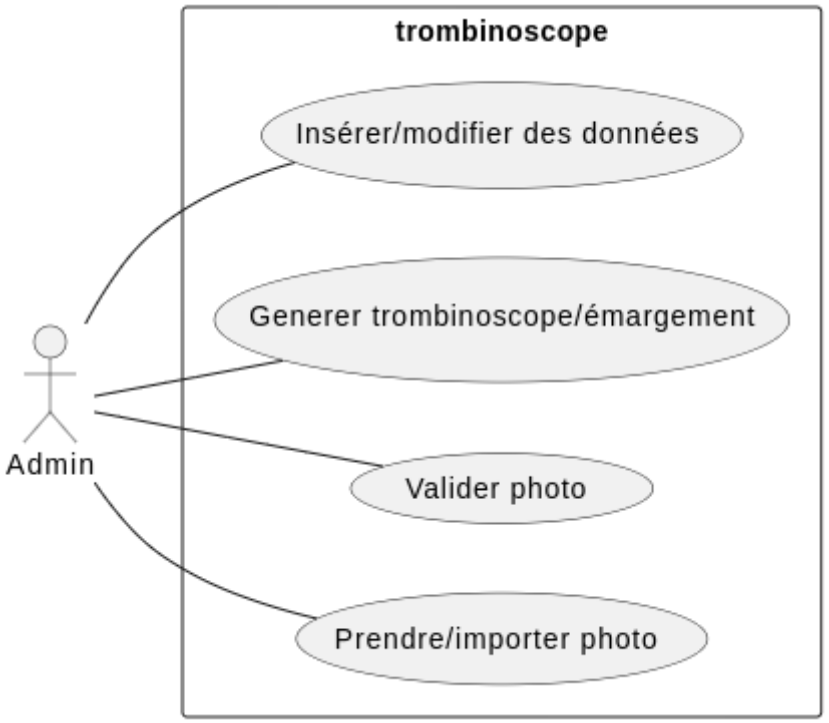

Figure 1 : Diagramme des cas d'utilisation

Notre solution a beaucoup évolué tout au long du semestre. Dans notre version finale, nous n'avons plus d'interaction qu'avec un seul acteur : une personne de l'administration. Nous avons supprimé la possibilité de se connecter à l'application pour en simplifier grandement le fonctionnement et l'implémentation. En effet, les séances de prise de photo se font toujours avec des personnes de l'administration, il n'est donc pas pertinent de permettre aux étudiant.e.s d'utiliser l'application dans une version sans connexion. La sécurisation des données se fait donc grâce à la nécessité de se connecter au réseau interne de l'université ainsi que de posséder les identifiants de connexion à la base de données MariaDB.

Dans notre version actuelle, il est donc possible de prendre et importer des photos, les associer aux étudiant.e.s grâce à leur adresse email universitaire. Il est possible de générer les trombinoscopes ainsi que les fiches d'émargement en PDF en fonction des conditions choisies (groupes, niveaux, etc.). Il est possible de choisir de lister ou non les adresses mail étudiantes à la fin des trombinoscopes. Depuis l'application, il est possible d'importer un fichier XLSX pour mettre à jour la base de données. Il est aussi possible de

modifier des données à la main directement sur l'application de façon plus ou moins précise (le nom d'une seule personne, l'année d'étude de tous les ESIR2, etc.).

2. Règles d'architecture

Le développement d'un projet informatique doit se faire dans le respect de plusieurs règles d'architecture.

Le choix de technologie ayant porté sur Java, le projet doit respecter le paradigme de la programmation orientée objet. Ainsi, la structuration du code devrait être basée sur la définition de classes et d'objets, favorisant l'encapsulation, l'héritage et le polymorphisme. De plus, il est essentiel de suivre les bonnes pratiques de programmation Java, telles que la gestion appropriée des exceptions, l'utilisation judicieuse des interfaces et la mise en œuvre de méthodes efficaces.

Tout bon projet informatique se doit de respecter les cinq principes de conception SOLID. Ces principes, acronyme de Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation et Dependency Inversion, fournissent des directives pour la création de logiciels modulaires et extensibles. Respecter ces principes contribue à éviter les dépendances excessives entre les différentes parties du code, à faciliter la maintenance et les évolutions, ainsi qu'à rendre le système plus robuste et évolutif. En intégrant ces principes dès les premières phases du développement, l'équipe peut garantir une architecture solide et durable pour le projet informatique, favorisant ainsi la qualité et la pérennité du logiciel final.

C'est avec ces différentes règles à l'esprit, que nous avons construit le diagramme de composants du projet :

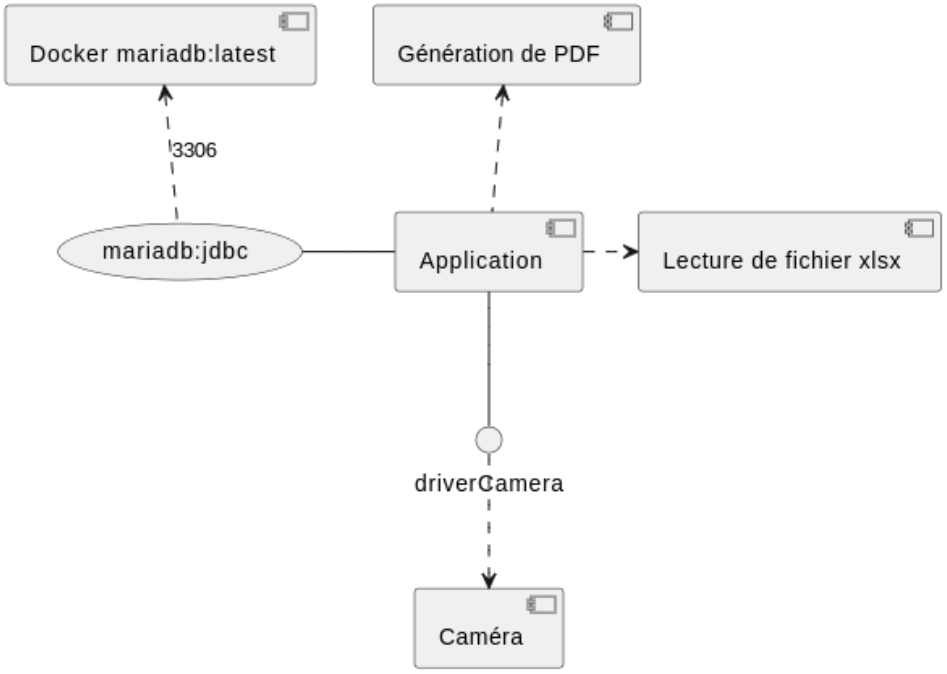

Figure 2 : Diagramme de composants du projet

Chacun des rectangles de ce schéma est un composant indépendant du logiciel. Une flèche en pointillés pointe vers un composant utilisé par celui à l'origine de la flèche. Les cercles sont des interfaces.

Le corps du projet est une application du bureau. Elle interagit avec une base de données distante hébergée sur une machine virtuelle de l'ISTIC et accessible par un tunnel SSH intégré directement dans l'application, la liste des paramètres requis pour cette connexion se trouve dans un fichier de configuration externe au programme, ce qui permet de changer de lieu de base de données sans reconstruire le logiciel. L'application se base sur la lecture de fichiers XLSX pour ajouter et mettre à jour des données. Elle stocke les photos des étudiant.e.s sur la base de données. Elle génère des fichiers PDF et des fiches d'émargement sur la machine l'hébergeant, à la racine du projet (donc dans le même dossier que le fichier .exe).

En pratique, en dehors de la caméra en elle-même, de la base de données et des fichiers de photographies, toutes les composantes du projet seront contenues dans une application monolithe. En effet, l'utilisation de microservices n'est pas pertinente pour ce projet. L'objectif est de développer un logiciel de bureau, répondant à des besoins simples et définis. Cette architecture pourrait être considérée comme un système d'information d'un style ancien, mais elle est bien plus adaptée à l'envergure du projet. Diviser le projet ne ferait que l'alourdir, impliquer des technologies supplémentaires, des besoins de sécurité supplémentaires, une installation plus compliquée pour des non informaticiens et davantage de travail pour un bénéfice nul.

3. Modèle statique

Le modèle statique du projet est dépeint dans le diagramme de classes suivant :

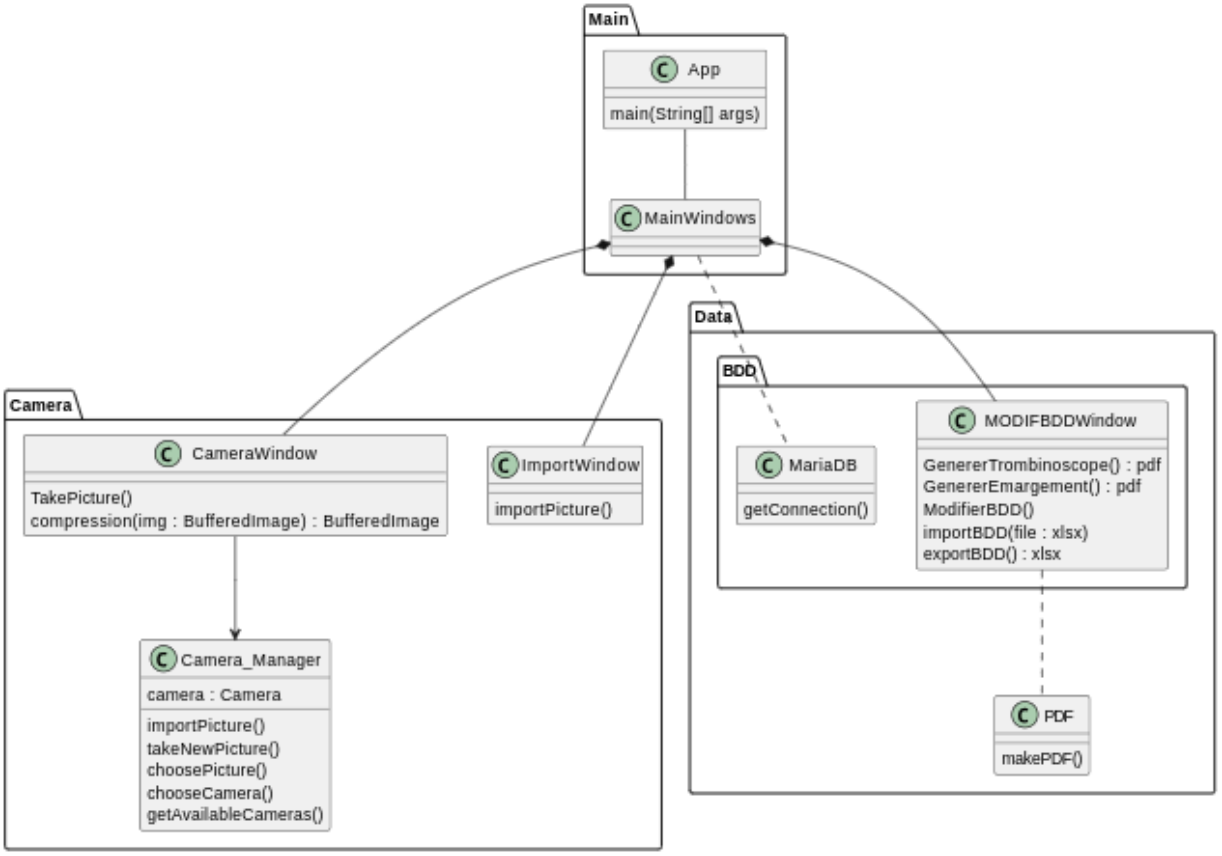

Figure 3 : Diagramme de classes

Différents packages y sont définis :

- Le package Main permet de relier et gérer les différents packages du projet, ainsi que de le lancer.
- Le package Data consiste en la gestion des données sous format XLSX, la communication avec la base de données MariaDB et la génération de PDF.
- Le package Camera permet de gérer la caméra utilisée, son affichage, l'import de nouvelles images et la compression.

Chaque package contient une ou plusieurs classes.

<u>Classes et responsabilités :</u>

- Caméra
    - CameraManager : gestion de la caméra active, du flux vidéo et des photographies.
    - CameraWindow : page d'affichage et de sélection de la photo de l'étudiant.
    - ImportWindow : page d'import d'une image externe pour un étudiant.
- Data
    - MariaDB : gestion de la base de données, création/importation BDD, mise à jour, mais aussi gestion des requêtes et création de ces requêtes.

    - ModifBDDWindow : interface ou l’on peut envoyer des nouvelles informations, ou importer/exporter la base de données.
    - PDF : production des différents documents en sortie du programme.
- Main
    - Application: lance l’interface et ferme tout le programme à sa fermeture.
    - MainWindows: gestion de l’interface de notre logiciel.

Le détail de la gestion des caméras, de la connexion et des requêtes en base de données n’est pas représenté ici, car il repose sur les dépendances du projet (décrites dans la partie cadrage de production).

4. Modèle dynamique

Les diagrammes de séquence de notre logiciel viennent détailler la réalisation des cas d’utilisation.
Dans ces diagrammes, le rôle d’administrateur sont les membres de l'administration de l’ESIR ayant les droits de connexion.
Les flux des événements sont présentés par les diagrammes de séquence suivants :

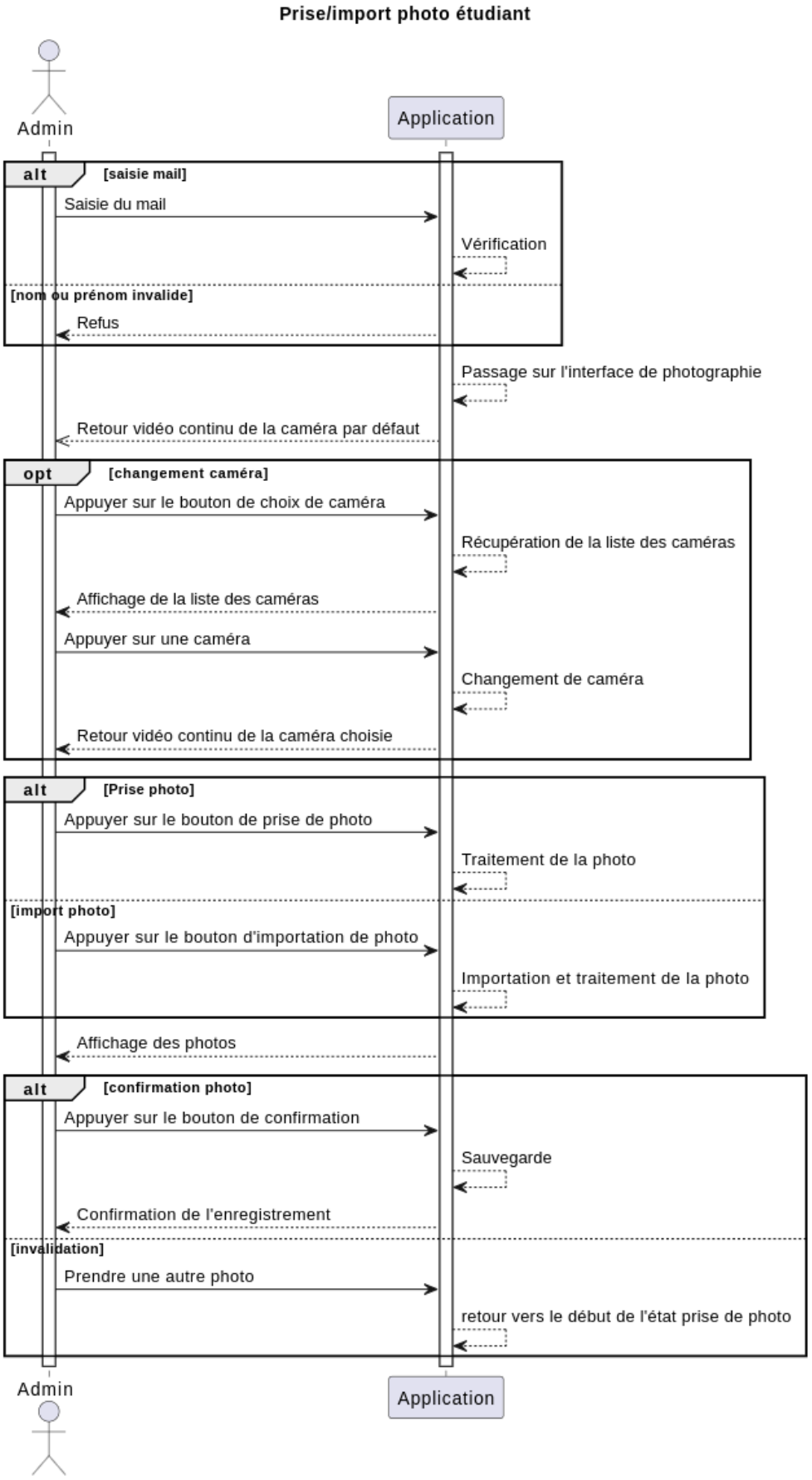

Figure 5 : Diagramme de séquence “Prise de photo”

Il y a 2 interfaces soeurs, la première liée à la caméra et permettant plusieurs essais, la seconde affichant simplement la photo importer. Les 2 permettent de voir les traitements mis en place comme le recadrage ou la compression. Une fois cela fait il suffit de rentrer le mail et l'étudiant et si il existe la photo lui sera attribué.

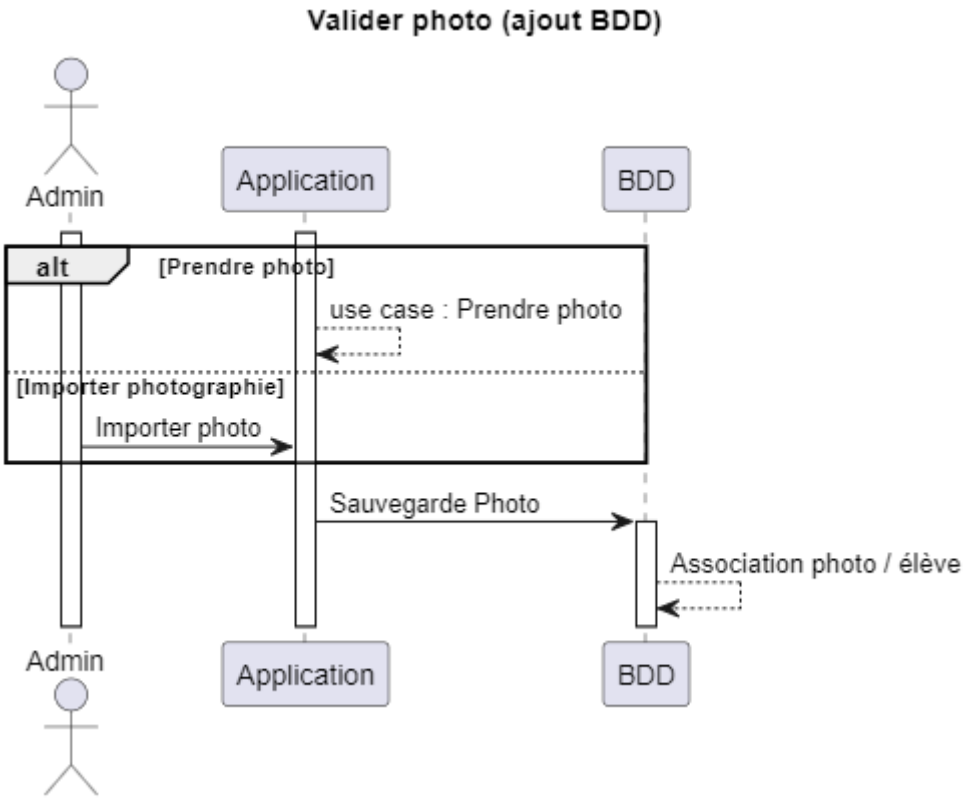

Figure 6 : Diagramme de séquence "Valider photo (ajout BDD)"

Les photographies prises par des étudiants doivent être validées par l'administrateur afin d'être disponibles dans les trombinoscopes. Cela permet d'éviter d'avoir des photographies indésirables utilisées.

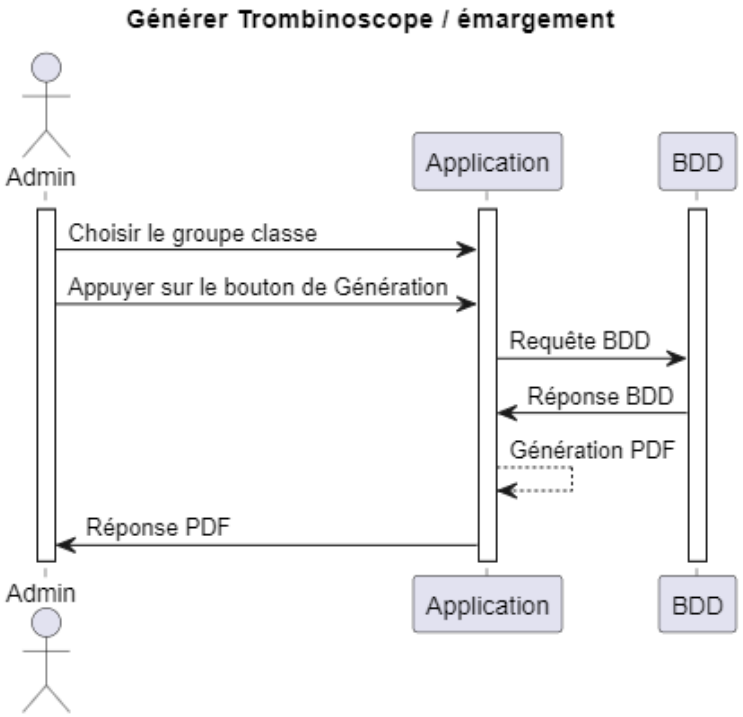

Figure 7 : Diagramme de séquence “Générer Trombinoscope”

Un administrateur peut générer un trombinoscope ou une feuille d'émargement à partir des informations en base de données et des photographies enregistrées.

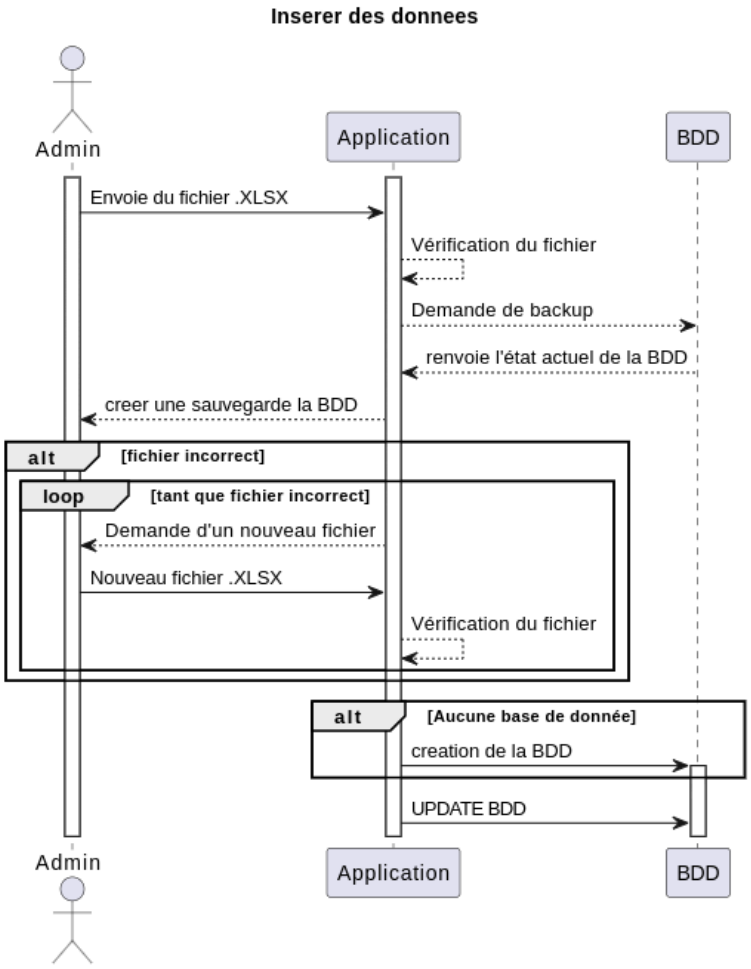

Figure 8 : Diagramme de séquence “Insérer des données”

En important un fichier XLSX, un administrateur peut mettre à jour les informations contenues en base de données. Avant toute modification, l’application génère une sauvegarde de l’état de la base de données. Des étudiants dont le mail n’existe pas encore en base seront créés, tandis que s’il existe déjà, ils seront mis à jour en cas de différence avec les informations en base de données.
L’utilisation du mail pour savoir si un étudiant est déjà présent dans la base de données permet de gérer les homonymes.

5. Prise en compte de contraintes

Une des contraintes majeures de notre projet concerne les machines sur lesquelles notre logiciel doit être déployé. Ces machines sont des ordinateurs de bureau utilisés par l'administration, et ne sont pas équipés de nombreuses dépendances logicielles. Il est donc crucial de toutes les fournir. Étant donné que les ordinateurs sont équipés de Java, nous avons décidé de programmer notre application en Java. De plus, c'est le langage avec lequel nous sommes le plus à l'aise et que nous connaissons le mieux.

Une autre contrainte concerne la supervision du processus de prise de photographies par les élèves. Afin d'éviter tout accès non autorisé à la base de données, le logiciel sera supervisé par un administrateur pendant la prise de photographies par les élèves.

Un défi supplémentaire concerne la possibilité que plusieurs élèves aient le même nom et prénom, ce qui pourrait entraîner des doublons dans la base de données. Cette problématique d'homonymie pourrait entraver les requêtes. Après discussion avec le client, il a été décidé que l'identifiant unique est l'adresse e-mail des étudiants générée en amont et unique, permettant donc d'éviter les doublons.

Notre application devant être simplement utilisable par des non informaticiens sur des ordinateurs Windows, nous la livrerons donc sous forme d'un exécutable .exe. Celui-ci sera disponible sur le dépôt Github et sera mis à jour lors de chaque build du projet.

Le respect des règles du RGPD (Règlement Général sur la Protection des Données) constitue une autre contrainte importante. Les photographies, noms et prénoms des étudiants représentent des données personnelles et leur traitement est réglementé. Bien que nous ne soyons pas responsables de la gestion des droits à l'image des étudiants, nous devons protéger la base de données pour limiter l'accès aux seules personnes autorisées. Nous avons donc protégé la base de données en limitant l'accès aux seules personnes connectées à eduroam. De plus, nous avons ajouté un fichier de configuration qui contient les mots de passe et l'adresse de la base de données, informations qui ne seront pas publiques, avec un faux fichier de configuration publié sur le dépôt Github.

6. Cadrage de production

Pour le cadre de production, nous avons plusieurs outils de développement : nos machines personnelles ainsi que plusieurs IDE notamment VSCode et IntelliJ.

Nous avons pu réaliser une veille technologique pour choisir comment implémenter notre modèle. La première décision technologique a été l'utilisation du langage Java. Par son utilisation de la JVM, le programme sera facilement déployable sur différentes machines. Pour le stockage des données, nous avons pris la décision d'utiliser du SQL plutôt qu'une concaténation de fichiers XLSX. En effet, cela permet une meilleure indexation des données et un requêtage plus optimisé. Dans le souci d'accéder à distance à la base de données et depuis plusieurs ordinateurs, nous sommes passés d'une base de données SQLite à MariaDB. Elle est hébergée sur une machine virtuelle de l'ISTIC, ce qui permet un accès depuis plusieurs ordinateurs et garantit la protection des données des étudiants.

A. Génération de PDF

La technologie utilisée pour la génération de PDF est une bibliothèque nommée iText qui fournit des méthodes pour créer et manipuler les documents PDF. Nous l'avons choisie car elle est open source, utilisée par un grand public (plus de 3000 utilisateurs) et assez appréciée (1700 étoiles). De plus, elle est régulièrement mise à jour. La dernière version date du 30 avril 2024.

B. Base de données et XLSX

Pour la lecture du fichier .xlsx d'entrée, nous utilisons Apache POI, une API Java dédiée aux documents Microsoft. Il s'agit d'une bibliothèque officielle et sécurisée, développée par Apache. Pour exécuter les requêtes générées, nous avons en premier utilisé la bibliothèque SQLite-jdbc. Nous l'avions choisie en raison de sa reconnaissance au sein de la communauté (2,6k étoiles). De plus, la version la plus récente est datée du 16 avril 2024. La documentation est également bien conçue, facilitant la compréhension du fonctionnement de la bibliothèque.

Nous avons, lors du sprint 3, migré la base de données locale en SQLite vers une base de données distante en MariaDB, hébergée sur les serveurs de l'ISTIC. La connexion avec cette base de données se fait à l'aide de MariaDB JDBC (dont la dernière release date du 17 mai 2024). On l'initialise à l'aide d'un fichier .xlsx fourni, action que l'on peut répéter par la suite si nécessaire, pour réinitialiser la base de données en ajoutant par exemple des colonnes si besoin. On peut également exporter la base de données actuelle dans un fichier .xlsx.

C. Caméra

Pour la caméra, nous avons choisi d'utiliser Webcam Capture API. Son avantage est d'être une interface supportant plusieurs frameworks de capture. Nous pourrons donc en changer sans que cela n'affecte notre code (nous utilisons celui par défaut pour le moment). Après l'avoir testée, nous avons découvert que la même librairie était utilisée dans le logiciel actuellement utilisé pour la génération des trombinoscopes, témoignant de son efficacité. Nous avons d'ailleurs découvert un bug négligeable (peu probable sur les machines de l'administration) lors de la sélection d'une caméra virtuelle qui fait planter les logiciels utilisant la librairie. Nous avons donc pris la décision de bloquer l'accès aux caméras virtuelles (non utiles pour notre projet).

D. Interface

Pour l'interface, nous avons d'abord utilisé la bibliothèque QtJambi, qui est une liaison Java du framework d'application Qt et qui permet de créer des interfaces plus complexes. Cependant, lorsque nous avons essayé de créer un exécutable, nous nous sommes rendus compte qu'il était quasiment impossible d'intégrer QtJambi à l'application, et qu'il aurait fallu l'installer et la configurer sur tous les ordinateurs avant de lancer l'application. Nous avons donc basculé sur JavaFX, une autre librairie graphique directement intégrée à l'exécutable, pour que le programme soit utilisable sur n'importe quelle machine, dès l'installation.