ECOLE SUPERIEURE
D'INGENIEURS DE RENNES

Université de Rennes

Rapport d’architecture

Equipe-projet :

Baptiste AMICE
Noam GEFFROY
Roland KOFFI
Moïra PERROT
Guillaume PINAULT
Théo VINCENT

Janvier 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 et l'implémentation.

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 des cas d'utilisation ci-dessous :

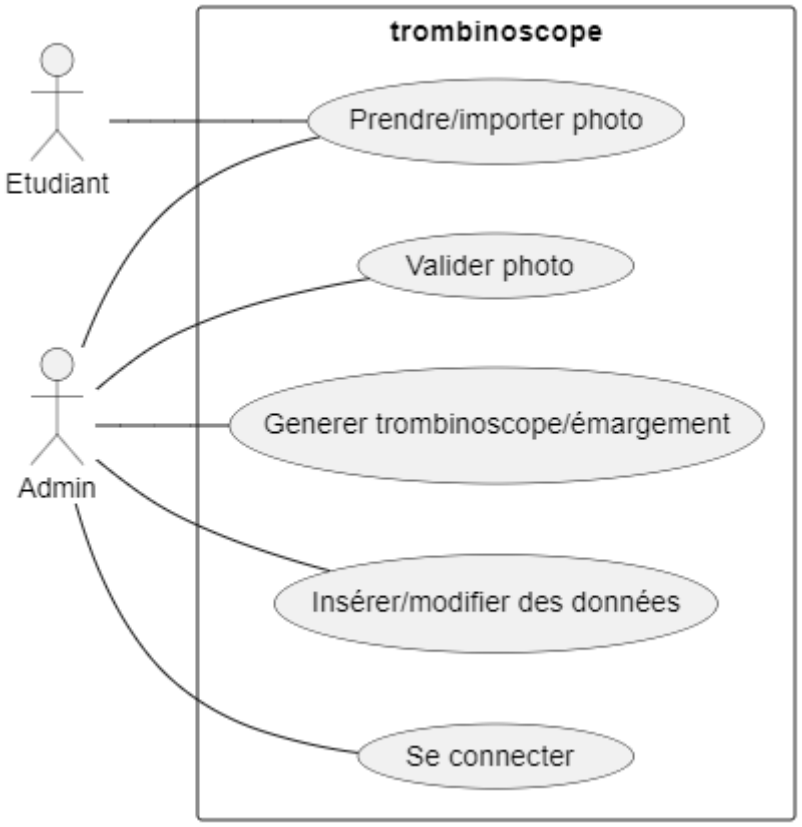

Figure 1 : Diagramme des cas d'utilisation

Deux types d'acteurs pourront interagir avec le logiciel. Un utilisateur quelconque (étudiant) pourra, sans se connecter, prendre ou importer des photographies. Elles devront toutefois être validées par un administrateur avant d'être ajoutées à la base de données. Un administrateur pourra également générer un trombinoscope ou une feuille d'émargement en fonction de paramètres choisis (groupes, niveaux, etc.). Un administrateur pourra modifier et insérer de nouvelles données par l'import d'un fichier XLSX.

Pour accéder aux fonctionnalités d'administration, un utilisateur devra se connecter. La fonctionnalité de connexion ne sera vraiment poussée que dans la deuxième version du logiciel. En effet, la base de donnée étant, dans la première version, localisée sur la machine, la sécurité de cette version en est donc par nature amoindrie et facilement contournable.

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 :

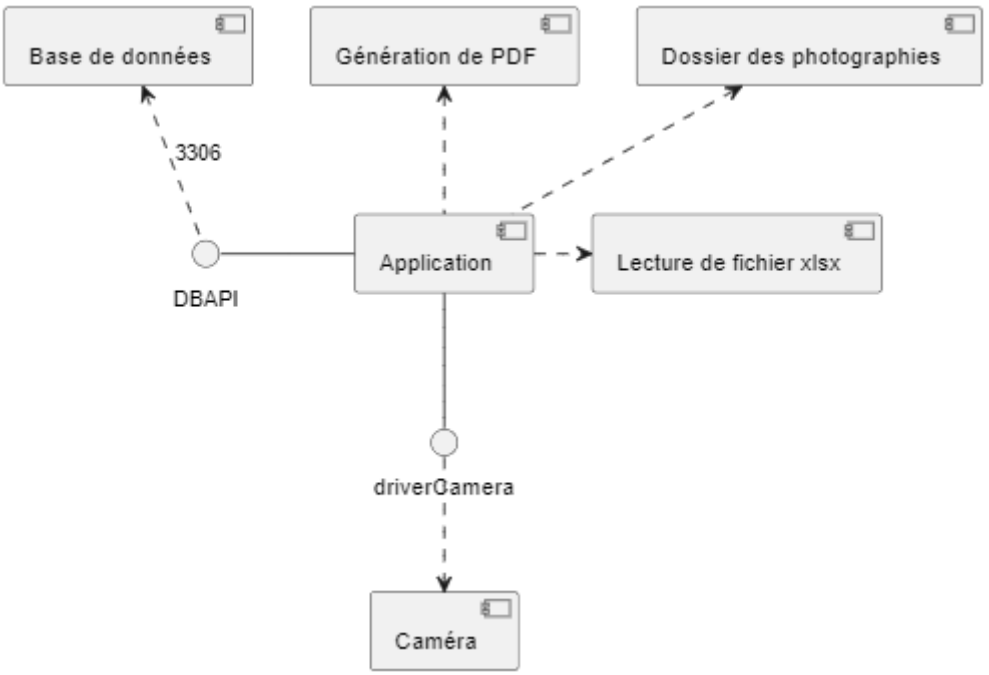

Figure 2 : Diagramme de composants du projet

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 (pour le moment en locale, mais à terme la communication se fera sur le port 3306). Elle se base sur le driver de caméra configuré par Webcam Capture API pour

communiquer avec les différentes caméras de la machine l'hébergeant. L'application se base sur la lecture de fichiers XLSX pour mettre à jour ses données. Elle stocke les photographies des élèves dans un dossier (pour le moment en local, mais une portabilité de l'application laisse supposer de les partager en drive, par exemple sur Microsoft SharePoint). Elle génère des fichiers PDF sur la machine l'hébergeant.

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 :

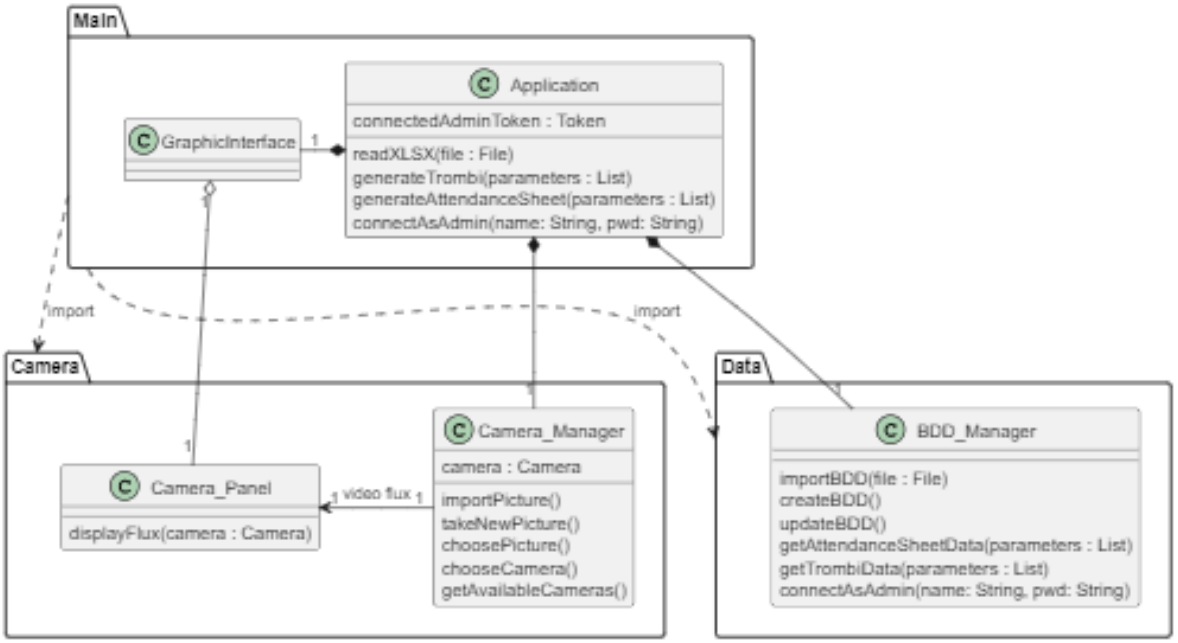

Figure 3 : Diagramme de classes

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 et de base de données SQLite.
- Le package Camera permet de gérer la caméra utilisée et son affichage.

Chaque package contient une ou plusieurs classes.

Classes et responsabilités:

- Caméra
    - CameraManager : gestion de la caméra active, du flux vidéo et des photographies
    - CameraPanel: Gestion de l'affichage du flux caméra.
- Data
    - BDD_Manager: Gestion de la base de données. création/importation, mise à jour, mais aussi gestion des requêtes.
- Main
    - Application: Lie toutes les fonctions entre elles, classe principale et entrée dans le programme.
    - Graphicinterface: 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 ici pas représenté, car il repose sur les dépendances du projet (décrites dans la partie cadrage de production).

4. Modèle dynamique

Les diagrammes de séquences de notre logiciel viennent détailler la réalisation des cas d'utilisation.
Dans ces diagrammes, le rôle d'étudiant correspond à tout élève ayant accès à l'application, tandis que les administrateurs 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 suivant :

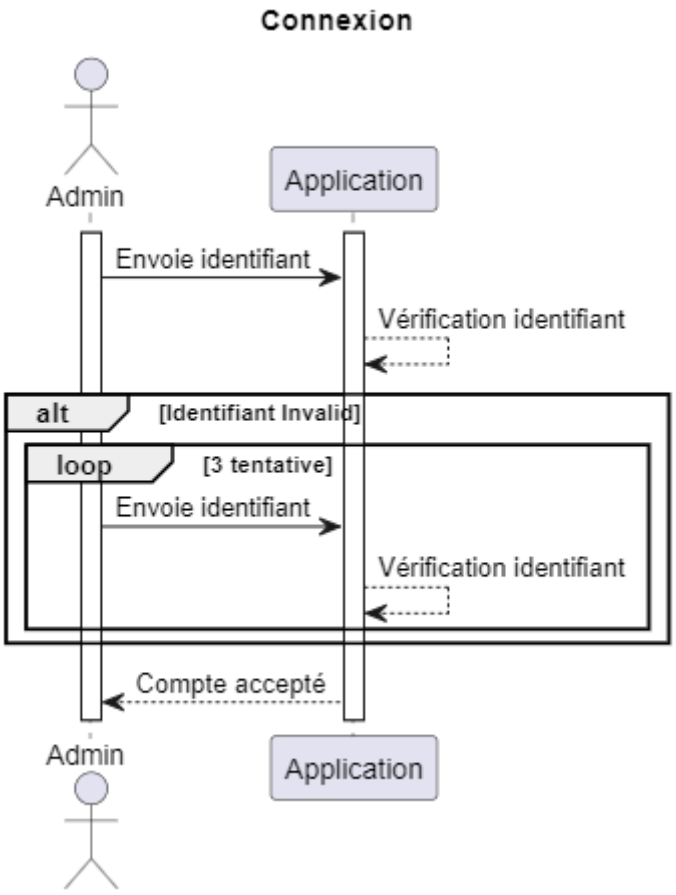

Figure 4 : Diagramme de séquence “Connexion”

Pour la connexion d’un administrateur, il doit saisir son identifiant et mot de passe. L'application vérifie alors si ils sont corrects. S’ils sont valides, une session est créée pour l’utilisateur. En cas de saisie invalide, l’utilisateur peut réessayer plusieurs fois la saisie.

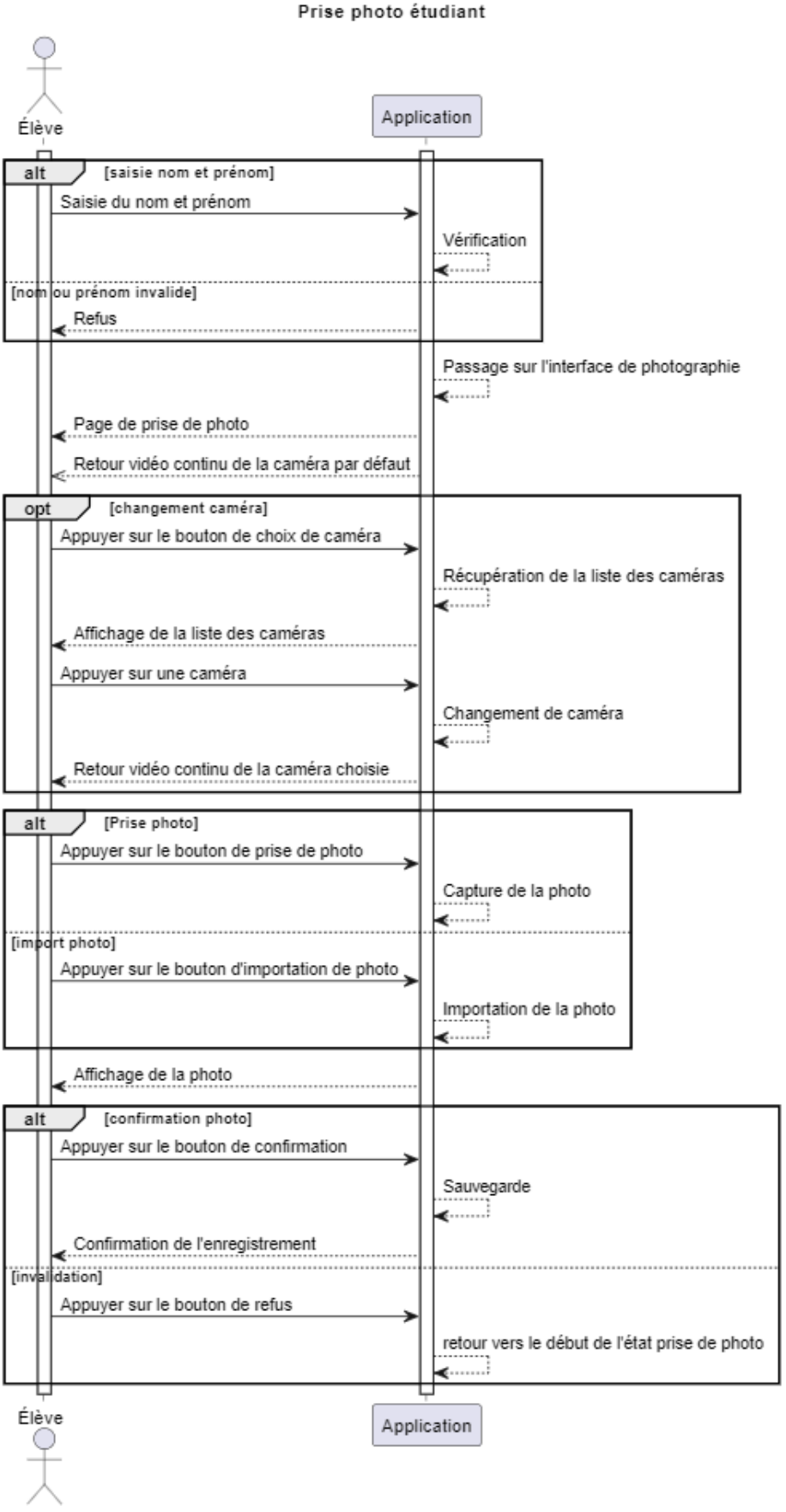

Figure 5 : Diagramme de séquence “Prise de photo”

Pour l’ajout d’une photographie, il faut tout d’abord sélectionner l’étudiant qu’elle concerne. On arrive ensuite sur l’interface de prise de photographie qui affiche un flux vidéo continue de la caméra active. Un changement de caméra active est disponible sur cette page. L’étudiant peut se prendre en photographie ou importer une image existante. Après une photographie, il peut confirmer s’il souhaite la conserver ou en reprendre une nouvelle.

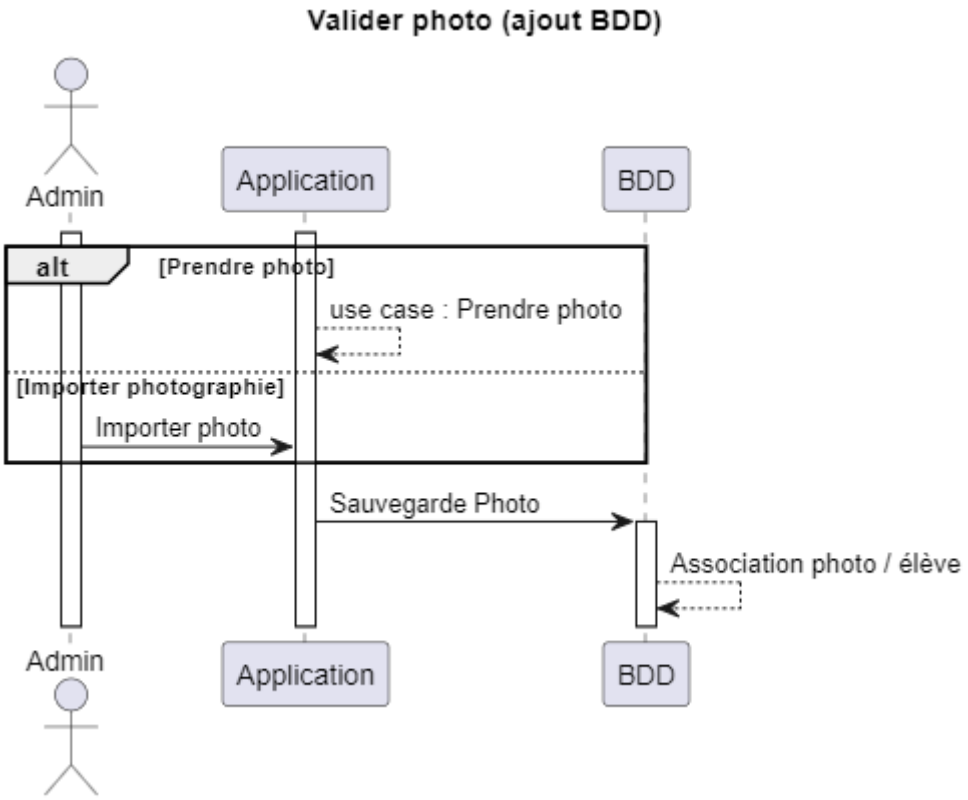

Figure 6 : Diagramme de séquence “Valider photo (ajout BDD)””

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.

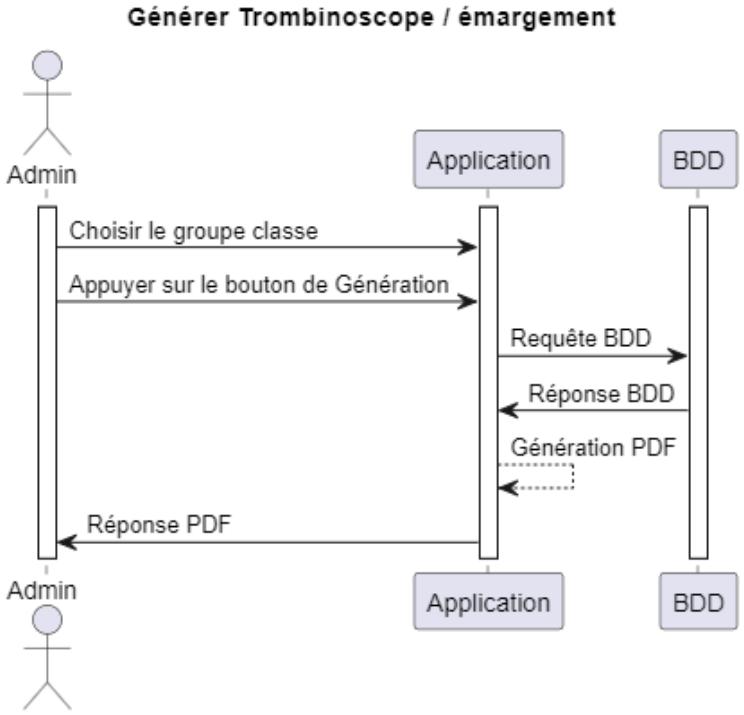

Figure 7 : Diagramme de séquence “Générer Trombinoscope”

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

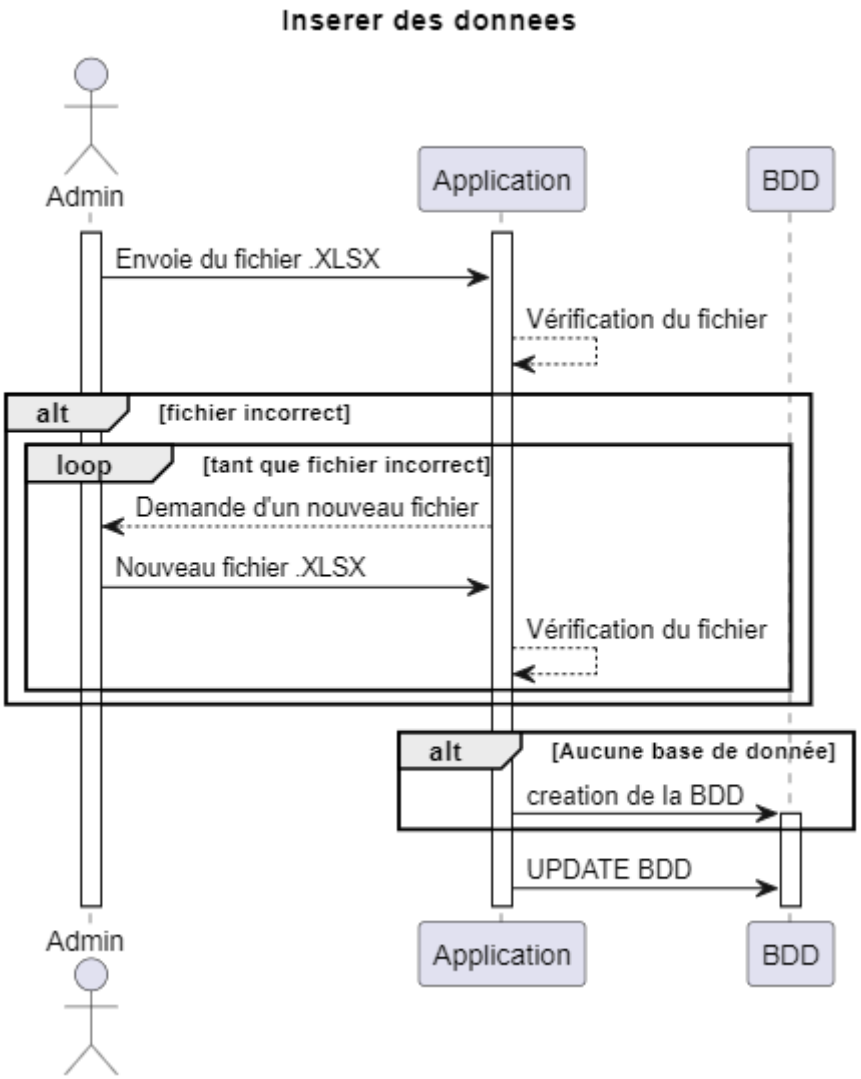

Figure 8 : Diagramme de séquence “Insérer des données”

En important un fichier XLSX, un administrateur peut mettre à jour les informations contenues en base de données. 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 en base permet de gérer les homonymes.

5. Design ergonomique

Nous avons réfléchi lors du premier sprint à la conception d'une maquette ergonomique permettant de localiser les différents éléments. Celle-ci n'a pas pour objectif d'être esthétique, étant donné que la charte graphique n'a pas encore été définie.

Cette maquette est divisée en trois parties, chacune représentant l'une des fonctionnalités principales de notre projet. Dans le but de simplifier la navigation pour l'utilisateur, nous utilisons une barre de navigation.

On a tout d'abord la page de prise de photo :

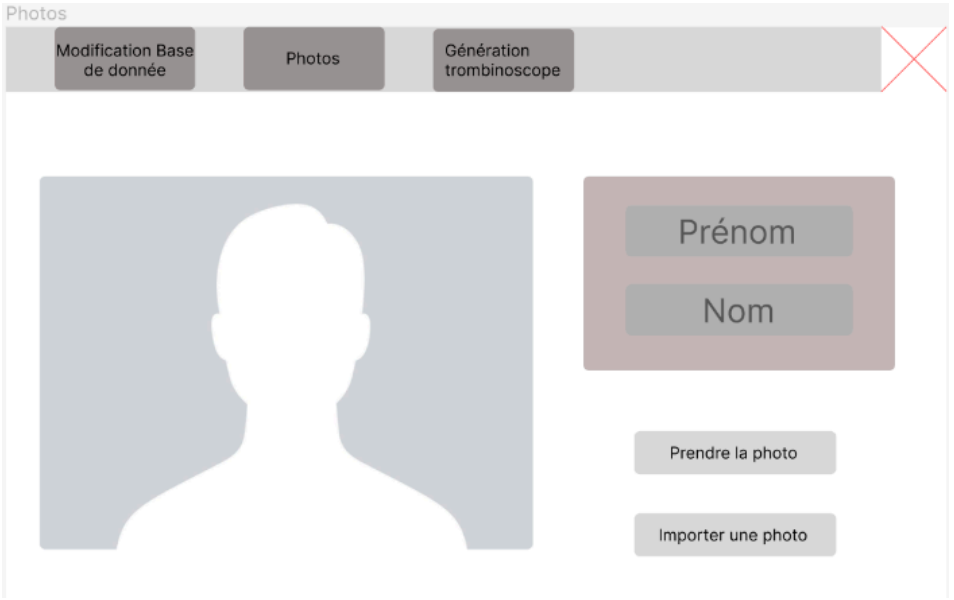

Figure 9 : Maquette de la page de prise de photo

La page où on peut importer un nouveau fichier XLSX, avec une fonction de glisser/déposer:

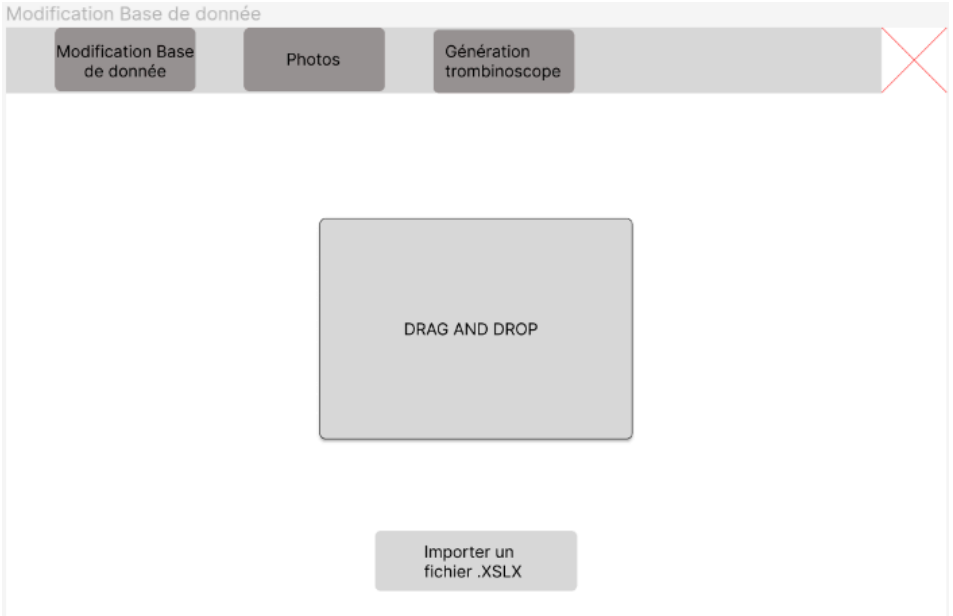

Figure 10 : Maquette de la page d'import de la base de donnée

Pour finir, nous avons la page de création de fichier PDF ou on peut choisir les paramètres de génération de pdf :

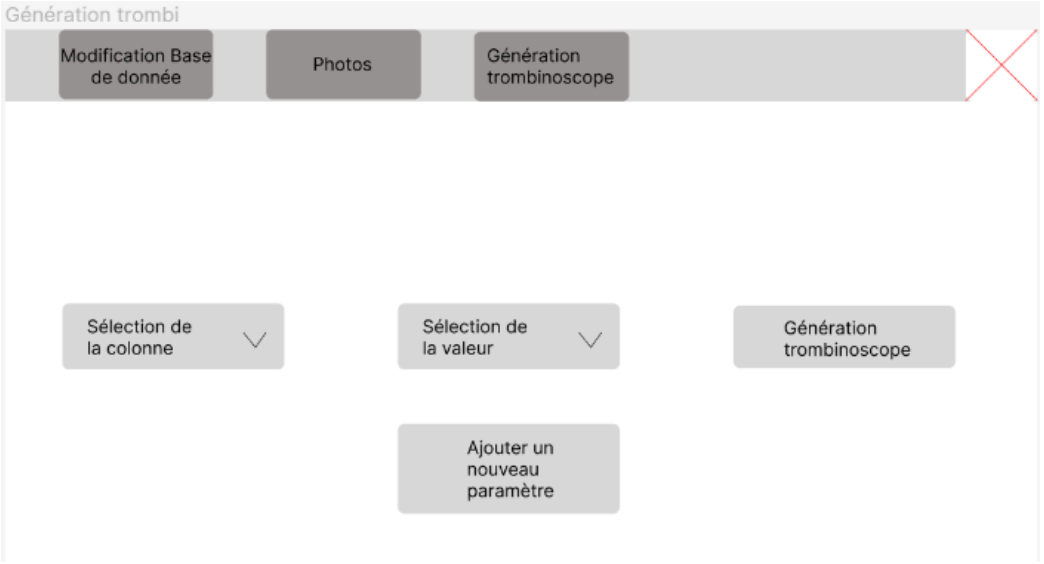

Figure 11 : Maquette de la page de génération de PDF

6. 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 elles ne sont pas équipées 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 utilisant un système d'exploitation Windows, nous la livrerons donc sous forme d'un exécutable .exe. Celui-ci sera disponible sur le github et sera mit 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. Ainsi, nous avons opté pour un logiciel sans accès à Internet, fonctionnant localement sur une machine sécurisée. De cette manière, la base de données est stockée localement, réduisant les risques de sécurité, en accord avec la pratique actuelle chez notre client où les données sont également stockées localement.

7. 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. En effet, nous avons déjà globalement défini les technologies et librairies utilisées. 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 SQL plutôt qu'une concaténation de fichiers XLSX. En effet, cela permet une meilleure indexation des données et un requêtage plus optimisé. L'utilisation de SQLite nous permet de nous contenter d'un fichier en local pour "héberger" la base de données.

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 11 janvier 2024.

B. Base de données et XLSX

Pour le prototypage, nous avons utilisé Apache POI, une API Java dédiée aux documents Microsoft. Il s'agit d'une bibliothèque officielle et sécurisée, développée par Apache. De plus, elle est régulièrement mise à jour, la dernière mise à jour datant du 25 novembre 2023, ce qui en fait un choix solide. Pour exécuter les requêtes générées, nous utilisons la bibliothèque SQLite-jdbc. Nous l'avons choisie en raison de sa reconnaissance au sein de la communauté (2,6k étoiles). De plus, la version la plus récente est datée du 27 novembre 2023. La documentation est également bien conçue, facilitant la compréhension du fonctionnement de la bibliothèque.

La technologie actuelle de manipulation de base de données utilise des requêtes SQL codées en dur dans l'application. Il s'agit d'une méthode peu sûre (nécessitant un assainissement des entrées) et peu pratique. Elle n'a été prise que pour la phase de prototypage, car nous ne sommes pas encore parvenus à choisir précisément la librairie ou le framework à utiliser pour la version finale. Nous avons décidé d'utiliser à terme une technologie de query builder (par exemple JOOQ).

Nous aurions également pu utiliser une technologie pour la persistance de mapping Objet-Relationnel en Java. Toutefois, cela nous a paru trop lourd en termes d'architecture par rapport aux données à stocker.

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 la première version de l'interface, nous avons choisi Swing afin de créer rapidement un prototype initial. Cependant, pour les versions ultérieures, nous envisageons de migrer vers une bibliothèque d'interface plus performante, telle que JavaFX. Cette dernière, qui est en open source, offre une richesse de fonctionnalités et utilise des fichiers FXML, CSS et Java.