Remise des diplômes - Rapport d'architechture

Février 2019

Sami Lazrak

Youness El idrissi

Walid Larabi

Jacques Rossel

Contents

1 Introduction

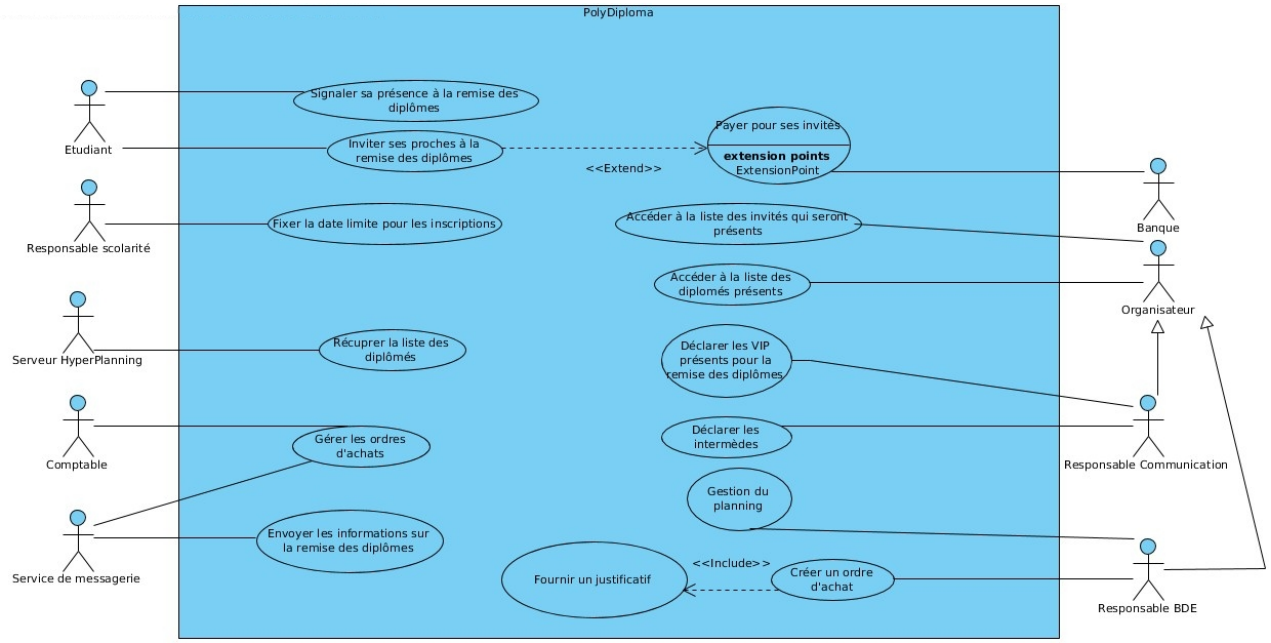
Face à la complexité de la gestion de la cérémonie de la remise des diplômes, l'école Polytech Nice-Sophia Antipolis nous a chargé de créer une application facilitant l'organisation de cette journée. Cette dernière est imaginée par client comme étant un outil qui centralisera l'organisation et automatisera les tâches lourdes et répetitifs. L'objectif étant de raccourcir le délais entre la fin des études et la remise des diplomes, minimiser les erreurs/incompréhensions entre acteurs de cette journée et enfin permettre aux acteurs de se focaliser sur des tâches plus importantes.

Suivant ces besoins, nous avons défini un produit qui répondra au mieux à ce qui est demandé. Nous allons vous présenter dans ce document les différentes fonctionnalités que fournira notre système, et ça à travers un diagramme des cas d'utilisations; puis l'organisation de notre système via le diagramme de composants; et enfin les objets fonctionnels de notre application via un diagramme de classes orienté métier.

2 Vue Fonctionnelle

2.1 Diagramme de cas d'utilisations

2.1.1 Diagramme haut niveau



2.1.2 Présentation des acteurs

Etudiant : Il s'agit d'un étudiant qui a validé son diplôme. Il peut s'inscrire à la cérémonie pour dire s'il sera présent, inscrire ses invités pour partager le moment avec eux et il peut payer les places de ses invités supplémentaires avec l'application.

Comptable : Elle se retrouve rapidement debordée face aux afflux d'ordres d'achat et d'écheances de paiement : l'application lui permettera de recevoir les ordres d'achat, de les valider ou les refuser en donnant un motif.

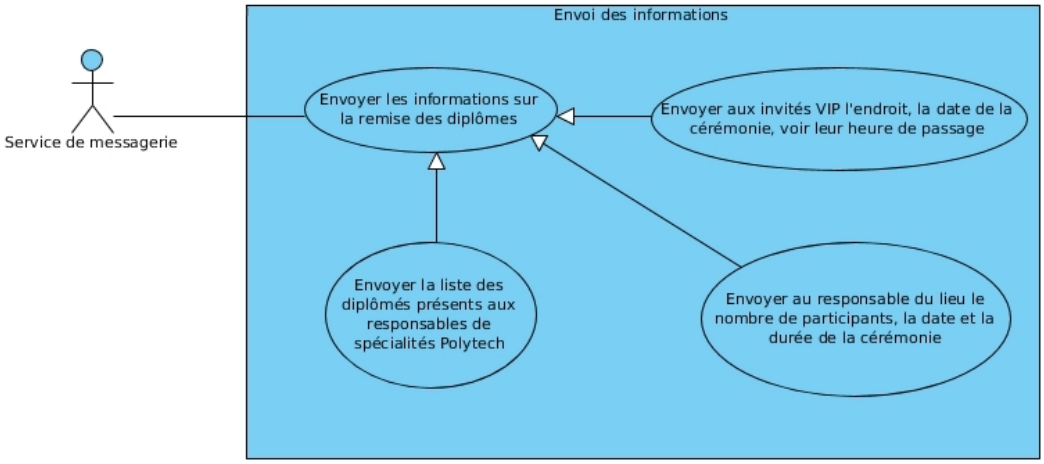
Organisateur : Il s'agit d'un individu qui se charge de mettre en place la cérémonie. Il a accès aux informations qui l'aident pour organiser et il peut créer des ordres d'achat pour la comptable

Responsable communication : Son rôle consiste à mettre en avant l'école, améliorer son image en mettant en avant les qualités des élèves en inscrivant des intermèdes mais aussi des invités de marque (PDG...).

Serveur hyperplanning : C'est un acteur secondaire et non-humain de notre système, il envoit la liste des étudiants diplomés dans l'application quand on fait la requête

Serveur de messagerie : C'est un acteur secondaire et non-humain de notre système. Il se charge d'envoyer les informations nécéssaires aux personnes jouant un rôle dans la cérémonie mais qui ne vont pas forcement télécharger l'application (ex : PDG).

2.1.3 Cas d'utilisation : envoyer les informations



Ce cas d'utilisation concerne l'envoi d'informations sur la cérémonie de remise des diplômes. L'application a pour but de faciliter la communication entre les différents acteurs et réduire les délais entre les échanges. Donc nous souhaitons mettre en place un système qui passe par un traitement automatique des tâches plutôt banales mais encombrantes (comme par exemple l'envoi d'un ordre d'achat au service comptabilité de Polytech), pour permettre une meilleure efficacité et un gain de temps considérable dans l'organisation de la remise des diplomes.

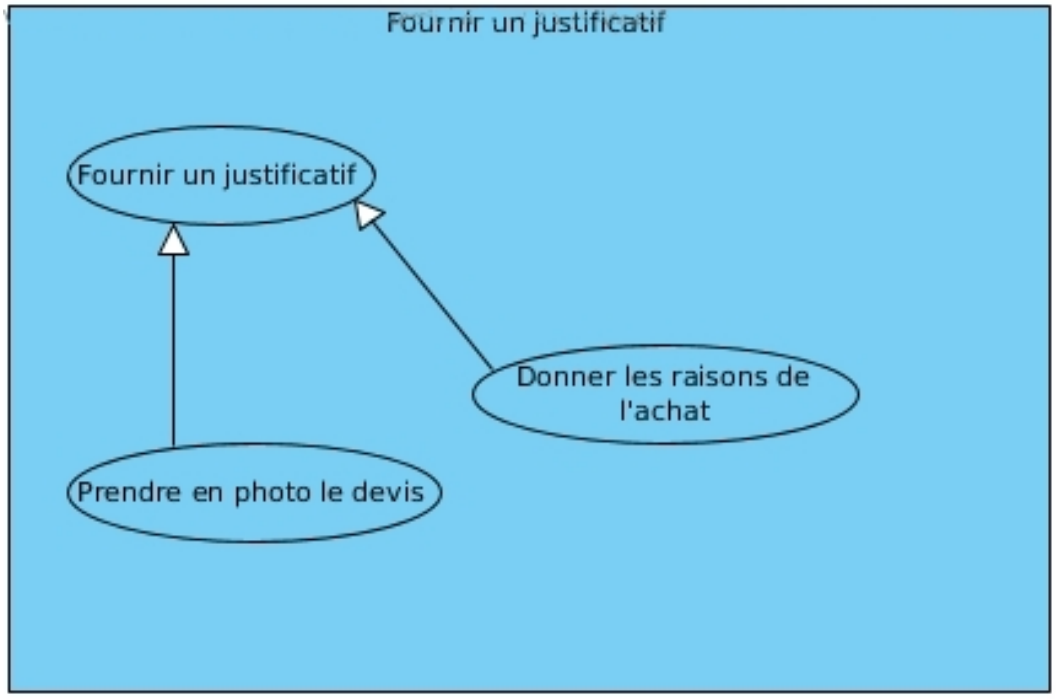
L'application doit pouvoir envoyer la liste des diplômés présents aux responsables de spécialité Polytech afin que ces derniers, s'ils sont conviés à prononcer un discours, puissent avoir les bons mots pour mettre en valeur les diplômés.

Elle doit également informer le responsable du lieu qui va accueillir la cérémonie de remise des diplômes du nombre de participants, de la date et de la durée, afin qu'elle puisse aménager le lieu et contacter son traiteur.

Enfin, les invités VIP qui n'ont pas forcément le temps d'assister à toute la cérémonie se verront envoyer un mail qui leur indiquera la date et le lieu, voir leur heure de passage s'ils sont conviés à prononcer un discours.

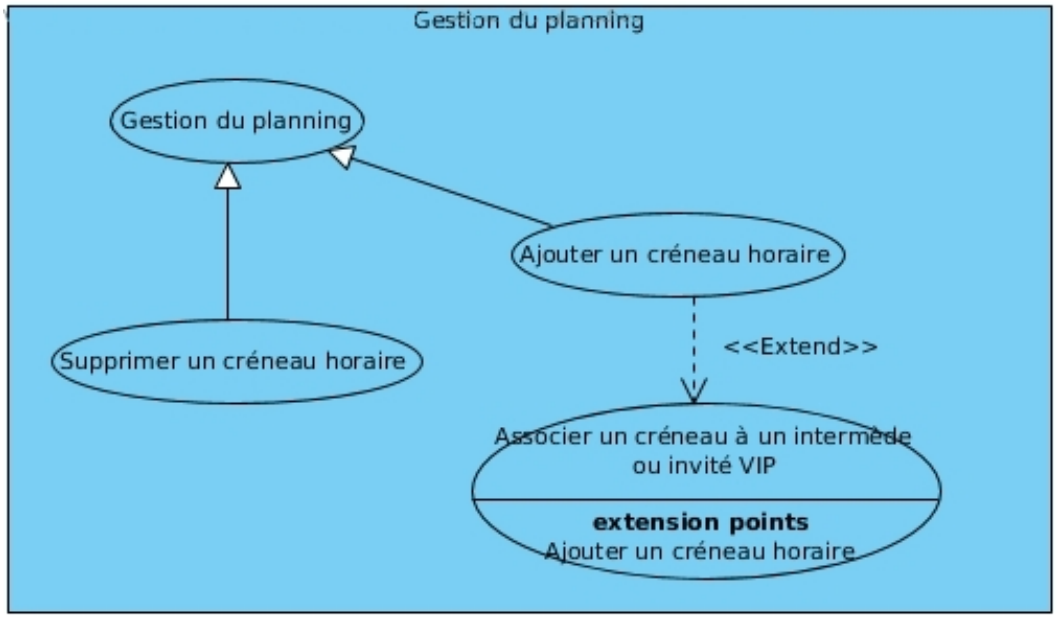
Toutes ces informations seront envoyés en même temps, juste après une date limite qui sera certainnement celle de la fin des inscriptions.

2.1.4 Cas d'utilisation : fournir un justificatif



Ce cas d'utilisation concerne la justification d'un ordre d'achat qu'un organisateur doit transmettre à la comptable : Comment un ordre d'achat pourrait se justifier via l'application? On a décidé que pour convaincre la comptable de valider l'ordre d'achat, il faudrait que l'organisateur puisse transmettre le document du devis à la comptable contenant l'ordre d'achat, ainsi que les raisons de cette achat, pour vérifier sa véracité et sa cohérence.

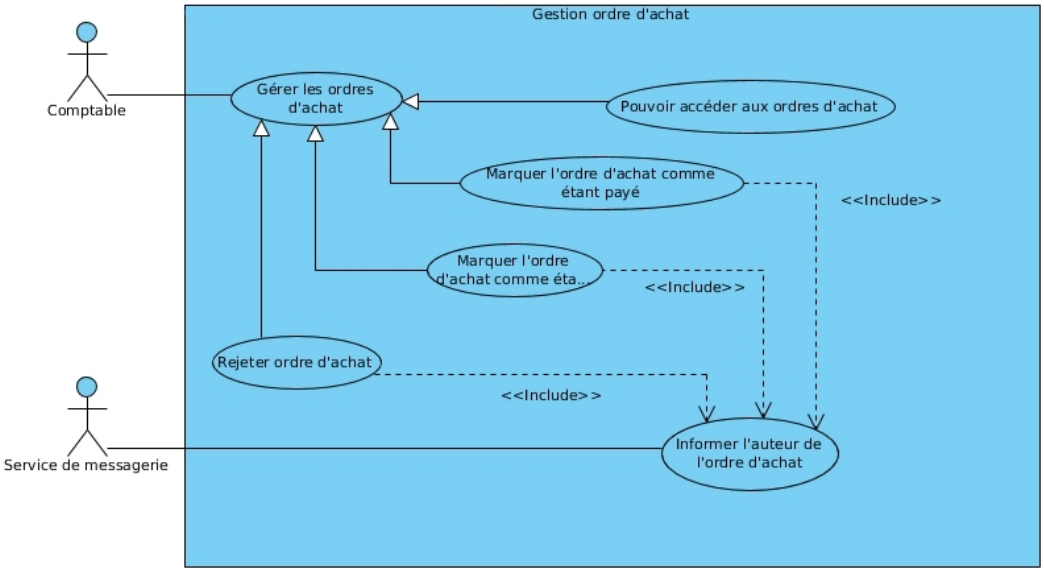
2.1.5 Cas d'utilisation : Gestion du planning



Ce cas d'utilisation concerne la gestion du planning, les fonctionnalités qui sont offertes au responsable du service communication de l'école pour pouvoir mettre en place le déroulement de la cérémonie.

La cérémonie sera divisée en créneau horaires, avec un début et une fin, et le responsable pourra soit en ajouter soit en supprimer pour aménager au mieux les créneaux. Chaque créneau horaire pourra être associée ou pas à un invité VIP (par exemple, un PDG qui prononcera un discours) ou un intermède pour montrer les qualités des étudiants.

2.1.6 Cas d'utilisation : Gestion d'ordre d'achat



Ce cas d'utilisation résume les fonctionnalités de l'application destinées au service comptable de l'école :

Elle peut accéder à tous les ordres d'achat via une interface afin de savoir quelles sont les prochaines échéances, et de mieux pouvoir les gérer en évitant les mauvaises surprises.

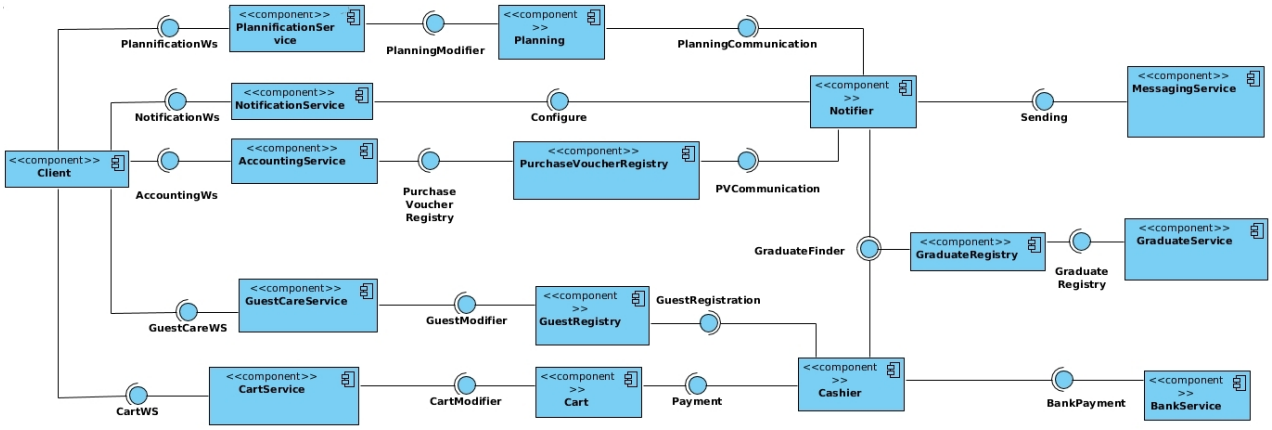
Elle peut aussi valider un ordre d'achat ou le rejeter si la justification est mal placée. De plus, elle a la possibilité de le marquer comme payé ou pas.

Chaque action de validation/rejet fait intervenir le service de messagerie qui devra annoncer à l'auteur (celui qui l'a envoyé à la comptable) de la validation ou pas de son ordre d'achat

3 Vue développement

3.1 Diagramme de composants

3.1.1 Diagramme



Notre diagramme dispose d'un composant Client qui est utilise les 5 services suivants à travers son interface utilisateur:
Le service de plannification, de notification, de comptabilité, de gestion des invités et de réservation des places.

Le service de notification est lié à un composant "Notifier", qui une fois configuré (on lui fixe une date limite d'inscription), récupère les informations dont il a besoin (via les interfaces "Finders") pour informer via le service de messagerie les acteurs de la cérémonie.

Le CartService sert à inscrire les invités de la cérémonie. Les diplomés l'utilisent pour s'inscrire avec leurs invités, à la validation, on passe à la caisse, si nous avons pas d'invités supplémentaire, on inscrit les invités au registre, sinon on paye et les invités seront ajoutés au composant Guest Registry et le paiement sera effectué via le service bancaire extérieur à l'application. Ici les responsables de communication peuvent inscrire des invités gratuitement (les VIP et intermèdes), des réductions peuvent être appliqués également.

Tout changement effectué après la commande pourra être effectué via le GuestService qui est lié au GuestRegistry via une interface de modifications d'informations sur les invités.

Le responsable du service de communication a son service de plannification avec lequel il peut organiser plus facilement la cérémonie.

La liste des diplômés est récupérée du serveur hyperplanning, qui symbolisée dans le diagramme par le composant GraduateService, qui a inscrire dans le GraduateRegistry cette liste.

Concernant la comptabilité, toutes les opérations listées dans les cas d'utilisations seront réalisées via l'interface PurchaseVoucherRegistry.

3.1.2 Interfaces et fonctions

Configure :
```
void setDeadline(Date newDeadLine);
void getDeadline();
```

PurchaseVoucherRegistry :
```
void addPurchaseVoucher(PurchaseVoucher purchaseVoucher);
void validate(PurchaseVoucher purchaseVoucher);
```

void reject(PurchaseVoucher purchaseVoucher, String reasons);

CartModifier :
void addGuestToCart(Guest guest);
void removeGuestFromCart(Guest guest);
Cart getCart();

Payment :
boolean process(Cart cart, int reductionPercentage);

BankPayment :
boolean pay(PaymentData senderPaymentData, int amount);

GuestRegistration :
void addGuest(Guest guest);

GuestModifier :
void cancelGuest(Guest guest);

GuestFinder :
Guest findGuestByName(String guestName);
list(Guest) getAllGuests();

GraduateRegistry :
List <GraduateUser> retrieveGraduateStudents();

GraduateFinder :
GraduateUser findGraduateByName(String graduateName);
list(GraduateUser) getAllGraduates();

Sending :
void send(String destinationMail, String subject, String message);

PlanningCommunication :
Date getCeremonyDate();
int getCeremonyDuration();
Planning getPlanning();

Planning Modifier :
void addTimeSlot(Date start, Date end, Guest guest, String description);
void deleteSlot(TimeSlot timeSlot);

PVCommunication :
void informValidation(String email);
void informDecline(String email);

CartWS :
void addGuestToCart(Guest guest);
void removeGuestFromCart(Guest guest);

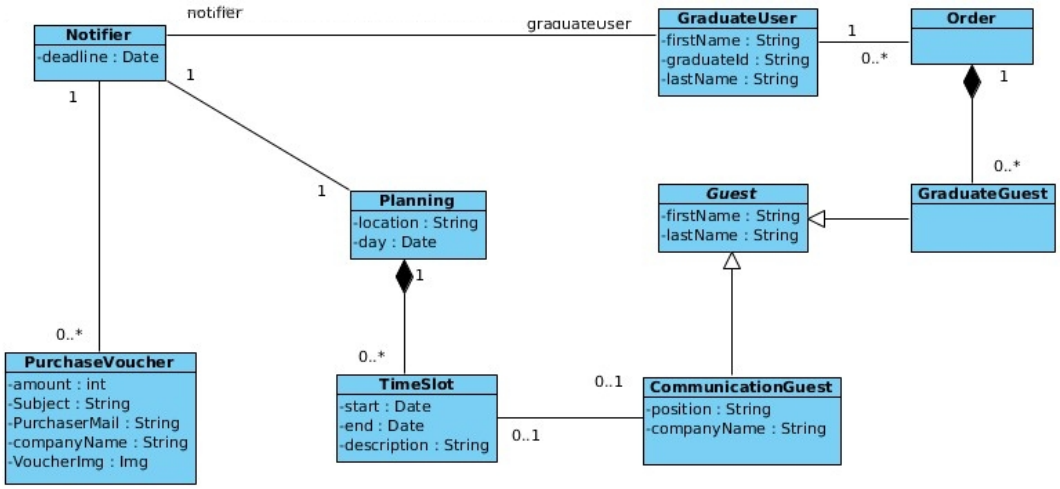
NotifierWS :
void configureDeadLine(Date deadline);

PlanningWS :
void addTimeSlot(Date start, Date end, CommunicationGuest guest, String description);
void deleteTimeSlot(int idTimeSlot);

GuestWS :
void addGuest(int idGuest);
void cancelGraduate(int graduateId);
void changeGuest(int idGuest, String firstName, String lastName);

AccountingWS :
void addPurchaseVoucher(PurchaseVoucher pv);
void validatePurchaseVoucher();
void declinePurchaseVoucher();
void markPurchaseVoucherAsPaid(PurchaseVoucher pv);

3.2 Diagramme de classe des objets métiers du système



4 Conclusion

Pour conclure, cette première phase du projet nous a permis de mieux identifier les besoins du client et ainsi définir l'ensemble des acteurs et leurs interactions mais aussi les fonctionnalités qu'offrira notre future application. Le diagramme cas d'utilisation donne un aperçu de haut niveau sur le rôle de chaque acteur dans le système.

Les diagrammes de composants et de classe orienté métier nous ont permis de creuser dans l'aspect développement du projet. L'approche de découpage par modules ré-utilisable et indépendants nous donne plus de flexibilité pour maintenir le code facilement et l'étendre au besoin.