# Spécification d'exigences logicielles

Gwendal Daniel & Robin Boncorps

Gestion d'emploi du temps

# Table des matières

# Introduction

## Objectif du document

Ce présent document a pour objectif de décrire les exigences fonctionnelles et non-fonctionnelles du système de gestion de calendrier.

On s'attachera à décrire le comportement utilisateur attendu pour chaque aspect du système au travers de use cases et de diagrammes de séquences.

## Portée du document

Le système de gestion d'emploi du temps doit être associable avec tout système de gestion de calendrier de type CALDAV. L'application pourra fonctionner sans connexion internet, mais la nécessitera pour se synchroniser avec l'application tierce.

## Définitions, acronymes et abréviations

| | |
|---|---|
| REST | REpresentational State Transfer |
| CALDAV | Calendaring Extensions to WebDAV |
| SAX | Simple API for XML |
| XML | Extensible Markup Language |

# Partie 1

# Description générale

## 1.1 Perspectives du produit

### 1.1.1 Interfaces système

Le système doit être capable de communiquer avec une application tierce de gestion de calendriers au format CALDAV. Il faudra en outre une interface de synchronisation avec le système tiers.

Bien que l'exigence soit reportée (voir 1.4), l'application doit prévoir l'ajout ultérieur d'un module permettant de récupérer un emploi du temps universitaire de type CELCAT.

### 1.1.2 Interfaces utilisateurs

L'interface utilisateur doit permettre une synchronisation simple avec le calendrier CALDAV de l'application tierce (bouton de synchronisation). L'affichage local de l'emploi du temps se fera dans une grille avec ajout et suppression de d'éléments intuitives (simple clic sur la grille), ou plus classiques (menus contextuels).

L'année universitaire étant organisée en semaine, il faudra permettre une navigation aisée de semaines en semaines.

### 1.1.3 Interfaces logicielles

L'application devra être développée en utilisant le langage de programmation c++. L'interface graphique utilisateur devrait être développée en utilisant les librairies Qt. Enfin, il conviendra de mettre en place une interface logicielle souple qui pourra permettre l'ajout de fonctionnalités dans le futur (exigence reportée, support d'autres formats de calendriers).

### 1.1.4 Interfaces de communication

Le système doit être capable de communiquer avec un gestionnaire de calendriers distant. Pour assurer cette communication l'application doit avoir un accès internet. Un haut débit n'est pas nécessaire.

L'interface de communication retenue est le protocole REST, qui permettra via l'envoi de requêtes http de créer, modifier et supprimer des événements sur le gestionnaire distant.

## 1.2 Fonctions du produit

Le système doit permettre l'affichage, la modification locale et la synchronisation de données avec un service de gestion de calendriers distant.

La synchronisation inclut à la fois la récupération d'un calendrier depuis un gestionnaire et la mise à jour depuis l'application.

## 1.3 Caractéristiques des utilisateurs

Les utilisateurs attendus du système sont l'ensemble des personnes susceptibles de consulter un emploi du temps de l'université de Nantes. Le système doit donc convenir aux étudiants mais aussi aux professeurs et au personnel non enseignant.

## 1.4 Contraintes

Le livrable de l'application doit être remis le 5 décembre au plus tard. Aucun délai ne sera autorisé. Il convient donc d'organiser la conception et l'implémentation en fonction de cette date limite.

## 1.5 Exigences reportées

Nous avons choisi de reporter l'exigence de récupération d'emploi du temps depuis le site de l'université de Nantes. Cette exigence a pour but d'alléger l'utilisation du système en permettant de récupérer directement l'emploi du temps d'une formation donnée à l'université. Elle a été reportée car le format utilisé n'est pas simple à manipuler (bien qu'en format XML) et parceque son développement ne nous semblait pas envisageable dans la période donnée.

# Partie 2

# Exigences spécifiques

## 2.1 Fonctionnalités

Cette partie détaillera la liste des exigences fonctionnelles du système. Pour établir cette liste, nous nous basons sur le diagramme de cas d'utilisations (figure 2.1). Chaque cas d'utilisation sera complété d'une brève description textuelle ainsi que d'un diagramme de séquence simplifié symbolisant le flot nominal du cas d'utilisation considéré. On détaillera également les exceptions possibles à l'exécution de ce cas d'utilisation.

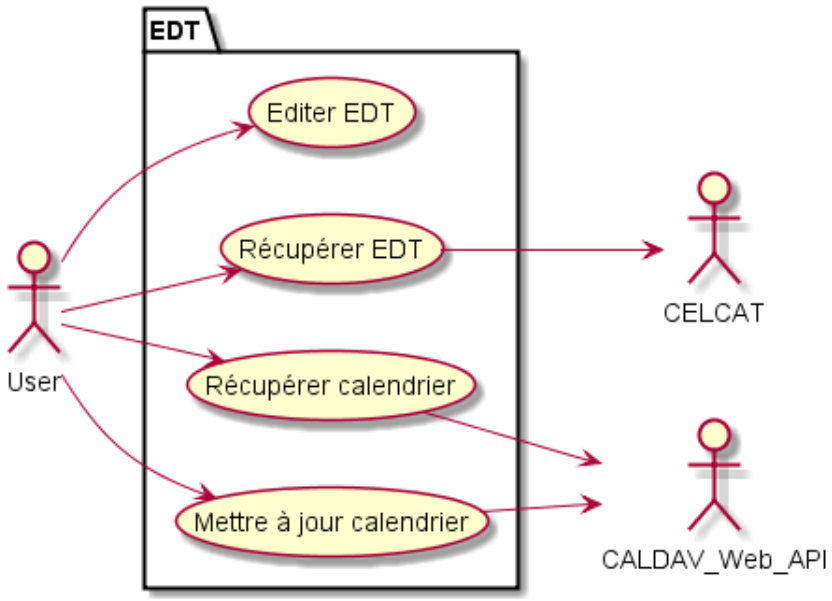


FIGURE 2.1 – Diagramme de cas d'utilisation du système

**Note :** Le cas d'utilisation "Récupérer EDT" depuis l'emploi du temps CELCAT de l'université a été reporté. Il a été ajouté comme aide pour la modélisation et ne doit pas être développé en priorité. Ce cas d'utilisation permettra de récupérer directement l'emploi du temps pour un groupe donné depuis le site de l'université de Nantes.

### 2.1.1 Récupération du calendrier

L'application doit permettre à l'utilisateur de récupérer un calendrier distant. Ce calendrier est traité et intégré au modèle de données local avant d'être affiché.

| | |
|---|---|
| Précondition | Le calendrier distant existe |
| Postcondition | le modèle de l'application est mis à jour et est affiché |
| Niveau | Fonction principale |
| Acteur principal | utilisateur |
| Acteur secondaire | API web du gestionnaire de calendriers. |

Flot nominal : voir figure 2.2.

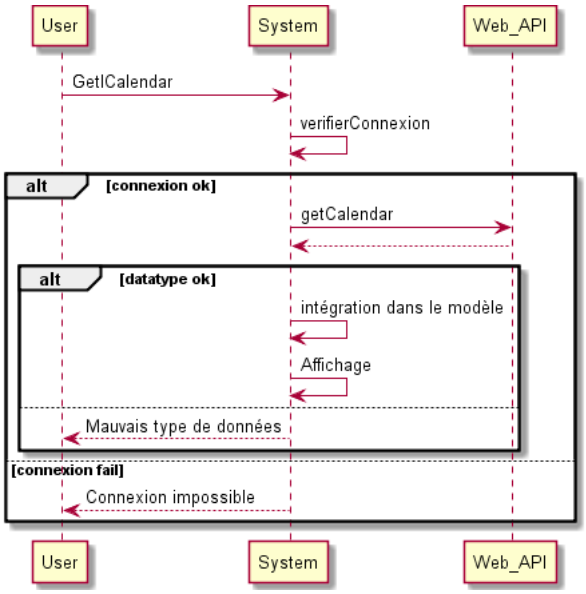


FIGURE 2.2 – Flot de récupération d'un calendrier

Exceptions possibles :
– Connexion impossible
– Le service distant n'est pas joignable
– Les données ne sont pas dans le bon format

**Note :** On portera une attention particulière à la gestion de conflit en cas d'importation après modification des données. Le programme doit avertir de l'écrasement des données.

### 2.1.2 Mise à jour du calendrier

L'application doit permettre à l'utilisateur de mettre à jour le calendrier sur le gestionnaire distant. La mise à jour est effectuée à partir des données du modèle local.

| | |
|---|---|
| Précondition | le modèle local n'est pas vide |
| Postcondition | le calendrier en ligne est mis à jour avec les données du modèle. |
| Niveau | Fonction principale |
| Acteur principal | utilisateur |
| Acteur secondaire | API web du gestionnaire de calendriers. |

Flot nominal : voir figure 2.3.

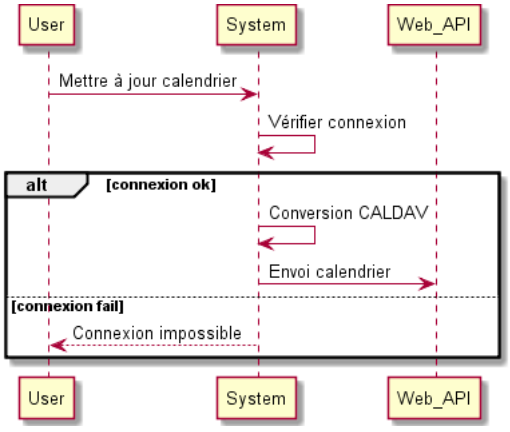


FIGURE 2.3 – Flot de mise à jour d'un calendrier

Exceptions possibles :
– Connexion impossible

– Le service distant n'est pas joignable

### 2.1.3 Edition du calendrier local

L'utilisateur doit pouvoir modifier la version locale du calendrier (soit de l'emplois du temps) : il doit pouvoir ajouter des créneaux de cours, en supprimer, les modifier (modification de salle, de professeur, d'heure ...)

| | |
|---|---|
| Précondition | le modèle local n'est pas vide |
| Postcondition | le calendrier local est mis à jour |
| Niveau | Fonction principale |
| Acteur principal | utilisateur |
| Acteur secondaire | vide |

Flot nominal : voir figure 2.4.

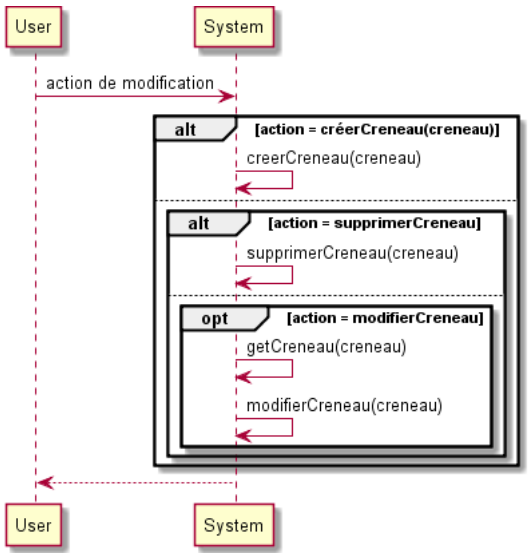


FIGURE 2.4 – Flot d'édition d'un calendrier local

Exceptions possibles :
– Le creneau à supprimer n'existe pas
– Le creneau à créer existe déjà
– Les données du creneau ne sont pas cohérentes

## 2.2 Exigences supplémentaires

Cette section détaillera les exigences non fonctionnelles de l'application. On y traitera notamment des contraintes de fiabilité, de performance et de maintenabilité.

### 2.2.1 Utilisabilité

L'utilisation de l'application doit être intuitive pour l'utilisateur : il ne doit pas nécessiter de formation afin de la maîtriser. De plus, il est demandé de respecter l'organisation et la charte graphique des menus des applications linux.

### 2.2.2 Fiabilité

Le système doit être suffisament robuste pour supporter une perte de connexion internet lors d'une demande de mise à jour vers le calendrier en ligne. Ce support doit inclure la sauvegarde locale de l'emploi du temps pour pouvoir permettre à l'utilisateur de réeffectuer une mise à jour ultérieurement.

Plus généralement, le système doit sauvegarder l'état de son modèle dans le cas d'une modification effectuée sur plusieurs périodes.

### 2.2.3 Performance

La question de la performance est cruciale dans un système utilisant internet comme support de communication. L'application doit permettre de récupérer et de mettre à jour un calendrier dans un laps de temps court (de l'ordre de la seconde). En cas de communication plus longue, il ne faut pas que le processus de récupération soit bloquant pour l'application.

### 2.2.4 Maintenabilité

Comme nous l'avons dit précédemment, l'application résultante de cette phase de développement ne sera pas une application finale et immuable, elle est ammenée à évoler et à incorporer de nouvelles fonctionnalités.

- Récupération du calendrier sur le site de l'université
- Support d'autre type de calendrier
- Extension à d'autres universités que l'université de Nantes
- ...

Il est donc très important de prévoir le développement en conséquence. Il faut que l'application puisse s'étendre facilement vers de nouveaux types de données et puisse intégrer de nouveaux modules (récupérateur de données sur le site de l'université avec parseur SAX par exemple).

### 2.2.5 Sécurité

L'application doit permettre à l'utilisateur d'entrer l'adresse et la clé API du calendrier qu'il souhaite importer. Cette information est stockée durant la session d'utilisation et est conservée dans le fichier de sauvegarde. La sécurité est en partie assurée par l'utilisateur qui se chargera de placer son fichier de sauvegarde dans un dossier sécurisé.

# Partie 3

# Classification des exigences fonctionnelles

| Fonctionnalité | Type | Etat |
|---|---|---|
| Récupérer Calendrier | Essentielle | |
| Editer EDT | Essentielle | |
| Mettre à jour calendrier | Essentielle | |
| Temps de réponse de l'ordre de la seconde | Essentielle | |
| Communication non bloquante | Essentielle | |
| Gestion des conflits d'import | Essentielle | |
| Affichage par semaines | Essentielle | |
| Identifiants de connexion | Essentielle | |
| Architecture modulaire | Essentielle | |
| Protocole REST | Essentielle | |
| Sauvegarde locale | Essentielle | |
| Modification locale par menus | Essentielle | |
| Respect des délais | Essentielle | |
| Respect charte graphique Linux | Souhaitable | |
| Modification locale par clics intuitifs | Souhaitable | |
| Support de perte de connexion | Souhaitable | |
| Récupérer EDT CELCAT | Optionnelle (reportée) | |
| Support d'autres formats de calendrier | Optionnelle (reportée) | |