Nom du groupe : Flyin' Muffins

Alexandra BEYRON
Loïc FOTSO
Thibault ROY
Frédéric THOMAS

GL02 : Cahier des charges du projet A – iCal

Version 1.0

Résumé :

Ce rapport présente le cahier des charges du projet iCal : il s'agit du développement d'une librairie pour la gestion des emplois du temps au sein de l'association ADOMY qui aura pour objectif de détecter tous les conflits et d'organiser des rendez-vous.

Responsable de l'UV : Matthieu Tixier
Semestre : Automne 2015

Table des matières

Préface

Ce cahier des charges a été établi à la demande de la mairie de Yers pour l'association d'aide au maintien à domicile de Yers (ADomY).
Dans une première partie, il s'agira de définir en langage naturel toutes les spécifications générales des exigences fonctionnelles mais aussi non-fonctionnelles.
Ensuite, une seconde partie sera consacrée aux différentes spécifications détaillées que nous aurons définies.

Il sera destiné au prestataire qui sera chargé de produire l'application et de développer la librairie en Javascript.

Règles de révision du document

Cette version est la version 1.X du cahier des charges que nous proposons. Le document sera sûrement amené à être modifié lors de ce projet en équipe pour apporter toute nouvelle contribution.
A chaque modification, la version sera incrémentée de 0.1 pour une modification minime et de 1 pour de plus grandes modifications.
Il n'y aura pas de restrictions ni de demande de permission concernant la modification de données.

Pour effectuer une modification sur le document (ensemble d'ajouts, de modifications et de suppression de données), il faudra que la personne active le mode modification pour garder en mémoire les éléments originaux qui vont être modifiés. Cela permet à toutes les personnes qui vont travailler sur le document de voir au fur et à mesure les modifications faites depuis le début. De ce fait, chaque élément ajouté sera d'une autre couleur avec le nom de l'auteur et chaque élément supprimé sera enregistré ou barré de la place où il était.
Si une personne souhaite faire une remarque sur certaines données, elle peut alors annoter le passage de la manière suivante : lors de l'insertion d'une note, il faut placer le curseur à l'endroit voulu. Dans le menu de révision, il faut alors ajouter la note et le commentaire pouvant être accompagné de la signature de l'auteur.
Il y aura alors un rectangle de couleur indiquant le commentaire correspondant qui s'affichera.

Glossaire

Union : L'union de plusieurs emplois du temps est la superposition de tous les évènements de chaque emploi du temps dans un même planning.

Intersection : L'intersection de deux emplois du temps est l'affichage de tous les créneaux communs de tous les emplois du temps.

iCalendar RFC 5545 : Standard pour les échanges de données de calendrier.

CSV : Comma-Separated values. Format informatique qui représente des données tabulaires sous formes de valeurs et séparées par une virgule.

Fichier .csv : C'est un fichier texte. Chaque ligne du texte correspond à une ligne du tableau et les virgules correspondent aux séparations entre les colonnes.

Format pivot : Format de transition utilisé dans l'application

Introduction

Contexte

L'association ADomY travaille auprès de personnes âgées, plus ou moins en situation de dépendance, avec l'aide de différents types d'intervenants : professionnels (aide-ménagère, auxiliaire de vie, etc.) ou libéraux (infirmiers, taxis, etc.). Cette association doit pouvoir coordonner les différents emplois du temps en fonction des disponibilités de tous les intervenants qui viennent fournir plusieurs services destinés aux personnes âgées.

Pour cela, l'association possède déjà un logiciel de gestion des ressources humaines qui édite des plannings de manière individuelle pour chaque intervenant et pour chaque domicile. Tous ces plannings sont ainsi exportables et importables au format CSV.

Cependant, la manière dont les gestionnaires identifient les conflits n'est pas assez efficace et reste empirique. Ils affichent plusieurs emplois du temps simultanément puis font la liste des conflits mais pas sous forme de planning. Pour attribuer un créneau qui conviendrait aux personnes concernées, il faut donc rouvrir leur planning pour permettre une vue des différents conflits.

Objectifs du logiciel

Le logiciel sera une librairie écrite en Javascript qui permettra de charger et convertir plusieurs types de fichiers (iCal, CSV) en format pivot iCal FRC 5545 contenant : date et heure d'intervention, durée, responsable, lieu.

Pour cela, le projet logiciel doit remplir plusieurs objectifs importants. Le logiciel à développer devrait donc devoir lister les conflits des emplois du temps automatiquement et générer un planning. Ces conflits devront être alors affichés sous forme d'intersection et d'unions.

L'association veut également que l'application puisse générer les complémentaires d'emplois du temps, c'est-à-dire les disponibilités et non les évènements.

De plus, les intervenants externes fournissent déjà leurs emplois du temps sous la forme iCalendar. Il faudrait que l'application puisse charger les autres fichiers de format CSV, iCal et les exporter en fichier pivot que nous définirons.

Les gestionnaires d'Adomy voudraient également plus d'informations sur le nombre d'interventions planifiées et du nombre horaire total pour chaque intervenant sur une semaine. Cela permettrait de comparer tous leurs conflits de disponibilités.

I - Spécifications générales des exigences

Dans cette première partie, l'objectif de notre tâche reste relativement simple : identifier les différentes exigences que doit fournir la future librairie. Ces exigences seront énoncées en ordre d'importance et catégorisées selon leur degré de fonctionnalité. Ici, la spécification se fera uniquement en langage naturel. Tout lecteur de ce cahier des charges, quel que soit son niveau de connaissances en ingénierie logiciel, devra donc être capable de reformuler correctement les exigences du produit voulu.

A) Les exigences fonctionnelles

1. L'ajout des fichiers iCal ou CSV à la librairie : SPEC_01

Tous les intervenants d'Adomy ont leur emploi du temps (edt) respectifs inscrits dans un fichier iCal ou alors regroupés dans des fichiers CSV. La librairie doit donc être capable d'ajouter un edt (sous forme iCal) ou une liste d'edt (sous forme CSV) à sa base de fichiers.

2. Mettre un fichier iCal au format pivot : SPEC_02.1

Les fichiers dans notre librairie sont définis sous un format très spécifique. Dès lors les fichiers enregistrés dans notre base doivent être transformés pour s'accommoder à ces spécificités. Tous les fichiers transformés doivent donc comportés les champs (remplis ou non) : *date et heure d'intervention*, *durée*, *responsable* et *lieu*.

3. Mettre un fichier CSV au format pivot : SPEC_02.2

Il est possible que l'entreprise reçoive une liste d'emplois du temps regroupés sous un seul fichier sous le format CSV. Après l'ajout dans sa base de cette liste, la librairie doit être capable de transformer ce fichier CSV en autant de fichiers correspondant à chaque emploi du temps présent dans le fichier d'origine. Tous ceux-ci doivent bien sûr comporter les champs obligatoires déjà mentionnés plus haut à savoir *date et heure d'intervention*, *durée*, *responsable* et *lieu*.

4. Générer l'intersection d'emploi du temps : SPEC_03

Le gestionnaire de la librairie doit être capable de comparer les emplois du temps de plusieurs intervenants de l'entreprise et de rendre utile les données issues de cette comparaison. Ainsi, il doit pouvoir ouvrir sur une interface les horaires de différents intervenants (au moins deux) et de voir en évidence les recoupements de celles-ci. Plus encore, il doit pouvoir créer, à la fin de cette comparaison, un fichier présentant les recoupements entre les horaires d'activité des intervenants. Ce fichier sera enfin

enregistré au format pivot de la librairie. Les champs responsables et *lieu* seront remplis similairement à ceux des fichiers d'origine. Les champs *date et heure d'intervention*, *durée* quant à eux dépendront de l'intersection des horaires.

5. Générer l'union d'emplois du temps : SPEC_04

Le gestionnaire de la librairie doit également être capable de visualiser l'emploi du temps cumulé de plusieurs intervenants (au moins deux) sur une même interface. Il doit être capable par la suite d'enregistrer ce cumul en un nouveau fichier au format pivot de la librairie.

6. Générer le complémentaire d'emplois du temps : SPEC_05

En plus de l'intersection et de l'union, le gestionnaire doit pouvoir visualiser le complémentaire de plusieurs emplois du temps différents. En d'autres mots, il s'agira de pouvoir mettre en évidence les horaires non utilisés par les différents intervenants. Ces horaires pourront être enregistrés dans un fichier au format pivot de la librairie.

7. Exporter les fichiers en format pivot au format iCal : SPEC_06.1

Tous les fichiers enregistrés par la librairie doivent pouvoir être utilisées par la suite par d'autres services de la compagnie. Ces services seront plus familiers à des formats classiques comme le iCal. Dès lors, notre librairie doit pouvoir transformer tous les fichiers enregistrés dans la librairie sous le format pivot (les fichiers normaux, les fichiers d'union, d'intersection, de complémentaire) en format iCal. Ces fichiers iCal pourront ainsi être exportés vers une autre base de fichiers. Le fichier iCal final présentera les mêmes champs que le fichier enregistré au format pivot c'est-à-dire : *date et heure d'intervention*, *durée*, *responsable* et *lieu*.

8. Exporter les fichiers en format pivot au format CSV : SPEC_06.2

D'autres services ou partenaires de l'entreprise sont peut-être plus habitués à l'utilisation de fichiers CSV. L'utilisateur de la librairie pourra donc sélectionner plusieurs fichiers au format pivot de notre base et sera capable de les fusionner en un nouveau fichier CSV. Ce fichier CSV pourra donc ensuite être exporté vers une autre base de fichiers.

9. Calculer le volume horaire d'un employé sur une semaine : SPEC_07

L'utilisateur de la librairie doit être capable de connaître le nombre total d'heures de travail d'un employé sur une période d'une semaine. Notre librairie doit donc pouvoir ouvrir un fichier donné enregistré au format pivot et additionner le nombre d'heures d'intervention de l'employé. Elle doit par la suite pouvoir afficher ce résultat à l'utilisateur.

De manière optimale, ce calcul pourrait même être automatisé. C'est à dire qu'à chaque fichier enregistré au format pivot, la librairie effectuerait ce calcul automatiquement et afficherait directement à l'utilisateur ce volume horaire à côté des autres champs du fichier sans qu'il n'ait à faire cette requête.

10. Relever le nombre d'interventions d'un employé sur une semaine : SPEC 08

L'utilisateur de la librairie doit être capable de connaître le nombre de fois qu'un intervenant est amené à travailler sur une période d'une semaine. Notre librairie doit donc pouvoir ouvrir un fichier donné enregistré au format pivot et relever le nombre d'occurrences dans le champ *date et heure d'intervention*.
Encore une fois, de manière optimale, ce calcul devrait pourrait être automatisé. L'utilisateur n'aurait pas besoin de faire une requête pour avoir cette information sur un fichier. En effet, lors de l'enregistrement de ce dernier dans le format pivot, la librairie relèverait cette donnée et l'afficherait à côté des autres champs du format.

B) Les exigences non-fonctionnelles

1. Le langage utilisé : SPEC_NF_1

L'équipe qui s'occupe de l'intégration de l'application au Système d'Informations d'*Adomy* a demandé à ce que la librairie soit entièrement développée en JavaScript.

2. L'accessibilité à la librairie : SPEC_NF_2

Pour des raisons de sécurité, la librairie ne devra être accessible qu'au gestionnaire de celle-ci et son équipe. Elle ne devra donc être exécutable que sur les postes de travail de l'équipe concernée ou alors à distance mais via une connexion sécurisée et des identifiants/mots de passe propres à chacun des membres de l'équipe de gestion.

3. La protection des fichiers : SPEC_NF_3

Le flux des fichiers passant par la librairie doit être complètement sécurisé. Optionnellement, elle pourrait même tenir un historique de toutes les personnes ou services ayant eu accès aux informations d'un fichier.

4. Maintenance : SPEC_NF_4

Des fonctionnalités pourront être rajoutées ultérieurement à la librairie soit par l'équipe de développement, soit par une autre. Le code source doit être suffisamment fourni en commentaires explicatifs afin de permettre à une équipe externe d'améliorer le service.

II – Spécifications détaillées

A) Détail de chacune des exigences fonctionnelles

Titre	Ajouter un fichier iCal ou CSV à traiter
Identifiant	SPEC_01
Objectifs	L'utilisateur doit charger dans l'application des fichiers iCal qui seront traités
Préconditions	Le fichier importé doit être au format iCal RFC5545 ou CSV
Post conditions	Le fichier est chargé et prêt pour traitement.
Entrées	Fichier, Intervenant, semaine
Traitements	Le fichier est chargé dans l'application et ajouté à la liste des fichiers à traiter.
Sorties	La liste des fichiers ajoutés est mise à jour et affichée
Gestion des erreurs	Message d'erreur en cas de saisie incomplète, erronée, de mauvais format de fichier ou de fichier déjà ajouté.
Remarques	Les entrées sont obligatoires

Titre	Mettre un fichier iCal au format pivot
Identifiant	SPEC_02.1
Objectifs	L'utilisateur doit obtenir des fichiers au format pivot à partir d'un fichier iCal importé
Préconditions	Le fichier importé doit être au format iCal RFC5545
Post conditions	Le fichier doit être conforme au format pivot
Entrées	Fichier iCal
Traitements	Le système converti les fichier de format iCal au format pivot en exportant chacune des entrées du format pivot : date et heure d'intervention, durée, responsable, lieu
Sorties	Fichier au format pivot.
Gestion des erreurs	Message d'erreur en cas de mauvais format de fichier ou d'erreur lors du processus de conversion.
Remarques	Les entrées sont obligatoires

Titre	Mettre un fichier CSV au format pivot
Identifiant	SPEC_02.2
Objectifs	L'utilisateur doit obtenir des fichiers au format pivot à partir d'un fichier CSV importé
Préconditions	Le fichier importé doit être au format CSV
Post conditions	Le fichier doit être conforme au format pivot.
Entrées	Fichier CSV
Traitements	Le système converti les fichier de format CSV au format pivot en exportant chacune des entrées du format pivot : date et heure d'intervention, durée, responsable, lieu
Sorties	Fichier au format pivot.
Gestion des erreurs	Message d'erreur en cas de mauvais format de fichier ou d'erreur lors du processus de conversion.
Remarques	Les entrées sont obligatoires

Titre	Générer l'intersection d'emplois du temps
Identifiant	SPEC_03
Objectifs	L'utilisateur doit pouvoir comparer des emplois du temps affichés par intersection
Préconditions	Au moins deux fichiers au format pivot sont chargés dans l'application (SPEC_02)
Post conditions	-
Entrées	Au moins deux fichiers au format pivot, numéro de la semaine
Traitements	Les événements ayant lieu au même moment de la semaine donnée dans les deux emplois du temps sont reportés dans un nouvel emploi du temps.
Sorties	Emploi du temps au format pivot présentant les intersections trouvées sur la semaine
Gestion des erreurs	Message d'erreur si moins de deux fichiers sont chargés, ou si une erreur est survenue lors de l'intersection.
Remarques	Les entrées sont obligatoires

Titre	Générer l'union d'emplois du temps
Identifiant	SPEC_04
Objectifs	L'utilisateur doit pouvoir comparer des emplois du temps affichés par union
Préconditions	Au moins deux fichiers au format pivot sont chargés dans l'application (SPEC_02)
Post conditions	-
Entrées	Au moins deux fichiers au format pivot, numéro de la semaine.
Traitements	Tous les événements de la semaine donnée dans les deux emplois du temps sont reportés dans un nouvel emploi du temps.
Sorties	Emploi du temps au format pivot présentant les unions trouvées sur la semaine
Gestion des erreurs	Message d'erreur si moins de deux fichiers sont chargés, ou si une erreur est survenue lors de l'union.
Remarques	Entrées obligatoires

Titre	Générer le complémentaire d'emplois du temps
Identifiant	SPEC_05
Objectifs	L'utilisateur doit pouvoir consulter les plages horaires communes qui sont disponibles
Préconditions	Au moins deux fichiers au format pivot sont chargés dans l'application (SPEC_02)
Post conditions	-
Entrées	Au moins deux fichiers au format pivot, numéro de la semaine
Traitements	Toutes les disponibilités communes des emplois du temps sont reportées dans un nouvel emploi du temps.
Sorties	Emploi du temps au format pivot présentant les disponibilités communes trouvées sur la semaine
Gestion des erreurs	Message d'erreur si moins de deux fichiers sont chargés, ou si une erreur est survenue lors du calcul du complémentaire.
Remarques	Entrées obligatoires

Titre	Exporter les fichiers en format pivot au format iCal
Identifiant	SPEC_06.1
Objectifs	L'utilisateur doit convertir les fichiers au format pivot en un fichier iCal plus facile à partager
Préconditions	Au moins un fichier pivot doit être chargé (SPEC_02)
Post conditions	Des fichiers iCal sont générés
Entrées	Fichiers au format pivot
Traitements	Le système converti les fichier de format pivot au format iCal en exportant chacune des entrées du format pivot : date et heure d'intervention, durée, responsable, lieu
Sorties	Fichier iCal
Gestion des erreurs	Message d'erreur si aucun fichier pivot n'a été chargé ou si une erreur est survenue lors de la copie des informations.
Remarques	Entrées obligatoires

Titre	Exporter les fichiers en format pivot au format CSV
Identifiant	SPEC_06.2
Objectifs	L'utilisateur doit convertir les fichiers au format pivot en un fichier CSV exploitable par un tableur
Préconditions	Au moins un fichier au format pivot doit être chargé (SPEC_02)
Post conditions	Des fichiers .csv sont générés
Entrées	Fichiers au format pivot
Traitements	Le système converti les fichier de format pivot au format csv en exportant chacune des entrées du format pivot : date et heure d'intervention, durée, responsable, lieu
Sorties	Fichier .csv
Gestion des erreurs	Message d'erreur si aucun fichier pivot n'a été chargé ou si une erreur est survenue lors de la copie des informations.
Remarques	Entrées obligatoires

Titre	Calculer le volume horaire d'un employé sur une semaine
Identifiant	SPEC_07
Objectifs	L'utilisateur doit pouvoir connaître le volume horaire d'un employé sur une semaine donnée
Préconditions	Au moins un fichier au format pivot doit être chargé (SPEC_02)
Post conditions	-
Entrées	Emploi du temps au format pivot, numéro de semaine, employé
Traitements	Calculer la somme des durées des interventions de l'employé sur la semaine donnée
Sorties	Le système retourne et affiche le volume horaire de l'employé sur la semaine donnée
Gestion des erreurs	Message d'erreur si aucun fichier pivot n'a été chargé ou si une erreur est survenue lors du traitement.
Remarques	Entrées obligatoires

Titre	Relever le nombre d'interventions d'un employé sur une semaine
Identifiant	SPEC_08
Objectifs	L'utilisateur doit pouvoir connaître le nombre d'interventions d'un employé sur une semaine donnée
Préconditions	Au moins un fichier au format pivot doit être chargé (SPEC_02)
Post conditions	-
Entrées	Emploi du temps au format pivot, numéro de semaine, employé
Traitements	Calculer le nombre d'interventions de l'employé sur la semaine
Sorties	Le système retourne et affiche le nombre d'interventions de l'employé sur la semaine
Gestion des erreurs	Message d'erreur si aucun fichier pivot n'a été chargé ou si une erreur est survenue lors du traitement.
Remarques	Entrées obligatoires

B) Spécification des formats de données pour l'application

L'application que nous voulons développer doit permettre de lire des fichiers de formats CSV, iCal RFC 5545. Premièrement, nous spécifierons le format CSV d'un employé et d'un intervenant qui sont similaires.

Format CSV employé et intervenant

Planning : 48* Ligne (car on compte en demi-heures)

Ligne : 7* Créneau (7 jours de la semaine)

Créneau : " vide " OU Intervention ';'

Intervention = Fonction '(' Nom OU Entreprise ')'

Fonction = 1* (VCHAR / WSP)

Nom = VCHAR '.' WSP 1*VCHAR

Entreprise = 1* (VCHAR / WSP)

Le format iCalendar (RFC 5545)

L'objet central de gestion d'agenda et de planification est une collection d'informations de gestion d'agenda et de planification. Ces informations constitueront typiquement un seul objet iCalendar. Toutefois, plusieurs objets iCalendar peuvent être regroupés en séquence. La première ligne de l'objet iCalendar et la dernière *doivent* contenir les éléments du couple de délimitation d'un objet iCalendar.

La syntaxe ABNF d'un objet iCalendar est la suivante :

```
icalobject = 1*("BEGIN" ":" "VCALENDAR" CRLF
                icalbody
                "END" ":" "VCALENDAR" CRLF)
```

Un objet iCalendar doit inclure les propriétés de calendrier "PRODID" et "VERSION". De plus, il doit inclure au moins un composant de calendrier (dans notre cas, le composant est un événement qui correspond à une intervention).

Voici un exemple simple d'objet iCalendar :

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Adomy Corporation//MonProduit v1.0//FR
BEGIN:VEVENT
DTSTART:20150714T170000Z
DTEND:20150715T035959Z
SUMMARY:Nom de l'intervention
DESCRIPTION:intervenant ou détails supplémentaires
LOCATION:Bâtiment A
END:VEVENT
END:VCALENDAR
```

Détail des propriétés du format iCal utilisés dans l'application

BEGIN : marque le début d'une entité. Dans notre application seulement deux entités seront utiles : VCALENDAR qui permet de définir le début du calendrier (ici de l'emploi du temps) et VEVENT, composant d'un calendrier qui permet de définir le début d'un événement.

VERSION : obligatoire, cette propriété définit l'identificateur correspondant au numéro de version le plus élevé, ou l'intervalle minimum et maximum, de la spécification iCalendar nécessaire pour interpréter l'objet iCalendar.

La propriété VERSION est définie par la notation ABNF suivante :

```
version    = "VERSION" verparam ":" vervalue CRLF
verparam   = *(";" xparam)
vervalue   = "2.0"         ;
           / maxver
           / (minver ";" maxver)
minver     = <Un identificateur de version iCalendar
enregistré auprès de l'IANA>
;  la version iCalendar minimum permettant d'interpréter
;  l'objet iCalendar
maxver     = <Un identificateur de version iCalendar
enregistré auprès de l'IANA>
;  la version iCalendar maximum permettant d'interpréter
;  l'objet iCalendar
```

PRODID : obligatoire, cette propriété définit l'identificateur du produit (application) ayant servi à créer l'objet iCalendar. Cette propriété ne *doit* apparaître qu'une seule fois dans un objet iCalendar.

La propriété PRODID est définie par la notation ABNF suivante :

```
  prodid     = "PRODID" pidparam ":" pidvalue CRLF
     pidparam   = *(";" xparam)
     pidvalue   = text

; une chaîne qui décrit le produit et sa version, et qui
; est assurée d'être unique.
```

DTSTART : Cette propriété indique le début du composant de calendrier. Le type de valeur est DATE-TIME. Dans le composant de calendrier "VEVENT", cette propriété définit la date et heure de début de l'événement. Elle est *obligatoire* dans le composant de calendrier "VEVENT". Les événements peuvent avoir une date et heure de début mais pas de fin. Auquel cas, l'événement n'occupe pas de temps.

La propriété DTSTART est définie par la notation ABNF suivante :

```
     dtstart    = "DTSTART" dtstparam ":" dtstval CRLF
     dtstparam  = *(

                ; les paramètres suivants sont optionnels,
                ; mais ne doivent pas apparaître plus d'une fois

                  (";" "VALUE" "=" ("DATE-TIME" / "DATE")) /
                  (";" tzidparam) /

                ; le paramètre suivant est optionnel
                ; et peut apparaître plus d'une fois

                    *(";" xparam)
                  )
     dtstval    = date-time / date
     ; la valeur doit correspondre au type de valeur
```

Pour simplifier, dans l'application la date de début d'un événement est au format <AnnéeMoisJourHeureMinutesSecondesZ>

Exemple : « DTSTART:20151123T153000Z » signifie qu'un événement commence le 23 novembre 2015 à 15h30.

Dans notre application on se place dans le fuseau horaire GMT+0 et on ne tient pas compte de l'heure d'été.

DTEND : propriété fonctionnant sur le même principe que DTSTART, représentant la date et l'heure de fin d'un événement dans un calendrier.

UID : Cette propriété indique l'identificateur unique et persistant de l'événement. Cette propriété est de type TEXT.
La propriété UID est définie par la notation ABNF suivante :

```
uid         = "UID" uidparam ":" text CRLF
uidparam    = *(";" xparam)
```

SUMMARY : Cette propriété de type TEXT indique un bref résumé ou un sujet pour l'événement.

La propriété SUMMARY est définie par la notation ABNF suivante :

```
summary    = "SUMMARY" summparam ":" text CRLF
summparam  = *(

           ; les paramètres suivants sont optionnels,
           ;  mais ne doivent pas apparaître plus d'une fois

              (";" altrepparam) / (";" languageparam) /

           ; le paramètre suivant est optionnel
           ;  et peut apparaître plus d'une fois

              (";" xparam)

              )
```

DESCRIPTION : Cette propriété de type TEXT fournit une description plus longue et complète de l'événement du calendrier que ne le permet la propriété "SUMMARY". Dans notre application elle sera utilisée pour renseigner des détails sur l'intervention.

La propriété DESCRIPTION est définie par la notation ABNF suivante :

```
description        = "DESCRIPTION" descparam ":" text CRLF
descparam  = *(

           ; les paramètres suivants sont optionnels,
           ;  mais ne doivent pas apparaître plus d'une fois

              (";" altrepparam) / (";" languageparam) /

           ; le paramètre suivant est optionnel
           ;  et peut apparaître plus d'une fois

              (";" xparam)

              )
```

LOCATION : Cette propriété de type TEXT définit le lieu prévu pour un événement.

La propriété LOCATION est définie par la notation ABNF suivante :

```
     location    = "LOCATION locparam ":" text CRLF
     locparam    = *(
                 ; les paramètres suivants sont optionnels,
                 ;  mais ne doivent pas apparaître plus d'une fois

                    (";" altrepparam) / (";" languageparam) /

                    ; le paramètre suivant est optionnel
                    ;  et peut apparaître plus d'une fois

                    (";" xparam)

                    )
```

END : balise qui marque la fin d'une entité précédemment ouverte par BEGIN.

C) Spécification algébrique du type Emploi du temps

Le rang du booléen dans le tableau permet de pouvoir y associer une case pleine à une intervention.

Titre	Emploi du temps
Sorte	P<boolean*7><boolean*48>
Référence	Boolean
Description	Définit un tableau non vide de booléens. Remarque : dans chaque emploi du temps il y a 7 jours x 24 heures x 2 demi-heures plages disponibles par semaine c'est à dire 336 booléens. Les opérations sont : Créer, Ajouter, Enlever, Union, Intersection, Complémentaire, Compter. L'opération **Créer** permet la création d'un nouvel emploi du temps vide. L'opération **Ajouter** permet l'ajout d'un événement dans l'emploi du temps, l'opération **Enlever** est donc son contraire. L'opération **Union** permet de rassembler tous les événements de plusieurs emplois du temps alors que l'**Intersection** permet de réunir les évènements qui se passent au même moment. L'opération **Complémentaire** ressort l'inverse de l'emploi du temps et affiche donc toutes les disponibilités. Ces trois opérations affichent ensuite le résultat dans un autre planning. L'opération **Compter** additionne tous les événements sur un même planning.

Signature

Créer : → P<boolean*7><boolean*48>

Ajouter : Créer X boolean → P<boolean*7><boolean*48>

Enlever : Créer X boolean → P<boolean*7><boolean*48>

Union : P<boolean*7><boolean*48> X P<boolean*7><boolean*48> → P<boolean*7><boolean*48>

Intersection : P<boolean*7><boolean*48> X P<boolean*7><boolean*48> → P<boolean*7><boolean*48>

Complémentaire : P<boolean*7><boolean*48> → P<boolean*7><boolean*48>

Compter : P<boolean*7><boolean*48> → Int

Axiomes

Compter(Créer) = 0
Compter(Ajouter(P1, Bool)) = compter(P1) + 1
Compter(Enlever(P1, Bool)) = compter(P1) - 1
Compter(Union(P1, P2)) = compter(P1) + compter(P2) - compter(intersection(P1, P2))
Compter(Intersection(P1, P2))= compter(P1) + compter(P2) - compter(union(P1,P2))

Conclusion

A travers ce cahier des charges, nous avons détaillé toutes les exigences nécessaires pour l'élaboration de cette librairie en Javascript.

Le contexte de la situation présente au sein de l'association Adomy nous a ainsi permis de définir toutes les spécifications fonctionnelles et non-fonctionnelles qui nous ont semblées importantes pour pouvoir réaliser les différents objectifs demandés par l'association.

La présentation des exigences en langage naturel a été faite selon la chronologie de ces dernières et dans le but qu'elles soient claires en réponse aux objectifs demandés.

La présentation des spécifications détaillées a permis de préciser leurs différentes contraintes d'implémentation. Nous avons ensuite défini tous les formats de données sous forme ABNF qui seront exploités par l'application.

Nous avons terminé par la définition des spécifications algébriques qui présente la signature et les axiomes de la syntaxe que nous avons décidé de prendre.