RAPPORT DE SOFTWARE ENGINEERING

GROUPE : YANN MARTIN D'ESCRIENNE ET YOHANN TOGNETTI

CONCERNANT LES METHODES DE TRAVAIL

REPARTITION DU TRAVAIL

Tout au long du projet, nous avons effectué le projet de façon équitable. Travaillant sur les mêmes plages horaires simultanément. Chacun travaillant à son tour sur tous les aspects du projet afin de couvrir tous les points vus en cours.

DECOUPAGE DU TRAVAIL

Le travail fut découpé en plusieurs phases.

Tout d'abord nous n'avons réaliser un **modèle** pour le calendrier et un ou plusieurs VEvent, la génération du fichier .ics par un **switch**, le **Xtext** et sa grammaire et enfin son implémentation graphique par **Sirius.**

Une fois le sujet du VEvent maitrisé et implémenté nous avons rajouté le **VTodo** , le **VJournal** ainsi que la **VTimeZone** en suivant les mêmes étapes que pour le VEvent (modèle -> switch -> xtext -> sirius).

FONCTIONNALITES IMPLEMENTEES

Restreint ou non restreint indique que les valeurs sont limitées à celle pouvant être affectées. Par exemple la méthode ne peut être que PUBLISH ou REQUEST.

Pour le calendrier :

- CALSCALE
- METHOD (non restreinte)
- PRODID
- VERSION

Les événements :

- SUMMARY
- UID
- DESCRIPTION
- STATUS (restreint)
- DTSTAMP
- DTSTART
- RRULE (restreint) avec FREQ, COUNT et BYDAY
- EXDATE
- CREATED
- LAST-MODIFIED
- DTEND
- TRANSP (non restreint)
- LOCATION

Les TODOS :

- SUMMARY

- UID
- DESCRIPTION
- STATUS (restreint)
- DTSTAMP
- DTSTART
- RRULE (restreint) avec FREQ, COUNT et BYDAY
- EXDATE
- CREATED
- LAST-MODIFIED
- DUE
- LOCATION
- PRIORITY

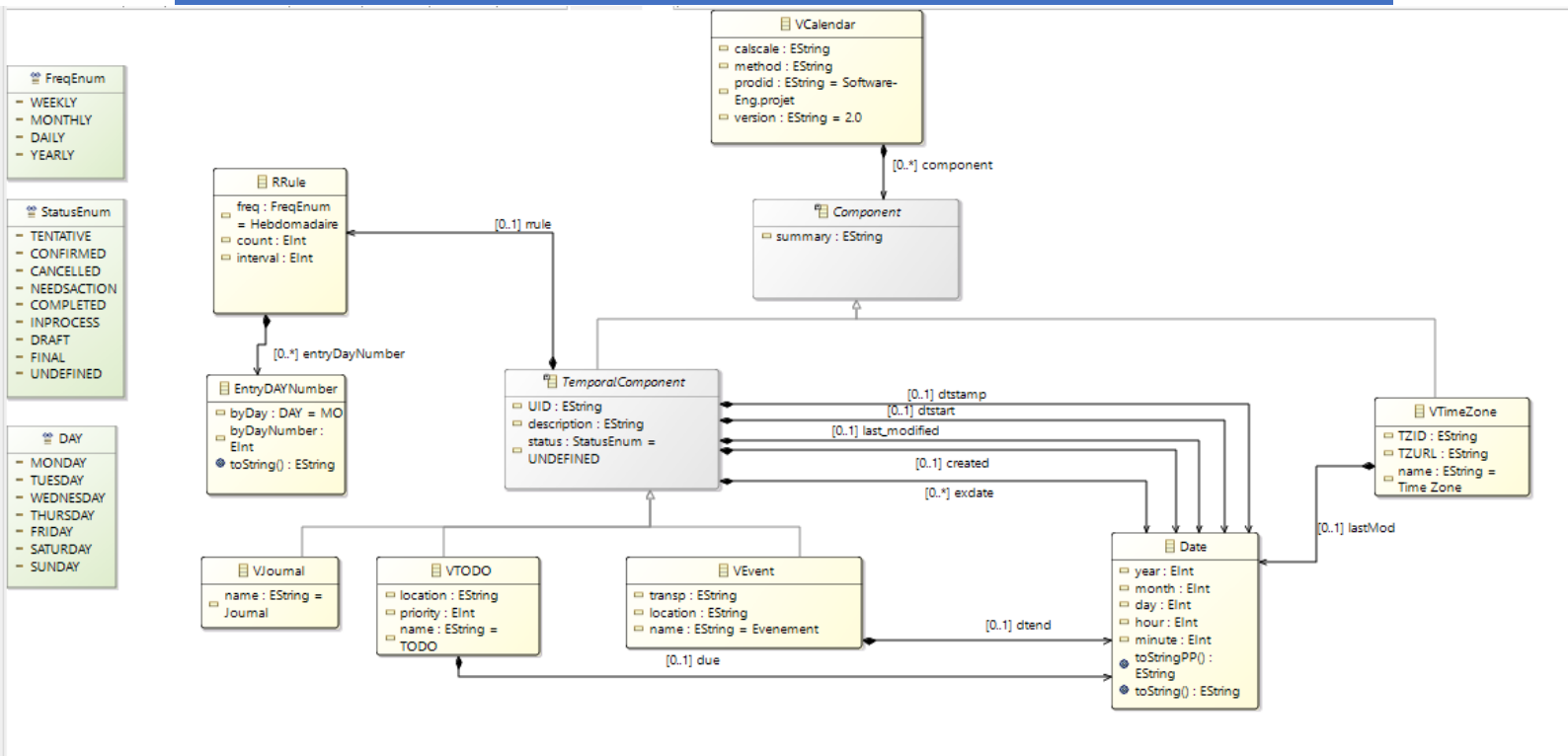
Les journaux :

- SUMMARY
- UID
- DESCRIPTION
- STATUS (restreint)
- DTSTAMP
- DTSTART
- RRULE (restreint) avec FREQ, COUNT et BYDAY
- EXDATE
- CREATED
- LAST-MODIFIED

Les time zone :

- TZID
- TZURL

LE MODELE



Dans l'optique de créer différent component ne partageant que très peu d'attribut, nous avons créer un niveau d'abstraction avec la classe abstract **Component** (Summary étant commun à tous les éléments)**.**

Viens ensuite une autre classe abstract **TemporalComponent** étant la classe mère de tous les components à teneur temporel.

Pour l'implémentation des **dates**, nous avons décider de créer une classe à part afin de pouvoir y ajouter des méthodes propres aux dates comme **toStringPP** : le toString pour le pretty-print respectant l'affichage voulu dans le RFC (*yyyymmddThhminsecZ*) et le **toString** : l'affichage user-friendly de la forme *dd/mm/yyyy, hh :min.* Mais aussi afin de les manipuler plus facilement.

Remarque : On voit que last modified ne se retrouve pas dans Component alors qu'il est commun à toutes les implémentations alors que ceci aurait tout à fait pu être le cas. Cela s'explique par le fait que nous voulions couvrir potentiellement plus de composant n'ayant pas last-modified comme properties.

Également, timezone possède un champ Summary alors qu'il n'en a pas, mais il reste inutilisé et bloqué dans la grammaire.

On remarque également que **RRule** possède également une classe à part, complémenter par **EntryDayNumber**. En effet étant une règle de répétition pouvant être complexe nous avons préférer sa séparation pour une manipulation aisée. Les dates pouvant être exclus se trouve néanmoins directement dans le TemporalComponent (exdate). **EntryDayNumber** correspond au BYDAY de la norme RFC (nous n'avons pas implémenter le BYMONTH et BYYEAR)

Certain champ comme prodid et version ont une valeur par défaut. (Car ne doivent pas être modifié normalement et sont toujours identique)

Chaque élément final possède d'ailleurs un champ name afin de pouvoir les identifiés. Cela est utilisé notamment dans notre représentation graphique avec Sirius.

On constate également de nombreuse énumération servant pour les restrictions des valeurs à choix fixe.

PRETTY-PRINT AVEC SWITCH

Il n'y a pas grand-chose à expliquer ici, on se contente d'afficher les informations données dans le format voulu dans la norme RFC 5545 en factorisant les parties communes des components dans la limite de EMF.

Le tout produit un fichier .ics qui est ouvrable depuis n'importe quelle application de calendrier.
L'emplacement ou le fichier à été créer est affiché dans la console de eclipse EMF.

XTEXT

```
CALENDRIER{
    METHODE : REQUEST
    SCALE : GREGORIAN
    TIMEZONE {
        TZID : "Europe/Paris"
    }
    TODO:"FINIR LE PROJET" {
        DESCRITPION : "Rendre le projet"
        CREER LE : 06/06/2021 à 14 h 30
        LIEU : "ent.unice.fr"
        POUR LE: 07/06/2021
        PRIORITE: 9
        STATUTS: EN COURS

    }


     EVENEMENT : Cours {
        UID : "89472b8b-785a-4f13-bd6b-e46d97f68161"
        DESCRITPION : "Un cours de master"
        STAMP : 06/06/2021 à 16 h 03
        DEBUT : 01/09/2020 à 14 h
        CREER LE : 6 / 6 / 2021 à 16 h 3
        RECURRENCE : TOUTES LES MOIS 6 FOIS SEULEMENT LE LUNDI ET LE 1 ER VENDREDI
        LIEU : Sophia
        FIN :01/09/2020 à 16 h 30
        STATUTS : ANNULE
     }

    EVENEMENT : "Cours 2" {
        UID : "a2c72b8b-785a-4f13-bd6b-e46d97f64761"
        DESCRITPION : "Un cours de master 2"
        STAMP : 06/06/2021 à 16 h 03
        DEBUT : 02/09/2020 à 14 h
        CREER LE : 6 / 6 / 2021 à 16 h 3
        RECURRENCE : TOUTES LES SEMAINES 6 FOIS
        SAUF LES 24/05/2020, 25/05/2020
```

Nous avons essayé de réaliser une grammaire sous forme de **pseudo-phrase** qui permet de générer facilement les différents composant du calendrier et leur propriété tout en restant un minimum structuré avec des ‘{‘ par exemple.

Tout les mots clefs sont en majuscules afin de différencier ce qui est « système » et « utilisateur »

Un effort à été réalisé sur **l’écriture des dates** dans un format plus compréhensible par un humain. De même pour les **règles de récurrence** pouvant être parfois complexe, ici sous forme de simple phrase regroupant d’un coup d’œil tous les éléments pour les règles de répétition.

L’utilisation des énumérations permet notamment une autocomplétions pour certaines propriétés (comme le statut)

De plus le statut n’a pas toujours les mêmes valeurs selon le type de components (event, todo…), nous avons alors restreint les choix possibles avec notre grammaire.

```
 95
 96 enum StatusVJournal returns StatusEnum:
 97     DRAFT = "BROUILLON" | FINAL = "FINAL" | CANCELLED = "ANNULE"
 98 ;
 99 enum StatusVEvent returns StatusEnum:
100     TENTATIVE = "TENTATIVE" | CONFIRMED = "CONFIRME" | CANCELLED = "ANNULE"
101 ;
102
103 enum StatusVTODO returns StatusEnum:
104     NEEDSACTION = "ACTION REQUISE" | COMPLETED = "COMPLETE" | INPROCESS = "EN COURS" | CANCELLED = "ANNULE"
105 ;
```

IMPORTANT : il est possible d’appeler la génération du pretty-print directement depuis le fichier .cal en faisant clic-droit dessus. C’est pour cela qu’il n’y a pas de projet .seProj dans le run-time eclipse.

SIRIUS

	TZID	Titre	Status	Description	Debut	Date Exclues	Fin	Lieu	Pour le	Priorité	Intervalle	Occur...	Seulement le
Time Zone	Europe/Paris												
TODO		FINIR LE PROJ	IN-PROCESS	*Rendre le projet*		[]		ent.unic	07/06/2021	9			
Evenement		Cours	CANCELLED	*Un cours de master*	01/09/2020 à 12h00	[]	01/09/2020 à 14h30	Sophia					
Mensuel											0	6	[Lundi, 1er Vendredi]
Debut													
fin													
Evenement		Cours 2	UNDEFINED	*Un cours de master 2*	02/09/2020 à 12h00	[24/05/2020, 25/05/202	02/09/2020 à 14h30	Sophia					
Hebdomadaire											0	6	[]
Debut													
fin													
date exclue													
date exclue													
Journal		Un journal	DRAFT	*le suivi du projet*	01/06/2021	[]							
Hebdomadaire											2	3	[]
Debut													
Journal		Un autre journ	UNDEFINED	*le suivi du projet*		[]							

Nous avons choisi la représentation de notre calendrier sous forme de tableau car elle est pour nous, la meilleure forme pour notre modèle.

Toutes les propriétés de chaque élément et toutes celle du calendrier n'y sont pas implémenter car cela ajouterait beaucoup de colonne et peu d'intérêt. Nous avons donc choisi les fonctionnalités les plus utiles et parlante pour un utilisateur.

Les règles de récurrence sont également affichées sous forme de sous-groupe du composant parent pour les cacher/afficher et simplifier la lecture du tableau.

Les dates se trouvent également dans ce sous-groupe afin de pouvoir modifier leur valeur, mais elles n'ont pas de colonne attribuée en plus dans le tableau (on les voit directement dans la ligne du composant)

Un code couleur rudimentaire permet de regrouper visuellement les informations :

- Violet pour les informations communes,
- Orange pour les informations d'un événement
- Blue pour les informations d'un to-do
- Rouge pour la time zone
- Gris pour ne pas lire les colonnes propres au Rrule affichées en jaune.

Tous les champs sont modifiables. Il se peut que quelques petites incohérences se produise comme « TOUT LES 1 EME LUNDI » dû à notre grammaire Xtext un peu trop laxiste et à la génération automatique.

Il est aussi possible d'ajouter des composants directement depuis le tableau. UID, LAST-MODIFIED et CREATED seront alors automatiquement remplis avec la date de création et un UID aléatoire (ce qui est aussi le cas dans le pretty-print pour UID)

Il est aussi possible de supprimer les lignes ou les éléments des lignes