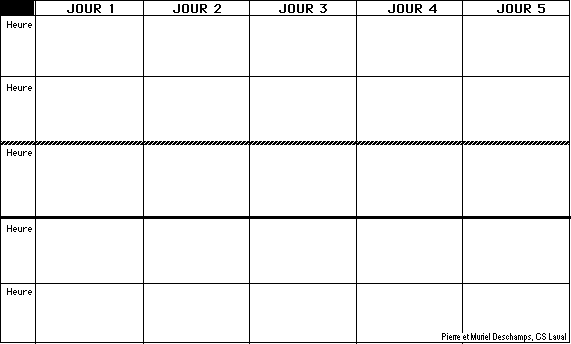


**Projet horaire**

*Rapport de conception*

**Présenté par : Sébastien Fournier, Frédéric Boivin, Émile Bergeron**

**Mise-à-jour : 04/10/2010**



Cégep de Chicoutimi

2010-09-30

Table des matières

[Introduction 3](#_Toc273967659)

[La base de données 3](#_Toc273967660)

[schéma de base de données 3](#_Toc273967661)

[Diagramme de classes 4](#_Toc273967662)

[Description 4](#_Toc273967663)

[Les ressources 4](#_Toc273967664)

[Les blocs 5](#_Toc273967665)

[Affichage de l’horaire 5](#_Toc273967666)

[L’onglet horaire 5](#_Toc273967667)

[L’onglet profil 5](#_Toc273967668)

[L’onglet paramètre 5](#_Toc273967669)

[La classe de gestion de la base de données 6](#_Toc273967670)

[L’Algorithme de génération d’horaire 6](#_Toc273967671)

[Le diagramme d’activités représentant l’algorithme 7](#_Toc273967672)

[Les maquettes 8](#_Toc273967673)

[Onglet HORAIRE 8](#_Toc273967674)

[Maquette de l’onglet Horaire 9](#_Toc273967675)

[Onglet EMPLOYÉ (profils) 9](#_Toc273967676)

[Maquette de l’onglet employé 10](#_Toc273967677)

[Onglet PARAMÈTRES 10](#_Toc273967678)

[Maquette de l’onglet paramètre - Généraux 11](#_Toc273967679)

[Maquette de l’onglet paramètre - Ressources 12](#_Toc273967680)

[Conclusion 13](#_Toc273967681)

Rapport de conception

# Introduction

Ce document contient tout le nécessaire pour pouvoir développer le projet. On y retrouve :

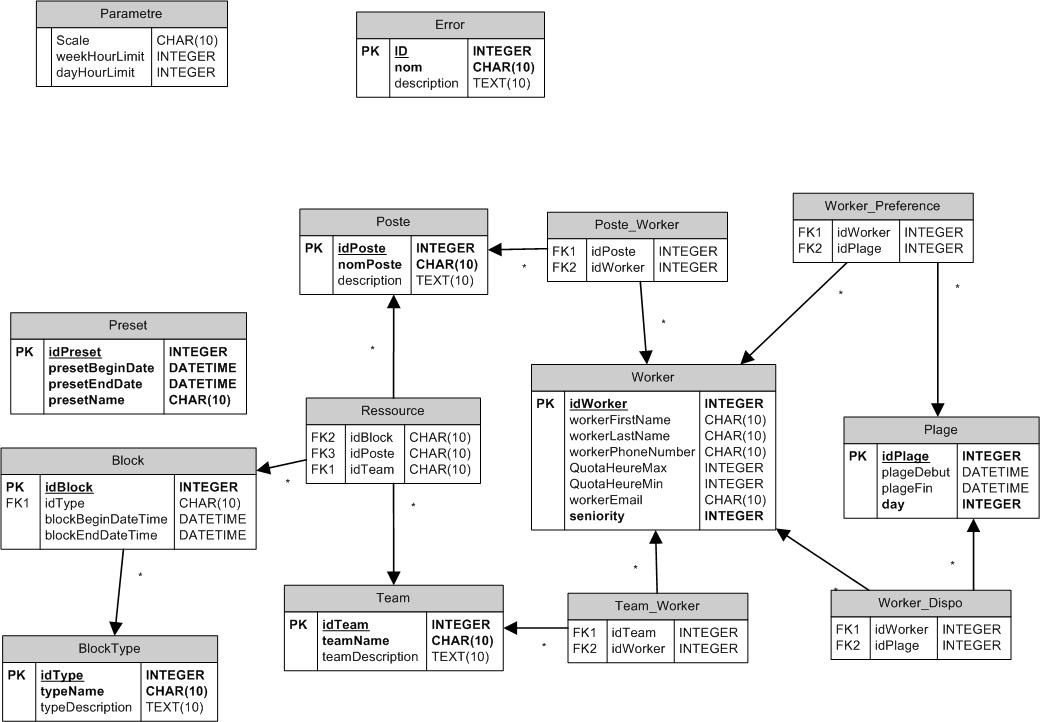
* Maquettes de l’application
* Schéma de la base de données
* Diagramme de classe de l’application
* Un diagramme d’activité représentant l’algorithme de génération d’horaire
* Et des descriptions éclaircissant chacune des figures

# La base de données

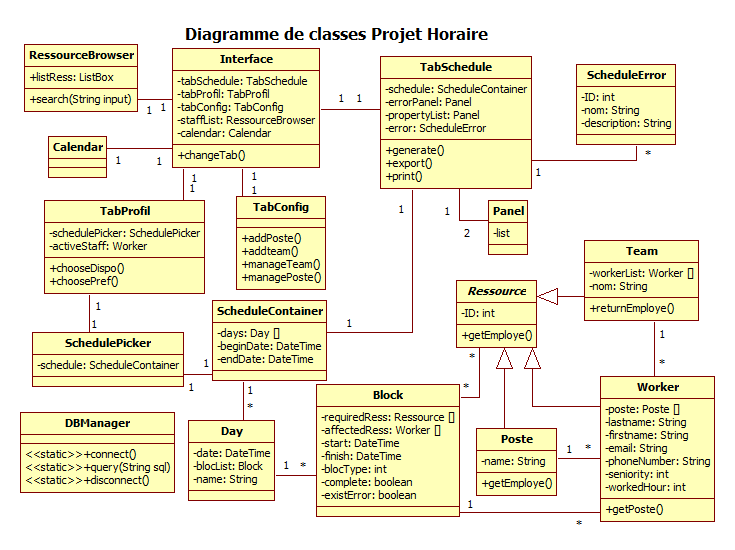
Dans la base de données, tout ou presque peut être relié à plusieurs éléments de d’autre table via des tables de jointures, ce qui donne des possibilités infinies pour le développement future de l’application.

Tout le diagramme tourne autour de la table Worker et Block, c’est les éléments fondamentaux de la base de données.

## schéma de base de données



# Diagramme de classes



## Description

Bien qu’à première vue, le diagramme de classe ressemble a une toile d’araignée, avec quelques explications, il devient très facile à comprendre. Le programme est divisé en 3 onglets principaux, on distingue facilement la classe Interface qui est l’interface principal contenant les onglets. Les classes des onglets se nomment TabSchedule, TabRessource et TabConfig. Chacune de ces classes possèdent des objets différents, bien que certains soit réutilisé.

Avant d’expliquer chacun des onglets en détails, voyons les classes de base pour bien comprendre leur utilité.

### Les ressources

Les ressources sont en fait composées de trois (3) types :

* Des employés : la classe Worker;
* Des équipes : la classe Team;
* Et des postes : la classe Poste.

Pas besoin d’explication pour employé, c’est explicite. Les équipes sont composées d’employés et les employés possèdent chacun un ou plusieurs postes selon leurs qualifications dans l’entreprise.

La classe Ressource est une classe abstraite qui est parent aux classes présentées ci-dessus.

### Les blocs

Lorsqu’il est question de placer des plages d’heure dans l’horaire, les blocs entrent en jeu. Les blocs sont représentés par la classe Block. Ils possèdent une date de début et de fin qui permet de déterminer leur longueur et leur emplacement dans le temps. Lorsque nécessaire[[1]](#footnote-1), les blocs peuvent être rempli de ressources à combler et éventuellement d’employés comblant la ressource à combler.

### Affichage de l’horaire

Pour cette fonction, c’est la classe ScheduleContainer qui entre en jeu. Elle est utilisé plus d’une fois dans le programme à des fins différentes mais touchant le placement de bloc.

Visuellement, la classe sera représentée par un tableau gradué en heures sur l’axe vertical et en jours sur l’axe horizontal. Les blocs seront placés vis-à-vis le jour et l’heure correspondant.

### L’onglet horaire

L’onglet horaire est représenté dans le diagramme par TabSchedule. Dans celui-ci, on y retrouve :

* un module d’affichage d’horaire, soit ScheduleContainer;
* une liste de recherche de ressource, soit RessourceBrowser;
* une zone pour la liste d’erreur (les erreurs sont représentées par ScheduleError mais la zone est un Panel);
* une liste de propriété des blocs (soit un Panel).

C’est dans cet onglet que l’horaire sera généré, les erreurs y seront affichées et gérées automatiquement ou manuellement par l’utilisateur selon le cas.

### L’onglet profil

L’onglet profil contient une interface représenté par TabProfil où il est possible de modifier les employés, c’est là que la classe ScheduleContainer est réutilisée pour paramétrer les disponibilités et les préférences d’un employé en y insérant des blocs vides.

### L’onglet paramètre

Dans cet onglet, il est possible de gérer les équipes, les postes et les paramètres généraux de l’application. Cet onglet est représenté par la classe TabConfig.

### La classe de gestion de la base de données

C’est la classe DBManager qui sert d’interface entre l’application et la base de données. Elle est statique et donc accessible de partout dans le code sans nécessiter d’appel ou de création d’objet. Elle permet la connexion, les requêtes et la déconnexion à la base de données.

# L’Algorithme de génération d’horaire

Lors de la génération d’horaire, le programme appelle la fonction generate() par l’entremise du bouton Générer dans l’onglet horaire. Celui-ciappelle alors la fonction *generate()* qui construit l’horaire de la période sélectionnée dans le calendrier.

La fonction commence par affecter les employés dans tous les blocs selon leurs disponibilités en les récupérant dans la base de données.

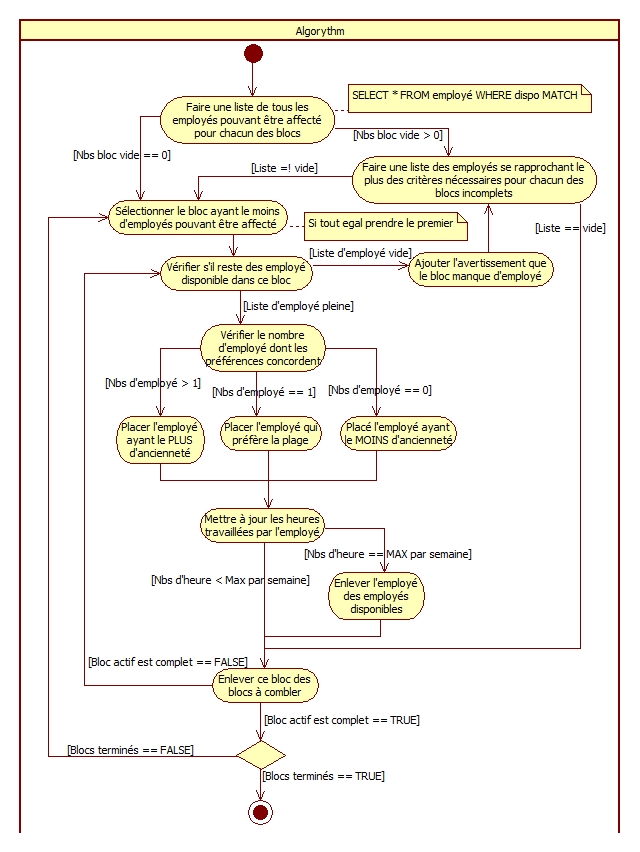
Par la suite generate() désigne le bloc possédant le moins d’employé disponibles et la fonction va trier les employés et en assigner un en prenant d’abord en compte l’ancienneté, puis les préférences.

Une fois l’employé assigné, la fonction ajuste son nombre d’heures et modifie les objets BLOCKS et WORKERS en conséquence.

Si un employé atteint son nombre maximum d’heures, il est retiré de la liste des employés disponibles. La fonction recommence l’opération jusqu’à ce que le bloc soit comblé, puis jusqu’à ce que tous les blocs soient comblés.

Lors d’une erreur, le système accède à la classe ScheduleError et affiche le message d’erreur dans la barre d’état au bas de l’horaire.

## Le diagramme d’activités représentant l’algorithme



# Les maquettes

## Onglet HORAIRE

Le bouton **Générer** appelle la fonction *generate()* qui construit l’horaire de la période sélectionnée dans le calendrier.

Le bouton **Imprimer** appelle la fonction *print()* qui va formater une page d’impression et **Exporter** permet d’exporter l’horaire en sélection en « JPG ».

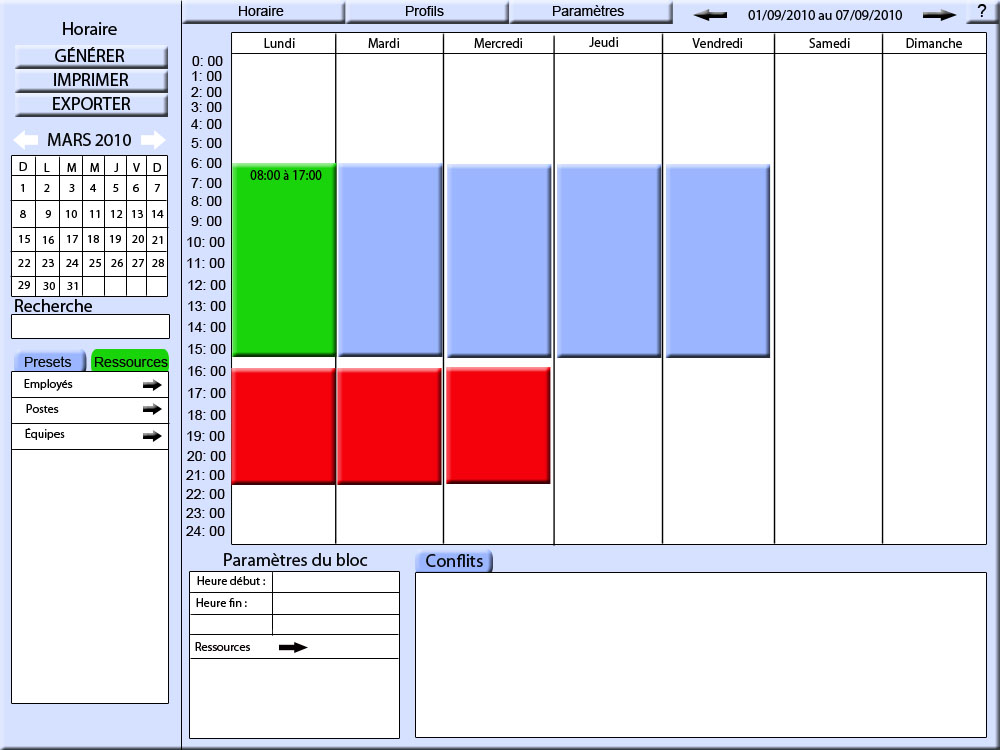
Le calendrier permet de naviguer à travers les mois et les années, il est basé sur la classe Calendar qui contient des fonctions génériques de calendrier. La classe est basée sur une librairie déjà existante. Si l’utilisateur clique sur un jour, l’écran affiche la semaine qui inclut la date, sinon l’utilisateur peut sélectionner une plage horaire.

La zone de recherche permet d’entrer un mot qui triera dynamiquement l’information dans le menu des ressources. Si l’utilisateur entre la lettre C, le menu affichera les mots débutants par C, si l’utilisateur rajoute « ui », le menu affichera tous les mots contenant Cui, par conséquent, à partir du champ de recherche, il parcoure la base de données et retourne les résultats qui correspondent.

Le menu Presets/Ressources permet de naviguer dans un menu contextuel qui contient les équipes/postes/employés. Si un bloc est sélectionné, cliquer sur une sélection l’ajoutera.

L’horaire affiche les blocs visuellement, et la graduation s’étend sur 24h par défaut mais est personnalisable.

### Maquette de l’onglet Horaire



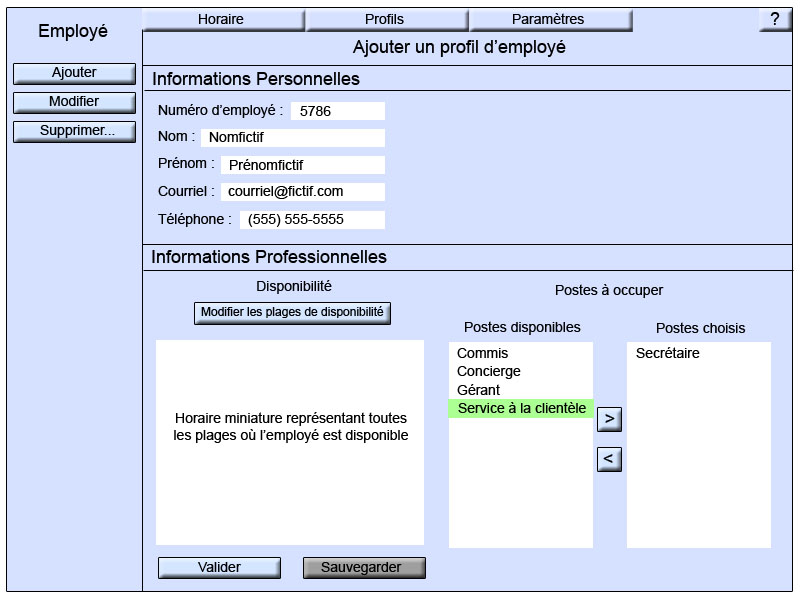
## Onglet EMPLOYÉ (profils)

Les boutons Ajouter/Modifier/Supprimer sont explicites. Ils permettent de manipuler les profils des employés en appelant leurs fonctions respectives. Un profil est ajouter/modifier ou supprimer de la base de données uniquement lorsque le formulaire est complet et validé, le programme envoie alors la requête correspondante afin de mettre à jour les tables nécessaires. Les informations personnelles permettent d’identifier et de contacter l’employé, principalement en cas de conflits d’horaire.

Les disponibilités et les préférences sont modifiables en cliquant sur l’Horaire miniature qui est une réutilisation des classes nécessaire à la génération d’horaire.

Les postes disponibles sont modifiables dans l’onglet paramètres.

### Maquette de l’onglet employé

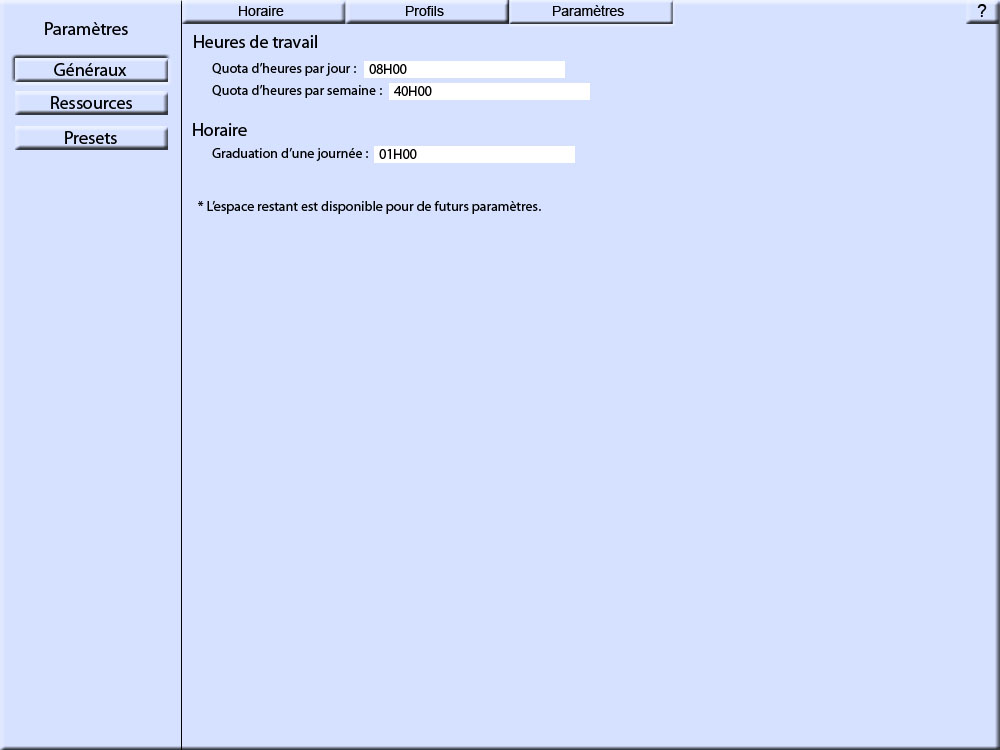


## Onglet PARAMÈTRES

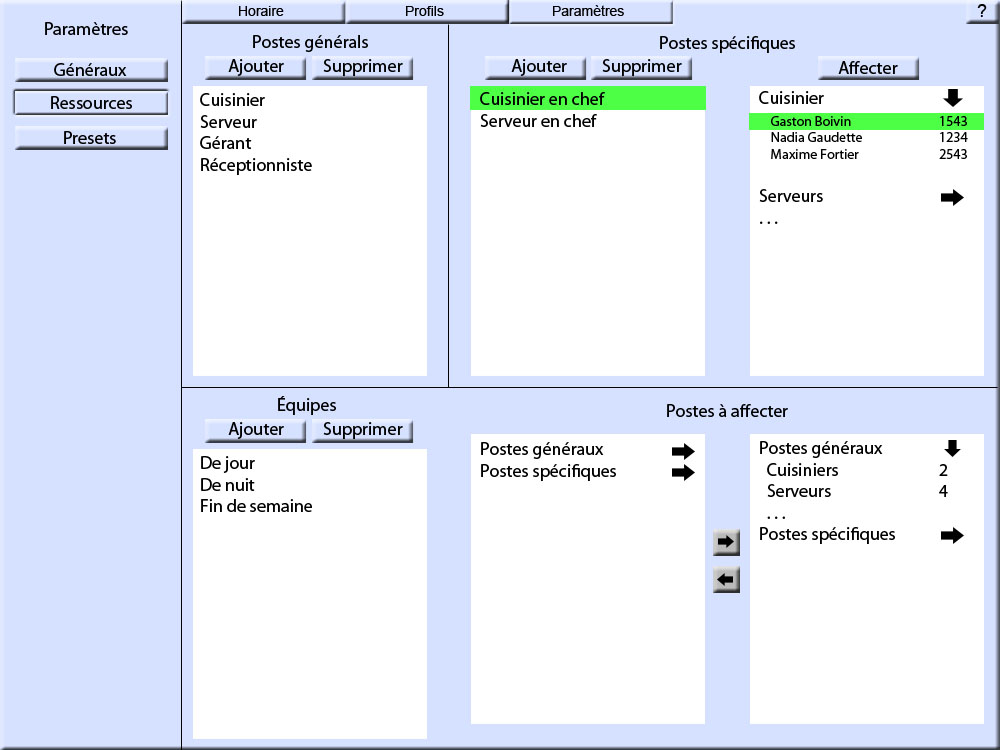
L’écran paramètre permet de personnaliser le logiciel selon certains critères. Tout d’abord, l’utilisateur doit choisir le quota d’heure par jour et par semaine par défaut pour les employés. Quant à l’horaire, l’utilisateur peut graduer l’échelle des heures.

Quant aux ressources (postes/équipes), ils doivent être ajouté dans l’onglet prévu à cet effet, un click sur le bouton ajouter ouvre un pop-up qui demande les informations (i.e. : nom) et une fois validé, la nouvelle ressource est accessible dans la base de données.

### Maquette de l’onglet paramètre - Généraux



### Maquette de l’onglet paramètre - Ressources



# Conclusion

En somme, l'application se résume à une gestion de données efficace et au placement de certaines de ces données dans un horaire à l'aide d'un algorithme. Au niveau des difficultés que l'on pourrait rencontrer en plein développement, Il se pourrait que l'interface soit complexe à produire. Vu son utilisation poussé de la souris, la sélection dynamique et une grande utilisation de boutons, il faudra s'assurer d'avoir la technologie pour tout supporter.

Un langage de programmation qui gère bien le graphisme serait de mise comme le C++ et ses dérivées (C#), Visual Basic, Java, etc. Au niveau de la base de données, étant donné le nombre de relations possibles entre les tables, Il faudra prévoir un système de gestion de base de données permettant les tables InnoDB ou bien tout autre moteur gérant les relations; SQLite ou autre gestionnaire de base de données léger peuvent suffirent.

Bref, si l'on réussit bien l'interfaçage et l'utilisation de l'algorithme, le tout reste entièrement faisable dans un délai acceptable.

1. Il est possible d’insérer un bloc sans ajouter de ressource juste pour délimiter une période de temps. [↑](#footnote-ref-1)