Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet collectif

2017-2018

Rapport de projet technique: Projet "Bienvenue au DI"

Analyse et conception de l'application sources ADE

Tuteurs académiques
Mohamed Slimane
Jean-Yves Ramel

Étudiant
Xavier Bouchenard (DI4)

07 Mai 2018

Liste des intervenants

Nom	Email	Qualité
Xavier Bouchenard	xavier.bouchenard@etu.univ-tours.fr	Étudiant DI4
Mohamed Slimane	mohamed.slimane@univ-tours.fr	Tuteur académique, Département Informatique
Jean-Yves Ramel	jean-yves.ramel@univ-tours.fr	Tuteur académique, Département Informatique

Avertissement

Ce document a été rédigé par Xavier Bouchenard susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Mohamed Slimane et Jean-Yves Ramel susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnait assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnait qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document

Xavier Bouchenard, *Rapport de projet technique: Projet "Bienvenue au DI": Analyse et conception de l'application sources ADE*, Projet collectif, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2017-2018.

```
@mastersthesis{
    author={Bouchenard, Xavier},
    title={Rapport de projet technique: Projet "Bienvenue au DI": Analyse et conception de
      l'application sources ADE},
    type={Projet collectif},
    school={Ecole Polytechnique de l'Université François Rabelais de Tours},
    address={Tours, France},
    year={2017-2018}
}
```

Document mis en forme sous LaTeX
grâce à la classe de document `polytech.cls` V2.2.5 (2017/05/02)

Table des matières

Table des figures

1 Introduction

Ce rapport a pour but d'expliquer et d'apporter des informations sur la structure ainsi que sur les choix techniques qui ont été faits quant au développement de cette application. Une documentation utilisateur sera fournie pour aider l'utilisateur à s'approprier l'application.

Ce rapport s'inscrit dans le cadre du projet collectif "Bienvenue au DI", dont le but est de développer un système basé client-serveur, qui permettent à des utilisateurs lambda de trouver le chemin le plus rapide pour se rendre à l'endroit voulu. Une autre utilisation est pouvoir trouver un professeur dans l'établissement, à partir du moment où il est présent.

L'application principale ou dite "serveur", a besoin de sources d'informations pour pouvoir ensuite renseigner l'utilisateur par rapport à la requête effectuée. D'où découle la nécessité de développer une application permettant de renvoyer les informations utiles de l'emploi du temps du département concerné, en se basant sur le fichier ADE récupérable à partir de l'URL disponible sur l'ENT. Cette application permet de construire l'emploi du temps journalier, consistant en une liste de cours ordonnés par heure de début, pour chaque département renseigné par l'utilisateur et d'envoyer ces informations au **serveur**.

Cette application a donc été développée en **Java** car une librairie de lecture et fait pour le parse de fichier "*.ics" existait déjà, il s'agit de **ical4J**, qui est une librairie open-source récupérée à partir d'un dépôt sur **SourceForge**, dont l'URL est le suivant :
https ://sourceforge.net/projects/ical4j/ .

Dans une première partie, nous verrons les spécifications caractérisant l'application à concevoir. Ensuite, une partie sera dédiée à l'architecture qui en découle, ainsi que les choix techniques effectués.

2 Spécifications de l'application

Cette application doit répondre à des besoins particuliers car elle n'a pour but que l'envoi de données pour alimenter l'application serveur.

Elle doit donc :

1. **être simple d'utilisation** : pour éviter de faire perdre du temps à l'administrateur

2. **avoir des options limitées** : permettre les options essentielles comme la création d'une base d'emploi du temps ou la modification. Il n'est donc pas nécessaire d'afficher après création, le contenu de la liste de cours pour le département.

3. **être exécuté en mode console** : il n'est pas nécessaire de développer un outil graphique autour de ces fonctionnalités car il n'y a aucun affichage spécifique à faire.

4. **être entièrement autonome** : c'est-à-dire qu'elle ne se bloquera pas, dans l'attente d'une action de la part de l'administrateur.

5. **envoyer les données automatiquement** : une fois qu'une base d'emploi du temps est créée et que la liste de cours est construite, il faut qu'elle soit automatiquement envoyée à l'application serveur.

6. **faire les mises à jour automatiquement** : nous avons décidé de fixer une durée de 2 heures comme contrainte pour faire mettre à jour de façon automatique les listes de cours pour chaque bases. Elle sera faite pour toutes les bases, pour vérifier si une modification de l'emploi du temps de la journée en cours a été faite ou non.

Comme dit précédemment, une librairie open-source pour la lecture des fichiers fourni par ADE existe déjà, **ical4J** le permet en Java et est plutôt facile à utiliser et à prendre en main.

La majorité des informations présentées précédemment, sont résumées dans le diagramme des cas d'utilisations ci-dessous :

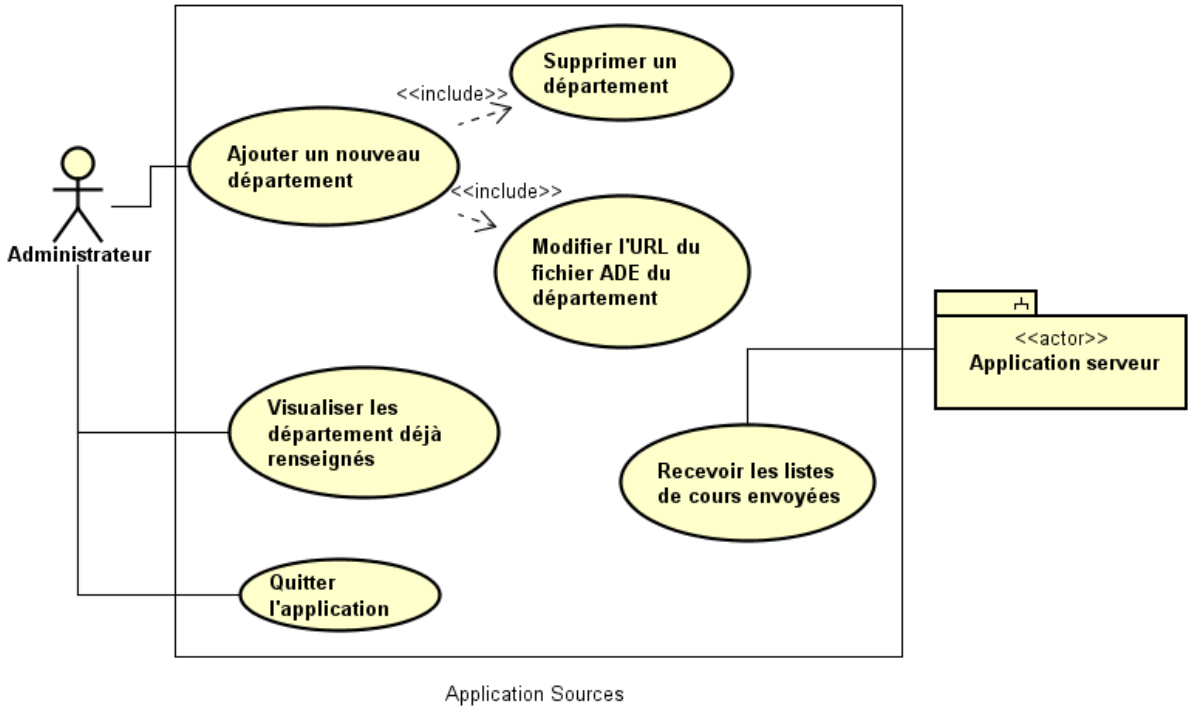

Figure 1 – *Diagramme des cas d'utilisations*

Comme on peut le voir, l'administrateur est le seul utilisateur de ce système. Il peut exécuter plusieurs différentes actions comme l'ajout d'une nouvelle base d'emploi du temps, dont le nom et l'URL du fichier ADE pour la récupération des informations. Un ajout inclut la modification ou la suppression de cette base s'il n'y a plus de raisons de la garder.

Il peut également voir le nom des bases déjà présentes, une par département, ou bien de quitter l'application. Le fait de quitter l'application permet aux informations qui ne sont pas enregistrées d'être sérialisées (écrites dans un fichier sous une forme particulière) pour éviter de perdre la trace de ce qui a déjà été construit.

Cette application va aussi avoir une liaison avec l'application serveur, pour laquelle elle construit le(s) liste(s) de cours pour le(s) département(s) présent(s) et le(s) envoie.

3 Architecture de l'application et choix faits

1 Choix effectués

Cette application tourne majoritairement autour des commandes faites par l'utilisateur, mais sans pour autant l'empêcher de fonctionner en autonomie.

Une interface textuelle, via l'invite de commandes, sera disponible pour traiter les éventuelles demandes de l'administrateur.

Le temps sera également gérée de façon indépendante, pour permettre la mise à jour automatique de toutes les bases de données stockées, ce qui assouplit le système concernant d'éventuelles demandes de mises à jour en boucle.

Un système de parseur et de constructeur de listes de cours a été implémenté grâce à l'utilisation de la librairie **ical4J**. Toutes les informations extraites par ce parseur seront stockées dans les listes de cours correspondantes.

Une partie dédiée à la communication TCP est prévue. Cette application hébergera un client TCP qui se connectera au serveur TCP de l'application serveur, seulement dans le cas où des informations sont à transmettre.

La partie suivante sera donc dédiée à l'architecture prévue et développée pour cette application.

2 Architecture de l'application

Le diagramme des classes correspondant à cette application est le suivant :

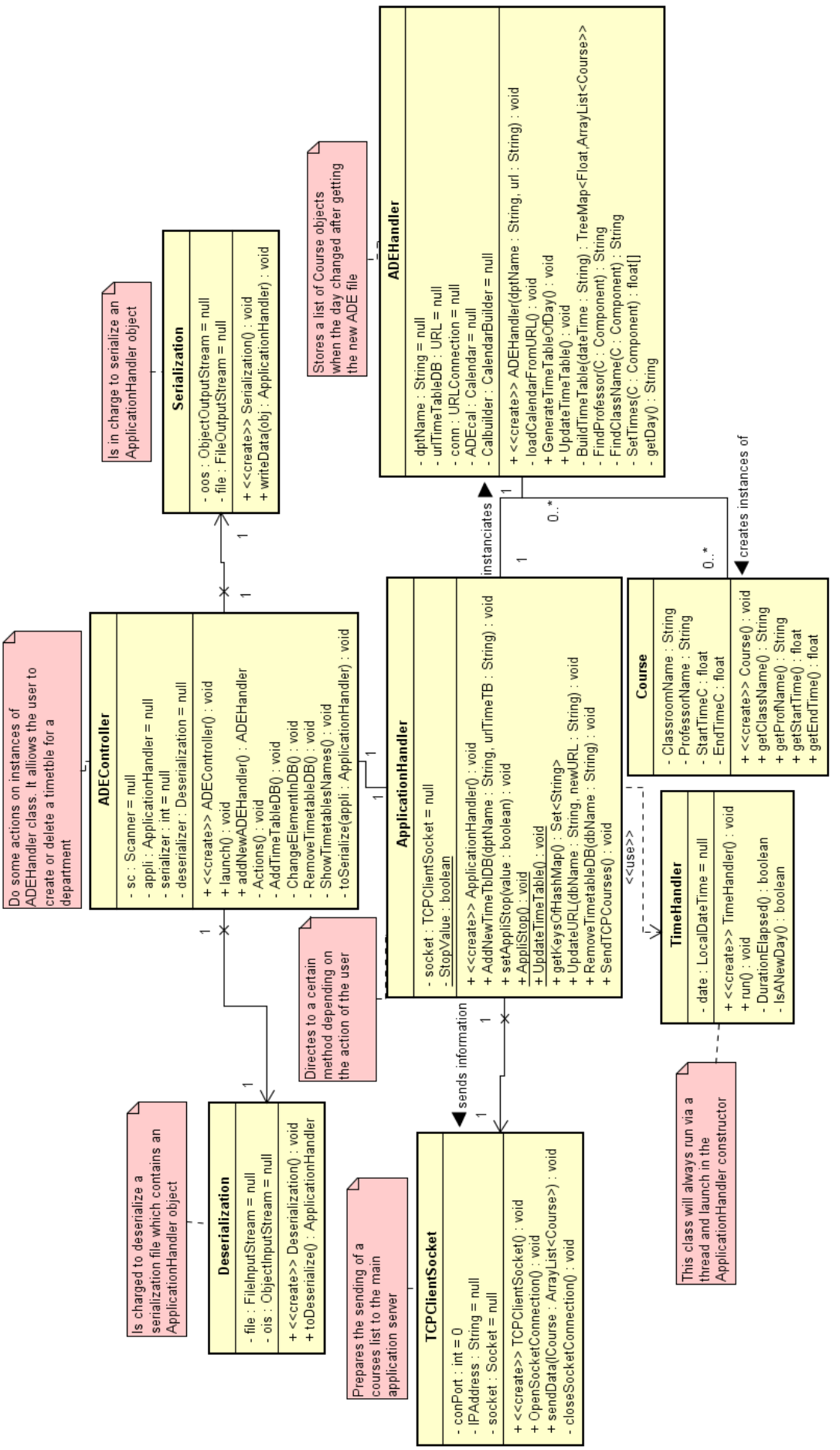

Figure 1 – *Diagramme des classes*

L'application est donc composée des classes :

1. **Course :** composée d'une heure de début, de fin et du nom du professeur

2. **ADEHandler :** va gérer l'ouverture, l'extraction des informations et va construire, par la suite, la liste de cours journalier sous la forme d'une **TreeMap** qui stocke une liste de cours construite par heure de début. C'est cette liste qui sera ensuite envoyée à l'application serveur.

3. **ApplicationHandler :** va gérer les différents états dans lesquels peut se trouver l'application. Il s'agit de la classe la plus importante, c'est le controlleur global de l'application qui coordonne toutes les actions. Elle est constituée d'une **HashMap** contenant le nom de la base d'emploi du temps (nom du département), ainsi que d'un objet de type **ADEHandler** contenant, lui, la liste de cours. Pour chaque département, un objet de type **ADEHandler** lui est affecté et lui est propre. Seules les listes de cours seront envoyés à l'application serveur.

4. **TCPClientSocket :** servira à envoyer les listes de cours ainsi que le nom du département associé, uniquement dans le cas d'un ajout ou d'une mise à jour de toutes les bases d'emplois du temps. Une connexion de ce type sera ouverte dans chaque cas présenté précédemment.

5. **TimeHandler :** va gérer le temps entre 2 périodes de temps distinctes et va mesurer le temps écoulé entre elles. Si la durée maximale est dépassée, alors une méthode statique du controleur est appelée pour effectuer une mise à jour globale de toutes les bases d'emplois du temps stockées dans **ApplicationHandler**.

6. **Serialization :** va gérer la sérialisation dans le cas où l'utilisateur demande la fermeture de l'application. A ce moment-là, les classes **ApplicationHandler, ADEHandler et Course** ayant implémenté l'interface **Serializable**, seront sérialisés. Ce système permettra de pouvoir récupérer le fichier et de recharger les bases et les informations liées directement sans devoir tout rentrer à nouveau, ce qui évite une perte de temps et d'informations.

7. **Deserialization :** va gérer la désérialisation des données présentes dans le fichier de sérialisation, pour pouvoir récupérer et reconstituer les bases d'emplois du temps avec la liste de cours. L'application cherche automatiquement à lire et à désérialiser les données à son démarrage pour les récupérer.

8. **ADEController :** va gérer les actions de l'utilisateur et va aiguiller vers les méthodes du controleur **ApplicationHandler** pour le traitement de ces actions.

Cette architecture permet une certaine souplesse car elle permet de gérer plusieurs bases d'emplois du temps simultanément et de manière automatique (lecture du fichier ADE, construction de la liste de cours,...). Un autre avantage est que l'utilisateur peut demander à faire des traitements en parallèle pendant que l'application fait une mise à jour globale de toutes les bases d'emplois du temps stockées, par exemple.

Rapport de projet technique: Projet "Bienvenue au DI"

Analyse et conception de l'application sources ADE

Résumé

Afin d'alimenter la base d'informations de l'application principale, une application permettant de récupérer et de filtrer les informations présents dans un fichier ADE a été développée. Cette application stocke toutes les informations et les URLs fournies par l'utilisateur pour construire une liste de cours pour la journée en cours pour tous les départements renseignés. Ces listes seront ensuite envoyées à l'application principale qui utilisera les données par la suite.

Mots-clés

Java, ical4J, librairie, architecture, utilisateur, communination

Abstract

To update the information base of the main application, this application alllows the system to read and get all the data sources. Then, it will build and store a new daily timetable per Polytech department which the user asked for and send it to the main application by TCP socket. The system administrator is the only one who can use this application. It is designed to be the most convenient to use as possible. After some actions, the application is able to do some processings without the need of the user.

Keywords

Java, ical4J, Jars sources, software architecture, TCP socket

Tuteurs académiques
Mohamed Slimane
Jean-Yves Ramel

Étudiant
Xavier Bouchenard (DI4)