FACULTÉ DES SCIENCES
UNIVERSITÉ D'ANGERS

Architecture distribuée

Loann Cady

Projet CRM
Master II Informatique CD

11 Janvier 2024

Professeur:
Martin Dieguez Lodeiro

Table des matières

1 Architecture globale

1.1 Module InternalCRM

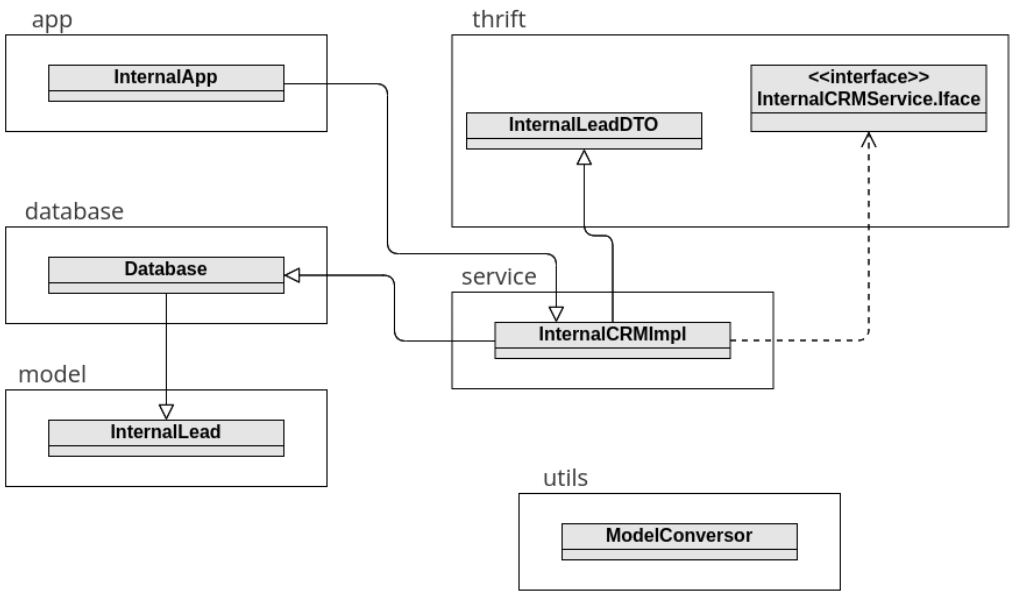

FIGURE 1 – Diagramme de classe du module InternalCRM

Le module InternalCRM est un module qui exploite l'API Apache Thrift pour mettre en œuvre des services spécifiques, notamment "findLeads", "findLeadsByDate", "addLead" et "deleteLead" dans la classe InternalCRMImpl qui implémente l'interface du service Thrift InternalCRMService. Ces services ont pour objectif de récupérer des prospects (leads) à partir d'une base de données.

findleads permet de chercher les leads dans la base de donnée qui ont un certains revenu et un certains état. findLeadsByDate permet quant à lui de chercher un lead en fonction de sa date. addLead et deleteLeads permettent enfin de rajouter ou d'enlever un Lead de la base de donnée.

La base de données en question est représentée par une simple classe Java (Database) qui contient une liste d'objets InternalLead. InternalApp, la classe principale, permet grâce à Apache Thrift de démarrer un serveur dans lequel on peux se connecter et communiquer pour appeler les services. Ainsi, en plus de la classe InternalCRMService un DTO (data transfer object) InternalLeadDTO est mis en place. La classe ModelConversor permet de simplement effectuer des conversion entre les beans InternalLead et les DTO InternalLeadDTO.

Au niveau de l'architecture, InternalCRM est composé de 6 packages, le package app principal contenant InternalApp, la classe main. Le package database avec une

classe Database stockant une liste d'InternalLead dans le package model. Un package utils utilisé pour stocker des fonctions utiles en majorité statiques. Enfin, un package service contenant l'implémentation des services dans la classe InternalCRMImpl. Il reste un package thrift généré automatiquement par Apache Thrift contenant notre DTO InternalLeadDTO et les services InternalCRMService.

1.2 Module VirtualCRM

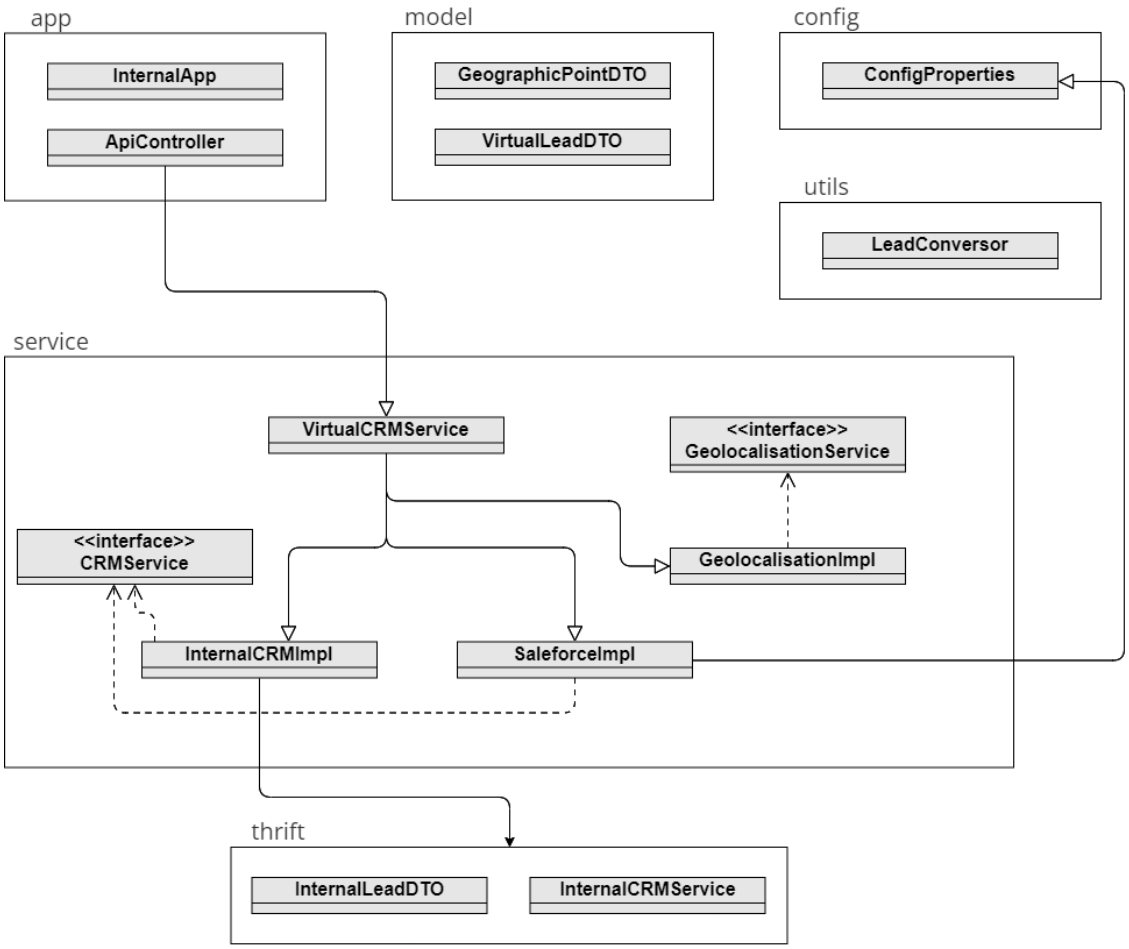

FIGURE 2 – Diagramme de classe du module VirtualCRM

Le module VirtualCRM est un module qui met en place un serveur Java Spring-Boot et met à disposition des services sous la forme d'une API.

VirtualCRM est un CRM qui ne dispose pas de donnée mais qui effectue des requêtes pour récupérer des données grâce à l'API SaleForce et sur l'InternalCRM. Les classes InternalCRMImpl et SaleforceImpl qui implémentent CRMService, implémentent les

méthodes findLeads et findLedsByDate afin de récupèrer respectivement les données de l'InternalCRM et de SaleForce d'après certains critères. SaleforceImpl permet aussi de récupérer un token de connexion avant d'effectuer une requête.

La classe VirtualCRMService implémentant aussi CRMService utilise InternalCRMImpl et SaleforceImpl afin de récupérer les données des deux services et de les rassembler. On utilise ensuite le service de géolocalisation dans la classe GeolocalisationImpl implémentant GeolocalisationService qui appelle l'API d'openstreetmap pour compléter les données de géolocalisation (latitude, longitude) des leads récupérés.

On renvoie ensuite, à celui qui a fait la requête, les données des leads au format JSON que l'on a récupéré.

Au niveau de l'architecture, VirtualCRM est composé de 6 packages, le package app principal contenant InternalApp, la classe main et l'ApiController. Le package model contenant nos beans VirtualLead et GeographicPoint sont utilisés par les services dans le package service contenant les implémentations des services de l'InternalCRM et Saleforce et l'implémentation de la géolocalisation dans la classe GeolocalisationImpl. Un package config contient une classe ConfigProperties utilisé par SaleforceImpl pour récupérer les informations Saleforce (ClientID, ClientSecret, ...) stockées dans un fichier application.properties. Les deux derniers packages thrift et utils sont utilisés de la même façon que dans l'InternalCRM.

1.3 Module CommandLineTool (Client)

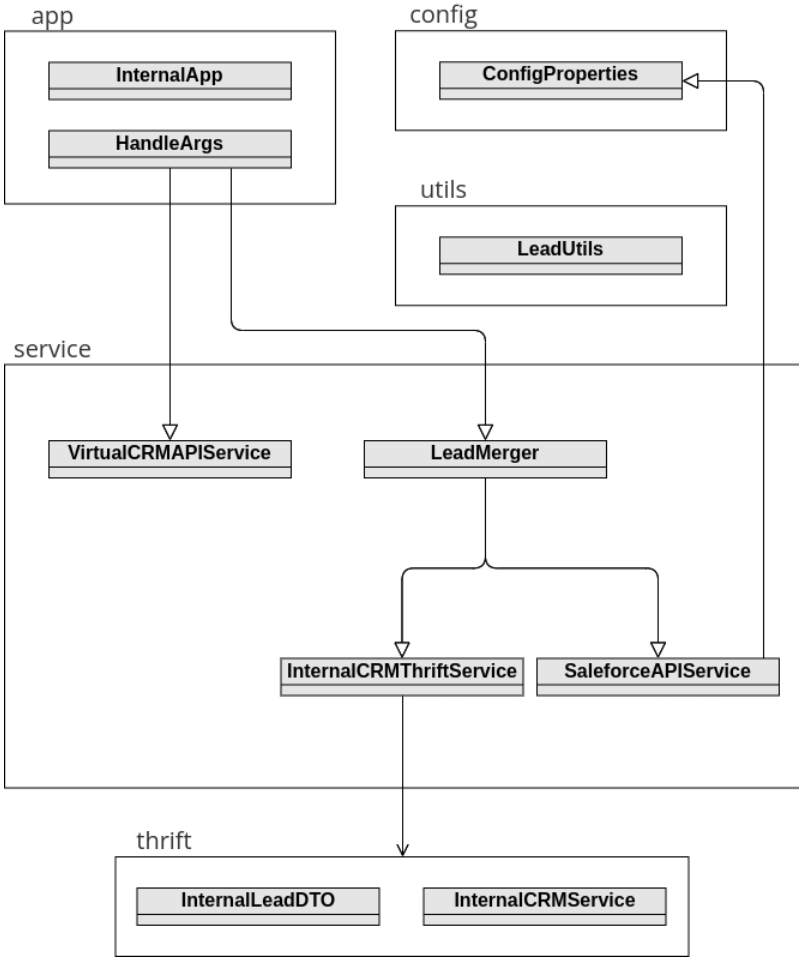

FIGURE 3 – Diagramme de classe du module CommandLineTool

Le module CommandLineTool permet pour la première partie d'effectuer des requêtes sur l'API du VirtualCRM, et donc de récupérer les leads de l'InternalCRM et de Saleforce. De plus, pour la deuxième partie, il contient la classe LeadMerger, qui permet de récupérer tout les leads sur Saleforce et de les ajouter à la base de donnée de l'InternalCRM.

2 Services

2.1 InternalCRM

InternalCRM met à disposition 4 services **findLeads**, **findLeadsByDate**, **addLead** et **deleteLead**. Ces services sont construits de sorte à ce que InternalCRMImpl implémente l'interface InternalCRMService.Iface généré par Apache Thrift.

En détail, findLead récupère dans l'objet Database une liste d'InternalLead qui correspondent à un revenu supérieur à lowAnnualRevenue et inférieur à highAnnualRevenue, et à un état. Ensuite, grâce à ModelConversor il transforme la liste d'InternalLead en liste d'InternalLeadDTO. Enfin, InternalCRMImpl retourne la liste de ces DTO et les envoie par l'API thrift au Client.

findLeadsByDate fonctionne de la même façon. La seule différence est qu'il récupère les leads étant entre deux dates prises en arguments de la fonction. addLead et deleteLead prennent en argument un InternalLeadDTO et servent à supprimer un InternalLead en particulier dans la base de donnée.

2.2 VirtualCRM

findLeads et **findLeadsByDate** sont assez similaire. Comme on peut le voir dans le diagramme de séquence de la fonction findLeads (ref 4), une fois la requête effectuée par le client, notre ApiController reçoit la requête puis invoque la fonction findLeads de la classe VirtualCRMService. Celui-ci instancie premièrement un objet InternalCRMImpl qui effectue grâce à thrift une demande au client de InternalCRMService afin de récupérer les leads de l'InternalCRM.

Une fois la liste d'InternalLeadDTO récupérée, on la transforme en liste de VirtualLeadDTO qui est retournée au VirtualCRMService. Ensuite, le VirtualCRMService instancie un objet SaleforceImpl, qui s'occupe de récupérer un token d'identification, puis d'effectuer une requête à Saleforce pour récupérer des leads en format JSON. Ce JSON est transformé en liste de VirtualLead et est retournée au VirtualCRMService.

Une fois les leads de l'InternalCRM et de Saleforce récupérés, nous les assemblons en une seule liste de VirtualLead. Un objet GeolocalisationImpl est instancié et on appelle la fonction setGeolocalisation sur la liste de VirtualLead. Cette fonction permet d'effectuer une requête OpenStreetMap récupérant et assigne la latitude et longitude de chaque VirtualLead. Une fois cette liste retournée au VirtualCRMService, elle est retournée à l'ApiController qui retourne au client une liste de VirtualLead en format JSON.

2.3 CommandLineTool (Client)

Le module CommandLineTool permet de récupérer des arguments via la fonction Main de la classe principale et d'effectuer des actions. Les arguments findLeads et findLeadsByDate permettent de faire des requêtes sur l'API du VirtualCRM et de récupérer les leads de l'InternalCRM de manière simple, similaire à un curl. Au niveau du mergeLeads (cf 5), la classe LeadMerger fonctionne comme un petit VirtualCRM, en instanciant deux variables SaleforceAPIService et InternalCRMThriftService. Comme dans le VirtualCRM, SaleforceAPIService permet la récupération d'un token d'authentification puis la récupération de leads. L'InternalCRMThriftService permet grâce à la technologie de Thrift d'appeler la fonction addLeads de l'InternalCRM et ajoute les leads que l'on avait récupéré dans Saleforce.

3 Compilation

(Voir README.md)

3.1 Run InternalCRM and VirtualCRM using gradlew

VirtualCRM :

```
./gradlew :VirtualCRM:appRun
```

InternalCRM :

```
./gradlew :InternalCRM:run
```

Run both VirtualCRM and InternalCRM :

```
./gradlew --parallel :VirtualCRM:appRun :InternalCRM:run
```

3.2 Run CommandLineTool (Client) using gradlew

Arguments

```
findLeads lowAnnualRevenue highAnnualrevenue State
findLeadsByDate startDate(YYYY-MM-DD) endDate(YYYY-MM-DD)
mergeLeads
```

Execution

```
./gradlew :CommandLineTool:run --args="arguments here"
```

Exemple :

```
./gradlew :CommandLineTool:run --args="findLeads 0 900000000 NC"
```

3.3 Run modules with bash-command-line-tool

Arguments

appRun : Run InternalCRM & VirtualCRM
findLeads lowAnnualRevenue highAnnualrevenue State
findLeadsByDate startDate(YYYY-MM-DD) endDate(YYYY-MM-DD)
mergeLeads

Execution

./bash-command-line-tool args

4 Liens

Github - CRMProjet
README.md

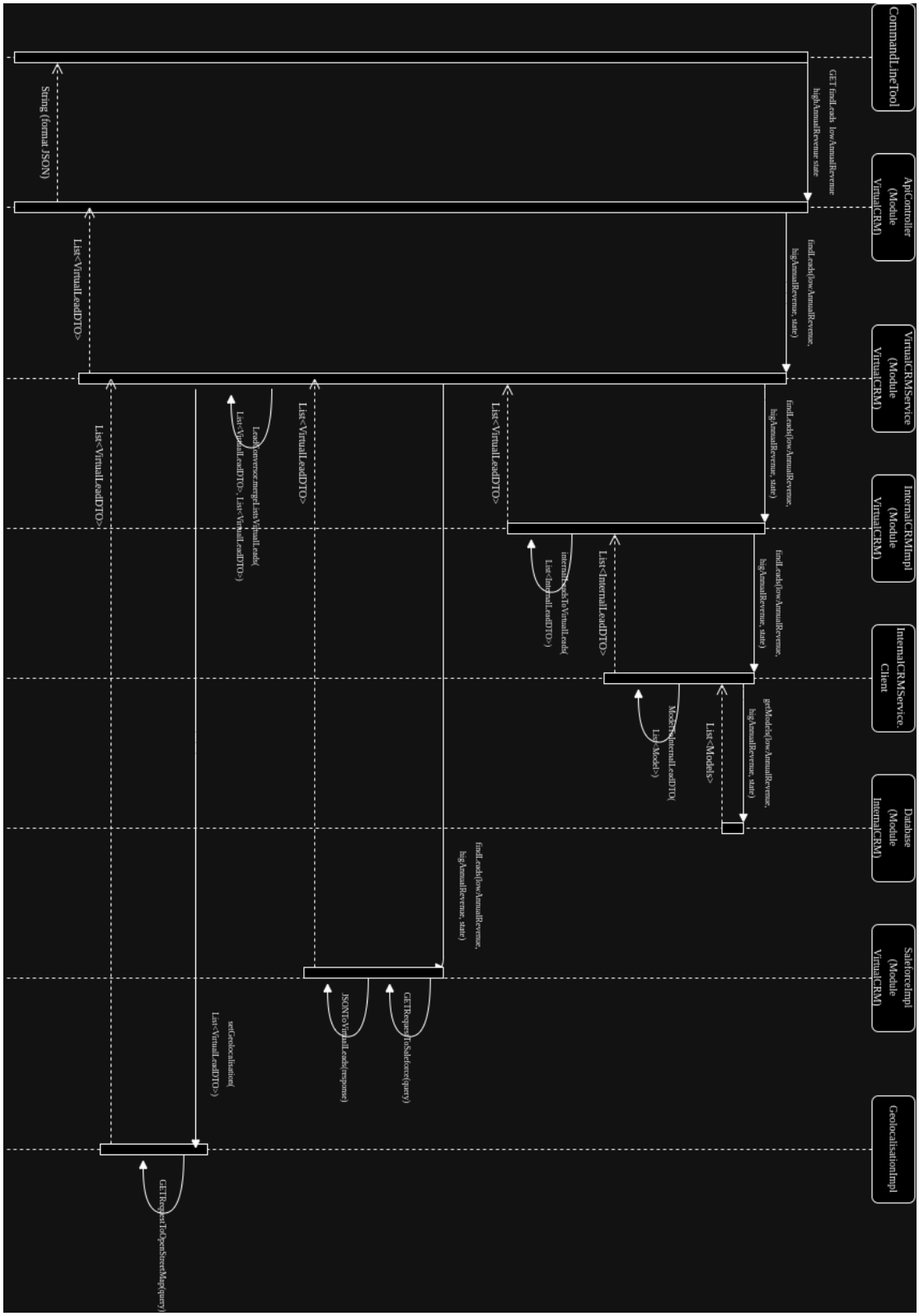

FIGURE 4 – Diagramme de séquence du service findLeads

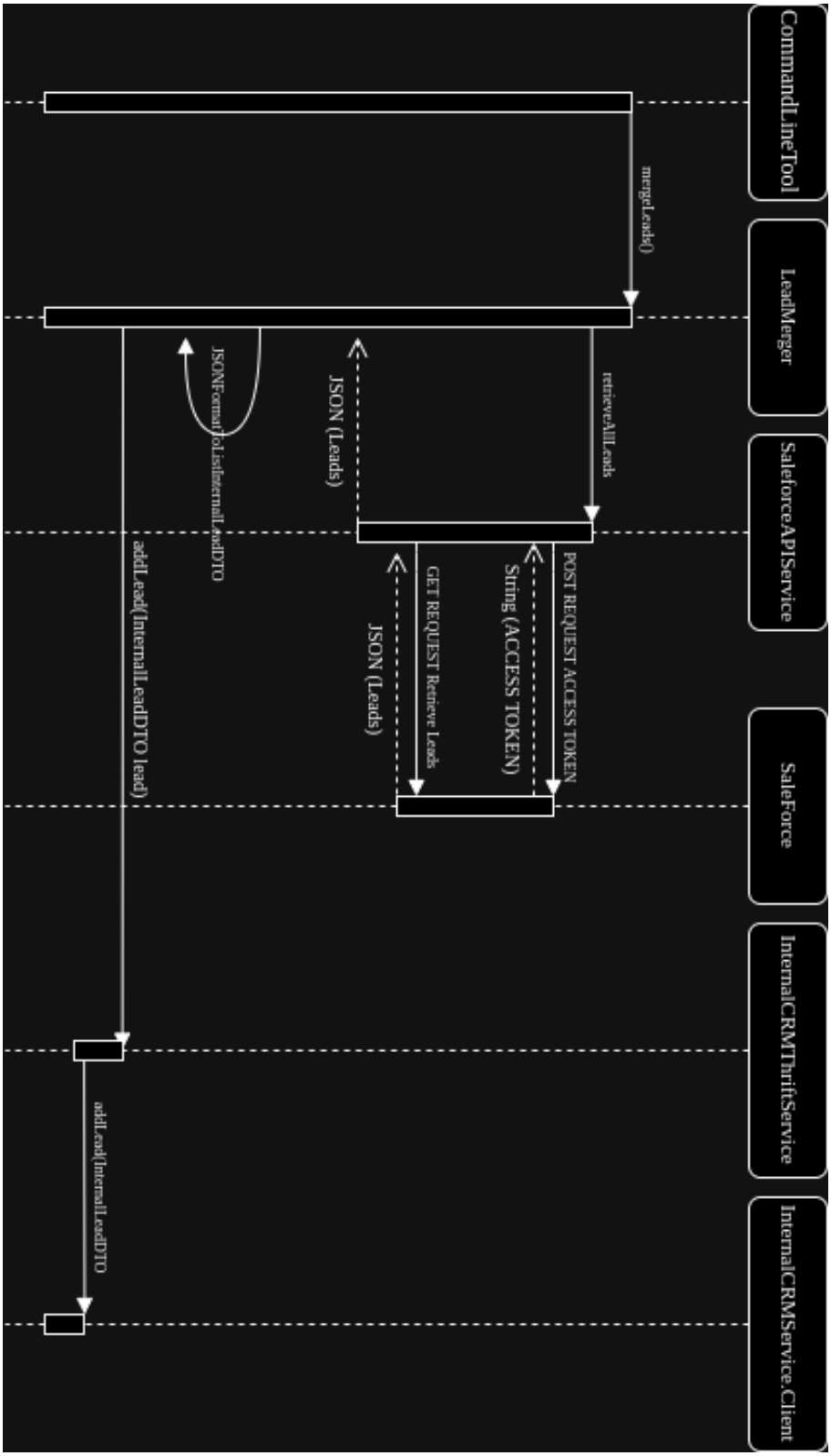

FIGURE 5 – Diagramme de séquence du service mergeLeads