

***TIC – Filière Informatique***

**Projet de semestre 6**

**2018**

**Rapport**

**S.A.S**

**Développement d'une application Web de surveillance  
et de soutien contre l'addiction en ligne**

Logo

**Nicolas Fuchs  
Grégory Ducrey**

Superviseurs : **Dr.** **Sandy Ingram**

**Dr. Houda Chabbi**

Mandant : **Dr. Claire Korkmaz**

**Dr. Sariah Haddad**



Fribourg, Mars 2018

Sommaire

[Historique des versions 4](#_Toc514342908)

[1. Introduction 5](#_Toc514342909)

[1.1. Contexte 5](#_Toc514342910)

[1.2. But du projet 6](#_Toc514342911)

[1.3. Document 6](#_Toc514342912)

[2. Analyse des besoins et des technologies 7](#_Toc514342913)

[2.1. Introduction 7](#_Toc514342914)

[2.2. Idéation 8](#_Toc514342915)

[2.3. Etat de l’art 9](#_Toc514342916)

[2.3.1. Self control 10](#_Toc514342917)

[2.3.2. Contrôle parental Mac os 11](#_Toc514342918)

[2.3.3. Rescue Time 12](#_Toc514342919)

[2.3.4. Moment 13](#_Toc514342920)

[2.3.5. Contrôle parental Swisscom 14](#_Toc514342921)

[2.3.6. Internet Security Swisscom 15](#_Toc514342922)

[2.3.7. Synthèse 16](#_Toc514342923)

[2.4. Tests technologiques 17](#_Toc514342924)

[2.4.1. Analyse des processus 17](#_Toc514342925)

[2.4.2. Analyse des sites web visités 19](#_Toc514342926)

[2.5. Conclusion 23](#_Toc514342927)

[3. Conception 24](#_Toc514342928)

[3.1. Introduction 24](#_Toc514342929)

[3.2. Cas d’utilisation 25](#_Toc514342930)

[3.2.1. Prototype 1, génération d’un fichier de log 26](#_Toc514342931)

[3.2.2. Prototype 2, création d’une application pour l’utilisateur 28](#_Toc514342932)

[3.2.3. Prototype 3, création d’une application pour le superviseur 32](#_Toc514342933)

[3.3. Structure des fichiers Json 39](#_Toc514342934)

[3.3.1. Json du temps passé par activité 39](#_Toc514342935)

[3.3.2. Json du temps passé par heure de la journée 40](#_Toc514342936)

[3.4. Conclusion 41](#_Toc514342937)

[4. Implémentation 42](#_Toc514342938)

[4.1. Introduction 42](#_Toc514342939)

[4.2. Prototype 1 43](#_Toc514342940)

[4.2.1. Extension Chrome 43](#_Toc514342941)

[4.2.2. Détection des processus 46](#_Toc514342942)

[4.2.3. Détection de la catégorie 47](#_Toc514342943)

[4.2.4. Ecriture du fichier log 50](#_Toc514342944)

[4.2.5. Diagramme de classes 52](#_Toc514342945)

[4.3. Prototype 2 53](#_Toc514342946)

[4.3.1. Génération des fichiers jsons 54](#_Toc514342947)

[4.3.2. Récupération des fichiers jSons 55](#_Toc514342948)

[4.3.3. Affichage des informations des Json dans des graphiques 56](#_Toc514342949)

[4.4. Prototype 3 58](#_Toc514342950)

[4.4.1. Détection des utilisateurs du même réseau 59](#_Toc514342951)

[4.4.2. Gestion de la liste des utilisateurs supervisés 59](#_Toc514342952)

[4.5. Interfaces homme machine 61](#_Toc514342953)

[4.6. Limites 63](#_Toc514342954)

[4.7. Conclusion 64](#_Toc514342955)

[5. Validation 65](#_Toc514342956)

[5.1. Introduction 65](#_Toc514342957)

[5.2. Tests d’utilisabilité 65](#_Toc514342958)

[5.3. Test utilisateur 65](#_Toc514342959)

[5.4. Test empiriques 65](#_Toc514342960)

[5.5. Conclusion 65](#_Toc514342961)

[6. Problèmes rencontrés 65](#_Toc514342962)

[7. Perspectives 66](#_Toc514342963)

[8. Conclusion 66](#_Toc514342964)

[8.1. Conclusions de groupe 66](#_Toc514342965)

[8.2. Conclusions personnelles 66](#_Toc514342966)

[8.2.1. Grégory Ducrey 66](#_Toc514342967)

[8.2.2. Nicolas Fuchs 66](#_Toc514342968)

[9. Annexes 66](#_Toc514342969)

[9.1. Déclaration d’honneur 66](#_Toc514342970)

[9.2. Bibliographie 66](#_Toc514342971)

# Historique des versions

Cet historique indique à quelles dates ont eu lieu les modifications du cahier des charges :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Semaine** | **Date** | **Personnes** | **Chapitres** | **Modification** |
| 0.1 | P3 | 05.03.18 | G. Ducrey N. Fuchs | 1, 1.1, 1.2, 1.3 | Création de la table des matières,  Introduction |
| 0.2 | P4 | 12.03.18 | G. Ducrey N. Fuchs | 2, 2.4 | Introduction à l’analyse, état de l’art et synthèse |
| 0.3 | P5 | 19.03.18 | G. Ducrey N. Fuchs | 2.4, 2.5 | Etat de l’art et tests technologiques |
| 0.4 | P6 | 29.03.18 | G. Ducrey N. Fuchs | 3 | Conception |
| 0.5 | P7 | 09.04.18 | G. Ducrey N. Fuchs | 3 | Conception |
| 0.6 | P8 | 16.04.18 | G. Ducrey N. Fuchs | 3 | Correction de la doc selon les retours |
| 0.7 | P9 | 23.04.18 | G. Ducrey N. Fuchs | 3, 4 | Fin de la conception et début de l’implémentation |
| 0.8 | P10 | 30.04.18 | G. Ducrey N. Fuchs | 4 | Implémentation |
| 0.9 | P12 | 14.05.18 | G. Ducrey N. Fuchs | 4 à 9 | Fin de l’implémentation  Validation  Problèmes rencontrés  Perspectives  Conclusions  Annexes |

# Introduction

Ce chapitre présente les aspects suivants du projet :

* Le contexte et les motivations de réalisation du projet
* Les objectifs
* Le document

## Contexte

Selon le monitorage suisse de l'addiction, un peu plus de 7% des jeunes Suisses entre 15 et 19 ans utilisent le Web de manière compulsive et problématique (addiction aux jeux de rôle en ligne, jeux d'argent, négligence des devoirs et des activités hors-ligne). L'objectif du projet est de concevoir et développer une application minimisant l'effet des stimulis addictifs, et redonnant le contrôle aux "consommateurs". Pour cela, un monitoring automatique des activités du consommateur (avec son consentement) doit être mise en place suivi de mesures de sensibilisation proposées en temps réel.

Pour détecter si une personne est dépendante aux jeux vidéos, il y a plusieurs critères qui sont les suivants selon wikipédia [1] :

* Monopolisation de la pensée par le projet de comportement addictif
* Intensité et durée des épisodes plus importants que souhaités à l'origine
* Tentatives répétées pour réduire, contrôler ou abandonner le comportement
* Temps important consacré à préparer les épisodes, à les entreprendre ou à s'en remettre
* L'engagement dans le comportement est tel que la personne ne peut plus accomplir des gestes élémentaires (se laver, se nourrir) et le conduit vers un désinvestissement social, professionnel et familial
* Poursuite du comportement malgré l'aggravation des problèmes sociaux et en dépit de la connaissance des conséquences négatives
* Tolérance marquée, c'est-à-dire besoin d'augmenter l'intensité ou la fréquence pour obtenir l'effet désiré, ou diminution de l'effet procuré par un comportement de même intensité
* Agitation, irritabilité et surtout angoisse si le passage à l'acte addictif est différé, empêché

On constate que pour ces gens-là, les conséquences néfastes peuvent être les suivantes :

* Appauvrissement de la vie sociale
* Manque de sommeil
* Trouble du comportement (saute d’humeur…)
* Mauvaise hygiène de vie
* Perte du sens des réalités
* Atteinte à la santé physique (maux de tête, yeux secs, douleurs au dos, manque d’activité physique, depression, suicide)

Ces liens contiennent quelques témoignages qui permettent de comprendre l’importance de la gestion des stimulis addictifs pour les joueurs. En effet, il n’est pas néfaste en soit de jouer, mais il est absolument nécessaire de garder le contrôle.

<http://www.jeuxvideo.com/dossiers/00018270/l-addiction-aux-jeux-video-temoignages-de-joueurs-006.htm>[2]

<https://www.vice.com/fr/article/ppvzd7/dependance-jeux-video-843>[3]

## But du projet

Le projet SAS (Supervision, Addiction, Soutien) consiste à développer un produit permettant aux utilisateurs de reprendre le contrôle lorsqu’ils sont soumis à des stimulis addictifs quand ils utilisent des jeux vidéos ou des réseaux sociaux.

L’enfant ou l’adolescent est en construction et il n’a pas encore tous les filtres lui permettant de comprendre les enjeux du temps passé sur un écran. C’est pourquoi cette application permettra de mettre un cadre où le joueur sera confronté à une responsabilisation de ses choix. A chaque âge le cadre doit être différent. Il sera donc plus restrictif pour des enfants que des adolescents.

Ce travail est réalisé dans le cadre du projet de semestre 6 en informatique à la haute école d’ingénieurs et d’architectes de Fribourg. Il est réalisé par M.Nicolas Fuchs et M.Grégory Ducrey pour les mandantes Dr. Claire Korkmaz, Dr. Sariah Haddad et il est supervisé par Mme Sandy Ingram ainsi que Mme Houda Chabbi.

## Document

Ce rapport fait partie intégrante du projet. Il contient des informations concernant les étapes de réalisation du produit final. La création de ce document se fait tout au long du projet. Il doit permettre au client de suivre les démarches de travail et si le projet devait être repris, le rapport donne toute les clés au développeur. Le document contient des informations sur les phases suivantes :

* Analyse
* Conception
* Implémentation
* Validation

Le rapport permet aux évaluateurs de vérifier la qualité et la rigueur du travail effectué.

# Analyse des besoins et des technologies

## Introduction

Nous ne connaissons pas énormément le sujet de la prévention des addictions aux jeux vidéos et aux réseaux sociaux, c’est pourquoi il est important d’étudier la base de ce sujet avant de réfléchir aux développement d’un produit pertinent.

Le client souhaite une application qui permet de monitorer et redonner le contrôle à l’utilisateur afin que ce dernier fasse des choix et en prenne l’entière responsabilité. Dans le chapitre 1.1 traitant du contexte du projet, des témoignages sont recueillis et ils attestent des dangers de ces addictions. En observant les critères d’addictions décrits, nous pouvons constater que le danger découle du temps passé sur les jeux ou les réseaux sociaux. C’est pourquoi nous avons en partie orienté nos recherches sur des moyens pour gérer le temps.

Nos recherches doivent nous aider à créer un produit répondant aux besoins de manière pertinente. Dans cette analyse nous avons effectué les travaux suivants :

* Exploration du sujet 🡪 Se renseigner sur le contexte, des statistiques
* Etat de l’art 🡪 Explorer les produits de prévention présents sur le marché
* Test de produits existants 🡪 Tester certains de ces produits
* Tests technologiques 🡪 Tester certaines méthodes qui pourraient nous aider à implémenter des fonctionnalités (Récupération des processus, récupération des URL…)
* Idéation 🡪 Sur la base du travail accompli, faire un brainstorming de ce qui serait pertinent d’implémenter pour ce produit
* Cahier des charges 🡪 Après ces 5 semaines d’analyse, établir un cahier des charges final décrivant le produit que nous allons réaliser en ayant une meilleure connaissance du sujet

## Idéation

Pour commencer ce projet, nous nous sommes dans un premier temps renseignés sur le sujet des addictions (voir chapitre « 1 introduction ») afin d’avoir une meilleure vue d’ensemble. Une fois fait, nous avons effectué un brainstorming [4] des idées de fonctionnalités qui pourraient être intéressantes lors du développement de l’application. Ce brainstorming est effectué avant les tests d’applications existantes et de tests technologiques. Il a pour but de laisser aller notre créativité sans être limité par des recherches déjà effectuées.

Voici les résultats du brainstorming, nous avons classé les idées par catégories :

**Concept de l’application**

* Contrôle des sites visités
* Gestion du temps
* Architecture client/serveur et paramétrisation de l’application native depuis la page web du serveur
* Système de récompense
* Contrôle parental
* Encouragement à la mobilité physique
* Blocage de certains processus en fonction d’une localisation GPS
* Validation par l’utilisateur

**Fonctionnalités concrètes**

* Monitoring du temps
  + - Paramétrer en fonction de l’âge
* Module statistique —> pour l’utilisateur et/ou contrôleur
  + - Temps passé sur une semaine, et sur quel sujet
* Alerte visuelles et sonores ET avertissement sur les dangers
  + - Forcer des pauses
    - Vidéo de sensibilisation
    - Blocage du processus durant la vidéo
    - L’utilisateur est prévenu de l’interruption
* Filtrer les sites visités (avec une liste, ou avec un thème)
* Vérification de l’USB (Si le jeu est lancé par disque)

**Moyens techniques**

* Récupération des processus
* Chronométrage du temps d’activité des processus
* Interruption de processus
* Récupération d’adresse IP et de localisation

**Plateforme**

* Application mobile multiplateforme
* Natif mac os
* Natif PC
* Natif Linux
* Jeux vidéos WEB

**Moyens de prévention**

* Liste des sites/processus sensibles à permettre/interdire
* Jeux

## Etat de l’art

Nous avons exploré plusieurs applications existantes permettant de mieux gérer le temps passé sur l’ordinateur ou le smartphone. Le but de cette observation est de se rendre compte de ce qui existe sur le marché et réfléchir à des fonctionnalités que nous pourrions apporter en plus. Pour pouvoir analyser correctement, nous avons examiné les aspects suivants :

* Le type de blocage
* Orienté web/natif/mobile 🡪 Si l’application travaille sur les navigateurs, sur les mobiles ou en natif sur l’ordinateur. Il est possible qu’une application native monitore des navigateurs.
* La plateforme sur laquelle le logiciel fonctionne
* Base volontaire ou restrictions 🡪 est-ce que l’application a tendance à interdire ou à sensibiliser
* Public cible 🡪 quel type de personne est susceptible d’utiliser le logiciel ?
* Paramétrer en fonction de l’âge 🡪 Est-ce possible de changer le monitoring ou les restrictions en fonction de l’âge de l’utilisateur ?
* Redonne le contrôle 🡪 est-ce que l’utilisateur peut décider ce qu’il fait ?
* Difficulté de prise en main 🡪 critère subjectif
* Administration depuis la même machine
* Fournit des statistiques d’utilisation
* Avis des utilisateurs

Ainsi, sur la base de ces observations nous pouvons comparer différentes applications et nous rendre compte de l’utilité des fonctionnalités pour notre client. Nous ne souhaitons pas plagier mais nous inspirer de ce qui existe déjà et créer un produit adapté aux besoins de nos clients.

Les applications testées proviennent des sites [5] à [12] de la bibliographie.

### Self control

|  |  |
| --- | --- |
| ../image%20applications/selfcontrol.png  **Self control [6]** | |
| **Utilisation** | |
| L'utilisateur choisit les sites webs ou les services mail à bloquer. Il choisit également la durée. Une fois fait il active la restriction et tant que le temps n'est pas écoulé c'est impossible d'accéder à ce qui est bloqué. (même en cas de suppression de l'application où de redémarrage). | |
| **Critères** | **Observations** |
| Blocage | Bloque les sites web (blacklist ou whitelist). Bloc les mails. Ne bloque pas des applications |
| Orienté web/natif/mobile | Application native orientée web |
| Plateforme | Fonctionne sur Mac OS |
| Base volontaire ou restrictions | L'utilisateur décide lui-même ce qu'il autorise ou non |
| Public cible | Personnes ayant la maturité de savoir ce qui est bon pour elles ou non. L'administration depuis une autre machine n'est pas possible |
| Paramétrer en fonction de l'âge | Il est possible d'adapter la liste des sites à bannir. |
| Redonne le contrôle | Redonne le contrôle au bout d'un certain temps |
| Difficulté de prise en main | Très facile à utiliser |
| Administration depuis la même machine | Il faut activer le logiciel sur la session à bloquer |
| Fournit des statistiques d'utilisation | Non, il ne fournit pas de statistiques |
| Avis des utilisateurs | Bon logiciel |
| Coûts | Le logiciel est gratuit |
| Opensource | Oui |
| Offline | Est inutile hors ligne puisqu’il monitore uniquement ce qui est en ligne |
| **Remarques complémentaires** | |
| L'outil est vraiment très simple. Il serait certainement utile pour se couper des distractions durant une journée de travail par exemple. L'utilisateur doit avoir une certaine maturité. **On a accès au code de l'application** | |

### Contrôle parental Mac os

|  |  |
| --- | --- |
| **../image%20applications/contrôleOS.png**  **Contrôle parental Mac os [7]** | |
| **Utilisation** | |
| L'utilisateur de la session administrateur donne des droits aux utilisateurs des autres sessions en activant le contrôle parental. | |
| **Critères** | **Observations** |
| Blocage | Assignation de plages horaires et de durée d'utilisation de l'ordinateur. Blacklist/whitelist) de sites web et limitation des sites web pour adultes. Blocage d'applications et de stores |
| Orienté web/natif/mobile | Application native |
| Plateforme | Fonctionne sur Mac OS |
| Base volontaire ou restrictions | Ne fonctionne que sur la base de restrictions. L'utilisateur de la session contrôlée ne peut pas changer quoi que ce soit aux restrictions, il subit des interdits. |
| Public cible | Les enfants particulièrement. Ce serait plutôt sur un ordinateur familial où il y a plusieurs sessions que le contrôle parental serait pertinent |
| Paramétrer en fonction de l'âge | Tout est adaptable en fonction de ce que l'administrateur souhaite pour l'utilisateur |
| Redonne le contrôle | Ne donne aucun contrôle. L'application interdit mais n'apprend pas à gérer |
| Difficulté de prise en main | Simple d'utilisation, mais beaucoup d'options |
| Administration depuis la même machine | Les restrictions sont élaborées depuis la session administrateur de la machine |
| Fournit des statistiques d'utilisation | Il conserve les historiques de consultation de site web |
| Avis des utilisateurs | Assez bon |
| Coûts | Le logiciel est compris dans le prix de l’OS |
| Opensource | Non |
| Offline | Le contrôle parental fonctionne avec ou sans connexion internet |
| **Remarque complémentaire** | |
| Ce logiciel est très solide et permet de bien gérer d'un point de vue administrateur. Mais il ne permet pas à l'utilisateur de prendre conscience du temps qu'il passe sur telle ou telle application. L'idéal serait qu'il sorte des statistiques d'utilisations et qu’il permette des alertes sans créer de blocage | |

### Rescue Time

|  |  |
| --- | --- |
| **../image%20applications/rescuetime.png[8]** | |
| **Utilisation** | |
| L'utilisateur installe l'application sur son appareil. Ensuite il paramètre la surveillance sur la page web. L'application répertorie tout le temps passé sur chaque application par l'utilisateur. Elle permet de fixer des objectifs et en fonction de ceux-ci, affiche des alertes. L'application ne bloque rien | |
| **Critères** | **Observations** |
| Blocage | Ne bloque rien, fournit des statistiques |
| Orienté web/natif/mobile | Application native dont les statistiques sont disponibles sur une page web |
| Plateforme | Fonctionne sur Mac OS, Linux, Windows et Android. Une version iOS est en cours de développement |
| Base volontaire ou restrictions | Base volontaire. Le "dashboard" ne bloque rien : il permet de fixer des objectifs donne des alertes. Ex: Objectif <2h sur facebook. Si le temps est atteint, une alerte approche |
| Public cible | Tout public |
| Paramétrer en fonction de l'âge | Oui, le parent pourrait fixer des objectifs pour l'enfant, et ce dernier ne verrait que les alertes. |
| Redonne le contrôle | Il redonne le contrôle grâce aux alertes, cela met l'utilisateur devant un choix. |
| Difficulté de prise en main | Il y a quelques options à décider, et il faut indiquer pour certain sites web s'ils sont sources de distraction ou non |
| Administration depuis la même machine | L'application est installée sur une machine, et on peut voir les statistiques sur une page web. |
| Fournit des statistiques d'utilisation | Donne des statistiques à la seconde près sur le temps passé sur les pages web, les applications. |
| Avis des utilisateurs | Très bon logiciel |
| Coûts | Parties gratuites et payantes |
| Open source | Non |
| Offline | Offline, l’utilisation n’est pas monitorée. Sauf en partie payante |
| **Remarque complémentaire** | |
| Cette application correspond beaucoup à ce que l'on souhaite faire : Redonner le contrôle.  **Une API est disponible** pour travailler avec les données récupérées par ce dashboard. Nous allons explorer les possibilités dans le chapitre « 2.5 tests technologiques ». La partie payante de l’application permet l’utilisation d’un bracelet paramétrable pour envoyer des décharges électriques en cas d’utilisation nocive d’application (à gérer en fonction des objectifs) [9]. Le nom de l’app est « Pavlok ». Plusieurs autres plugins utilisent les données. | |

### Moment

|  |  |
| --- | --- |
| **../image%20applications/moment.png**  **Moment [10]** | |
| **Utilisation** | |
| L'utilisateur télécharge l'application sur son produit iOS. Il l'ouvre et il autorise les accès par l'app aux photos ainsi qu'à la localisation. Une fois fait, il doit encore activer le traçage des apps utilisées. | |
| **Critères** | **Observations** |
| Blocage | Ne bloque rien, monitore |
| Orienté web/natif/mobile | Application native pour iOS. Monitore les applications, non pas les sites web. |
| Plateforme | iOS |
| Base volontaire ou restrictions | Base volontaire. Ce n'est que du monitoring. Pas de blocage. |
| Public cible | Toute personne utilisant un produit avec iOS |
| Paramétrer en fonction de l'âge | Non |
| Redonne le contrôle | Oui, car il peut donner des alertes |
| Difficulté de prise en main | Très simple |
| Administration depuis la même machine | Les statistiques apparaissent sur le mobile |
| Fournit des statistiques d'utilisation | Oui |
| Avis des utilisateurs | Bonne application (4/5 sur l'apple store) |
| Coûts | Gratuit |
| Opensource | Non |
| Offline | L’utilisation est la même offline |
| **Remarque complémentaire** | |
| L'application affiche le temps passé par app. Et elle donne un fil rouge pour la journée. Cependant elle n'est pas capable de filtrer le contenu web. Par contre, étant donné que les réseaux sociaux ont une app en général, c'est tout de même intéressant si les gens l'utilisent. Le produit est très semblable à Rescue Time, sauf qu’il est créé pour iOS. Une version Android est en cours de développement. L’application compte le nombre de fois que l’on dévérouille le smartphone. | |

### Contrôle parental Swisscom

|  |  |
| --- | --- |
| **../image%20applications/Logo_Swisscom.png[11]** | |
| **Fonctionnement** | |
| L'utilisateur choisit les appareils sur lesquels il veut activer le contrôle parental. Celui-ci consiste dans cette version standard à limiter les heures de navigation sur internet individuellement. Le temps total peut être défini, ainsi que les crénaux horaires selon les jours de la semaine et du weekend. Ce contrôle est paramétrable sur la swisscom TV box. Les émissions contenant des limites d'âge peuvent également être bloquées par un PIN que seules les personnes autorisées connaissent. | |
| **Critères** | **Observations** |
| Blocage | Bloque la navigation sur internet et certains contenus |
| Orienté web/natif/mobile | Orienté web |
| Plateforme | Fonctionne pour tout ordinateur, tablette et smartphone pouvant se connecter à internet |
| Base volontaire ou restrictions | Ne fonctionne que sur la base de restrictions. L'utilisateur du réseau contrôlé ne peut pas changer quoi que ce soit aux restrictions, il subit des interdits. |
| Public cible | Les enfants particulièrement et les adolescents |
| Paramétrer en fonction de l'âge | Navigation internet -> Pas directement. Si l'enfant utilise toujours les mêmes appareils, alors le contrôle est indirectement associé à l'âge. Blocage de contenu lié à l'âge -> Oui |
| Redonne le contrôle | Ne donne aucun contrôle. Le service interdit mais n'apprend pas à gérer |
| Difficulté de prise en main | Très facile à utiliser |
| Administration depuis la même machine | La gateway est paramétrée via une interface web donc administrable depuis n'importe quelle machine |
| Fournit des statistiques d'utilisation | Non, il ne fournit pas de statistiques |
| Avis des utilisateurs | Aucun avis disponible |
| Coûts | La fonctionnalité est comprise dans l'abonnement swisscom |
| Opensource | Non |
| Offline | Est inutile en hors ligne puisqu’il monitore uniquement ce qui est en ligne |
| **Remarque complémentaire** | |
| Ce contrôle parental est facile d'utilisation. Par contre, il ne permet pas de filtrer spécifiquement le contenu (mis à part le blocage du contenu limité par l'âge). Internet security permet un contrôle beaucoup plus approfondi, c'est le prochain service analysé. | |

### Internet Security Swisscom

|  |  |
| --- | --- |
| **../image%20applications/Logo_Swisscom.png[12]** | |
| **Fonctionnement** | |
| L'utilisateur installe cette application sur la machine que l'enfant ou l'adolescent utilise depuis le compte administrateur. L'application définit le contenu non-accessible par catégories et par sites spécifiques. Elle permet aussi de définir les horaires de navigation sur internet et d'accès à la machine même. | |
| **Critères** | **Observations** |
| Blocage | Bloque l'accès à l'ordinateur, les heures de navigation sur internet et certains contenus |
| Orienté web/natif/mobile | Application native |
| Plateforme | Fonctionne sur ordinateur, tablette et smartphone (fonctionnalités différentes) |
| Base volontaire ou restrictions | Ne fonctionne que sur la base de restrictions. L'utilisateur du réseau contrôlé ne peut pas changer quoi que ce soit aux restrictions, il subit des interdits. |
| Public cible | Les enfants particulièrement et les adolescents |
| Paramétrer en fonction de l'âge | Non |
| Redonne le contrôle | Ne donne aucun contrôle. L'application interdit mais n'apprend pas à gérer |
| Difficulté de prise en main | Facile à utiliser |
| Administration depuis la même machine | Les restrictions sont élaborées depuis la session administrateur de la machine |
| Fournit des statistiques d'utilisation | Non, il ne fournit pas de statistiques |
| Avis des utilisateurs | Bonne application (4.4/5 sur google play) |
| Coûts | La fonctionnalité est gratuite pendant 6 mois et ensuite payante |
| Opensource | Non |
| Offline | Utile car restreint également l'accès à la machine et pas seulement internet. |
| **Remarque complémentaire** | |
| Cette application est également très utile pour la protection des données et contre les sites webs dangereux ainsi que les virus. | |

### Synthèse

Durant ces recherches nous avons pu constater que les applications existantes permettent de réaliser déjà un bon monitoring ou contrôle par rapport à ce que fait l’utilisateur. Nous trouvons des produits qui interdisent l’accès à des ressources, d’autres qui informent l’utilisateur de son utilisation.

La plupart de ces produits sont gratuits, voir même opensource. L’application « Rescue Time » nous semble être la plus complète. Elle permet dans sa version gratuite de :

* Sortir des statistiques d’utilisation des applications desktop ou encore des pages web visitées (rapports journaliers) sur une page web
* Créer des alertes
* Classer les occupations par niveaux de productivité (mail = productif, Spotify != distraction), selon l’appréciation de l’utilisateur

La version payante [13] permet de :

* Monitorer le temps Offline
* Bloquer des sites web pendant une durée déterminée (comme self control)
* Avoir des rapports d’utilisations plus détaillés
* Créer des alertes personnalisées

Le tout pour 9$ par mois ou 72$ par année.

Nous trouvons que cette application est puissante, complète et efficace. Nous allons nous inspirer des fonctions suivantes pour notre projet :

* Récupération du temps que l’utilisateur passe sur chacune de ses activités
* Création de graphiques d’utilisation
* Création d’objectifs et d’alertes personnalisées

Voici un tableau semblable à ceux créés pour les technologiques mais il est rempli ce qui caractérise l’application SAS :

|  |  |
| --- | --- |
| **SAS** | |
| **Fonctionnement** | |
| Le superviseur envoie une invitation de monitoring à l’utilisateur. Ce dernier accepte l’invitation. Le logiciel récupère les informations des activités de l’utilisateur. Le superviseur et l’utilisateur peuvent consulter ces informations. Le superviseur peut mettre un temps limite d’utilisation, et l’utilisateur verra une barre passant du vert au rouge évoluer en fonction du temps. | |
| **Critère** | **Observation** |
| Blocage | Ne bloque pas |
| Orienté web/natif/mobile | Fonctionne comme application desktop |
| Plateforme | Windows, Mac OS |
| Base volontaire ou restrictions | Base volontaire |
| Public cible | Enfant et adolescents |
| Paramétrer en fonction de l'âge | Oui, possibilité de créer des objectifs adaptés à l’âge (barre d’utilisation) |
| Redonne le contrôle | Oui |
| Administration depuis la même machine | Non, c’est possible depuis une autre machine |
| Fournit des statistiques d'utilisation | Oui |
| Offline | Non |

## Tests technologiques

Cette section présente les différents tests technologiques menés lors de la phase d'analyse afin de déterminer les fonctionnalités techniquement faisables. La première partie concerne les processus tournant sur la machine de l'utilisateur du système. La seconde partie se penche sur la récupération du trafic internet. Deux petits programmes écrits en langage java ont été développés afin d'effectuer ces tests technologiques. Trois aspects différents des programmes sont mentionnés dans ce contexte : la compatibilité des devices et systèmes d'exploitation, les informations récupérées par le programme et l'utilisation d'éventuelles librairies.

### Analyse des processus

Le premier test implémente la récupération d'une liste des processus tournant sur un ordinateur portable. Concrètement pour le projet, les noms d'exécutables ainsi que leur chemin sur la machine sont analysés pour connaître la nature des applications qui tournent sur la machine (jeux,réseaux sociaux, ...).

#### Compatibilité des devices et systèmes d'exploitation

Le programme a été testé et fonctionne sur les systèmes d'exploitation suivants :

* Windows 10
* Mac OS X
* Kali Linux Rolling

Une liste des processus peut également être récupérée sur des appareils mobiles, notamment avec le langage C#. Ce langage est utilisé par Xamarin pour le développement d'application cross-plateform. Sur Android, on peut aussi utiliser le langage java. En ce qui concerne iOS, l'accès aux informations système semble plus complexe car à partir de la version iOS 9, l'accès aux données des applications est très restreint.

En estimant que la récupération de la liste des processus sur iOS prendrait trop de ressources par rapport à la durée du projet et qu'un appareil android peut aussi obtenir la liste des processus avec du java ou du C# (très similaire à java), il n'a pas été jugé nécessaire d'implémenter des tests technologiques sur un téléphone mobile.

#### Versions/Utilisation de librairie

Deux façons de récupérer les processus ont été implémentées.

La première solution consiste à lancer une commande système depuis le programme java. Cependant, cette solution n'est pas générique. Il faut donc créer deux versions différentes : Une version pour Windows et une autre pour les systèmes UNIX (Mac OS X et Linux). Le programme de test étant petit, la gestion des OS (Operating System) est gérée au travers de conditions dans le code. Dans le cas où cette option est choisie dans le cadre de ce projet, le code doit être séparé en plusieurs fichiers, un fichier par système d'exploitation.

La deuxième solution utilise une librairie java nommé *oshi* qui fournit aux développeurs des informations sur le système d'exploitation. Cette solution est moins rapide mais offre des options de tri intéressantes de même que d'autres données importantes liées aux processus détectés, telles que le processID ou le chemin de l'exécutable par exemple. La figure ci-dessous affiche les deux résultats obtenus.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

First way

services.exe­

svchost.exe

MySQLNotifier.exe

Telegram.exe

acrotray.exe

chrome.exe

...

explorer.exe

SearchUI.exe­

chrome.exe

eclipse.exe

javaw.exe

tasklist.exe

Total number of processes : 192

Deuxième solution

Première solution

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Second way

Process name : System Idle Process

Path : unknown

ProcessID : 0

ParentProcessID : 0

Process name : javaw.exe

Path : C:\Program Files\Java\jre1.8.0\_91\bin\javaw.exe

ProcessID : 1152

ParentProcessID : 10912

...

Process name : eclipse.exe

Path : C:\Users\Nicolas\eclipse\java-mars\eclipse\eclipse.exe

ProcessID : 10912

ParentProcessID : 1884

Total number of processes : 187

Résultats de la récupération des processus []

La figure ci-dessous montre le contenu du fichier ListProcesses.java contenant les deux solutions du test.

**package** test;

**import** oshi.SystemInfo;

**import** oshi.software.os.OSProcess;

**import** oshi.software.os.OperatingSystem;

**import** java.io.\*;

**import** java.util.regex.\*;

**public** **class** ListProcesses {

**public** **static** **void** main(String[] args) {

**try** {

String username = System.*getProperty*("user.name");

System.***out***.println("Username detected : " + username);

SystemInfo si = **new** SystemInfo();

OperatingSystem os = (OperatingSystem) si.getOperatingSystem();

Process p = **null**;

String operatingSystem = os.getFamily();

System.***out***.println(operatingSystem + " OS detected !\n");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("First way\n");

**if** (operatingSystem.equals("Windows")) {

p = Runtime.*getRuntime*().exec("tasklist.exe");

} **else** **if** (operatingSystem.equals("MacOS") || operatingSystem.equals("Kali GNU/Linux")) {

p = Runtime.*getRuntime*().exec("ps -eo comm"); //another command is "top" on mac (live update)

}

BufferedReader input = **new** BufferedReader(**new** InputStreamReader(p.getInputStream()));

String line = ""; **int** firstCounter = 0;

**if** (operatingSystem.equals("Windows")) {

Pattern pattern = Pattern.*compile*(".\*.exe");

**while**((line = input.readLine()) != **null**) {

Matcher matcher = pattern.matcher(line);

**if** (matcher.find()) System.***out***.println(matcher.group(0));­

firstCounter++;

}

} **else** **if** (operatingSystem.equals("MacOS") || operatingSystem.equals("Kali GNU/Linux")) {

**while**((line = input.readLine()) != **null**) {

System.***out***.println(line);

firstCounter++;

}

}

System.***out***.println("\n Total number of processes : " + firstCounter);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("Second way\n");

OSProcess[] processes = os.getProcesses(0,OperatingSystem.ProcessSort.***CPU***);

**int** secondCounter = 0;

PrintWriter pwn = **new** PrintWriter(**new** FileWriter(**new** File("Processes\_Names\_list.txt")));

PrintWriter pwp = **new** PrintWriter(**new** FileWriter(**new** File("Processes\_Pathes\_list.txt")));

**for** (**int** i = 0; i < processes.length; i++) {

System.***out***.println("Process name : " + processes[i].getName());

System.***out***.println("Path : " + processes[i].getPath());

System.***out***.println("ProcessID : " + processes[i].getProcessID());

System.***out***.println("ParentProcessID : " + processes[i].getParentProcessID());

System.***out***.println();

pwn.write(processes[i].getName());

pwp.write(processes[i].getPath());

secondCounter++;

}

pwn.close();

pwp.close();

System.***out***.println("\n Total number of processes : " + secondCounter);

} **catch** (Exception e) {

System.***out***.println("No command line opened");

e.printStackTrace();

}

}

}

Première solution

Deuxième solution

Code de la récupération des processus []

#### Informations récupérées

Comme cité plus haut dans le sous-chapitre précédent, la quantité et la forme des informations récupérées sont ­­dépendantes de l'utilisation d'une librairie et du système d'exploitation. Les informations récupérables sont les suivantes : Name, Path, ProcessID et parentProcessID. Comme on peut le voir dans le code de la première solution de la figure 2, Windows 10 requiert un parsing pour ne récupérer que le nom de l'exécutable et non le chemin complet.

### Analyse des sites web visités

Le deuxième test implémente la récupération d'une liste des sites web sur un ordinateur portable. Concrètement pour le projet, les sites web avec leurs métadonnées et leur temps de visite sont analysés pour distinguer les catégories des sites visités (jeux en ligne,réseaux sociaux, ...).

#### Compatibilité des devices et systèmes d'exploitation

Le programme (première solution) a été testé et fonctionne sur le système d'exploitation Windows 10. Les autres systèmes d'exploitation (Linux et Mac) et devices n'ont pas été testés car cette solution a vite été considérée comme inadaptée pour ce projet. La deuxième solution utilise un navigateur internet donc elle est générique au niveau des systèmes d'exploitation des ordinateurs. Elle a également été testée sur windows 10. Pour ce qui est des téléphones mobiles, cette solution ne fonctionne que sur le navigateur Firefox sur un appareil Android. Estimant que le trafic web ne valant pas la peine d'être analysé sur des smartphones, aucun test technologique n'a été fait dans ce sens.

#### Versions/Utilisation de librairie

Deux façons de récupérer les sites web ont été implémentées.

La première solution utilise la librairie libpcap (utilisée par l'outil wireshark) enrobée dans un wrapper java. Il en existe plusieurs sortes plus ou moins efficaces et faciles d'intégration dans un projet. La première librairie testée se nomme pcap4j. Elle n'est malheureusement pas utile pour ce projet car les protocoles que l'on désire capturer sont HTTP et HTTPS de la couche 7 du modèle OSI. Ces protocoles ne sont pas supportés par cette librairie. La seconde librairie testée se nomme jnetpcap. Elle est capable de capturer les protocoles HTTP et HTTPS, un protocole sécurisé et chiffré. Du fait qu’il est sécurisé et chiffré, HTTPS est capturable mais non-exploitable. C'est pour cette raison principalement que cette solution n'a pas été envisagée pour la suite du projet. De plus, la manière de capturer le traffic web a été remise en cause. Lors de la capture de paquets sur le réseau, on obtient uniquement les entêtes HTTP mais aucune informations concernant les onglets sur lesquels l'utilisateur est actif ainsi que le temps passé sur ceux-ci individuellement. Grâce aux applications Rescue Time et le contrôle parental de Swisscom, on a pu découvrir une autre façon bien plus efficace pour contrôler les sites sur lesquels navigue l'utilisateur. Cette alternative est la deuxième solution décrite en dessous.

Cette deuxième méthode fonctionne grâce à une extension google chrome, composée de fichiers html, css et js. Cette extension est également développable sur d'autres navigateurs comme firefox par exemple. L'extension a accès directement aux onglets du navigateur ainsi qu'aux URLs entrées. Le temps aussi peut être analysé. A chaque changement/ouverture/fermeture d'onglet, l'extension récupère le temps passé sur l'onglet précédent et envoie ces informations à un serveur http local (dans ce test, codé en java).

**package** test;

**import** java.io.\*;

**import** com.sun.net.httpserver.HttpExchange;

**import** com.sun.net.httpserver.HttpHandler;

**public** **class** SniffHandler **implements** HttpHandler {

@Override

**public** **void** handle(HttpExchange he) **throws** IOException {

InputStreamReader isr = **new** InputStreamReader(he.getRequestBody(), "utf-8");

BufferedReader br = **new** BufferedReader(isr);

String line = br.readLine();

System.***out***.println(line);

String response = "OK";

he.sendResponseHeaders(200, response.length());

OutputStream os = he.getResponseBody();

os.write(response.toString().getBytes());

os.close();

}

}

Deuxième solution de sniffing web - SniffHandler.java []

**package** test;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.jnetpcap.Pcap;

**import** org.jnetpcap.PcapIf;

**import** org.jnetpcap.packet.PcapPacket;

**import** org.jnetpcap.packet.PcapPacketHandler;

**import** org.jnetpcap.protocol.tcpip.Http;

**import** org.jnetpcap.protocol.tcpip.Http.Request;

**import** org.jnetpcap.protocol.tcpip.Tcp;

**public** **class** SniffNetwork {

**public** **static** **void** main (String[] args) {

List<PcapIf> alldevs = **new** ArrayList<PcapIf>(); // List of all capture interfaces

StringBuilder errbuff = **new** StringBuilder(); // Used for error messages

**int** statusDevs = Pcap.*findAllDevs*(alldevs, errbuff);

**if** (statusDevs == Pcap.***NOT\_OK*** || alldevs.isEmpty()) {

System.***err***.println("Network devices list issue");

**return**;

}

**for** (PcapIf device : alldevs) {

System.***out***.println("Name : " + device.getName());

System.***out***.println("Description : " + device.getDescription() + "\n");

}

PcapIf device = alldevs.get(2);

**int** snaplen = 64 \* 1024; // Capture all packets, no truncation

**int** flags = Pcap.***MODE\_PROMISCUOUS***; // Capture all packets

**int** timeout = 10 \* 1000; // 10 seconds in milliseconds

Pcap pcap = Pcap.*openLive*(device.getName(), snaplen, flags, timeout, errbuff); // Open the device for sniffing

**if** (pcap == **null**) {

System.***err***.println("Opening sniffing device issue : " + errbuff);

**return**;

}

PcapPacketHandler<String> packethandler = **new** PcapPacketHandler<String>() {

**private** **final** Tcp tcp = **new** Tcp();

**private** **final** Http http = **new** Http();

@Override

**public** **void** nextPacket(PcapPacket packet, String user) {

**if** (!packet.hasHeader(tcp) || !packet.hasHeader(http) || http.isResponse()) **return**;

String host = http.fieldValue(Request.***Host***);

**if** (host != **null**) System.***out***.println("Host : " + host);

}

};

pcap.loop(-1, packethandler, "jNetPcap rocks!");

pcap.close();

}

}

Première solution de sniffing web []

**var** port = 19119;

**var** previousURL = "";

**var** startTime = Date.now() / 1000;

**var** endTime = Date.now() / 1000;

chrome.tabs.onUpdated.addListener(**function**(tabId, changeInfo, tab) {

endTime = Date.now() / 1000;

**if** (**typeof** tab !== **undefined** && changeInfo.url !== **undefined**) {

**if** (previousURL !== "" && previousURL !== "chrome://newtab/") {

sendTimedURL();

}

previousURL = changeInfo.url;

startTime = endTime;

}

});

chrome.tabs.onActivated.addListener(**function**(activeInfo) {

endTime = Date.now() / 1000;

chrome.tabs.get(activeInfo.tabId, **function**(tab) {

**if** (previousURL !== "" && previousURL !== "chrome://newtab/") {

sendTimedURL();

}

previousURL = tab.url;

startTime = endTime;

});

});

**function** sendTimedURL() {

**var** http = **new** XMLHttpRequest();

**var** description = getMetaContentByName('description');

**var** TimedUrl = previousURL + " " + (endTime - startTime) + " " + description;

http.open("POST", "http://localhost:" + port + "/sniff", **false**);

http.setRequestHeader("Content-type", "text/plain; charset=utf-8");

http.setRequestHeader("Content-length", TimedUrl.length);

http.setRequestHeader("Connection", "close");

http.send(TimedUrl);

}

**function** getMetaContentByName(mn) {

returnContent = "";

**var** m = document.getElementsByTagName('meta');

**var** metadata = "Metadata\n";

alert(document.querySelector('meta[property="description"]').getAttribute('content'));

**for** (**var** i = 0; i < m.length; m++) {

**try** {

alert("Content : " + m[i].getAttribute("content"));

**if** (m[i].name.toLowerCase() == mn.toLowerCase()) {

returnContent = m[i].content;

**break**;

}

} **catch** (err) {

**continue**;

}

}

**return** returnContent.trim();

}

*Deuxième solution de sniffing web – sniff.js []*

**package** test;

**import** java.io.IOException;

**import** java.net.InetSocketAddress;

**import** com.sun.net.httpserver.HttpServer;

**public** **class** ExtensionServer {

**public** **static** **void** main(String[] args) {

**int** port = 19119;

HttpServer server;

**try** {

server = HttpServer.*create*(**new** InetSocketAddress(port), 0);

System.***out***.println("server started at " + port);

server.createContext("/sniff", **new** SniffHandler());

server.setExecutor(**null**);

server.start();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

*Deuxième solution de sniffing web - ExtensionServer.java []*

#### Informations récupérées

Avec cette extension, on peut récupérer les URLs recherchées par l'utilisateur avec le temps passé sur chaque onglet comme le montre la figure ci-dessous. Les métadonnées sont également récupérables mais n'apparaissent pas dans la figure.

server started at 19119

chrome://extensions/?id=gnkebgfnodnhdhghmbgldmpkgdgbjaed 16.388000011444092

https://www.google.ch/ 2.064000129699707

chrome://extensions/?id=gnkebgfnodnhdhghmbgldmpkgdgbjaed 22.98099994659424

https://www.youtube.com/watch?v=Fi4YmLiC18E 2.875999927520752

Résultat de la récupération des URLs et leur temps de navigation []

## Conclusion

Cette partie d’analyse nous a permit de mieux nous réaliser de la portée du projet et de la manière d’aborder les fonctionnalités que nous pourrions implémenter. D’un point de vue humain, en lisant des témoignages et des informations sur le sujet des addictions aux réseaux sociaux et aux jeux vidéos, nous nous sommes rendu compte que la gestion du temps passé est un point central pour les générations en cours et à venir. Cela nous motive pour la phase d’implémentation.

Techniquement, il existe déjà beaucoup de produits avec des fonctionnalités performantes. Parmis ces logiciels figure notamment Rescute Time. Nous estimons qu’il est efficace. Nous avons observé sa manière de monitorer l’activité web des utilisateurs : nous pensons qu’il utilise les extensions des navigateurs afin de connaître les URLs. Cela permet de ne pas avoir de problème de décodage en utilisant les couches réseaux. Nous nous en sommes inspirés pour créer la récupération de l’activité web de l’utilisateur.

Cette application monitore toutes les activités effectuées sur l’ordinateur. De notre coté, il a été convenu avec les clientes que nous nous concentrerions uniquement sur les thèmes des jeux vidéos et des réseaux sociaux (web et desktop). C’est pourquoi nous allons récupérer uniquement les informations relatives à ces thèmes.

Lors de l’analyse technologique, nous avons constaté que les extensions Google Chrome ne sont pas installables sur les mobiles et que les extensions Firefox sont disponibles pour Android. Nous allons nous concentrer sur l’activité des ordinateurs, plus particulièrement des OS Mac et Windows et nous utiliserons les extensions Chromes pour monitorer les activités web de l’utilisateur.

La valeur ajoutée de notre produit par rapport aux produits existants est le fait que le projet est plus personnalisé par rapport aux souhaits des clientes. En effet, nous ciblons le monitoring sur un thème précis et nous ne nous occupons pas du reste. Il est possible que cela améliore les performances du logiciel.

Suite à ces recherches nous pouvons donc fixer le cadre qui projet ; nous allons mettre en place une application qui  :

* Monitore uniquement l’activité sur les réseaux sociaux et jeux vidéos sur l’ordinateur de l’utilisateur (nous n’allons pas implémenter la surveillance sur les smartphones)
* Sort un fichier de log contenant les informations de l’activité de l’utilisateur
* Permet de consulter ces informations :
  + Sous forme de graphiques
  + Sur une application desktop
  + Sur le web

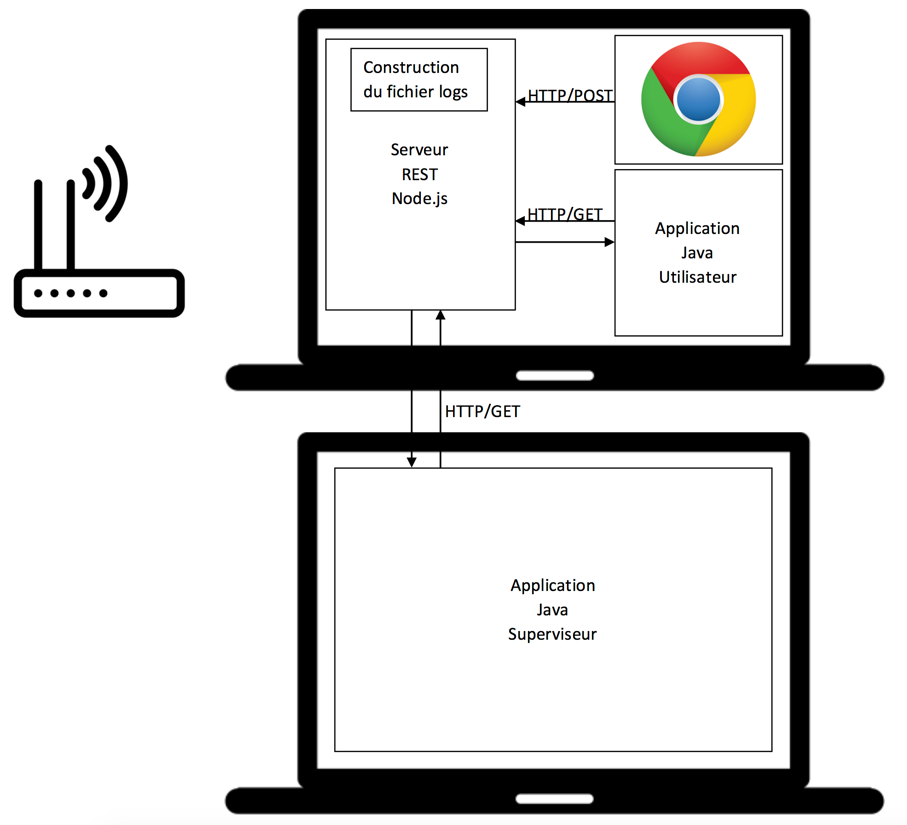
Nous avons ainsi le « scope » que nous allons donner à notre produit final.

# Conception

## Introduction

La partie d’analyse de ce projet nous a permit de fixer le cadre du projet. Dans ce chapitre, c’est la manière de mettre en œuvre ce projet qui est présentée. Un diagramme de cas d’utilisation décrit les actions que le logiciel est capable d’effectuer. Pour chacun de ces cas il y a une fiche descriptive qui le caractérise, ainsi qu’un diagramme de séquence et les besoins en IHM. Il est possible que le fonctionnement final du logiciel diffère de ce qui est prévu à la conception, car en cette phase le développeur exprime comment il souhaite construire programme, mais parfois la phase d’implémentation révèle que ce qui est prévu doit être réalisé autrement.

Le schéma ci-dessous représente l’architecture de l’application:



Dans un même réseau se trouvent plusieurs machines :

* La machine de l’utilisateur a monitorer
* La machine du superviseur

Sur la machine utilisateur tourne un serveur local en NodeJS. Celui-ci récupère les informations sur l’activité utilisateur envoyées par le browser ainsi que par l’OS. Avec ces informations il crée un fichier de log. L’application java utilisateur récupère ce fichier de log et affiche des statistiques d’utilisation sur la machine monitorée.

Sur la machine du superviseur, il y a une application Java (qui est intégrée dans une application web) où le superviseur peut connaître les statistiques d’utilisation pour les machines qu’il monitore.

Les informations entre les machines et les composants communiquent par des requêtes HTTP/GET et HTTP/POST.

## Cas d’utilisation

Le schéma ci-dessous représente le diagramme de cas d’utilisation de l’application. Ceux-ci ont été établis sur la base des recherches et des discussions effectuées dans les premières semaines du projet. Il exprime une large palette des fonctionnalités que l’utilisateur pourra effectuer.

Les différents cas sont implémentés dans les prototypes suivants :

1. En bleu, le prototype 1
2. En jaune, le prototype 2
3. En rouge, le prototype 3
4. En vert, les perspecitves d’amélioration

Note : Les cas implémentés dans le prototype 1 le sont également dans le prototype 2 et 3. Ceux implémentés dans le 2 le sont aussi dans le 3.



### Prototype 1, génération d’un fichier de log

#### Enregistrer les activités desktop et web

**Description** : Le serveur local / extension Chrome récupère les activités desktop et web de l’utilisateur.

**Acteurs** : Serveur local, Utilisateur, Extension Chrome et Application Java Utilisateur

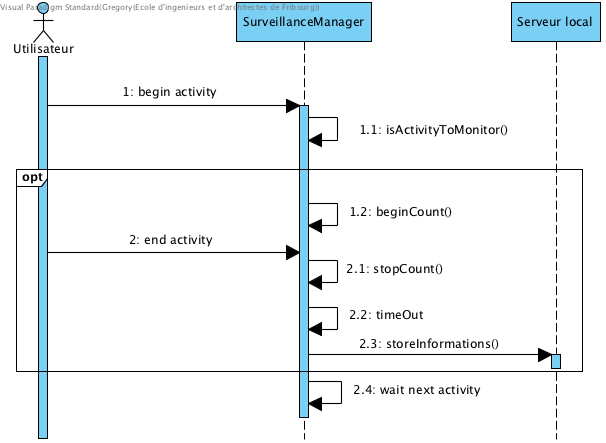
**Scénario nominal :**

1. L'utilisateur commence une activité sur l’ordinateur
2. La surveillance détecte s’il s’agit d’un jeu ou d’un réseau social (desktop ou page web)
   1. Si c’est le cas, un compteur démarre et se termine à la fin de l’activité
3. La surveillance est en attente jusqu’à la prochaine activité, auquel cas elle retourne au cas 1 du scénario nominal
4. La surveillance envoie les activités enregistrées au serveur local après un timeout
5. Reprend au point 1 du scénario nominal

**Remarques (optionnel) :**

La surveillance (extension Chrome / serveur local) est constamment en train de vérifier les activités de l’utilisateur. Il est possible que le serveur local ne détecte pas tous les processus comme étant des jeux vidéos.

##### Diagramme de séquences

****

#### Générer un fichier de log

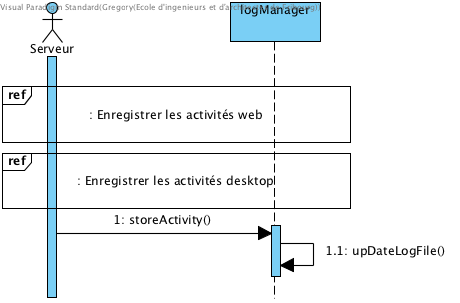
**Description** : Les activités ont été récupérées par les cas <<UC : enregistrer les activités desktop>> et <<UC : enregistrer les activités web>>. Un fichier de log est généré à partir de ces informations.

**Acteurs** : Serveur local

**Scénario nominal :**

1. <<UC : Enregistrer les activités web>>
2. <<UC : Enregistrer les activités desktop>>
3. Le serveur local stocke les activités reçues dans un fichier log mis à jour régulièrement
4. Reprend au point 1 du scénario nominal

##### Diagramme de séquences



### Prototype 2, création d’une application pour l’utilisateur

#### Se logger

**Description** : Un utilisateur ou un superviseur souhaite se logger à l’application afin de consulter les statistiques.

**Acteurs** : Serveur local, Utilisateur, Superviseur, Applications java

**Scénario nominal**

1. La personne ouvre l’application java
2. L’application java vérifie si la personne est loggée
3. L’application java affiche une page de login
4. La personne entre son login et son mot de passe
5. L’application java vérifie et valide les données de connexion
6. Fin du scénario nominal

**Scénario alternatif**

A1 : La personne est déjà loggée

* Démarre au point 3 du scénario nominal

1. Reprend au point 6 du scénario nominal

A2 : Les données de connexion sont incorrectes

* Démarre au point 6 du scénario nominal

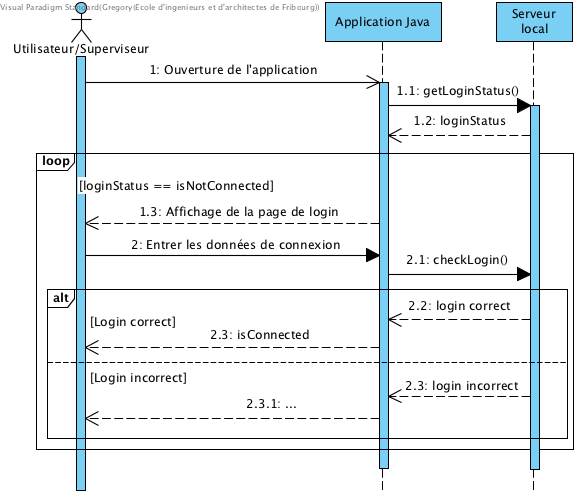
1. L'application affiche un message d'erreur
2. Reprend au point 4 du scénario nominal

**Maquettes :**

****

**Remarques :** Le login et le mot de passe sont stockés en dur dans le code. Nous ne gérons pas les aspects de sécurité ainsi que de timeout de session.

##### Diagramme de séquences



#### Consulter ses statistiques

**Description** : Un utilisateur souhaite consulter ses statistiques.

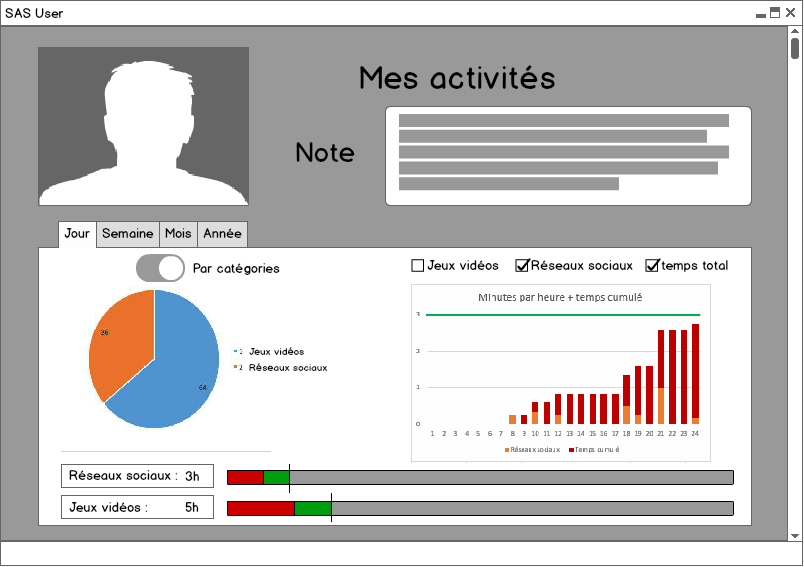
**Acteurs** : Utilisateur, Application Java Utilisateur, Serveur local

**Scénario nominal :**

1. <<UC : Se logger>>
2. L'application Java Superviseur récupère les informations de l'utilisateur sur son serveur local
3. L'application Java Superviseur affiche les statistiques récupérées
4. L'utilisateur consulte ses informations

**Maquettes :**

****

****

**Remarques :** l’utilisateur peut consulter ses activités sur différentes périodes (Jour, semaine, mois, année). Nous n’implémenterons l’affichage que pour une journée à la fois, mais le logiciel est prévu pour permettre l’affichage des autres périodes.

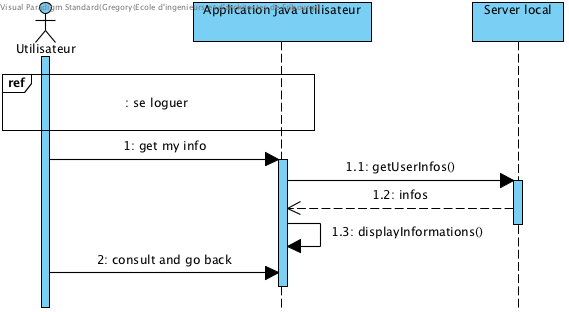
**Le graphique circulaire (en rouge)** affiche le temps passé par activité (par exemple 5h sur Facebook, 2h sur Instagram, 1h sur Crazy Bird), ou par catégories **(en jaune)** en activant ou non le toggle en dessus du graphique. Si l’on reprend le même exemple le graphique afficherait 1h sur les jeux vidéos, et 17h sur les réseaux sociaux. Ainsi il est possible de voir la proportion de l’utilisation.

**Le graphique à barres (en bleu)** affiche le temps passé par heure de la journé. Pour chaque heures de la journée, le temps est maximum de 60 minutes. Ainsi il est possible de voir à quelle période de la journée l’utilisateur à consommé soit des réseaux sociaux, soit des jeux vidéos. En effet, ce graphique n’affiche que 2 catégories. À l’aide des checkbox, l’utilisateur peut choisir ce qu’il affiche. Ce graphique permet également d’afficher le temps cumulé sur l’ensemble de la période.

**Les barres d’objectifs (en vert)** affichent le temps passé par rapport à l’objectif fixé par le superviseur. L’entier de la barre représente 24h, la partie rouge représente le temps passé, la partie verte le temps restant et la barre perpendiculaire représente la durée totale de l’objectif. Ainsi, si l’utilisateur dépasse l’objectif, on peut le détecter et voir de combien de temps l’objectif à été dépassé. La barre verte dans le graphique **bleu** représente l’objectif.

**Le champ des notes** affiche des commentaires écris par l’utilisateur.

##### Diagramme de séquences



### Prototype 3, création d’une application pour le superviseur

#### Afficher la liste des utilisateurs

**Description** : Le superviseur souhaite administrer ses utilisateurs et consulter les statistiques des informations enregistrées. Il veut aussi pouvoir supprimer ou ajouter des utilisateurs à surveiller et éventuellement leurs créer des objectifs.

**Acteurs** : Serveur local, Superviseur, Application Java Superviseur

**Pré conditions** : : La machine du superviseur doit être connecté sur le réseau local où se trouvent les machines et utilisateurs qu’il souhaite superviser (utilisation de requêtes HTTP).

**Scénario nominal**

1. <<UC : se logger>>
2. L'application Java Superviseur affiche la liste des utilisateurs
3. Le superviseur consulte la liste des utilisateurs monitorés
4. Le superviseur quitte l’application
5. Fin du scénario nominal

**Scénario alternatif**

A1 : Consulter les statistiques d'un utilisateur monitoré

* Démarre au point 2 du scénario nominal

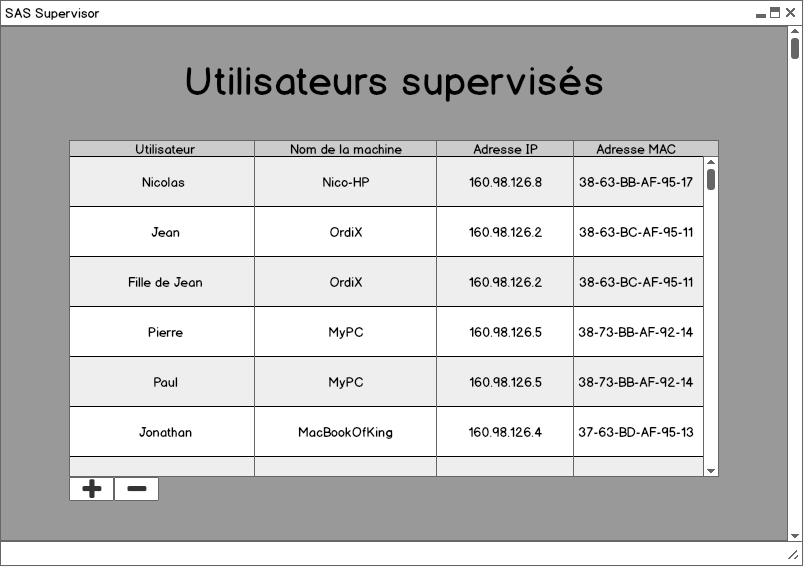
1. <<UC : Consulter les statistiques>>
2. Reprend au point 2 du scénario nominal

A2 : Ajouter un utilisateur

* Démarre au point 2 du scénario nominal

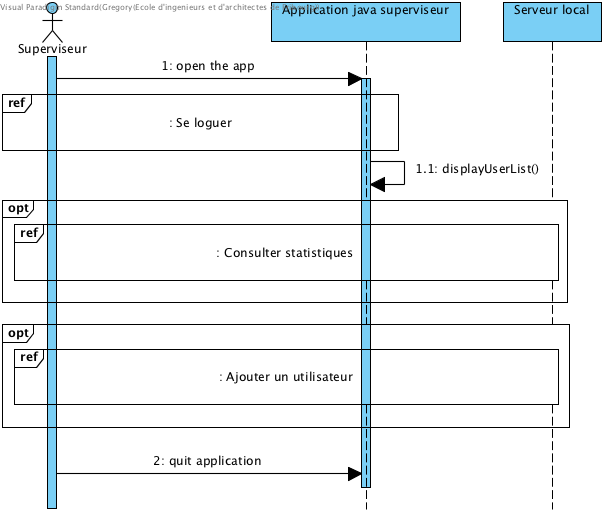
1. <<UC : Ajouter un utilisateur>>
2. Reprend au point 2 du scénario nominal

**Maquettes :**

****

**Remarques :** C’est la vue qui est affichée à l’ouverture de l’application coté superviseur (une fois le login effectué). Depuis là le superviseur peut ajouter, supprimer et consulter un utilisateur. L’application permet de suivre différents utilisateurs qui sont sur une même machine (Pierre et Paul dans notre exemple).

##### Diagramme de séquences



#### Supprimer un utilisateur

**Description** : Le superviseur souhaite ne plus monitorer un utilisateur.

**Acteurs** : Superviseur, Application Java Superviseur, Serveur local

**Description des enchaînements :**

**Pré conditions** : -

**Scénario nominal**

1. Le superviseur clique sur le bouton "Supprimer un utilisateur"
2. L'application Java Superviseur affiche une demande de confirmation de suppression
3. Le superviseur clique sur le bouton "Supprimer"
4. Le serveur local de l'utilisateur à ne plus monitorer supprime le droit
5. Fin du scénario nominal

**Scénario alternatif**

A1 : La suppression est annulée

* Démarre au point 3 du scénario nominal

1. Le superviseur clique sur le bouton "Annuler"
2. Fin du scénario alternatif

##### Diagramme de séquences

#### ../../../../../../../../Library/Containers/com.apple.Preview/Data/Desktop/Supprimer%20un%20ut

#### Ajouter un utilisateur

**Description** : Le superviseur souhaite ajouter un utilisateur à monitorer.

**Acteurs** : Superviseur, Application Java Superviseur, Serveur local

**Pré conditions** : La machine du superviseur doit être connecté sur le réseau local où se trouvent les machines et utilisateurs qu’il souhaite superviser (utilisation de Nmap).

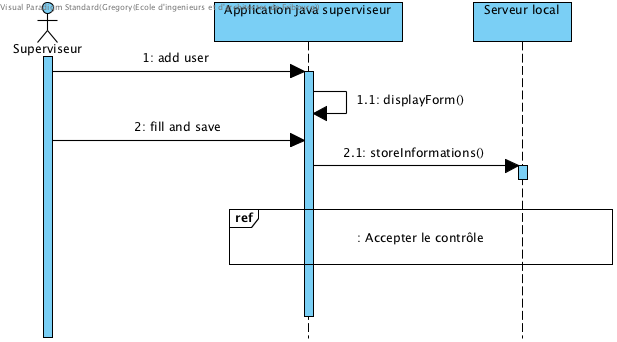
**Scénario nominal**

1. Le superviseur clique sur le bouton "Ajouter un utilisateur"
2. L'application Java Superviseur affiche le formulaire pour ajouter un utilisateur
3. Le superviseur remplit les champs puis valide
4. Le serveur local de l'utilisateur à monitorer enregistre les informations
5. <<UC : Accepter le contrôle>>
6. Fin du scénario nominal

**Maquettes :**

****

##### Diagramme de séquences



#### Accepter le contrôle

**Description** : Un utilisateur autorise le contrôle de ses statistiques à un superviseur

**Acteurs** : Utilisateur, Application Java Utilisateur, Serveur local

**Description des enchaînements :**

**Pré conditions** : -

**Scénario nominal**

1. <<UC : Se logger>>
2. L'utilisateur clique sur l'onglet "Demandes de monitoring"
3. L'application Java Utilisateur récupère les demandes de monitoring sur le serveur local
4. L'application Java Utilisateur affiche la liste des demandes
5. L'utilisateur clique sur le bouton "Accepter" à côté d'une demande
6. L'application Java Utilisateur met à jour la liste des demandes
7. L’application java envoie une confirmatin/infirmation au serveur local
8. Fin du scénario nominal

**Scénario alternatif**

A1 : L'utilisateur refuse la demande de monitoring

* Démarre au point 4 du scénario nominal

1. L'utilisateur clique sur le bouton "Refuser" à côté d'une demande
2. Fin du scénario alternatif

**Post conditions :** -

**Besoin d’IHM:**

**Remarques (optionnel) :**

##### Diagramme de séquences

#### ../../../../../../../../Library/Containers/com.apple.Preview/Data/Desktop/Accepter%20le%20

#### Consulter les statistiques

**Description** : Le superviseur souhaite consulter les statistiques des informations enregistrées des utilisateurs monitorés.

**Acteurs** : Superviseur, Application Java Superviseur, Serveur local

**Description des enchaînements :**

**Pré conditions** : -

**Scénario nominal**

1. Le superviseur sélectionne un utilisateur dans la liste
2. L'application Java Superviseur récupère les informations de l'utilisateur sur son serveur local
3. L'application Java Superviseur affiche les statistiques de l'utilisateur sélectionné
4. Le superviseur consulte les informations
5. Le superviseur revient à la liste des utilisateurs
6. Fin du scénario nominal

**Scénario alternatif**

A1 : Créer un objectif pour un utilisateur monitoré

* Démarre au point 4 du scénario nominal

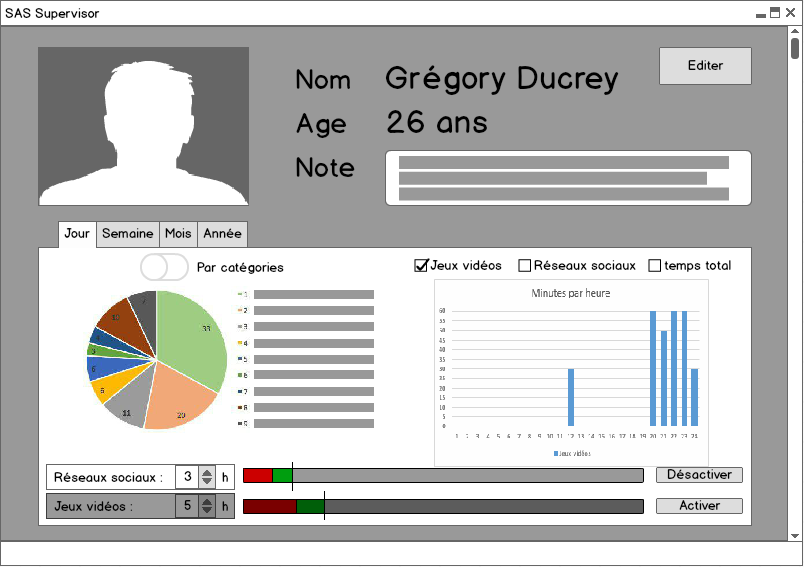
1. <<UC : Créer un objectif>>
2. Reprend au point 4 du scénario nominal

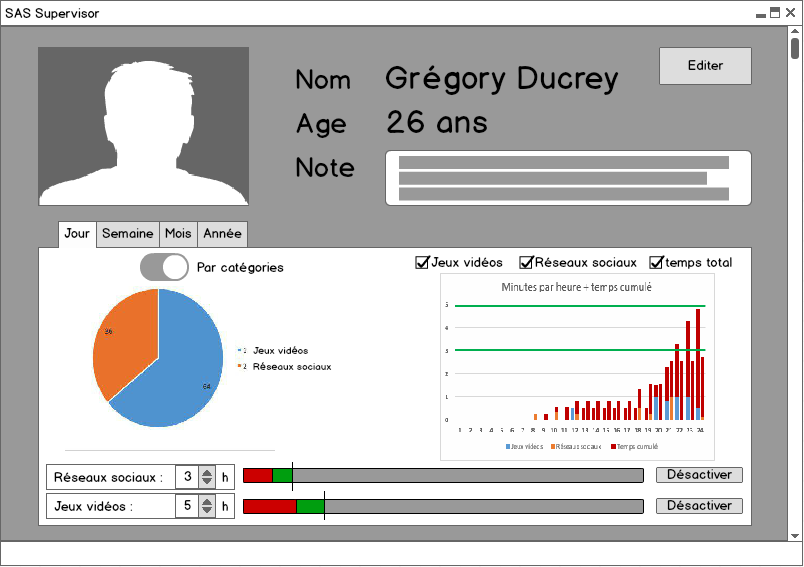
A2 : Supprimer un utilisateur monitoré

* Démarre au point 4 du scénario nominal

1. <<UC : Supprimer un utilisateur>>
2. Reprend au point 4 du scénario nominal

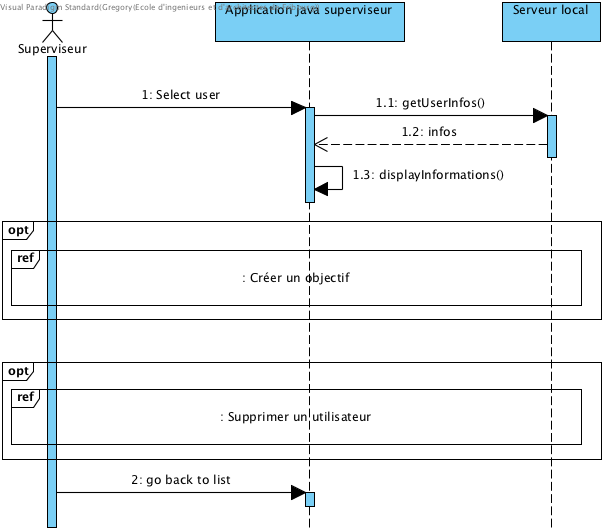
**Maquettes :**

****

****

**Remarques :** Lorsque le superviseur sélectionne un utilisateur qu’il supervise, il arrive sur cette page qui est semblable à celle de de l’utilisateur. Il peut en plus de consulter les activités, activer ou désactiver les objectifs et éditer les champs (Nom, âge, note).

##### Diagramme de séquences



#### Créer un objectif

**Description** : Le superviseur souhaite créer un objectif pour un utilisateur monitoré.

**Acteurs** : Superviseur, Application Java Superviseur, Serveur local

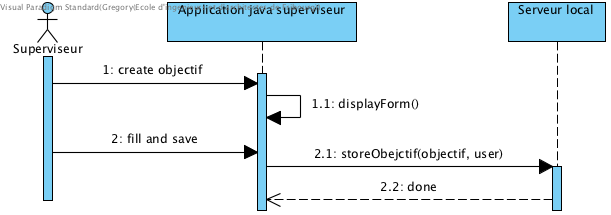
**Pré conditions** :

**Scénario nominal**

1. Le superviseur clique sur le bouton "Créer un objectif"
2. Le système affiche le formulaire de création d’objectif
3. Le superviseur remplit le formulaire et clique sur le bouton "Sauvegarder"
4. Le serveur local de l'utilisateur monitoré enregistre l’objectif

**Maquettes :** La création d’objectif n’a pas été faite dans un formulaire à part comme sous entendu dans la fiche descriptive. La fonctionnalité se trouve dans la maquette de supervision d’un utilisateur.

##### Diagramme de séquences



#### Consulter la barre d'utilisation

**Description** : Un utilisateur souhaite consulter sa barre d'utilisation.

**Acteurs** : Utilisateur, Application Java Utilisateur, Serveur local

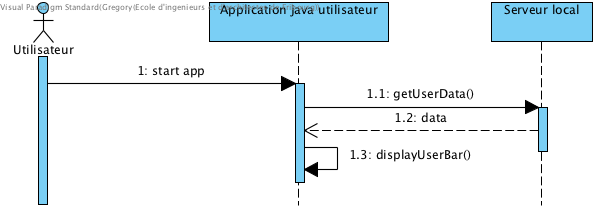
**Scénario nominal**

1. L'utilisateur clique sur l'icône de l'application sur le bureau
2. L'application Java Utilisateur récupère les données sur le serveur local
3. L'application Java Utilisateur affiche les données récupérées
4. Fin du scénario nominal

**Scénario alternatif**

**Maquettes :** La consultation se fait sur la page de supervision.

##### Diagramme de séquences



## Structure des fichiers Json

Le serveur NodeJS permet de récupérer les activités de l’utilisateur sur Google Chrome ainsi que les applications ouvertes sur le bureau. Une fois que l’entier de celles-ci sont récupérées, il est nécessaire de les trier et de garder celles qui sont pertinentes :

1. Les activités de jeux
2. Les activités de réseaux sociaux

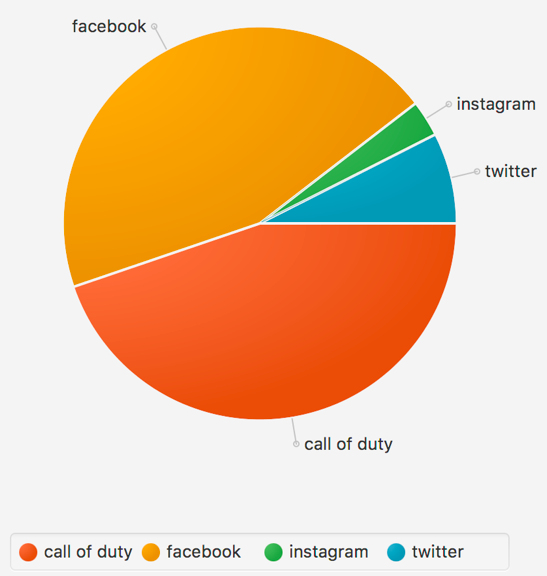
Une fois ces informations triées, il faut les mettre dans un fichier JSon que l’application java sera en mesure de traiter afin d’en afficher des statistiques. Il y a 2 fichier Json qui sont doivent être générés.

### Json du temps passé par activité

Ce fichier Json contient les informations du temps passé par l’utilisateur pour chaque activité qu’il a effectué (et qu’il est nécessaire de monitorer). Le temps est toujours affiché en seconde. Il y a pour chaque jour un fichier qui est généré.

Grâce à ce fichier, il nous est possible de générer le diagramme suivant :

{

 "date": "22.02.1299",

"games" : [ 🡪 Catégorie

« jeux vidéos »

{

"name" : "call of duty", 🡪 Nom du processus ou du site

"totalTime" : 3000 🡪 Temps total passé au moment de la génération du fichier

}

]

,

"networks" : [

{

"name" : "facebook",

"totalTime" : 3000

},

{

"name" : "instagram",

"totalTime" : 200

},

{

"name" : "twitter",

"totalTime" : 500

}

]

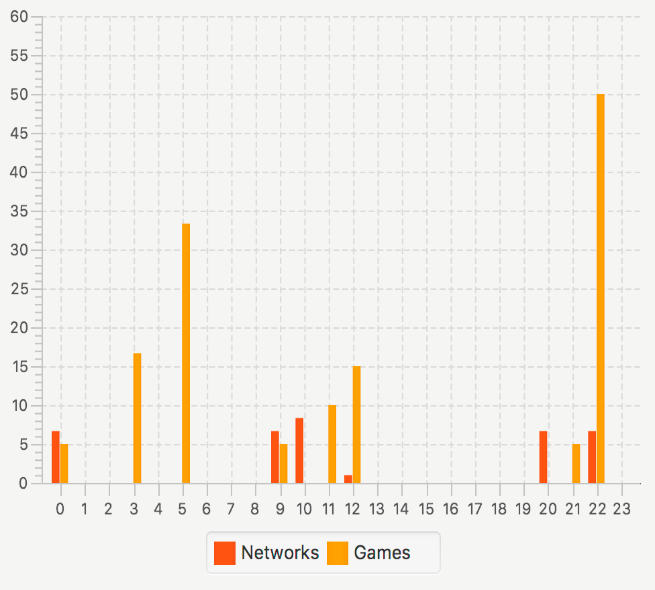
}

### Json du temps passé par heure de la journée

Dans le fichier d’exemple ci-dessous, nous avons récupéré pour chaque heure le temps passé par catégories (réseau sociaux ou jeux vidéos). Il y a pour chaque jour un fichier qui est généré.

Grâce à ce fichier, il nous est possible de générer le diagramme suivant :

{

 "date": "22.02.1992",

"0" : 🡪Heure de la journée

{

"games": 300, 🡪 Temps passé sur les jeux

"networks" : 400 🡪 Temps passé sur les réseaux sociaux

},

"1" :

{

"games": 0,

"networks" : 0

},

"2" :

{

"games": 0,

"networks" : 0

},

"3" :

{

"games": 1000,

"networks" : 0

},

"4" :

{

"games": 0,

"networks" : 0

},

Jusqu’à 23 pour compléter la journée.

"23" :

{

"games": 0,

"networks" : 0

}

}

## Conclusion

Cette phase de conception oblige le développeur à penser concrètement à la manière dont il souhaite développer l’application. A ce stade la manière présumée de réaliser le logiciel est donc connue et il ne devrait y avoir qu’à appliquer ce qui à été réfléchi pour créer le produit final. Plus concrètement, cette phase de conception a permis :

1. D’établir les uses-cases de l’application et pour chacun d’eux :

* Les fiches descriptives
* Les besoins IHM
* Les enchaînements des actions
* Les intervenants (système et humain)

1. De créer la structure des fichiers JSon qui seront générés à partir du fichier de log des informations sur les activités de l’utilisateur

Avec ces informations, il est possible de réaliser le logiciel en ayant une vision concrète du produit final tout en étant souple sur les besoins techniques inhérents à la phase d’implémentation.

# Implémentation

## Introduction

La phase de conception a permit de décider concrètement comment l’on souhaite que l’application fonctionne. La phase d’implémentation est celle qui réalise le logiciel de manière pratique. Voici un rappel de la portée du travail effectué par le logiciel de monitoring:

* Il monitore les activités web et desktop de l’utilisateur
* Les activités filtrées sont celles qui concernent les jeux vidéos et les réseaux sociaux
* Les plateformes systèmes supportant le monitoring sont Mac OS et Windows
* Le browser permettant la surveillance des activités web est Google Chrome
* Le logiciel récupère l’activité de machines se trouvant sur le même réseau

L’implémentation est découpée en 3 parties. Il y a 3 prototypes qui sont construits successivement en réutilisant le prototype précédent :

1. Récupération, filtrage et enregistrement dans un fichier de log des activités web et desktop de l’utilisateur
2. Création de fichier Json à partir des informations de log et affichage dans une application Java tournant sur la machine monitorée
3. Affichage des fichier Json dans une application Java superviseur afin que le superviseur soit en mesure de voir les informations de plusieurs utilisateurs supervisés dans le même réseau local

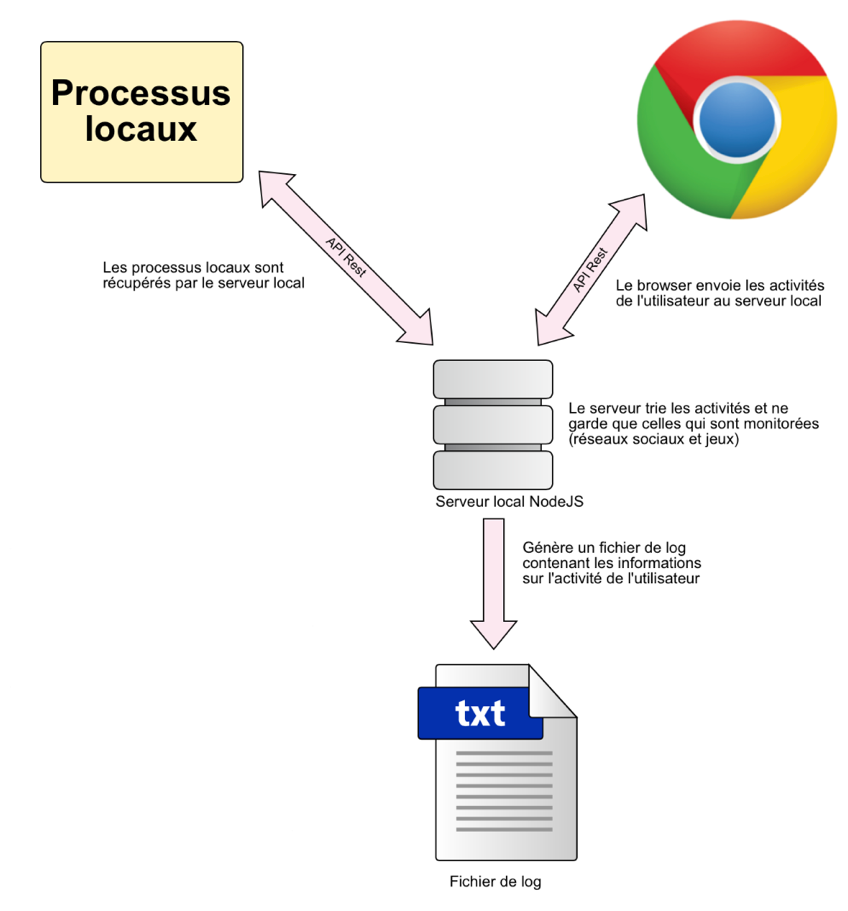
Comme expliqué précédemment, le lien entre ces prototypes est très fort car tous ont besoin du prototype précédent pour fonctionner (sauf le 1).

## Prototype 1

Ce premier prototype permet de récupérer les informations de l’utilisateur. Le schéma ci-dessous décrit ce qu’il est capable de faire. À ce stade il n’y a que le serveur local ainsi que l’extension chrome qui tournent pour pouvoir générer le fichier de log.

Fonctionnement du serveur :

* Récupération des activités de l’utilisateur
* Création et écriture de ces informations dans un fichier de log



### Extension Chrome

Comme décrit précédemment, l’extension chrome permet de détecter les activités web de l’utilisateur. Pour ce faire, le principe est simple :

Chrome met à disposition des fonctions permettant de connaître les URL visitées ainsi que les actions telles que l’ouverture d’un nouvel onglet ou d’une nouvelle fenêtre du navigateur. Cela fonctionne aussi lorsque le mode de navigation privée est activé. La stratégie pour enregistrer le temps des activités est la suivante :

* Détecter l’ouverture d’un nouvel onglet ou d’une nouvelle fenêtre du navigateur
* Récupérer l’URL visitée
* Récupérer les métadonnées
* Récupérer le temps du début
* Détecter la fin de l’utilisation de la fenêtre ou de l’onglet
* Récupérer le temps de fin
* Envoyer les informations au serveur
* Le serveur écrit dans le log ce qu’il s’est passé, combien de temps et dans quelle catégorie doit aller l’information (Games ou Networks)

Voici les fonctions NodeJs permettant de le faire :

* Evénement capturé lorsque l'utilisateur change d'onglet

chrome.tabs.onActivated.addListener(function(activeInfo) {  
 *endTime* = new *Date*();  
 chrome.tabs.get(activeInfo.tabId, function(tab) {  
 if (*previousURL* !== "") {  
 *sendTimedURL*();  
 }  
 *getMetaContent*();  
 *previousURL* = tab.url;  
 *startTime* = *endTime*;  
 });  
});

* Evénement capturé lorsqu'un onglet est créé ou mis à jour

chrome.tabs.onUpdated.addListener(function(tabId, changeInfo, tab) {  
 *endTime* = new *Date*();  
 if (typeof tab !== undefined && changeInfo.url !== undefined && changeInfo.url !== "chrome://newtab/") {  
 if (*previousURL* !== "") {  
 *sendTimedURL*();  
 }  
 *getMetaContent*();  
 *previousURL* = changeInfo.url;  
 *startTime* = *endTime*;  
 }  
});

À partir de ces deux fonctions il est possible de savoir si l’utilisateur est actuellement sur le navigateur ou non. Chaque fois que l’événement est déclenché, le temps de fin d’utilisation est mis à jour. Cela permet de :

* Savoir si le navigateur est toujours actif

var *hasBrowserFocusOut* = false;  
*setInterval*(function() {  
 var http = new *XMLHttpRequest*();  
 http.open("GET", "http://localhost:" + *port* + "/active", false); *// false for synchronous request* http.send();  
 if (!*hasBrowserFocusOut* && *previousURL* !== "" && http.responseText !== "" && http.responseText !== "chrome.exe") {  
 *hasBrowserFocusOut* = true;  
 *endTime* = new *Date*();  
 *sendTimedURL*();  
 } else if (*hasBrowserFocusOut* && http.responseText === "chrome.exe") {  
 *hasBrowserFocusOut* = false;  
 *startTime* = new *Date*();  
 }  
}, 1000);

* Envoyer la donnée de navigation au serveur local via une requête HTTP.

function *sendTimedURL*() {  
 var http = new *XMLHttpRequest*();  
 var day = *startTime*.getDate();  
 var month = *startTime*.getMonth() + 1;  
 var date = *startTime*.getFullYear() + "." + ((month < 10)?"0":"") + month + "." + ((day < 10)?"0":"") + day;  
 var time = (*endTime*.getTime() - *startTime*.getTime()) / 1000;  
 var TimedUrl = {date:date, url:*previousURL*, start:*startTime*, end:*endTime*, time:time, description:*previousDescription*};  
 http.open("POST", "http://localhost:" + *port* + "/sniff", false); *// false for synchronous request* http.setRequestHeader("Content-type", "application/json");  
 http.send(*JSON*.stringify(TimedUrl));  
}

### Détection des processus

En parallèle de la détection de l’activité web, il est nécessaire de déceler l’activité des processus Desktop. La stratégie est la suivante pour chaque activité :

* Récupérer le processus courant
* Récupérer le temps de début
* Détecter le changement de processus courant
* Récupérer le temps de fin
* Envoyer les informations au serveur
* Le serveur écrit dans le log ce qu’il s’est passé, combien de temps et dans quelle catégorie doit aller l’information (Games ou Networks)

Le code permettant de le faire est le suivant :

* Récupération du processus courant

function *setCrtProcess*() {  
 crtProcess = aw.sync();  
 *writeProcess*();  
}  
function *getCrtProcess*() {  
 return crtProcess.owner.name;  
}

* Détection du changement de processus courant et écriture dans le fichier log

function *writeProcess*() {  
 endTime = moment(new *Date*());  
 if (endTime.hours() === 0 && endTime.minutes() === 0 && endTime.seconds() === 0) {  
 *console*.log("new day merge");  
 var filename = os.userInfo().username + "\_processes";  
 var ws = fs.createWriteStream(filename, {flags: 'a'});  
 var entry = date + "|" + lastProcess.owner.name + "|" + lastProcess.owner.path + "|" + lastProcess.owner.processId + "|" + startTime.format() + "|" + endTime.format() + "|" + lastProcess.title;  
 ws.write(entry + "\n");  
 ws.end();  
 startTime = endTime;  
 lastProcess = crtProcess;  
 lm.*mergeLogs*(function() {  
 });  
 } else if (lastProcess === "") {  
 startTime = endTime;  
 lastProcess = crtProcess;  
 } else if (crtProcess.owner.name !== lastProcess.owner.name) {  
 var filename = os.userInfo().username + "\_processes";  
 var ws = fs.createWriteStream(filename, {flags: 'a'});  
 var entry = date + "|" + lastProcess.owner.name + "|" + lastProcess.owner.path + "|" + lastProcess.owner.processId + "|" + startTime.format() + "|" + endTime.format() + "|" + lastProcess.title;  
 ws.write(entry + "\n");  
 ws.end();  
startTime = endTime;  
 lastProcess = crtProcess;  
 }  
}

### Détection de la catégorie

Dans l'optique d'afficher des graphiques pour chaque catégorie d'activité (jeux vidéos et réseaux sociaux), il faut pouvoir déterminer à quelle catégorie appartient chaque entrée. Pour cela, plusieurs tests ont été implémentés, différents selon la catégorie à tester. Tous les tests sont menés dans le fichier charts.js qui utilise en grande partie des listes, toutes placées dans le dossier checkFiles.

#### Tests pour les jeux vidéos

Certains tests s'adressent aux sites web visités alors que d'autres s'adressent aux jeux installés en local.

#### Test par dictionnaire (méthode checkLists)

Le premier type de test est la recherche dans un dictionnaire. Ce type de test apparaît trois fois pour les jeux vidéos.

Le premier test s'applique aux sites web comme aux processus locaux. Le système recherche si l'url ou le nom de l'exécutable se trouve dans la liste des jeux déjà détectés par le système. Cette liste se nomme gameEntries.txt.

Le deuxième test s'applique uniquement aux processus locaux. Le système vérifie si le nom du deuxième dossier parent dans le chemin de l'exécutable est un nom de site de téléchargement de jeux, autrement dit, si le nom du dossier se trouve dans la liste GamesDownloadSitesList.txt.

#### Test par mot clé (méthode checkKeywords)

Ce type de test est assez similaire au précédant. Il apparaît deux fois pour les jeux vidéos dans notre application.

Le premier test s'applique aux sites web uniquement. Le système vérifie si l'url ou la description contiennent les mots clés définits dans le tableau gameKeywords du fichier charts.js.

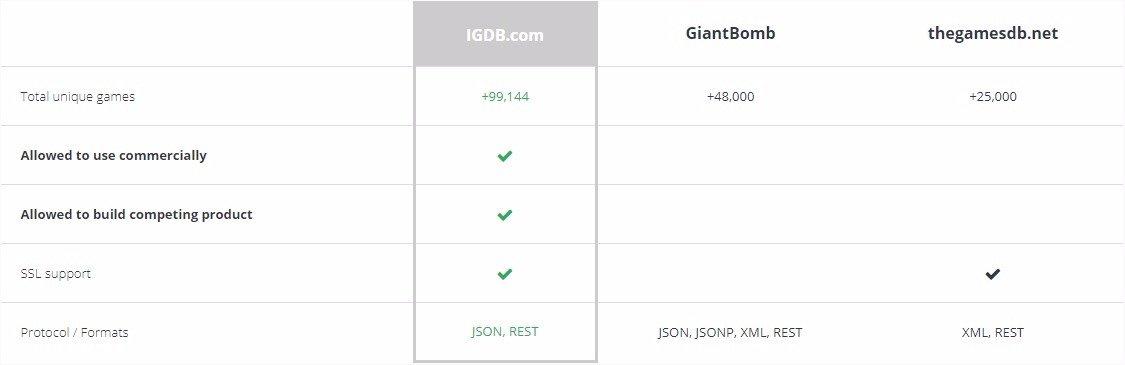
Le deuxième test s'applique aux processus locaux. Le système vérifie si le chemin de l'exécutable ou le titre de la fenêtre contiennent les mêmes mots clés que dans le premier test.

La difficulté liée à cette recherche par mot clé est que certains mots clés sont contenus dans d'autres mots qui n'ont aucun lien entre eux. On peut citer l'exemple du mot clé "jeu" qui peut prendre les formes suivantes :

"Jeux", "jeux", " jeux", "enjeu", "jeudi", " jeune". C'est pour cette raison que l'on peut obtenir des faux positifs de jeux vidéos.

#### Test par API

Plusieurs API de jeux sont actuellement disponible. On peut voir dans la Figure XXXXXXXXX ci-dessous un tableau comparatif des 3 services mis à disposition des développeurs.



La restriction d'IGDB s'élève à 3000 requêtes simples et 100 requêtes étendues par mois pour une utilisation gratuite. Celle de GiantBomb ne limite pas le nombre de requêtes mais l'espace de temps entre chaque requête qui doit être au miminum une seconde. En ce qui concerne thegamesdb.net, aucune restriction n'a été annoncée dans la version 1. Les restrictions seront mises en place dans l'API version 2.

Pour le cadre de ce projet, nous avons décidé d'utiliser la première API (IGDB.com) car elles recense le plus de jeux et le quota pour une utilisation gratuite nous a semblé suffisant. De plus, un wrapper Node.js est également disponible pour requêter plus facilement sur cette API.

Dans ce test, le système vérifie si le dossier parent de l'exécutable ou le titre de la fenêtre ou le nom de l'exécutable se trouve dans la base de données IGDB. Il est évident que cette recherche ne donne pas toujours des résultats satisfaisants. En effet, la base de données ne contient pas tous les jeux existants et le nom de l'exécutable par exemple ne correspond pas toujours au nom du jeu.

Dans la Figure XXXXXXXXX ci-dessous, on peut voir l'appel à l'API IGDB.

function checkGameAPI(sitename, splittedEntry, callback) {  
 var splittedPath = splittedEntry[2].split((process.platform === "win32")? "\\" : "/");  
 var parentDirectory = splittedPath[splittedPath.length-2];  
 if (splittedPath[2] === "Google" || splittedPath[1] === "Window" || parentDirectory === "Application" || parentDirectory === "bin") {  
 callback(false);  
 } else {  
 client.games({  
 limit: 1,  
 offset: 0,  
 search: parentDirectory  
 }, ['name']).then(function (res) {  
 if (res.body !== "[]" && res.body.startsWith(parentDirectory)) {  
 fs.appendFileSync("checkFiles/gameEntries.txt", splittedEntry[1] + "\r\n");  
 lastCategory = "game";  
 callback(true);  
 } else {  
 client.games({  
 limit: 1,  
 offset: 0,  
 search: splittedEntry[7]  
 }, ['name']).then(function (res) {  
 if (res.body !== "[]" && res.body.startsWith(splittedEntry[7])) {  
 fs.appendFileSync("checkFiles/gameEntries.txt", splittedEntry[1] + "\r\n");  
 lastCategory = "game";  
 callback(true);  
 } else {  
 client.games({  
 limit: 1,  
 offset: 0,  
 search: sitename  
 }, ['name']).then(function (res) {  
 if (res.body !== "[]" && res.body.startsWith(sitename)) {  
 fs.appendFileSync("checkFiles/gameEntries.txt", splittedEntry[1] + "\r\n");  
 lastCategory = "game";  
 callback(true);  
 } else {  
 callback(false);  
 }  
 }).catch(function (err) {});  
 }  
 }).catch(function (err) {});  
 }  
 }).catch(function (err) {});  
 }  
}

#### Tests pour les réseaux sociaux

Des tests par dictionnaire et des tests par mot clé sont appliqués de la même manière que pour les jeux vidéos. Ce sont simplement les listes qui changent (socialEntries.txt et SocialNetworkslist.txt et socialKeywords).

On observe dans la Figure XXXXXXXXX ci-dessous un exemple de code de vérification par mot clé pour les réseaux sociaux.

if (!categoryFound) {  
 for (var j = 0; j < socialKeywords.length; j++) { // Social Network description test  
 if (isWebsite && splittedEntry[5].toLowerCase().includes(socialKeywords[j].toLowerCase())) {  
 fs.appendFileSync("checkFiles/socialEntries.txt", splittedEntry[1] + "\r\n");  
 lastCategory = "socialNetwork";  
 response = lastCategory;  
 categoryFound = true;  
 break;  
 } else if ( !isWebsite && splittedEntry[2] !== "" && splittedEntry[2].toLowerCase().includes(socialKeywords[j].toLowerCase()) ||  
 !isWebsite && splittedEntry[7] !== "" && splittedEntry[7].toLowerCase().includes(socialKeywords[j].toLowerCase())) {  
 fs.appendFileSync("checkFiles/socialEntries.txt", splittedEntry[1] + "\r\n");  
 lastCategory = "socialNetwork";  
 response = lastCategory;  
 categoryFound = true;  
 break;  
 }  
 }  
}

### Ecriture du fichier log

#### Structure du fichier de log

Les activités de l'utilisateur sont récupérées et écrites dans un fichier de log selon le format suivant :

"Nom de l'utilisateur système"\_log\_"Date des entrées du fichier"

On peut donc récupérer un fichier de log par jour par utilisateur.

Le contenu de ce fichier log est composé des entrées des fichiers :

"Nom de l'utilisateur système"\_extension (Récupération des activités sur chrome)

"Nom de l'utilisateur système"\_processes (Récupération des processus locaux)

Les entrées du fichier extension ont le format suivant :

Date de capture|url complète|Date début|Date fin|Temps (Fin - début)|Metadonnées de l'url (Description)

Exemple d’une entrée du fichier extension :

2018.05.11|http://www.jeuxjeuxjeux.fr/jeu|2018-05-11T11:40:43+02:00|2018-05-11T12:02:43+02:00|1320|  
Makes a robust determination of a user's timezone through Javascript.

Les entrées du fichier processes ont le format suivant :

Date de capture|Processus|Chemin du processus|ProcessID|Date début|Date fin|Temps (Fin - début)|Titre de la fenêtre

Exemple d'une entrée du fichier processes :

2018.05.11|Telegram.exe|C:\Users\Nicolas\AppData\Roaming\Telegram Desktop\Telegram.exe|6076|  
2018-05-11T18:10:26+02:00|2018-05-11T18:10:27+02:00|1.002|TelegramDesktop

L'unité du champ Temps est indiquée en secondes.

Le contenu du fichier log n'est pas organisé de manière séquentielle. C'est une simple concaténation des deux fichiers sources (extension et processus). Chaque source par contre est organisée de manière séquentielle dans le fichier log. La structuration des entrées est exécutée lors de la génération des fichiers jsons qui sont eux directement exploitables pour l'affichage des graphiques dans l'interface java.

#### Stratégie de fusion des deux fichiers sources

Le code de fusion des deux fichiers sources est appelé périodiquement selon un timeout définit dans le code. C'est le fichier logMerger.js qui est chargé de cette tâche. La méthode mergeLogs récupère le contenu des fichiers log, extension et processes s'ils existent et applique une simple concaténation des fichiers sources (extension et processes) avec le fichier log. Aucun traitement particulier n'est lancé après la concaténation. Les données contenues dans le log sont brutes.

La fusion automatique est arrêtée 5 minutes avant minuit. La fusion sera à ce moment là prise en charge par la méthode writeProcess du fichier processWatcher.js.

Dans la Figure XXXXXXXXX ci-dessous, on voit que les méthodes setCrtProcess et mergeLogs sont lancées à chaque interval de temps respectif.

// detection process  
var pw = require('./routes/processWatcher');  
setInterval(pw.setCrtProcess, 1000);  
  
// file merging process  
var lm = require('./logMerger');  
setInterval(lm.mergeLogs, 600000); //1000(ms) = 1s

On observe dans la Figure XXXXXXXXX ci-dessous que la méthode mergeLogs est interrompue lorsque l'horloge indique 5 minutes avant minuit.

function mergeLogs() {  
 console.log("Merging process");  
 var blockingDate = moment(new Date());  
 if (blockingDate.hours() === 23 && blockingDate.minutes() > 55) {  
 return;  
 }  
  
 var source1 = os.userInfo().username + "\_extension";  
 var source2 = os.userInfo().username + "\_processes";

Dans la Figure XXXXXXXXX ci-dessous, on remarque que la fusion des fichiers est lancée à minuit.

function writeProcess() {  
 endTime = moment(new Date());  
  
 if (startTime !== null) {  
 var day = startTime.date();  
 var month = startTime.month() + 1;  
 var date = startTime.year() + "." + ((month < 10) ? "0" : "") + month + "." + ((day < 10) ? "0" : "") + day;  
 }  
 if (endTime.hours() === 0 && endTime.minutes() === 0 && endTime.seconds() === 0) {  
 writeToProcesses(date);  
 lm.mergeLogs();  
 } else if (lastProcess === "") {  
 startTime = endTime;  
 lastProcess = crtProcess;  
 } else if (crtProcess.owner.name !== lastProcess.owner.name) {  
 writeToProcesses(date);  
 }  
}

#### Cas de minuit

Pour avoir le fichier de log du jour précédent à jour, une fonction inclue dans l'extension chrome vérifie s'il est une seconde avant minuit (la fusion est exécutée à minuit). Si c'est le cas, l'extension considère que l'activité de l'utilisateur se termine à minuit pour le jour précédent et débute à minuit pour le jour suivant, comme on peut le voir dans la Figure XXXXXXXXX ci-dessous.

// Check si l'heure est 11h59  
setInterval(function() {  
 var date = new Date();  
 if (!hasBrowserFocusOut && date.getHours() === 23 && date.getMinutes() === 59 && date.getSeconds() === 59) {  
 endTime = new Date(date.getTime()+1000);  
 sendTimedURL();  
 startTime = endTime;  
 }  
}, 1000);

Le cas critique se produit lorsque l'utilisateur change d'onglets très rapidement dans chrome une seconde avant minuit. Ce cas critique a été traité de la manière citée ci-dessus pour éviter un accès concurrentiel au fichier extension. En voici un scénario :

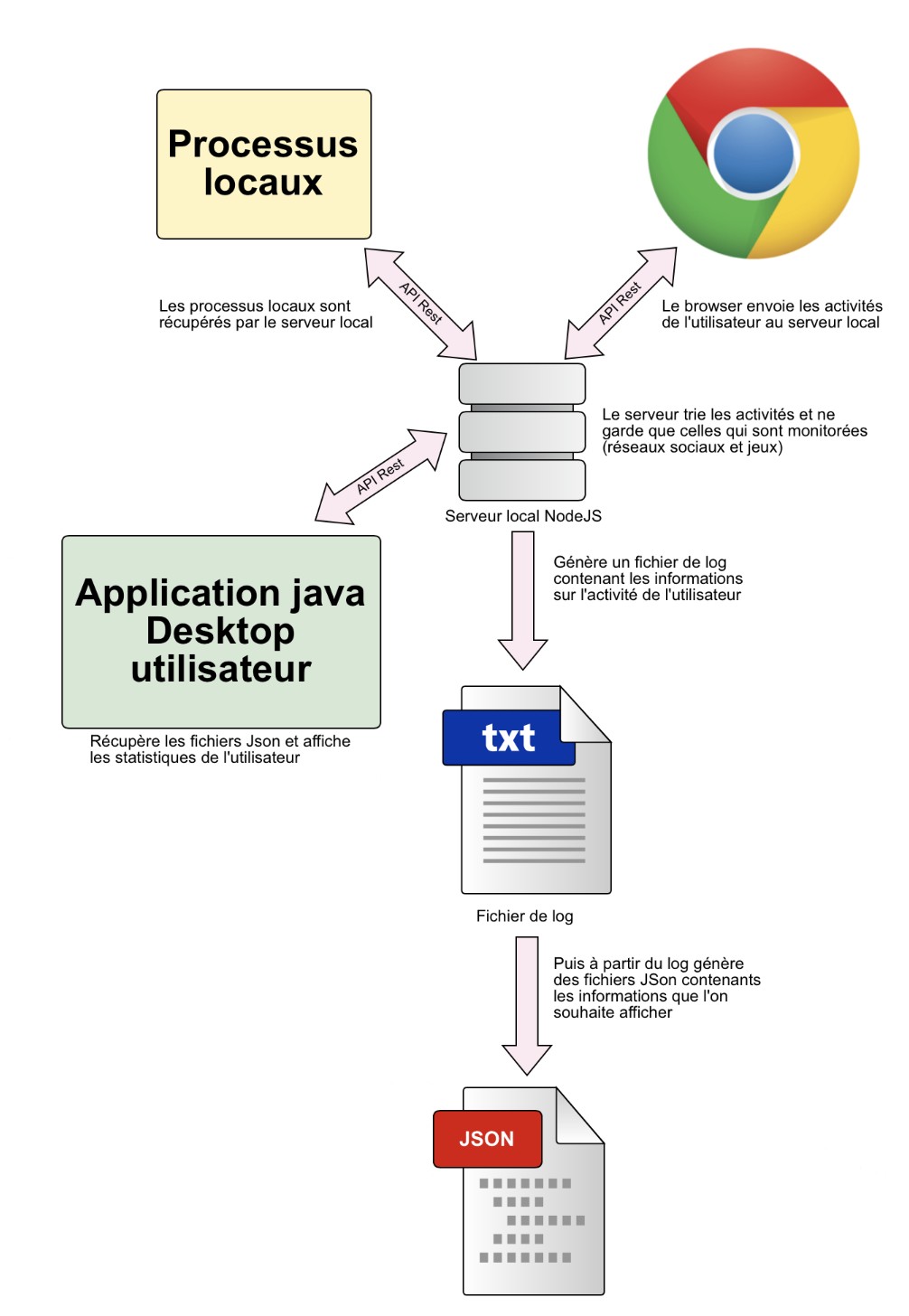
L'utilisateur ouvre un nouvel onglet pour voir un film à 11h30 du soir. Il reste sur la page de son film en continu jusqu'à 00h45 du matin. L'entrée serait envoyée au serveur qui lui séparerait le temps passé avec la date de minuit. La première partie serait concaténée sans problème avec le fichier précédent qui n'est plus accédé par personne. En revanche, la deuxième partie ne nous garantit pas qu'une fusion n'est pas en cours.

### Diagramme de classes

## Prototype 2

Le prototype 1 est complété par les fonctionnalités suivantes, ce qui constitue le prototype 2. Son fonctionnement est le suivant :

* La partie serveur
  + Génère des Json à partir du fichier de log
  + Fournit une interface permettant à une application Java de récupérer ces fichiers à l’aide de requêtes HTTPS
* L’application Java
  + Récupère les fichiers à l’aide l’interface fournie
  + Parse les fichiers
  + Stocke les informations dans des objets
  + Affiche les informations dans des graphiques



### Génération des fichiers jsons

Deux fichiers jsons séparés ont été générés pour créer les deux graphiques dans l'interface java. Le premier json calcul le temps total de chaque activité différente (sites web et processus). C'est la génération de ce premier json qui s'occupe de catégoriser chaque entrée. Pour cela, le fichier de log est trié par ordre alphabétique. Toutes les entrées concernant un site web ou un processus sont consécutives. Le second json se sert de la catégorisation du premier json pour trier ses données. Ce deuxième fichier lui divise la totalité des activités (sites web et processus) en heure sur un jour. Pour faire cela, la structure de chaque entrée est modifiée pour avoir en deuxième et troisième position les dates de début et de fin. Un tri par ordre alphabétique est ensuite appliqué pour finalement obtenir un fichier séquentiel.

Voici le code de la génération du premier json dans la Figure XXXXXXXXX ci-dessous. On voit bien que c'est dans cette première génération de json que l'appel à getCategory est lancé.

router.get("/pieChart", function(req, res) {  
 date = getDate();  
 var filename = req.query.user + "\_log\_" + date;  
 entries = fs.readFileSync(filename, "utf8").split("\n");  
 entries = entries.sort();  
  
 getCategory(entries, 0, function () {  
 var index = activityName.indexOf(lastSite);  
 if (index === -1) {  
 activityName.push(lastSite);  
 activityTime.push(accumulatedTime);  
 activityCategory.push(lastCategory);  
 } else {  
 activityTime[index] += accumulatedTime;  
 }  
  
 var piejson = {date: date, games: [], networks: []};  
 for (var i = 0; i < activityName.length; i++) {  
 if (activityCategory[i] === "socialNetwork") {  
 piejson.networks.push({name: activityName[i], totalTime: activityTime[i]});  
 } else if (activityCategory[i] === "game") {  
 piejson.games.push({name: activityName[i], totalTime: activityTime[i]});  
 }  
 }  
  
 console.log(piejson);  
 res.send(piejson);  
 });  
});

### Récupération des fichiers jSons

Lorsque l’on souhaite consulter les activités d’un utilisateur (depuis l’application utilisateur ou l’application superviseur), il est nécessaire de récupérer les fichiers Json générés précédemment. Pour ce faire, le fonctionnement est le même pour les 2 applications : il faut envoyer des requêtes HTTP au serveur afin que ce dernier envoie les fichiers pour les deux diagrammes. Les applications envoyent deux requêtes successivement.

**Côté client**

Une classe JsonReader est disponible depuis les applications Java. Cette classe possède une instance de la classe ServerConnection qui permet d’envoyer des requêttes HTTP. Dans le constructeur de la classe JsonReader cet attribut ServerConnection est instancié. Ensuite les requêtes sont appelées.

Dans la Figure XXX, le constructeur de la partie Java utilisateur. Nous constatons que la requête HTTP est envoyée en localhost, car c’est le serveur sur sa propre machine qui est appelé. De plus, le nom de session de l’utilisateur est utilisé pour construire cette requête. C’est pourquoi il effectue l’appel à la commande System.getProperty(« user.name »).

public JsonReader(){  
 serverConnection = new ServerConnection();  
 myUserName = System.getProperty("user.name");  
 userInfosPieChart = serverConnection.getUserInfo("http://localhost:"+Main.port+"/charts/pieChart?user="+myUserName);  
 userInfosBarChart = serverConnection.getUserInfo("http://localhost:"+Main.port+"/charts/barChart?user="+myUserName);  
}

Pour la partie Java superviseur, le constructeur de JsonReader est différent : il récupère l’utilisateur à superviser à travers une variable statique et il récupère ses champs. La Figure XXX qui crée les requêtes HTTP pour la partie superviseur :

public JsonReader(){  
 serverConnection = new ServerConnection();  
 userInfosPieChart = serverConnection.getUserInfo("http://"+supervisedUsersList.*hostToSend*.getIp()+":"+Main.*port*+"/charts/pieChart?user="+supervisedUsersList.*hostToSend*.getUser()); userInfosBarChart = serverConnection.getUserInfo("http://"+supervisedUsersList.*hostToSend*.getIp()+":"+Main.*port*+"/charts/barChart"); }

Dans la classe ServerConnection, c’est la méthode « getUserInfo » qui envoie les requêtes au serveur. Elle reçoit l’URL en paramètre. Ce paramètre est donné par la class JsonReader lors de l’appel à la méthode de l’object ServerConnection. Le code se trouve dans la Figure XXX :

public String getUserInfo(String targetURL){  
 connection = null;  
 String userInfos="";  
 try {  
 url = new URL(targetURL);  
 connection = (HttpURLConnection) url.openConnection();  
 connection.setRequestMethod("GET");  
 BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));  
 userInfos = br.readLine();  
 } catch (MalformedURLException e) {  
 e.printStackTrace();  
 } catch (ProtocolException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return userInfos;}

**Coté serveur**

La requête appelée sur le serveur pour récupérer le Json du graphique à barre est la suivante :

router.get("/barChart", function(req, res)

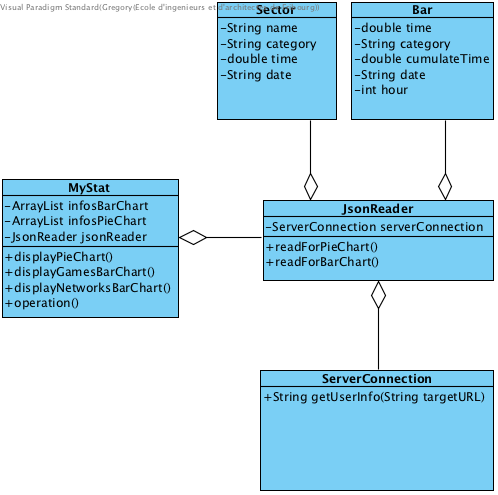
La requête appelée sur le serveur pour récupérer le Json du graphique circulaire est la suivante :

router.get("/pieChart", function(req, res)

Le code de ces requêtes est expliqué dans le chapitre 4.3.1

### Affichage des informations des Json dans des graphiques

Pour mieux comprendre cette partie, il faut se référer à la Figure XXX ci-dessous :



Pour faciliter la lecture, ce sont uniquement les méthodes et attributs principaux qui ont été affiché. Il existe d’autre méthodes et attributs internes mais le but est d’illustrer le fonctionnement général de cette partie du logiciel.

Explications du diagramme de classes :

* Main : Point d’entrée de l’application, permet à l’utilisateur de se logger
* MyStat : Classe qui affiche les graphiques et les autres informations de l’utilisateur. Elle contient une ArrayList de Secteurs (pour le graphique à secteurs) et de Bars (pour le graphique à barres). Elle possède un JsonReader qui lui permet de récupérer récupérer les fichier Json, de les parser et de créer des objets avec les informations
* Sector : Contient les informations de chaque secteurs pour le graphiques à secteurs
* Bar : Contient les informations de chaque barre pour le graphique à barres
* JsonReader : Permet de parser les fichier Json et de créer les objets nécessaires pour l’affichage dans les graphiques
* ServerConnection : Permet d’envoyer des requêtes HTTPS au serveur afin de récupérer les fichier Json que celui-ci à générer

Le code permettant l’affichage des informations dans les graphiques est le suivant :

Cette méthode récupère la liste de secteurs puis pour chaque entrée elle ajoute une données au graphique à secteur (JavaFX PieChart). Le PieChart crée ensuite lui-même la proportion entre les différentes entrées.

public void displayPieChart(){  
 ObservableList<PieChart.Data> pieChartData = FXCollections.*observableArrayList*();  
 for (int i = 0; i<infosForPieChart.size();i++){  
 pieChartData.add(new PieChart.Data(infosForPieChart.get(i).getName(),infosForPieChart.get(i).getTime()));  
 }  
 dailyPieChart.setData(pieChartData);  
}

La liste de barres contient à la fois les barres concernant la partie « Jeux vidéos » que la partie « réseaux sociaux ». Ainsi pour chaque entrées la méthode doit vérifier de quelle catéégorie il s’agit, et en fonction de ça, créer des barres pour l’une ou l’autre série du graphique à barre (JavaFX BarChart). Le graphique gère ensuite lui-même l’affichage des données en fonction de l’échelle choisie. Les axes sont les suivant :

* X : Heures de la journée (de 0 à 23)
* Y : Temps passé par heure de la journée (de 0 jusqu’à 60 minutes au maximum)

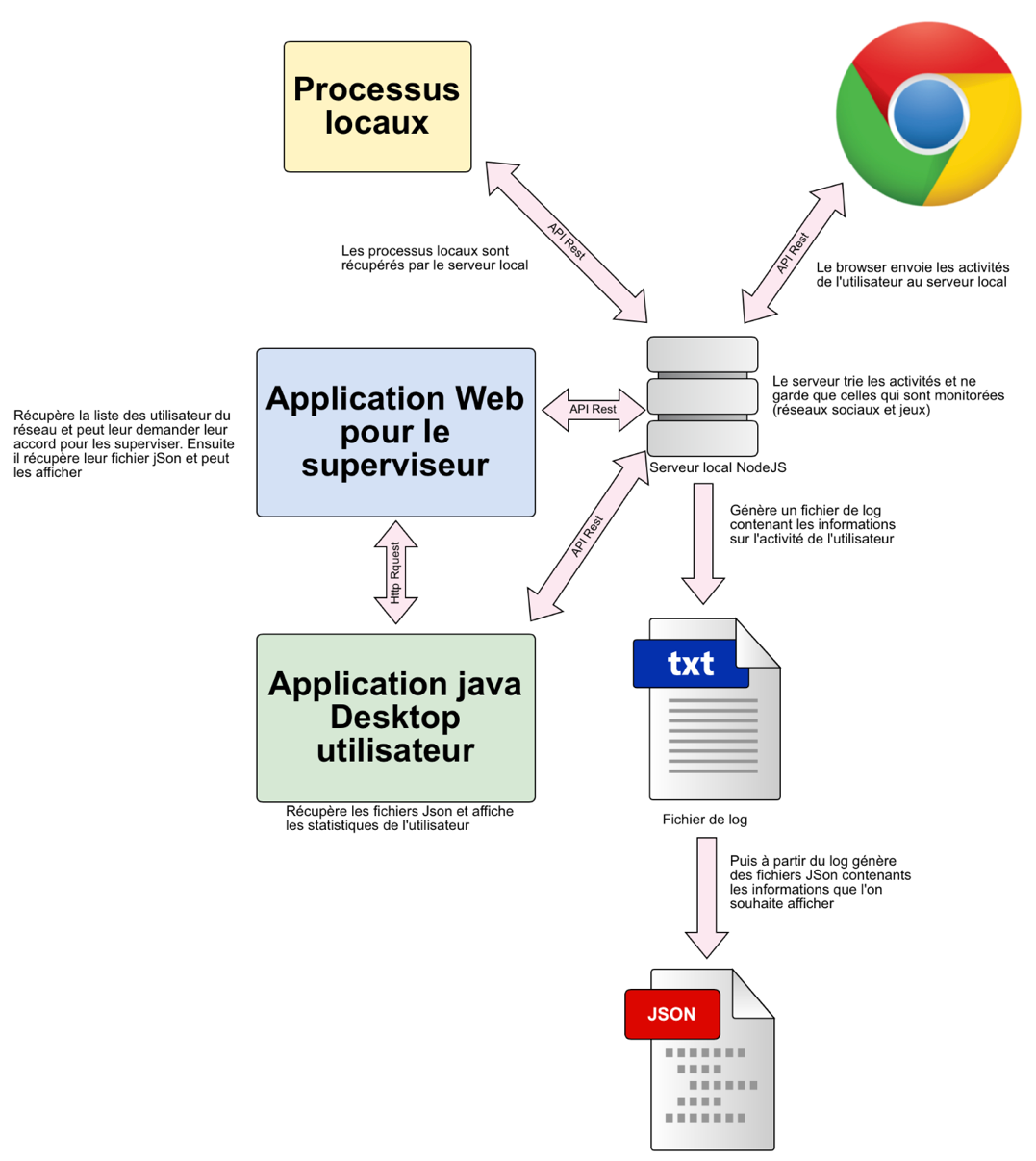
public void displayGamesBarChart(ArrayList<Bar>infosBarChart){  
 series1 = new XYChart.Series();  
 series1.setName("Games");  
 double time;  
 for(int i=0; i<infosBarChart.size();i++){  
 if(infosBarChart.get(i).getCategory().equals("games")){  
 time = (infosBarChart.get(i).getTime()/60);  
 series1.getData().add(new XYChart.Data(String.*valueOf*(infosBarChart.get(i).getHour()), time));  
 cumulateGamesTime += time;  
 }  
 }  
 gamesDisplayed = true;  
 dailyBarChart.getData().addAll(series1);  
}

Dans le code ci-desssus, c’est l’affichage pour la catégorie « jeux vidéos ». Le code pour la partie « réseaux sociaux diffère par les conditions du getCategory(). Il aurait aussi été possible de faire une seule méthode et passer une condition en paramètres. Cependant, cela n’a pas été dans le but de garder une séparation plus claire des méthodes.

## Prototype 3

Le prototype 3 contient la partie de récupération des utilisateurs du même réseau afin de pouvoir les surveiller. Cette partie concerne l’application Java du superviseur. Le fonctionnement général est le suivant :

* Détecter et afficher les utilisateurs du réseau
* Stocker les utilisateurs supervisés dans un fichier
* Permettre l’affichage des données d’un utilisateur supervisé



### Détection des utilisateurs du même réseau

Dans cette partie, on a besoin de connaître les informations des machines dans le même réseau local. Les machines détectées sont celles sur lesquelles est installée l'application utilisateur. Pour obtenir ces informations, on lance la commande nmap. Sur Mac OS X, le mot de passe administrateur est requis pour lancer la commande avec sudo. Si la commande n'est pas lancée avec les droits administrateur, le résultat est différent d'une machine Windows. Les adresses mac ne sont pas retournées dans la réponse. Cela impliquerait un traitement différent pour chaque système d'exploitation. C'est pour cette raison que le mot de passe administrateur est demandé sur Mac OS X et non sur Windows.

Voici dans la Figure XXXXXXXXX ci-dessous le début de la méthode getMachines qui permet de récupérer les informations des machines sur le même réseau local.

private List<Host> getMachines(NetworkInfo networkInfo) throws Exception {  
 Process p = null;  
 if(System.*getProperty*("os.name").equals("Mac OS X")) { *// do a readline because there is an additionnal line with mac os* displayPasswordDialog();  
 String[] command = new String[] {"/bin/bash","-c","echo "+password+" | sudo -S nmap -sP " + networkInfo.ip + "/" + networkInfo.subnetMask};  
 p = Runtime.*getRuntime*().exec(command);  
  
 }else{  
 String command = "nmap -sP " + networkInfo.ip + "/" + networkInfo.subnetMask;  
 p = Runtime.*getRuntime*().exec(command);  
 }  
 p.waitFor();  
 BufferedReader reader = new BufferedReader(new InputStreamReader(p.getInputStream()));  
 String line = "";

### Gestion de la liste des utilisateurs supervisés

Pour pouvoir gérer les utilisateurs supervisés, il est nécessaire de les stocker dans un fichier et pouvoir éditer celui-ci. Ce fichier est au format JSon. Lors de son ouverture, l’application va vérifier s’il existe un Json contenant les utilisateurs qu’elle supervise. Si c’est le cas, elle affiche ces utilisateurs et il est possible de les sélectionner pour afficher leur données. Pour que l’affichage des données fonctionne, il faut que le serveur soit enclanché. En effet, s’il ne tourne pas, l’application Java ne peut pas le requêter pour récupérer les informations de l’utilisateur. Si le fichier n’existe pas, rien n’est affiché dans la liste.

Le superviseur peut appuyer sur le bouton d’ajout. C’est à ce moment que le logiciel détecte les utilisateurs du même réseau. S’il y en a, ils sont affichés dans une liste. À ce moment là, le superviseur peut sélectionner un ou plusieurs utilisateurs à superviser. Lorsqu’il appuye sur le bouton « superviser », un fichier est créé avec les utilisateurs déjà supervisé s’il y en a et avec les nouveaux utilisateurs supervisés. Ensuite ce fichier est créé et il écrase le fichier d’utilisateurs supervisés précédants (s’il existe).

La Figure XXX montre un fichier Json avec un utilisateur supervisé.

{"users":

[{"ip":"160.98.127.214",

"name":"Nico-HP",

"user":"Nom : Nicolas\nNom complet : Nicolas Fuchs",

"mac":"D0-7E-35-66-A9-54"}

]

}

Le code dans la Figure XXX représente la récupération des utilisateurs supervisés actuels puis l’écriture de ceux-ci et des nouveaux utilisateurs à superviser dans le Json et ensuite la création de ce fichier Json.

hostList = discoveryUsersList.getSelectionModel().getSelectedItems();  
JSONObject jsonObject = new JSONObject();  
JSONArray jsonArray = new JSONArray();  
JSONObject user;  
if(existingHostList!=null){  
 for (int i=0; i<existingHostList.size();i++){  
 user = new JSONObject();  
 user.put("name", existingHostList.get(i).getName());  
 user.put("ip", existingHostList.get(i).getIp());  
 user.put("mac", existingHostList.get(i).getMac());  
 user.put("user",existingHostList.get(i).getUser());  
 jsonArray.add(user);  
 jsonObject.put("users",jsonArray);  
 }  
}  
for (int i=0; i<hostList.size();i++){  
 user = new JSONObject();  
 user.put("name", hostList.get(i).getName());  
 user.put("ip", hostList.get(i).getIp());  
 user.put("mac", hostList.get(i).getMac());  
 user.put("user",hostList.get(i).getUser());  
 jsonArray.add(user);  
 jsonObject.put("users",jsonArray);  
}  
discoveryUsersList.getItems().removeAll(hostList);  
FileWriter jsonWriter = new FileWriter(supervisedUsersList.*SUPERVISEDUSERSFILE*);  
jsonObject.writeJSONString(jsonWriter);  
jsonWriter.close();

La liste des utilisateurs supervisés a des écouteurs sur chaque entrées. Si l’on double clic sur une entrée, les données de l’utilisateurs sont affichées. Il est ensuite possible de revenir à cette liste.

Lors de la suppression, l’utilisateur a supprimer disparaît de la liste. Puis les éléments restants de cette liste sont écrits dans un nouveau fichier Json et ce dernier écrase l’ancienne liste d’utilisateurs supervisés.

## Interfaces homme machine

Dans le chapitre de la conception au points 3.2.2 et 3.2.3 sont présentées les maquettes de l’application. Ce sont les interfaces qui sont prévues. Lors de l’implémentation, nous avons implémenté les Ihm suivantes :

Figure xxx affichage des utilisateurs lors de l’ouverture de l’application superviseur

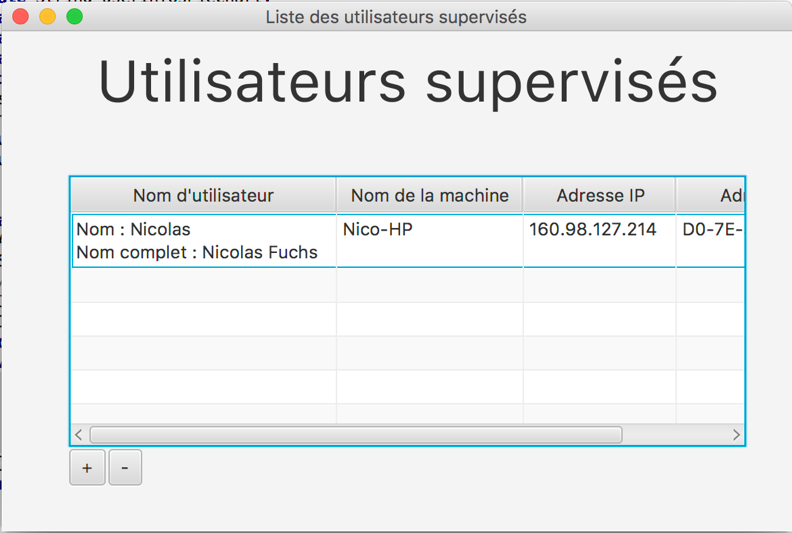


Figure XXX, demande de mot de passe administrateur pour lancer la recherche d’utilisateurs du même réseau. Un « nmap » est lancé et sur le mac il est nécessaire de donner le mot de passer admin de la session pour pouvoir lancer en Sudo.

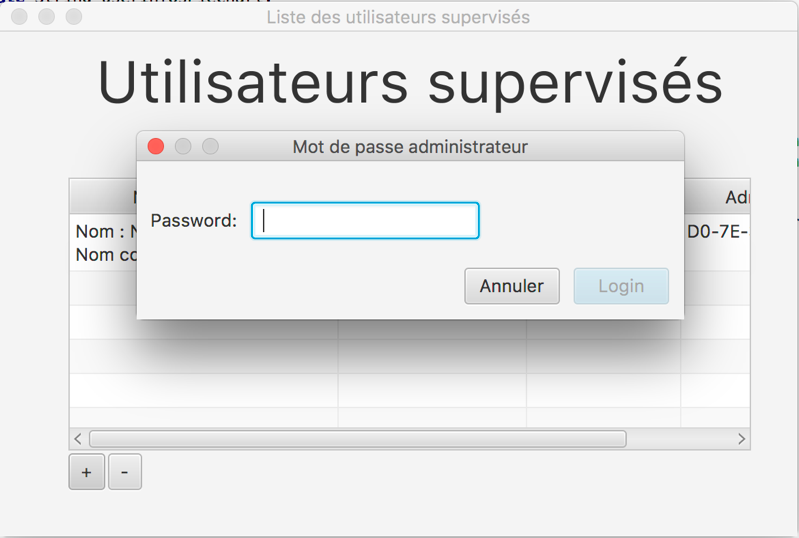


Figure XXX, liste des utilisateurs du même réseau

Figure XXX, la page d’affichage des informations de l’utilisateurs depuis l’application superviseur. Depuis la partie utilisateur, on voit les mêmes informations mais l’utilisateur ne peut pas éditer ses notes ni activer ou désactiver ses objectifs.

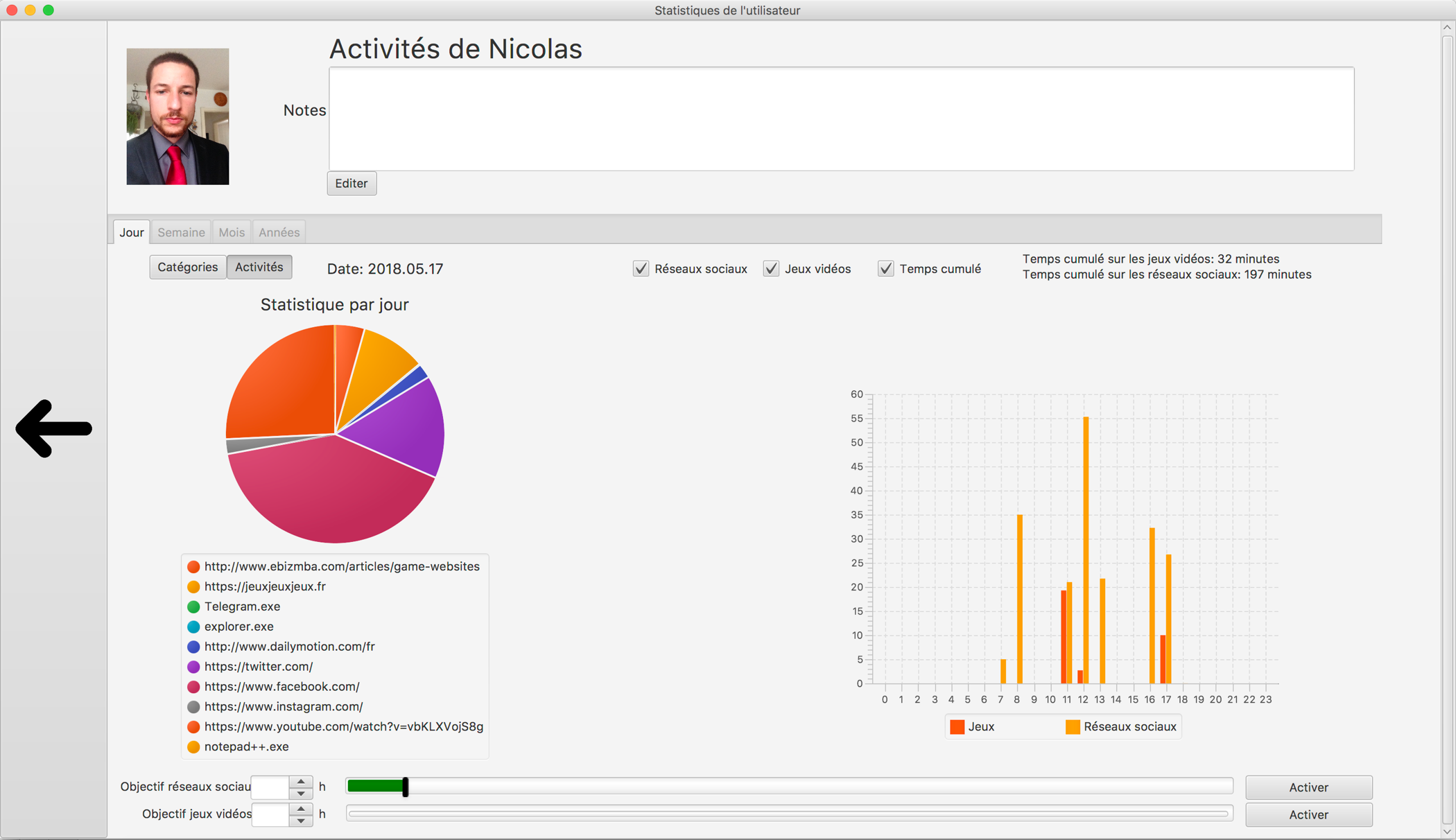


Figure XXX, lorsque l’on s’arrête sur un secteur, il est possible de voir le nom de celui-ci.

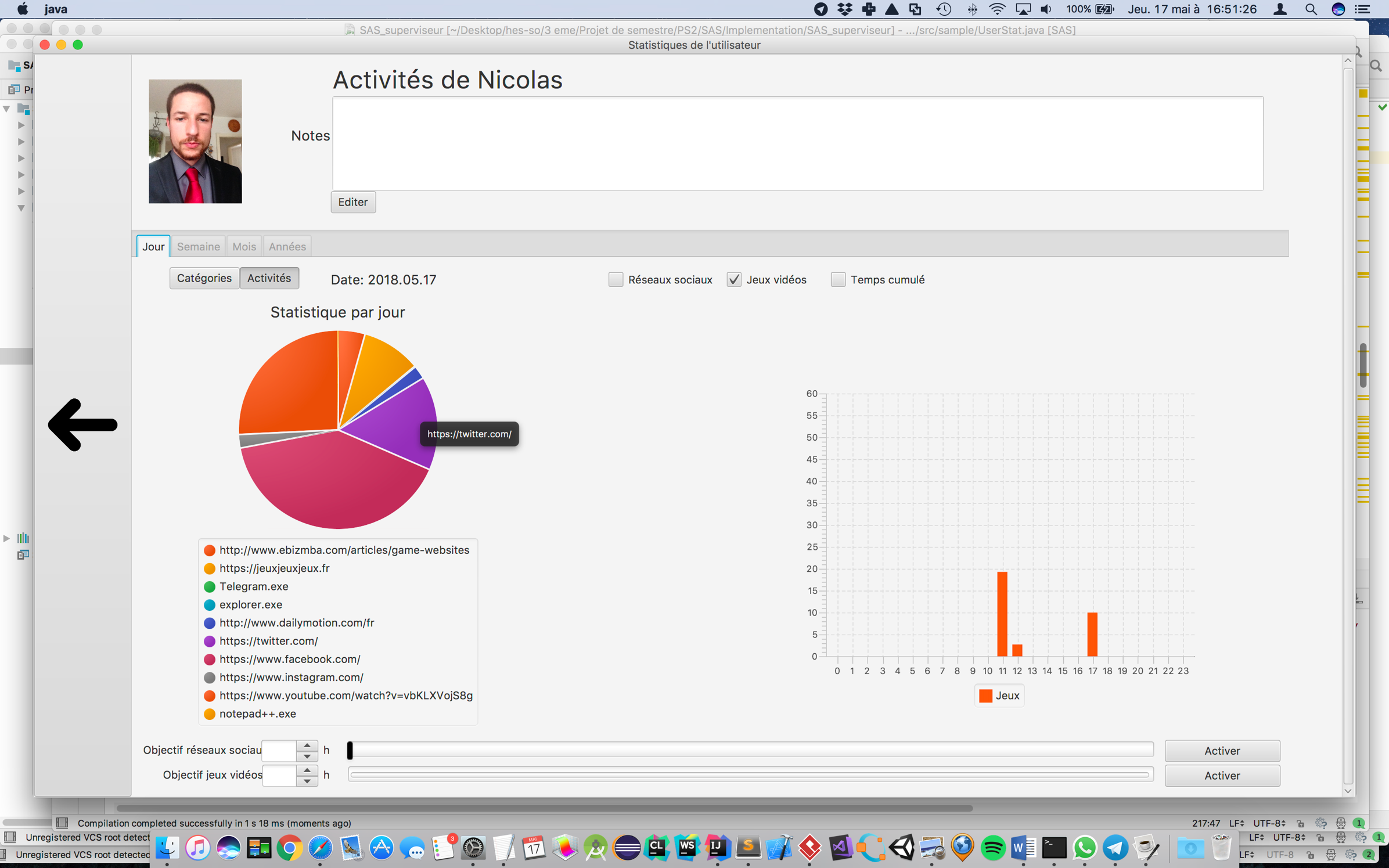
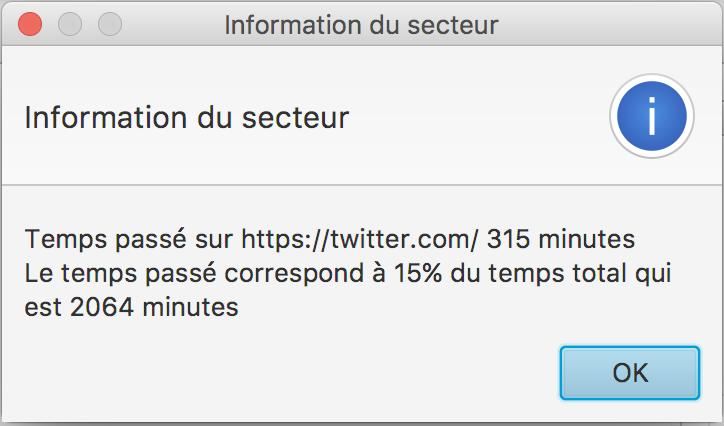


Figure XXX, si l’on clique sur un secteur, il est possible d’afficher des informations complémentaires sur celui-ci.



Les fonctionnalités des Figures XXX et XXX sont gérées par la création Tooltips et de boites de dialogues au moment du remplissage du graphique.

## Limites

**Cas limites**

Certains cas limites comme par exemple 3 jours d'affilés sur une activité n'ont pas été testés.

**Mac OS X vs Windows vs Linux**

Le système d'exploitation de la machine du superviseur peut être windows ou Mac OS X mais celui de la machine de l'utilisateur supervisé doit être windows en raison d'un bug non résolu avec active-win lorsque l'utilisateur replace le focus sur l'application java. Certaines méthodes spécifiques pour Mac OS X comme par exemple getUsers dans le fichier OSXSupervise.js n'a pas été implémentée.

Linux n'a pas du tout été inclu dans ce projet pour d'une part réduire le scope et d'autre part car nous avons estimé que les utilisateurs de linux sont moins nombreux et moins touchés par le thème du projet.

**Focus**

Notre application détecte le changement de focus, mais celui-ci ne détermine pas forcément la fin de l'activité. L'utilisateur peut lancer une vidéo dans une fenêtre, ouvrir une nouvelle fenêtre juste à côté et faire autre chose en parallèle. Le focus étant placé sur la nouvelle fenêtre, le temps passé sur l'onglet de la vidéo toujours active pourtant est stoppé. Si on voulait résoudre ce problème, il faudrait détecter la position des yeux pour voir où l'utilisateur regarde.

**Divers**

Actuellement, le logiciel de supervision (les 3 modules, serveur, extension chrome et application Java) doit être installé sur chaque sessions que nous souhaitons superviser. Cela signifie que le logiciel peut se retrouver à double sur une même machine.

## Conclusion

Dans cette phase du projet nous avons mis en œuvre ce que nous avions décidé dans la phase de conception. Au niveau architecture, la solution est composée de 4 parties :

* Un serveur NodeJS
* Une extension Chrome
* Une application Java utilisateur
* Une application Java superviseur

Avec ces parties, le logiciel est capable de :

* Récupérer et stocker des informations sur les activités de l’utilisateur (serveur et extension Chrome)
* Les filtrers et créer des fichiers d’informations pertinentes (serveur)
* Mettre à disposition ces fichiers pour des applications java (serveur)
* Les applications java sont capables de mettre en valeur les informations contenues dans les fichiers en les affichant dans des graphiques (application Java superviseur et utilisateur)

Les applications java se ressemblent mais certaines fonctionnalités sont différentes. Nous avons dédoublé le code commun et créé deux projets différents afin d’avoir une meilleure structure. Cela a parfois été un contretemps lorsqu’il y avait des modifications à effectuer sur la partie commune. Au niveau du planning, nous avons été dans les temps et nous sommes satisfait de notre partie de réalisation.

# Validation

## Introduction

## Tests d’utilisabilité

## Test utilisateur

## Test empiriques

## Conclusion

# Problèmes rencontrés

**Gestion des dates**

Les dates sont implémentées différement entre le javascript dans l'extension chrome et le node.js du serveur local. Toutes les dates sont décalées et les données deviennent incohérentes.

Solution implémentée : Utilisation de la librairie moment du côté serveur, emploi de méthode telle que getUTCDate, etc... Cette solution n'est cependant pas homogène et pas très propre.

Solution idéale : Pour rester cohérent dans toute l'application, il aurait fallu obtenir la date du serveur depuis l'extension chrome en exposant une méthode getDate par exemple dans notre API Rest.

**Filtres de catégorisation (Tests)**

Les tests de catégorisation d'activité ne sont pas fiables à 100%, en particulier la recherche dans l'API des jeux ainsi que la recherche de mot clé. Les filtres ne sont pas assez restrictifs et précis pour obtenir de bons résultats.

Solution (non-implémentée) : Consacrer plus de temps à cette partie de catégorisation et affiner les filtres.

**Active-win**

Comme mentionné dans le chapitre 4.6 Limites, un bug est survenu sur Mac OS X avec l'utilisation de la librairie active-win lorsque l'on replace le focus sur l'application java alors que le serveur tourne. Nous avons également essayé d'utiliser en premier lieu une autre librairie, active-window. Se rendant compte qu'elle n'était ni stable ni fonctionnelle, nous avons décidé d'utiliser la librairie active-win.

Solution (non-trouvée) : Résoudre le bug

**Mac OS X vs Windows**

Certaines informations comme les utilisateurs détaillés du système d'exploitation ne sont pas disponibles sur Mac OS X. Les données récupérables sur un OS ou l'autre sont parfois différentes, parfois indisponibles.

Solution (en partie implémentée) : Il faut écrire un traitement spécifique pour chaque système d'exploitation si disponible.

# Perspectives

Liste d'améliorations possibles de l'état actuel du système :

* Affichage des statistiques en fonction d'un jour choisi
* Précision des filtres de catégorisation
* Rendre le système multi-plateforme (Windows et Mac)
* Factoriser le code (format des dates et code superviseur/utilisateur)
* Rendre l'IHM responsive

Beaucoup de perspectives futures sont envisageables dans le cadre d'une suite de ce travail de semestre :

* Gestion des autorisations de supervision
* Gestion de la semaine, mois et année
* Extension du système pour Linux
* Mise en place d'un système de récompenses

# Conclusion

## Conclusions de groupe

Ce travail de semestre a été très intéressant dans le sens où plusieurs aspects d'un système informatique ont été abordés tels que l'IHM, le serveur REST et la partie réseau. La bonne entente au sein du groupe a également permis d'obtenir un résultat à la hauteur de nos attentes.

## Conclusions personnelles

### Grégory Ducrey

### Nicolas Fuchs

# Annexes

## Déclaration d’honneur

Nous, soussignés, Nicolas Fuchs et Grégory Ducrey, déclarons sur l'honneur que le travail rendu est le fruit de travaux personnels. Nous certifions ne pas avoir eu recours au plagiat ou à toutes autres formes de fraudes. Toutes les sources d'information utilisées et les citations d'auteur ont été clairement mentionnées.

## Bibliographie

Ces sites ont été visités sur la période du 03.2018 au 05.2018.

[1] [https://fr.wikipedia.org/wiki/Dépendance\_au\_jeu\_vidéo](https://fr.wikipedia.org/wiki/D)

[2] [http://www.jeuxvideo.com/dossiers/00018270/l-addiction-aux-jeux-video-temoignages-de-joueurs-006.htm](%22)

[3] <https://www.vice.com/fr/article/ppvzd7/dependance-jeux-video-843>

[4] <http://www.ot-lab.ch/?p=5605>

[5] <http://humanetech.com/take-control/>

[6] <https://selfcontrolapp.com>

[7] <https://support.apple.com/kb/PH25799?viewlocale=fr_FR&locale=hu_HU>

[8] <https://www.rescuetime.com/dashboard>

[9] <https://www.rescuetime.com/anapi/setup/overview>

[10] <https://itunes.apple.com/us/app/moment-screen-time-tracker/id771541926?mt=8>

[11] <https://www.swisscom.ch/fr/clients-prives/aide/internet/kinder-und-jugendschutz.html>

[12]<https://www.swisscom.ch/fr/business/pme/internet-reseaufixe-television/internet/services-supplementaires/internet-security.html>

[13] <https://www.rescuetime.com/rescuetime-pro>