



Léo Guilpain
TICB - IoT
Promotion 2019

SMARTVISER
Avenue des Champs Blancs 35510 - Cesson
Seigné
02 99 31 42 08

Johann Bourcier
Professeur des universités
à l'université de Rennes 1

Pascal Vilboux
Développeur Embarqué Android

CONFIDENTIEL

Rapport de Stage de Léo Guilpain

Année universitaire 2017-2018



Résumé

J'ai effectué un stage de trois mois chez SmartViser en tant que développeur embarqué Android. Smartviser est une jeune entreprise qui a créé une application permettant de faire de nombreux tests sur des téléphones portables. Lors de ce stage ai eu pour mission d'implémenter une nouvelle fonctionnalité dans cette application. Elle consistait à utiliser un nouveau type de lecteur vidéo (ExoPlayer). Une fois ce travail fait, j'ai été formé afin de pouvoir réaliser les mêmes tâches que le développeur Android présent au sein de l'entreprise. Après cette formation, j'ai poursuivi l'implémentation de nouvelles fonctionnalités dans l'application et j'ai également corrigé les différents problèmes rencontrés lors de l'utilisation de Viser.

I did an internship during three months at Smartviser and I worked as an android developer. SmartViser is a new company which has created an app which allows to do many tests on cellphones. During this internship, I had to implement a new fonctionnality in Viser. This fonctionnality allows users to choose between two medias player : MediaPlayer or ExoPlayer. I had to implement the fonctionnality ExoPlayer. Then, when this jobs was finished, I have been trained in order to realize the same job as the company's Android Developer. After this training, I continued to implement new functionalities and I began to correct different problems in the app.

Table des matières

Résumé.....	2
Introduction.....	4
Remerciements	5
Présentation de la société.....	6
Présentation de Smartviser	6
Explication de l'application Viser (annexe 2)	7
Organisation du travail chez SmartViser	8
Daily Meeting	8
Sprint	8
YouTrack (annexe 4).....	9
Fossil (annexe 5).....	9
Sprint 1	10
ExoPlayer	10
Réaliser un POC	10
Ajouter l'activité ExoPlayer dans Viser.....	12
Étude sur Android Go	13
Rétrospective.....	14
Sprint 2	14
Read & Write File	14
Viser Dummy App.....	16
Correction de Bugs	18
Rétrospective.....	18
Conclusion	19
Références.....	20
Annexes	21
Annexe 1 : L'équipe	21
Annexe 2 : Viser	22
Annexe 3 : ViserWebService (VWS)	24
Annexe 4 : YouTrack.....	26
Annexe 5 : Fossil.....	27

Introduction

C'est chez Smartviser que j'ai effectué mon stage. Ayant eu du mal à trouver un stage, je me suis rendu au forum du recrutement à Rennes en ayant au préalable fait une présélection des différentes entreprises répondant à ce que je recherchais. Smartviser à qui j'avais déjà envoyé un mail, m'intéressait. Sur place, la rencontre avec la personne représentant la société s'est agréablement passée. J'ai été retenu pour un entretien et lors de celui-ci, les deux CV que j'avais donnés (mail et forum) étaient en leur possession, signe de nos intérêts respectifs. La mission qui m'a été proposée sur l'application de l'entreprise m'a plu. À la vue du professionnalisme et de la considération montrés envers moi pendant l'entretien, j'ai accepté ce stage.

J'ai recherché un travail dans le développement Android étant donné que ce sujet traité en cours m'a beaucoup intéressé. Ceci explique le fait que j'ai postulé pour un poste de développeur Android.

L'objectif de ce stage chez Smartviser était pour moi d'approfondir le développement Android et donc la programmation. Il s'agissait également de mieux apprendre toutes les particularités du téléphone puisque l'entreprise est beaucoup basée sur les spécificités techniques de cet objet. De plus, mes quelques expériences en temps qu'intérimaire, n'ont jamais été en lien avec ma spécialité et donc là, je pouvais découvrir le monde de l'entreprise dans un domaine qui m'intéresse.

Dans un premier temps, je vais expliquer le fonctionnement de l'entreprise en parlant des différents logiciels et des méthodes de travail. Puis, je détaillerai les différentes tâches que j'ai réalisées tout au long de mon stage.

Remerciements

Tout d'abord, je souhaite remercier les personnes qui ont contribué au bon déroulement de ce stage et qui en ont fait une expérience enrichissante et passionnante.

J'adresse mes remerciements à mon maître de stage, **Mr Pascal Vilboux** qui m'a toujours considéré comme un membre à part entière de l'entreprise. Grâce à sa patience et ses explications, j'ai pu progresser de jour en jour à ses côtés.

Je remercie **Mr Renaud Grandbien** et **Mr Gilles Ricordel** qui m'ont accordé leur confiance en me permettant d'intégrer Smartviser.

Arrivé en même temps que moi dans la société en tant que stagiaire, je remercie **Pierre-Damien Denis**, avec qui je me suis très bien entendu et qui a donc facilité mon intégration.

Je tiens à remercier **toute l'équipe de chez Smartviser** pour son accueil et pour le soutien qu'elle m'a apporté dans les différentes tâches à accomplir.

Enfin, je souhaite également remercier les personnes qui m'ont apporté leur aide quant à la rédaction de ce rapport.

Présentation de la société

Présentation de Smartviser

Mon stage a eu lieu chez SmartViser [1]. Cette entreprise a été créée en 2014. L'équipe, composée de 12 personnes (**voir annexe 1**) a mis au point une solution automatisée basée sur un Test Bot embarqué, la seule solution qui place les Smartphones au cœur de l'expérience utilisateur.

Ce système est capable de générer automatiquement une suite de « User Actions » (Action Utilisateur) afin d'améliorer la précision des tests. Il fournit une évaluation plus approfondie et permet ainsi de faire des comparaisons plus fiables.

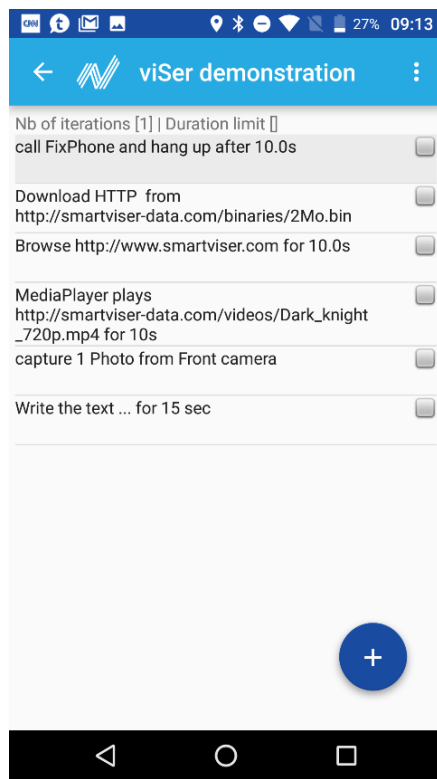
Le but de l'entreprise est d'optimiser mais également de simplifier le travail des opérateurs de tests. Ainsi, grâce à cette solution, tous les tests sont automatisés.

Leur système de tests collecte automatiquement les données afin de calculer le plus précisément possible les KPI (Key Performance Indicator).

En plus de tester les performances des téléphones, Viser permet aussi de connaître les performances réseaux. Ainsi, les opérateurs de communication sont très intéressés par cette solution automatisée que leur permet d'améliorer leur couverture réseau.

Explication de l'application Viser (annexe 2)

Dans cette application, l'utilisateur, en fonction de ses besoins, crée un scénario composé de plusieurs UA (User Action). Parmi ces scénarios, on peut trouver :



- **MO Call** : L'application génère un appel

- **Run Install App** : L'application lance une application déjà installée sur le téléphone

- **Camera** : L'application prend une photo ou une vidéo à l'aide de la caméra avant ou arrière.

Figure 1 : Exemple de scénario

Une fois que l'utilisateur a défini toutes les UA qu'il souhaite, il ne lui reste plus qu'à lancer le scénario. Les actions sont ensuite générées par l'application et à la fin de chacune d'elle, un résultat est donné : PASS ou FAIL.

Une fois le scénario terminé, l'utilisateur a accès à un résultat global en % de UA réussies. Il y a également le taux de batterie perdu lors du déroulement du scénario. (**voir annexe 2**)

Sur le téléphone, ce ne sont que des résultats synthétiques qui sont affichés. Pour avoir plus d'informations, l'utilisateur doit se rendre dans les fichiers de l'application lui permettant d'avoir accès au fichier JSON et au fichier CSV. Ces deux fichiers possèdent des informations plus précises quant au déroulement du scénario.

Il existe également un outil web pour pouvoir analyser les résultats : VWS (**annexe 3**)

Organisation du travail chez SmartViser

Daily Meeting

Tous les matins, l'équipe se réunit durant 15 minutes afin de discuter des tâches que chacun a accomplies la veille, ainsi que celle à venir pour la journée. Cela permet de savoir où en est chaque collègue mais également de s'entraider si une personne rencontre une difficulté.

Sprint

Depuis l'arrivée de Christophe (Technical Product Owner) dans la société, les tâches à réaliser sont plus distinctes et ordonnées. Les périodes de travail sont divisées sous forme de sprints qui se déroulent sur 3 semaines.

Au début du sprint a lieu une réunion afin de planifier les tâches finales, qui sont découpés sous forme de tâches intermédiaires.

Tout au long du sprint, un tableau est tenu à jour par chacun d'entre nous afin que tout le monde ait un regard sur l'avancée des différentes tâches en cours.



Figure 2 : Tableau permettant de suivre l'avancée des tâches

Sur la gauche de ce tableau on peut voir les différentes « story » mis en place par Christophe. Chaque story est affectée à une personne (magnet). À la suite, 4 colonnes permettent d'informer où en sont les tâches :

- **To do** : tâches planifiées mais non commencées
- **In Progress** : tâches en cours de déroulement
- **Done** : tâches terminées
- **To be reviewed** : partie devant être revue avec le pôle marketing

Enfin sur la droite, les bugs sont référencés avec différents ordres de priorité mais également les services qui correspondent aux travaux à réaliser avec les clients.

À la fin de ce sprint, une réunion est programmée afin que chacun expose son travail. Cette présentation se fait devant tous les membres de l'entreprise, ce qui permet à chacun d'être au courant du travail des autres et de donner son avis sur certains points.

Après cette réunion, il y a une rétrospective au cours de laquelle on discute de ce qui s'est bien passé ou non.

YouTrack (annexe 4)

Définition : *YouTrack est un système de suivi des bogues, un système de suivi des problèmes et un logiciel de gestion de projet développé par JetBrains.* [3]

Nous sommes deux à travailler en tant que développeur Android. Ainsi, lorsque la validation de l'application est faite, il y a souvent des bugs qui apparaissent. Ces derniers doivent être soumis dans YouTrack sous forme de ticket. Ces tickets permettent d'expliquer le problème en transmettant toutes les informations nécessaires à sa résolution. On peut aussi voir l'avancé de la correction du bug.

Fossil (annexe 5)

Définition : *Fossil est un logiciel de gestion de versions décentralisé intégrant, par le biais d'une interface web, un logiciel de suivi de problèmes et un wiki* [4]

En tant que développeur Android, lorsque l'on effectue une modification sur une partie du code, il ne faut pas qu'elle impacte les autres développeurs travaillant sur cette même partie. C'est pour cela que l'on utilise Fossil. Fossil fonctionne de la même façon que GitHub, c'est-à-dire que chaque développeur possède un répertoire local dans lequel il « stock » ses

modifications avant de les mettre en ligne sur le serveur. Ce logiciel permet ainsi aux autres personnes de ne pas être impactées par les modifications en cours mais également de les récupérer quand elles sont faites.

Sprint 1

ExoPlayer

Avant mon arrivée dans l'entreprise, l'application possédait une UA (User Action) « MediaPlayer ». Cette dernière permettait à l'utilisateur de simuler la lecture d'un média vidéo ou d'un média audio. Cependant, après plusieurs tests, des bugs ont été remontés par certains clients concernant cette UA. Afin de remédier à cela, l'entreprise a effectué des recherches et elle a trouvé un autre player : « ExoPlayer ». Il a donc fallu que je fasse des recherches sur ce Player pour pouvoir l'implémenter dans l'application.

Il faut savoir que l'UA MediaPlayer permettait à l'utilisateur de jouer une vidéo en local (parmi les fichiers existant sur le téléphone) mais également en remote (via un URL saisi par l'utilisateur).

De plus, différentes options s'offraient à l'utilisateur :

- **Loop back** : cela permettait à la vidéo de repartir à zéro une fois qu'elle avait été lue entièrement.
- **Time limit** : deux cas étaient possibles, si le temps n'était pas défini, la vidéo devait être jouée entièrement, si le temps était fixé, la vidéo stoppait au moment demandé.
- **Loading Time Out** : cela correspondait au temps de chargement de la vidéo. Si le temps de chargement dépassait le temps défini dans cette option alors la UA était FAIL.

Pour pouvoir intégrer ma version dans Viser, il m'a donc fallu réaliser une activité identique. Afin de préparer au mieux l'intégration dans l'application, j'ai suivi plusieurs étapes.

Réaliser un POC

Au départ, il ne s'agissait pas de copier l'UA Media Player mais seulement de réaliser un projet en dehors de Viser. Le but était de me familiariser avec ExoPlayer. J'ai donc, pendant

plusieurs semaines, fait des recherches afin de pouvoir créer une simple application permettant de lire un média. Dans un premier temps, l'URL était prédéfini dans le code de l'application et, dans un deuxième temps, j'ai créé un champ texte permettant de rentrer l'URL de la vidéo. Dans l'activité MediaPlayer se trouvait un code permettant de faire une recherche de fichiers sur le répertoire local. J'ai utilisé ce code pour moi aussi gérer les fichiers présents sur le téléphone, dans le stockage interne.

Après avoir fait ceci, il me restait encore à régler quelques détails tels que l'orientation de l'écran et la durée de la vidéo.

Une fois cette application réalisée, j'ai dû faire une présentation de mon travail à tous les salariés de l'entreprise afin d'avoir un retour sur ce que j'avais fait. Il faut savoir que l'utilisation d'ExoPlayer permet d'obtenir des nouveaux KPI tels que la qualité de la vidéo, les frames drop, le temps d'initialisation du décodeur,L'utilisation d'ExoPlayer est donc un plus pour l'entreprise car ces nouveaux KPI vont pouvoir être vendus aux clients.

Sur cette implémentation, j'ai travaillé en autonomie car personne n'avait de connaissances sur ExoPlayer. À la fin de cette première application, j'ai fait, avec Pascal mon tuteur et développeur Android, une revue de codes. Celle-ci m'a permis de me former aux normes de l'entreprise en ce qui concerne les noms de variables ou encore la place des différentes fonctions dans une activité.

Ajouter l'activité ExoPlayer dans Viser

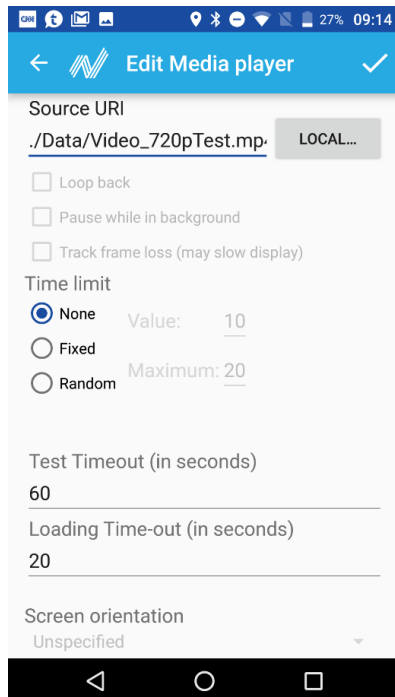


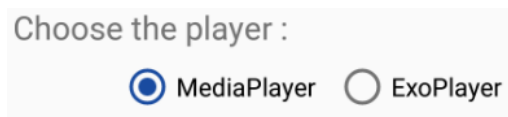
Figure 3 : UA Media Player

Dans un premier temps, mon activité ExoPlayer avait pour but de remplacer l'activité MediaPlayer. Ainsi, je devais être capable de gérer toutes les exceptions présentes dans MediaPlayer. J'ai donc dû recoder mon application à l'identique de MediaPlayer en utilisant les bonnes fonctions correspondant à ExoPlayer.

La première étape était de pouvoir faire en sorte que, lorsque toutes les conditions étaient validées, l'UA soit PASS. Une fois cette étape passée, il me suffisait qu'à gérer les erreurs telles que le retour en arrière ou les problèmes de connexion.

J'ai testé mon application sur de nombreux smartphones et j'ai pu constater qu'elle était fonctionnelle. Cependant, pour pouvoir conclure quant à la validité de l'application, il fallait qu'elle passe par une phase de tests plus précise organisée par l'ingénieur en tests.

Une fois validé, le pôle marketing a décidé qu'il ne fallait pas supprimer l'UA MediaPlayer, j'ai donc créé une CheckBox permettant à l'utilisateur de choisir entre MediaPlayer et ExoPlayer. Les scénarios prédéfinis auparavant vont utiliser le player par défaut qui est MediaPlayer. Ainsi, l'ajout d'ExoPlayer n'influe pas sur le reste de l'application.



Certaines des fonctionnalités présentes dans l'UA MediaPlayer n'ont pas été implémentées dans ExoPlayer. Comme l'interface graphique reste la même pour les deux UA, il a donc fallu griser les fonctionnalités qui n'étaient pas utilisées sous ExoPlayer.

Après avoir implémenter cette activité dans l'application, j'ai dû faire du refactoring sur les activités déjà existantes concernant les médias. Il s'agissait de remettre au propre le

code des activités concernées, tout en supprimant les différentes fonctions qui n'étaient plus utilisées.

Étude sur Android Go

Une fois l'activité ExoPlayer terminée, une nouvelle tâche m'a été confiée. Je devais faire une étude sur Android Go. Il faut savoir qu'Android Go est un nouveau type d'Android. Il est conçu pour les téléphones d'entrée de gamme. Ces derniers ne possèdent pas un mémoire vive importante et donc ne supportent pas tous les types d'applications. C'est pourquoi, il existe des applications « lite » fournies par google pour fonctionner sous Android Go tels que YouTube Lite, Gmail Lite, Google Lite, etc.

L'équipe a rencontré un problème au lancement de Viser sur des téléphones munis d'Android Go. J'ai donc dû analyser les résultats des traces renvoyées par le téléphone pour comprendre d'où venait le problème. Après plusieurs recherches, je me suis rendu compte que le problème venait de la RAM. En effet, lorsque Viser exécute des UA, la plupart d'entre-elles fonctionnent et les scénarios se finissent correctement. Cependant, lorsque les UA nécessitent l'ouverture d'un nouvel APK (format de fichiers pour Android), Viser est alors mis en arrière et se fait « killer » par le système. Ce mécanisme présent dans Android libère de la mémoire vive et permet ainsi au téléphone de fonctionner correctement. Il ne concerne pas seulement Viser mais également toutes les applications qui se retrouvent en arrière-plan. Sur Android Go, il est difficile de faire tourner plus d'une application à la fois.

Comme Viser est « killer », il n'est plus capable de finir correctement son scénario. Cela pose donc problème. Malheureusement, il n'existe pas de réelle solution pour résoudre ce problème. Ces téléphones sont des téléphones d'entrée de gamme et c'est pour cela qu'ils ne peuvent pas tout faire. Les seules solutions envisageables sont :

- Refaire l'application Viser de zéro et en faire un service capable de fonctionner en tâche de fond. A noter que nous n'avons aucune certitude quant à la faisabilité de ceci ainsi qu'à son bon fonctionnement.
- Avertir l'utilisateur avec une fenêtre popup au lancement de Viser sous Android Go que certaines UA peuvent faire crasher l'application et donc qu'il faut les éviter.

Rétrospective



Figure 4 : Rétrospective 1, on refait le match

Comme expliqué précédemment, nous faisons rétrospective tous ensemble à la fin du sprint. Sur cette image, on peut y voir en jaune les points positifs, en rose les points négatifs et en orange les améliorations possibles. En fin de séance, nous devions d'ailleurs en retenir trois pour leur mise en place dans le prochain sprint.

Dans les points négatifs, il est ressorti la difficulté à séparer les stories en tâches distinctes mais également le fait que certaines demandes des clients font changer les plans initiaux du sprint.

Suite à cette rétrospective, nous avons tous été d'accord pour dire que les démos présentées par chacun ont été très intéressantes. La mise en place de ce sprint a permis d'être plus cadré et d'avoir une meilleure visibilité en ce qui concerne le temps de développement.

Sprint 2

Read & Write File

Pour cette tâche, le fonctionnement est différent par rapport à ExoPlayer. En effet, lors de mon implémentation d'ExoPlayer, je me suis contenté de modifier une User Action déjà présente dans Viser. Or, ici, il s'agissait d'implémenter une UA totalement nouvelle dans Viser. J'ai donc dû suivre une procédure bien particulière afin de gérer l'ajout.

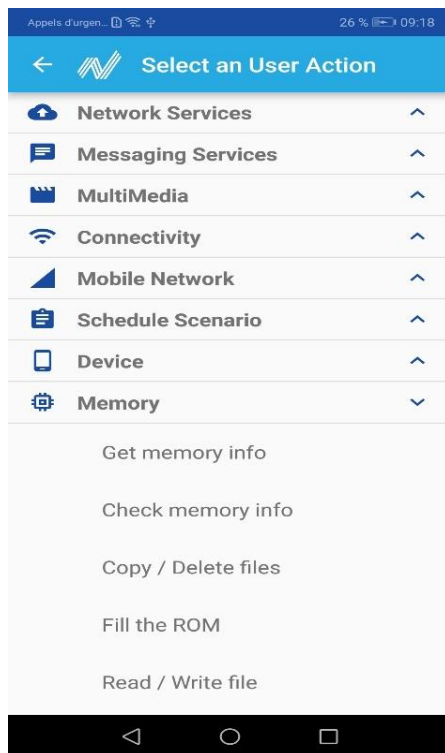


Figure 5 ; Ajout de la catégorie Memory

En plus de ceci, j'ai dû ajouter une nouvelle catégorie dans Viser, la catégorie « Memory ». Cela permet de regrouper toutes les UA un peu plus spécifiquement. La nouvelle User Action a été rédigée sur YouTrack durant l'année mais elle n'avait pas été traitée jusque-là. Dans un premier temps, il m'a fallu lire la documentation concernant cette story. Le but de cette nouvelle UA est de mesurer le débit d'écriture ainsi que le débit de lecture d'un fichier que l'on crée dynamiquement. La demande a été faite par un client spécifique, c'est pourquoi l'unité de mesure de la taille du fichier et du buffer ne sera pas en MegaByte mais en MebiByte (1 MiB = 1.048576 MB).

Dans la User Action, l'utilisateur doit renseigner 3 paramètres :

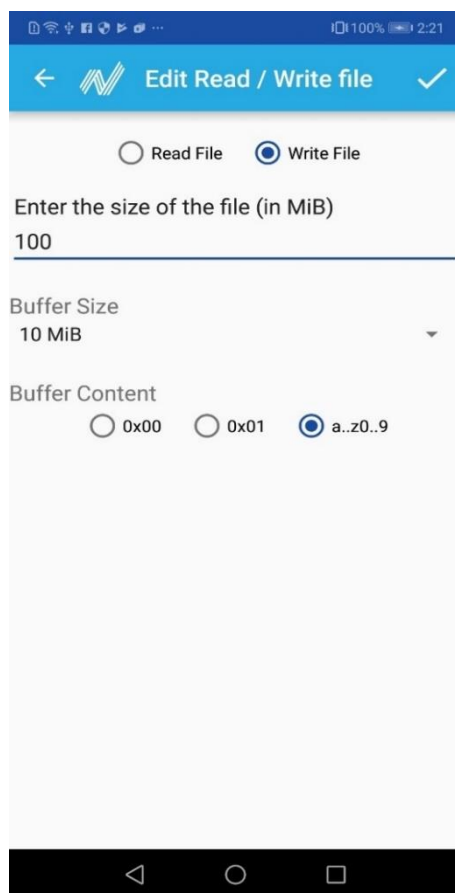


Figure 6 : UA Read / Write File

- **Lire ou écrire :** Le débit de lecture et d'écriture n'étant pas le même, l'utilisateur doit choisir ce qu'il souhaite analyser.
- **Taille du Fichier :** L'utilisateur devra renseigner la taille du fichier qu'il souhaite lire ou écrire.
- **Taille du Buffer :** En informatique, une mémoire tampon est une zone de mémoire vive ou de disque utilisée pour entreposer temporairement des données, notamment entre deux processus ou matériels ne travaillant pas au même rythme. Ainsi, nous allons pouvoir régler cette taille et gérer l'influence de ce dernier par rapport à la taille des différents fichiers. [2]
- **Contenu du Buffer :** Le buffer sera rempli par des bits de 0, des bits de 1 ou des caractères alphanumériques.

Il faut savoir que si l'utilisateur choisit la lecture de fichier, ce dernier sera préalablement créé puis supprimé, tout se fait dynamiquement. À la suite de cette UA, on récupère un « Throughput » en MiB/s. Cela indique à l'utilisateur le temps de lecture ou d'écriture d'un fichier. Ce débit varie en fonction des performances du téléphone. Plus le téléphone est performant et puissant, plus ce débit sera élevé.

La difficulté de cette implémentation concerne la gestion des valeurs de sortie (débit). En effet, aucune information fournie par le constructeur ou par internet ne permet de savoir si les valeurs calculées et donc renvoyées à l'utilisateur sont correctes. Nous avons donc comparé nos valeurs à différentes applications présentes sur le Play Store pour pouvoir valider nos mesures.

Viser Dummy App

Dans cette nouvelle User Action, le but est de simuler l'utilisation d'un téléphone lorsque celui-ci a de nombreuses applications en fonctionnement, c'est-à-dire lorsque la RAM est fortement utilisée. Pour les constructeurs de mobiles, il est important de mesurer les performances de leur téléphone lorsque ce dernier a beaucoup de RAM utilisée. Afin de simuler ceci, il a fallu créer des Dummy Application capables d'utiliser de la RAM. Ces applications ont pour simple but de remplir la RAM à l'aide d'un tableau de bits. La différence entre le temps calculé avant le remplissage et celui calculé après informe l'utilisateur du temps de remplissage de la RAM. Grâce à cela, on se rend compte que plus la RAM est utilisée, plus ce temps de remplissage est grand, plus cela signifie que beaucoup de RAM est utilisée et donc que les performances du téléphone sont diminuées.

J'ai donc créé une User Action permettant à l'utilisateur de saisir différents paramètres :

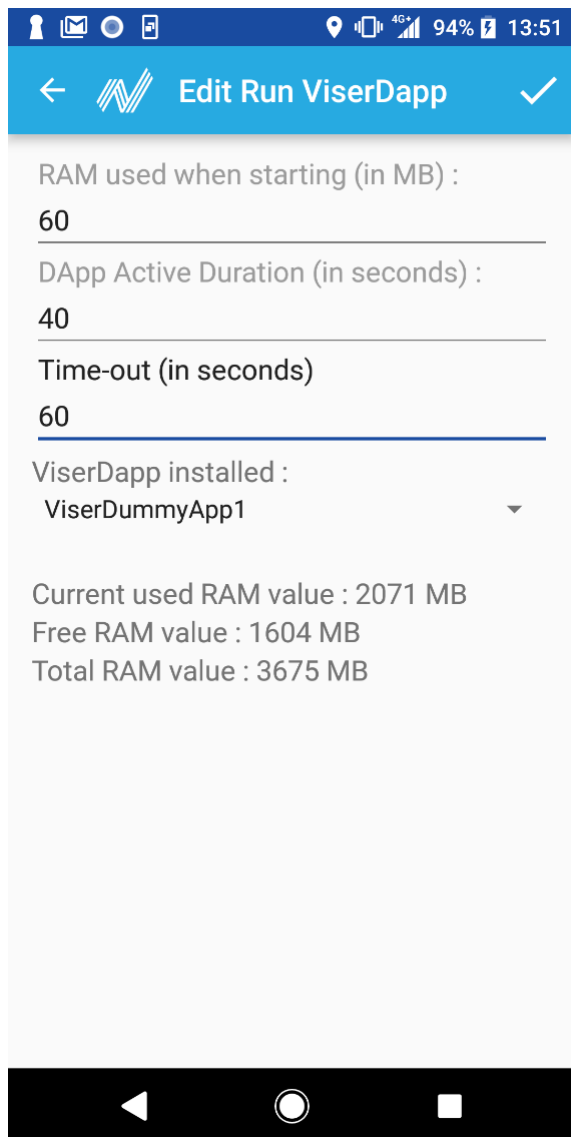


Figure 7 : UA Run ViserDapp

- **RAM used when starting** : Lors du démarrage de la DummyApp, elle va utiliser la valeur indiquée en MB.
- **Dapp Active Duration** : Le temps d'activité de l'application, c'est-à-dire le temps pendant lequel la Dummy App va être active et donc va consommer de la RAM.
- **Time-out** : Le Time-out permet, en cas de disfonctionnement, de terminer la User Action.
- La **Dummy App** que l'on souhaite lancer. Il existe 3 Dummy App avec le même fonctionnement.

On peut voir aussi que dans cette application, l'utilisateur a accès à la RAM utilisée, la RAM libre et la RAM totale du téléphone. Ces données sont indiquées au moment de l'édition et restent fixes. Si l'utilisateur veut voir en temps réel l'évolution de ces données, il est obligé de quitter l'édition

et revenir.

Lors de la création de cette UA, j'ai rencontré plusieurs difficultés. En effet, gérer la RAM est un mécanisme complexe puisque les Dummy App doivent avoir un impact sur la RAM utilisée en temps réel. Or, cette dernière fluctue constamment, il est donc difficile de cibler une taille précise pour l'utilisateur. De plus, comme j'ai pu l'expliquer dans l'étude d'Android Go, lorsque la RAM utilisée est trop grande, Viser se fait « killer » par Android. Ici, le problème peut être similaire si l'utilisateur choisit de trop remplir la RAM.

Correction de Bugs

Après un premier sprint dans l'entreprise, je me suis familiarisé avec l'application. J'ai donc pu commencer à corriger les problèmes remontés par les testeurs de l'application. Les bugs sont de tous types, allant d'un problème visuel jusqu'à un crash complet de l'application. Ils sont divisés en 4 catégories :

- **Show Stopper** : Ce sont des bugs qui nécessitent l'arrêt de toutes les tâches en cours et qui doivent être corrigés impérativement.
- **Critical** : Ces derniers sont importants à corriger mais ne nécessitent pas l'arrêt d'autres tâches.
- **Normal** : Ces bugs doivent être corrigés durant le temps accordé à la correction de bugs.
- **Best effort** : Lorsque les différentes tâches à réaliser sont finies, nous pouvons corriger ces bugs. Ce sont « des corrections bonus ».

Rétrospective

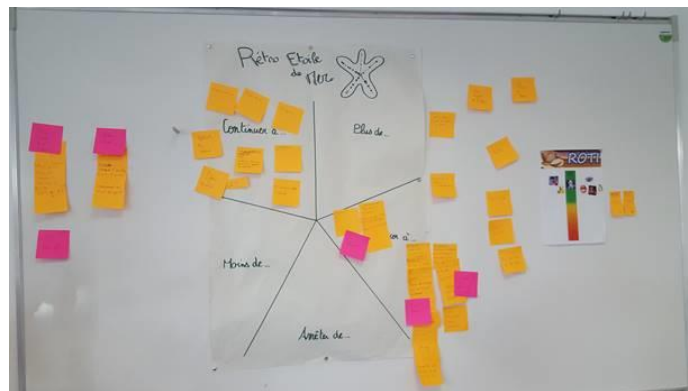


Figure 8 : Rétrospective 2, Etoile de mer

La rétrospective différée de la première puisqu'il ne fallait pas dire ce que nous avions aimé ou pas mais plutôt les activités à continuer à continuer ou à arrêter.

Durant ce sprint, il a été plus difficile de communiquer les uns avec les autres puisque du personnel était parti en vacances ou alors en revenait. Les personnes concernées n'étaient pas toujours présentes pour pouvoir communiquer et expliquer correctement les demandes et donc, cela a pu entraîner une perte de temps lors du développement de nouvelles UA.

Conclusion

Durant l'année scolaire, j'ai eu une formation au développement Android qui m'a très intéressé. Ceci explique mon choix où j'ai effectué mon stage en tant que développeur Android. Mon expérience dans cette entreprise a été très positive et m'a conforté dans l'idée de travailler plus tard dans ce secteur.

Le statut de Smartviser m'intéressait car c'est une startup et j'avais l'envie de découvrir son mode de fonctionnement. Ce dernier a parfaitement répondu à mes attentes. J'ai beaucoup aimé l'ambiance « familiale » et le fait que l'on puisse échanger avec tout le personnel de l'entreprise.

De plus lors de mon arrivée au sein de Smartviser, des changements en interne venaient d'être effectués et j'ai pu en voir les répercussions dans le travail. L'arrivée d'une personne qui a cadré les tâches à effectuer a été très bénéfique pour l'entreprise. Au cours de mon stage j'ai pu côtoyer 2 stagiaires et une personne nouvellement embauchée. Trois autres personnes vont également arriver au sein de Smartviser pour occuper différents postes. Cela montre que la société est en plein développement et il est intéressant pour moi de participer à sa croissance.

Comme mon stage s'est bien passé, une alternance m'a été proposée pour continuer à travailler en tant que développeur Android. C'est avec plaisir que j'ai accepté, je vais pouvoir ainsi continuer à enrichir mes connaissances au sein de Smartviser.

Références

- [1] «Home | Smartviser,» [En ligne]. Available: <http://www.smartviser.com/>.
- [2] «Mémoire tampon,» [En ligne]. Available: https://fr.wikipedia.org/wiki/M%C3%A9moire_tampon.
- [3] «Youtrack,» [En ligne]. Available: <https://en.wikipedia.org/wiki/YouTrack>.
- [4] «Fossil,» [En ligne]. Available: [https://fr.wikipedia.org/wiki/Fossil_\(logiciel\)](https://fr.wikipedia.org/wiki/Fossil_(logiciel)).

Annexes

Annexe 1 : L'équipe

SmartViser's organisation

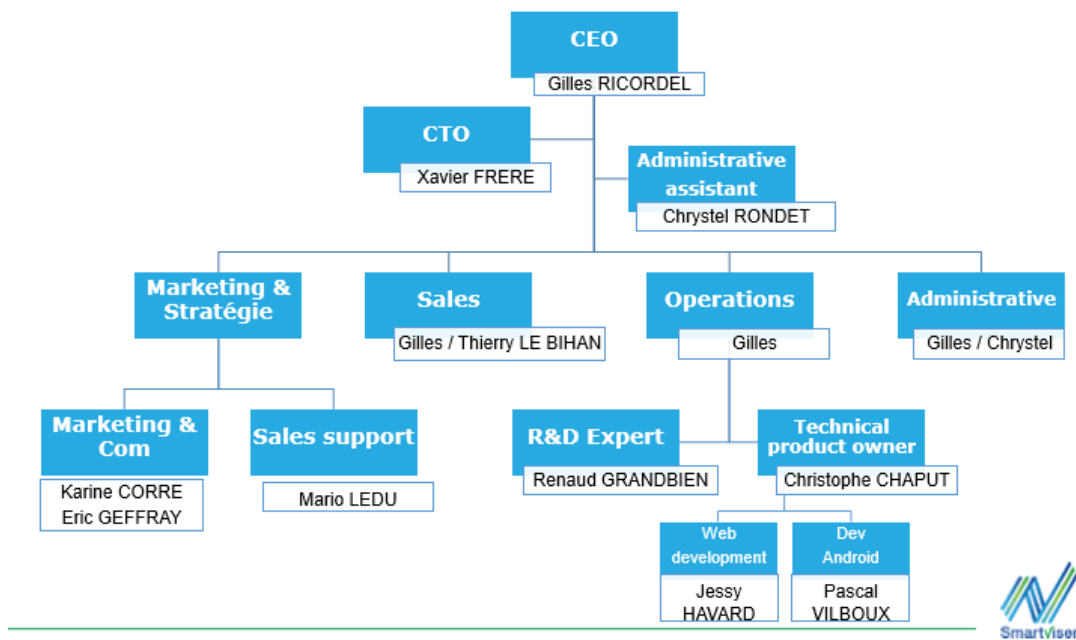


Figure 9 : Organigramme officiel



Figure 10 : Photo de l'équipe en Juillet avec les stagiaires

Annexe 2 : Viser

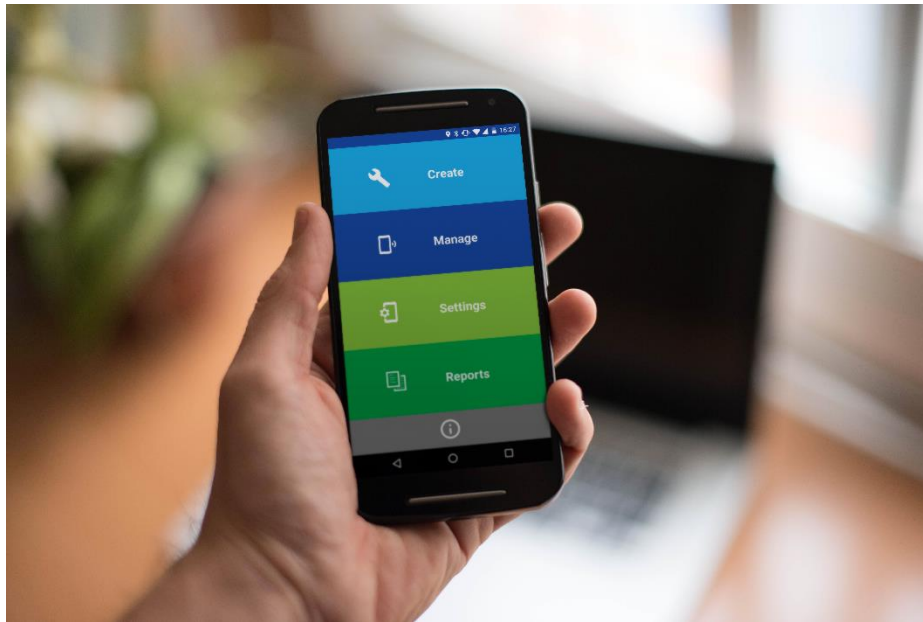


Figure 11 : Application Viser



Figure 12 : Architecture de Viser

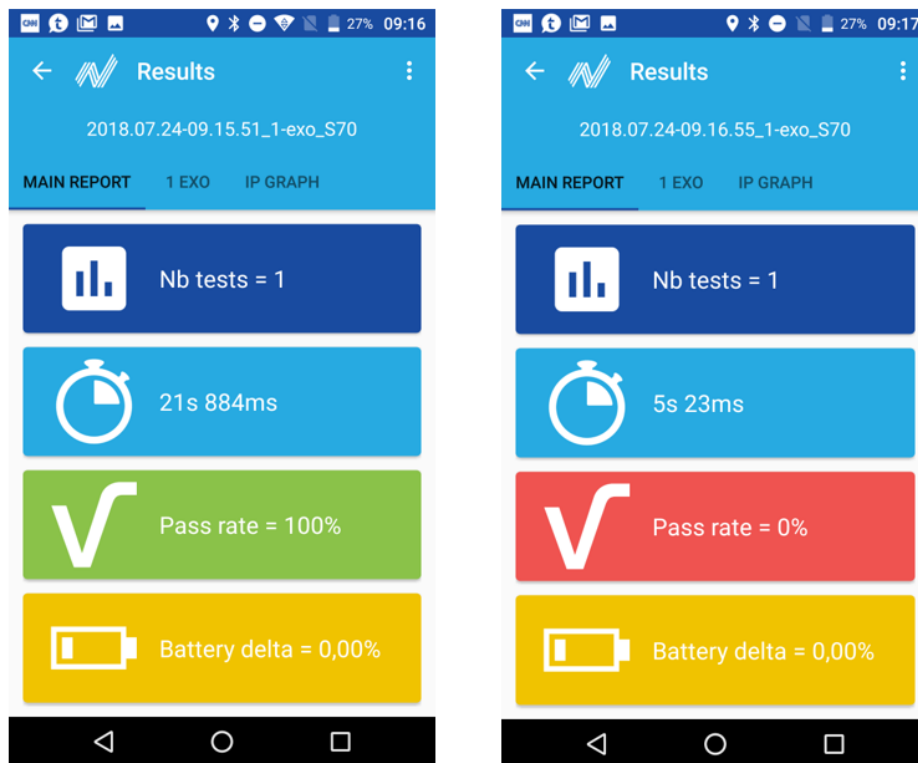


Figure 13 : Résultat des scénarios dans Viser

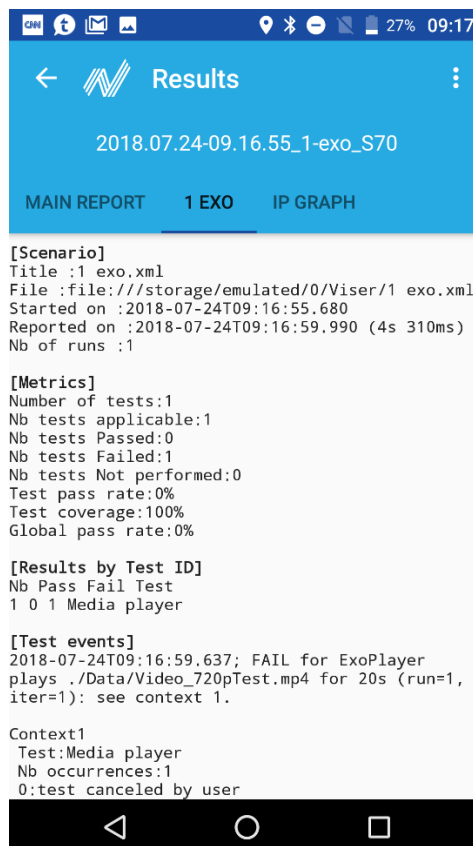


Figure 14 : Résultat des scénarios dans Viser 2

Annexe 3 : ViserWebService (VWS)

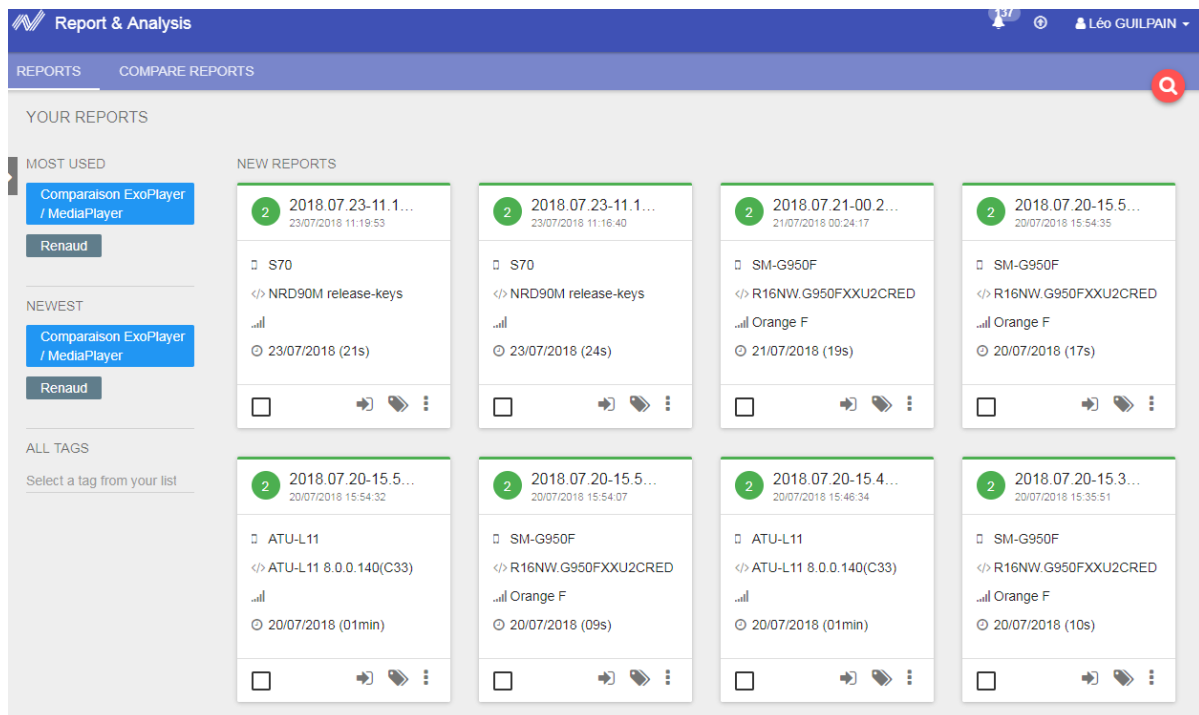


Figure 15 : Résultat des différents scénarios

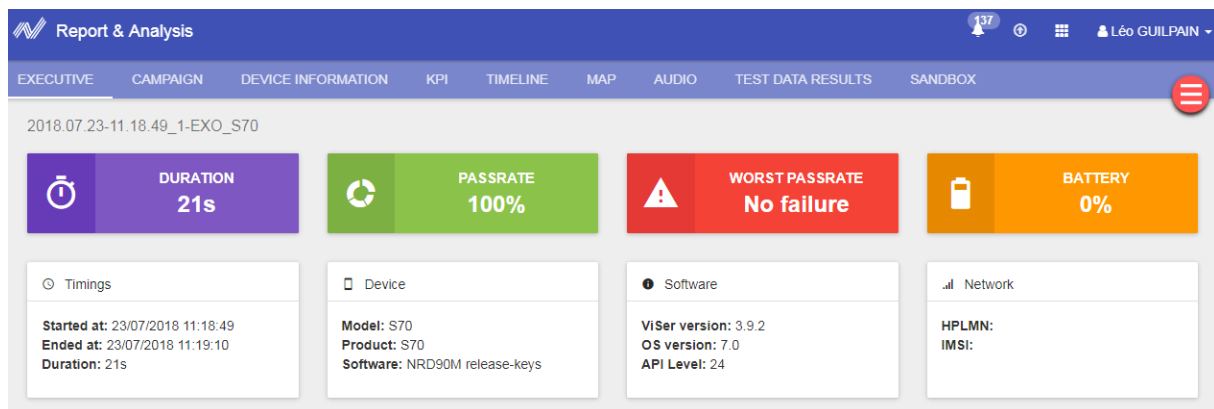


Figure 16 : Exemple d'un résultat

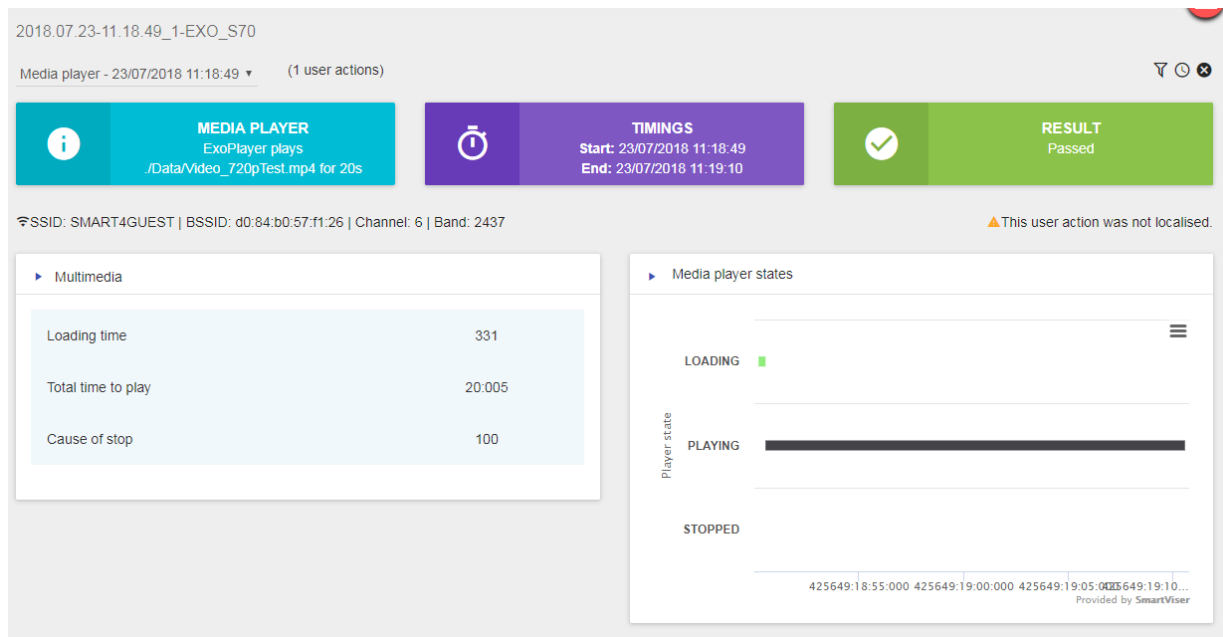


Figure 17 : Exemple d'un résultat détaillé

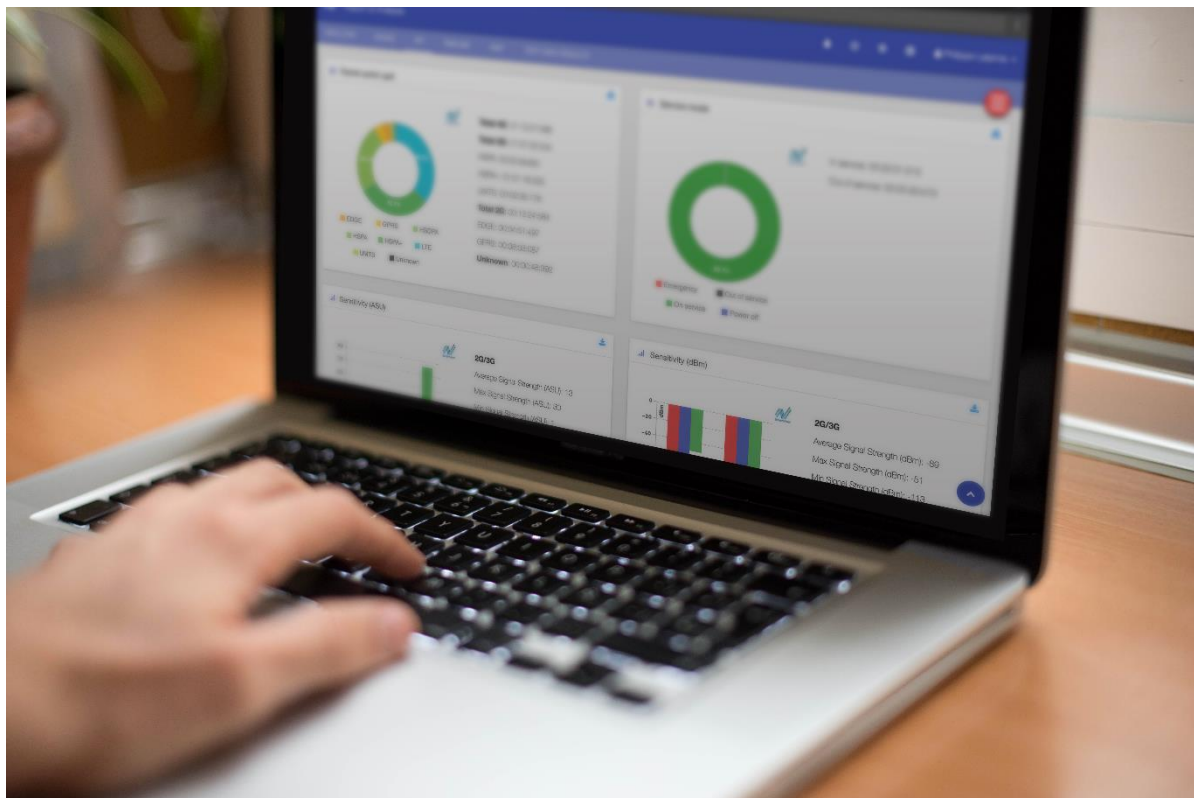


Figure 18 : Viser Web Service

- Issues**
- Dashboards
- Agile Boards
- Reports
- Projects
- Create Issue

Viser (viser)

Enter search request

Found 2258 issues. The results are sorted by: Updated. Save search.

Compact view

?

LG

PROJECTS 3

- Synchro-Bench
- Viser
- VWS

TAGS 1 (+)


- Star
- 8 MORE ▼

SAVED SEARCHES 3

- Assigned to me
- Commented by me
- Reported by me

FILTERS

ID	Type	Status	Description	Assignee	Due Date	Comments
viser-2677	Bug	Resolved	[Android P]Crash when performing a Wifi connect	Pascal Vilboux	sprint 2 - 18W30	3
viser-2758	Bug	Analysed	"Fill the ROM" UA blocks Viser sometimes when the target is too far	Unassigned	Unscheduled	2
viser-2759	Bug	Closed	the scenario with the reboot ua seems to never end	Pascal Vilboux	sprint 1 - 18W27	2
viser-2758	Bug	Major	Sleep UA is shorter than expected	Pascal Vilboux	sprint 1 - 18W27	3
viser-2793	User Story	In Progress	Run Viser on rooted phones	Unassigned	sprint 2 - 18W30	
viser-2436	User Story	Analysed	Dummy application to assess multitasking performance	Unassigned	sprint 2 - 18W30	4
viser-2794	User Story	Analysed	More accurate exit criteria (study)	Pascal Vilboux	sprint 2 - 18W30	
viser-2630	User Story	New	read / write file user action	Léo Guilpain	sprint 2 - 18W30	
viser-2778	Bug	Closed	[Android P] : PROCESS_OUTGOING_CALLS permission (mandatory for MO Call) is not granted => cannot perform MO Call	Pascal Vilboux	sprint 1 - 18W27	5
viser-2774	Bug	Closed	Run installed app failed	Léo Guilpain	sprint 1 - 18W27	
viser-2761	Bug	Closed	while screen rotation, there was a viser has stopped popup	Renaud Grandbien	Jul 20	5



Panda

Timeline

Logged in as [lgl](#)

Home Timeline Files Branches Tags Logout

Older Unhide Max: 200 Check-ins Without Files

200 most recent check-ins

2018-08-21	[ab7c23a7e3] Leaf: PR2874 [Dialer][KPI]Duration in dialing and active state are exactly the same, when performing several calls (user: pvb , tags: trunk)
17:20	
14:21	[c7c7724200] viSer-2878 [GPS]Localisation is never reactivated, after a location off/on for all devices (user: lgl , tags: trunk)
12:43	[e287dd2c] viSer-2828 : Allow devices with API less than 24 to change the language (user: lgl , tags: trunk)
12:36	[118651554d] PR2800 [Dialer]Impossible to exit to Viser, when viser is still the default app (user: pvb , tags: trunk)
11:18	[5f904b5431] viSer-2878 [GPS]Localisation is never reactivated, after a location off/on (devices with an API version lower than 24) (user: lgl , tags: trunk)
2018-08-20	
15:02	[0ea865f1f5] PR2884 [Camera]Impossible to capture more 2 photos (user: pvb , tags: trunk)
13:52	[843743b29a] PR2802 [Proxy] Several system apps crash when proxy is installed + upgrade viser & proxy versionName (user: pvb , tags: trunk)
2018-08-17	
08:34	[34526abb09] viSer-2828 Set the system language doesn't work (user: lgl , tags: trunk)
2018-08-14	
11:36	[20a2fe4d15] Viser 2868 - [Camera]Set a default for "Nbr photos to take" and "Take photo delay" (user: lgl , tags: trunk)
08:48	[7fbb18d638] Disable the User Action Run ViserDapp (user: lgl , tags: trunk)
2018-08-09	
15:53	[01fe3d93dc] Merge from Sprint2 branch : 3.9.2 (user: pvb , tags: trunk , 3.9.2)
14:32	[fff09ec411] Closed-Leaf: Fix issues since last commit on ViserDummyApp (user: lgl , tags: Sprint_2)
13:07	[411c84efa7] viSer-2630 : read / write file user action => new implementation with randomaccessfile class (user: lgl , tags: Sprint_2)
09:59	[01cec3c6e5] Closed-Leaf: ReadWriteFile is fonctionnal on Viser (user: lgl , tags: CR2630)
2018-08-08	
15:58	[ea9c436f01] CR2436 : Dummy application : compute new kpiDuration = reactivity duration (user: pvb , tags: Sprint_2)
2018-08-07	
08:51	[7c22721fb2] ReadWriteFile : New fonctionnal step for the Edit (user: lgl , tags: CR2630)