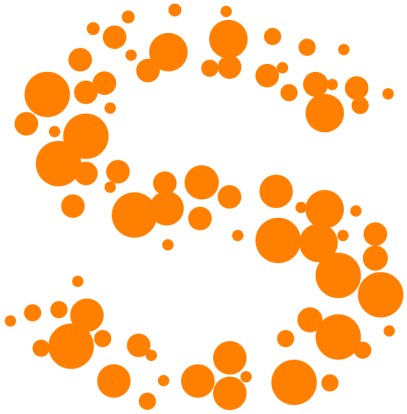
iICT
INSTITUTE FOR INFORMATION AND COMMUNICATION TECHNOLOGIES



saver

Smart Access to Versatile Emergency Resources

Institute for Information and Communication Technologies
University of Applied Sciences
HEIG-VD
Route de Cheseaux 1
CH-1401 Yverdon
Tel.: +41 24 423 22 89
Fax: +41 24 425 00 50
info@iict.ch
www.iict.ch

GEBERT RÜF STIFTUNG
WISSENSCHAFT.BEWEGEN

CHUV

sysmosoft

HESAV
HAUTE ÉCOLE DE SANTÉ VAUD

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Hes·so

L'auteur voudrait ici remercier tous les acteurs de ce projet, aussi bien ceux qui ont contribué à la mise sur pied de la solution technique que ceux qui ont permis, par le financement, par leur soutien administratif et technique et par leur bonne volonté à ce projet de voir le jour et de parvenir au résultat décrit dans le présent document.

Table des matières

CHAPITRE 1 Introduction

1.1 Le problème

Le service des urgences (URG) est équipé d'une plate-forme informatique abritant une application spécifiquement développée pour ce service. Cette application, parfaitement pertinente dans le cadre de son utilisation au service des urgences, présente toutefois un certain nombre de lacunes structurelles (donc inhérentes à la conception même de l'application) qu'il n'est pas aisé de combler sans modifications profondes de l'application et de l'utilisation même de l'outil.

1.1.1 Accès au service

- Les postes d'accès sont dispersés dans le service des urgences, mais ce sont des postes fixes, vers lesquels il faut se déplacer pour renseigner l'application. Ce déplacement est souvent incompatible avec le temps dont disposent les protagonistes au service des urgences; d'un autre côté, un interface Web mobile n'est guère pratique, URG n'allant pas être équipé de WiFi, mais de GSM/GPRS, probablement trop lents pour un accès HTTP, avec des documents codés en HTML.

- L'utilisation de comptes génériques permet l'accès aux ressources de manière simple, rapide, sans avoir à passer à chaque fois une procédure de login fastidieuse et sujette à erreurs. Le service des urgences doit avoir un accès rapide à l'application, incompatible avec une authentification répétée.

- Il est impossible actuellement de laisser aux patients un accès au système informatique, qui leur permettrait de renseigner le système sur certains paramètres (la douleur, par exemple). Leur seul interface de communication est une sonnette, actuellement.

1.1.2 Alarmes, indications asynchrones

- L'infrastructure disponible permet de générer des alarmes, mais du fait des comptes génériques, elles n'atteignent généralement pas la personne concernée. Renoncer aux comptes génériques impliquerait une complexité accrue pour un personnel souvent très volatil.

1.1.3 Sécurité

- Les applications ne sont que médiocrement sécurisées, car faites pour résider dans un intranet par lui-même sécurisé. Il n'est donc pas souhaitable d'offrir un accès semi-public, qui permettrait par exemple à des ambulanciers de renseigner certains champs par avance, pendant le trajet en ambulance vers le CHUV.

- De nombreux paramètres auxiliaires (par exemple des données pouvant par la suite servir à renseigner d'autres applications, comme Tarmed) ne sont actuellement qu'imparfaitement renseignés, en raison de la difficulté qu'il y a d'introduire des données sur des postes fixes dans le stress du service.

- Certains appareils de mesure pourraient éventuellement stocker directement leurs données de mesure dans le système d'informations du CHUV : mais les applications concernées n'ont pas été conçues dans cette optique.

1.2 Terminaux mobiles

L'introduction de terminaux mobiles de type GSM permet de pallier partiellement aux limitations constatées; ces terminaux, du genre PDA, tablette ou smartphone, peuvent se transporter dans la poche de tout un chacun, permettent l'introduction rapide, à une main, de paramètres importants (douleur, pouls, fièvre, pression, indicateurs de gravité, etc...) et peuvent grâce à leur accès à la téléphonie mobile GSM aussi être utilisé en dehors de l'enceinte de l'hôpital (ambulanciers, secouristes). Par ailleurs, l'utilisation de GSM est aussi en accord avec la politique actuelle du CHUV qui a remplacé le service de recherche de personnes par des terminaux mobiles GSM. Si les utilisateurs ont été équipés de terminaux portables très basiques, il n'est pas interdit à un service particulier de s'équiper de terminaux plus sophistiqués pour leurs besoins spécifiques.

L'utilisation de terminaux mobiles est en revanche liée à un certain nombre de problèmes potentiels qu'il convient de bien évaluer avant toute prise de décision :

1.2.1 Mobilité

- Les terminaux sont justement ... mobiles (!) ce qui implique qu'ils puissent voyager avec leurs possesseurs. Ces derniers peuvent sortir du périmètre du CHUV pour aller fumer une cigarette[1] à l'extérieur ou boire un verre aux Falaises; ou plus simplement passer à la PMU ou dans les bâtiments du Bugnon. Dans le cas d'une utilisation de ces terminaux pour un accès à l'Intranet, la mobilité étendue implique une ouverture partielle de l' Intranet vers l'extérieur, ce qui n'est pas souhaitable sans autres.

1 *Nuit gravement à la santé ;-)*

- La petite taille et la mobilité vont de pair; mais ces deux caractéristiques rendent l'objet aussi très facile à dérober, ou à perdre. Ce qui implique aussi que cet objet a quelque chance d'être utilisé par une personne non autorisée.

1.2.2 Sécurité

- Les concentrés de technologie que sont les smartphones et PDA sont aussi les objets informatiques les moins protégés que l'on connaisse actuellement. Ce qui signifie que les données que l'on stocke sur ces objets sont très vulnérables. De plus, les connaissances sur les virus et les attaques visant ces terminaux mobiles sont très lacunaires.

- Les terminaux mobiles sont difficiles à gérer. Idéalement, on voudrait distribuer aux utilisateurs les dernières versions des applications dont ils ont besoin, et celles-ci uniquement. En pratique, cela s'avère souvent difficile: il faut gérer aussi bien les utilisateurs que le rôle qu'ils remplissent au sein de l'organisation, de manière à leur distribuer la bonne version du logiciel adéquat.

1.2.3 Gestion

- La mise à jour de ce genre de terminaux est complexe. Il n'est pas pensable de rappeler ces terminaux pour en effectuer la mise à jour, un par un. Il faut pouvoir effectuer une mise à jour à distance, ce qui implique des choix technologiques préalables.

- La technologie est très réactive dans ce domaine. La durée de vie de ce genre d'appareils est relativement faible; il est très difficile de se baser sur une solution purement logicielle, car le matériel évolue très rapidement. Il est plus prudent de limiter les choix de matériels possibles pour obtenir une meilleure stabilité de la solution à moyen terme.

Ces problèmes ont été adressés par certains constructeurs; le plus connu est sans doute RIM (Research In Motion) avec le produit Blackberry. Mais les spécificités du monde biomédical n'ont pas fait l'objet en l'occurrence de préoccupations particulières. D'autre part, ces appareils sont souvent coûteux, et ne sont guère adaptés au développement d'applications destinées aux tâches médicales.

1.3 Considérations plus génériques

Si ce projet est par hypothèse plus orienté vers le service des urgences, on peut néanmoins se poser la question des possibilités plus génériques de l'introduction d'outils de mobilité dans un périmètre de soins. En effet, l'informatique a été traditionnellement orientée vers un modèle où l'utilisateur vient quérir l'information dont il a besoin. On lui installe un terminal de visualisation et d'interactions, et il est censé se déplacer vers ce point pour obtenir les renseignements désirés.

Par opposition, la médecine est une discipline beaucoup plus ancienne, où le soignant se déplace vers le patient pour exercer son métier. Soit que le patient est incapable de se déplacer vers le soignant, soit que le soignant a besoin d'observer et de diagnostiquer l'état du patient dans un cadre donné, il n'y a guère dans les professions de la santé de possibilités d'amener le problème vers l'endroit où traditionnellement l'informatique fournit l'information. De fait,

l'introduction de l'informatique dans le domaine des soins, actuellement, se heurte à cette contradiction qui fait que l'information n'est pratiquement jamais convoyée au bon endroit. Le bon endroit, c'est bien sûr là où le soignant exerce sa profession, c'est-à-dire auprès du malade.

On peut donc dire sans crainte de se tromper excessivement que l'introduction de la mobilité est une caractéristique qui dépasse le simple cadre du domaine des urgences. SAVER est de ce fait un projet exploratoire, certes défini dans le contexte du service des urgences du CHUV, mais qui acquiert du coup une portée beaucoup plus grande et plus fondamentale.

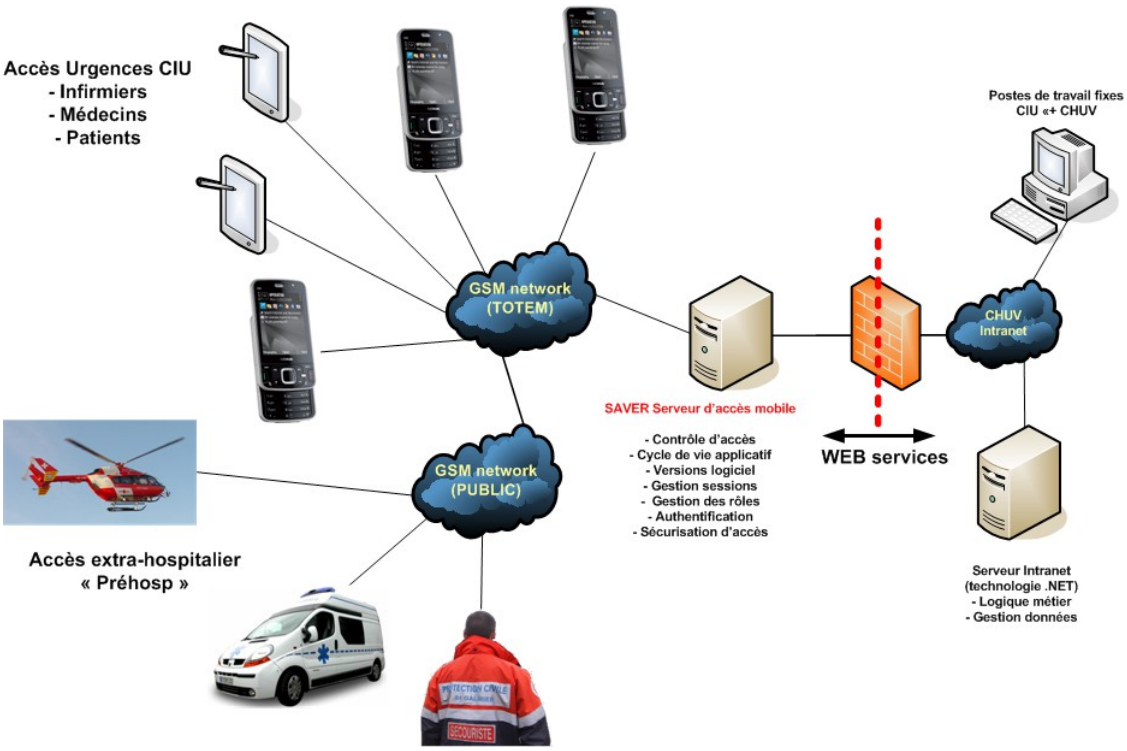
Le problème de la mobilité en milieu hospitalier est à considérer comme un problème tout à fait générique, et SAVER peut aussi s'appliquer à d'autres secteurs du domaine hospitalier, comme nous espérons le démontrer dans le présent exposé.

CHAPITRE 2 Architecture proposée

L'institut IICT a développé une infrastructure de gestion de flux de données dans le cadre du projet IMINET (http://useraware.iict.ch/fr/sante/iminet/index.html) qui permettrait, grâce aux fonctionnalités présentes, un interfaçage relativement aisé à la plate-forme existante au service des urgences, tout en amenant une gestion des utilisateurs et une sécurisation adaptée aux données manipulées et aux diverses tâches des utilisateurs. Ce même institut a également développé dans le cadre du projet OSMOSYS (http://useraware.iict.ch/fr/mobilite/osmosys/index.html) une interface mobile sécurisée pour PDA ou smartphones qui gère les utilisateurs et les terminaux mobiles, et permet d'introduire des services mobiles de manière largement transparente aux applications existantes. La mise en relation des deux plate-formes (OSMOSYS et IMINET) sur un serveur unique (que nous appelerons « serveur SAVER » dans la suite de cet exposé) ne constitue pas un problème, les deux se fondant sur des technologies similaires, et OSMOSYS ayant à l'origine été conçu pour fonctionner avec IMINET.

Par ailleurs, l'infrastructure OSMOSYS est actuellement entre les mains d'une société indépendante, spécialisée dans la sécurité mobile, ce qui permet de garantir le suivi après un développement conjoint par des milieux académiques.

La connexion à l'infrastructure informatique du CHUV repose idéalement sur l'utilisation de services WEB, relativement faciles à implémenter sur une architecture .NET aussi bien que sur une architecture Java J2EE. La définition des interfaces communes se fait au moyen du langage WSDL (Web Services Description Languages), et l'interconnexion des plate-formes est réalisée grâce à un protocole SOAP (Simple Object Access Protocol) ou similaire, lui-même basé sur HTTP. Cette architecture permet aussi de disposer d'un accès au service en « zone démilitarisée », et ainsi d'ouvrir de manière sécurisée l'application du service des urgences vers l'extérieur, vers l'accès préhospitalier (ambulances, secouristes ...).



Par ailleurs, le protocole OTA (Over The Air) de OSMOSYS est très léger, bien que sécurisé; ceci lui permet d'être parfaitement efficace aussi sur des connexions basse vitesse, si bien que les utilisateurs dans le cycle préhospitalier disposeront du même confort d'utilisation (mais pas forcément des mêmes applications, bien évidemment) que le personnel du CIU.

Il serait bien entendu possible d'intégrer le serveur d'accès mobile à l'un des serveurs abritant les applications hospitalières. Techniquement, le problème ne présente pas de difficultés particulières. Pour des raisons de sécurité et d'exploitation, nous aimerions pourtant déconseiller cette démarche pour les raisons suivantes :

- Tant IMINET que OSMOSYS constituent des systèmes hautement sécurisés, prévus pour un accès externe, ce qui n'est pas forcément le cas des applications médicales en exploitation au CHUV, conçues pour une utilisation dans l'intranet du CHUV, et non ouvertes sur un accès semi-public. Ouvrir une porte vers l'extérieur peut se faire sans risques particuliers pour IMINET ou OSMOSYS, mais pourrait représenter un investissement non négligeable pour la mise à jour des serveurs existant. La connexion avec deux serveurs séparés permet d'offrir un accès vers l'extérieur sécurisé sans nécessiter une révision des autres infrastructures logicielles de l'hôpital.

- Du point de vue du développement, il est beaucoup plus simple de séparer physiquement les deux produits. Par ailleurs, les contraintes de montée en charge ne seraient pas les mêmes pour SAVER que pour les autres applications hospitalières; SAVER serait en effet soumis directement à des sollicitations en provenance de terminaux mobiles,

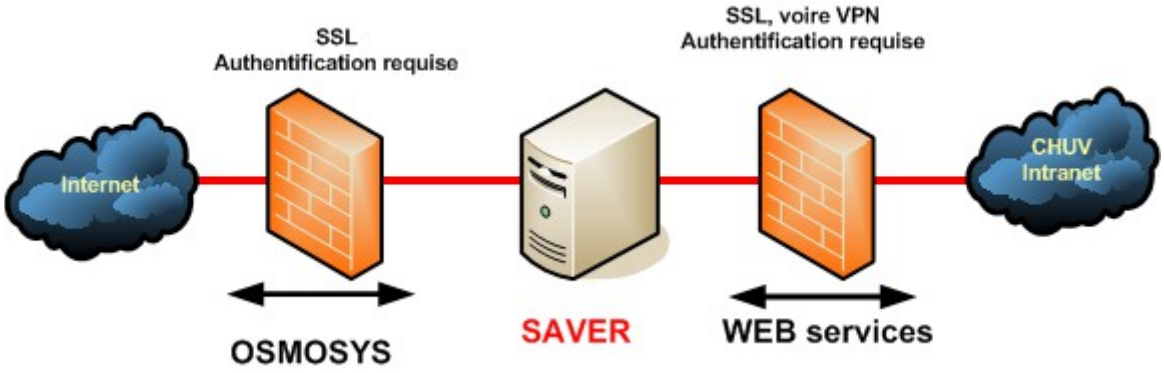
donc potentiellement moins maîtrisables que celles provenant de terminaux fixes.

- Les applications existantes constituent un système stable, éprouvé, en utilisation au CIU. Modifier ces applications pour leur ajouter des fonctionnalités nouvelles constitue une modification majeure, risquant de déstabiliser inutilement le produit. Une démarche utilisant des services WEB et un serveur séparé constitue une modification mineure, sans risques notables pour l'existant.

- Du point de vue coût, il est certainement moins coûteux d'adjoindre un serveur supplémentaire plutôt que d'entreprendre de coûteuses modifications de paquets logiciels existant.

- Au niveau de l'exploitation, la présence d'un serveur séparé constitue un inconvénient. Cet inconvénient peut en revanche se trouver diminué grâce au fait que ce nouveau serveur se trouve dans la zone démilitarisée : il peut assez simplement être délocalisé, confié à la gestion par un entreprise externe de confiance, le cas échéant. Ceci favorise aussi le développement, en ce sens que lors de la phase de développement, le serveur nécessaire peut résider dans l'enceinte de l'équipe de développement.

Dans cette configuration, le schéma de sécurité de SAVER relativement au réseau intranet de l'hôpital et au réseau Internet global est le suivant :



Les accès fournis sous forme de services WEB par l'infrastructure informatique de l'hôpital sont en principe inaccessibles depuis l'extérieur, sauf pour le serveur SAVER, qui utilise un canal protégé TLS (SSL) pour se connecter sur un protocole d'application HTTPS. On pourrait aussi imaginer, si cela semble préférable, relier les systèmes par un tunnel VPN, cela ne change pas grand-chose à la solution.

SAVER est lui-même protégé par un pare-feu (firewall) qui ne laisse passer que les services requis pour l'application SAVER; ces services sont implémentés sur la base de OSMOSYS qui utilise également un canal SSL pour sécuriser la liaison. Tous les accès à SAVER sont authentifiés: seules des applications dûment identifiées comme telles, pilotées par des utilisateurs autorisés, résidant sur des terminaux enregistrés peuvent accéder à SAVER. L'infrastructure offre de surcroît un chiffrement applicatif garantissant une authentification mutuelle entre le serveur et les terminaux mobiles.

Ce schéma laisse beaucoup d'options ouvertes pour la gestion du matériel et du logiciel.

SAVER peut bien sûr être localisé dans l'enceinte du CHUV, et géré par le SI à l'instar d'autres applications, par l'intermédiaire d'une interface Web classique; mais il est également possible d'externaliser le service, complètement ou partiellement. Le choix retenu dépendra essentiellement de considérations politiques vis-à-vis des backups et de la gestion des utilisateurs et des postes mobiles. A noter que même dans une solution externalisée, cette gestion peut (et devrait) se faire par le SI de l'hôpital par le biais d'un accès protégé HTTPS.

Cette configuration semble la plus flexible, et la mieux à même de protéger l'infrastructure informatique de l'hôpital contre des accès malveillants tout en permettant un accès extra-hospitalier par les personnes qui ont besoin de cet accès (comme c'est en principe le cas pour des solutions de grande mobilité). De plus, cette solution se caractérise par une très grande indépendance logicielle et matérielle vis-à-vis des infrastructures existantes et à venir de l'établissement hospitalier, ce qui nous paraît constituer un gage de pérennité vis-à-vis des évolutions prévisibles de part et d'autre.

CHAPITRE 3 Généricité de SAVER

La solution SAVER étant conçue autour d'une application native résidant sur le client, on peut craindre une dépendance assez forte vis-à-vis d'un matériel ayant une durée de vie relativement réduite, et sujet à de forts effets de mode. Plusieurs réponses peuvent être apportées à cette objection par ailleurs tout à fait justifiée.

- Si les objectifs poursuivis sont aussi bien une sécurité élevée qu'un confort d'utilisation maximal, il paraît difficile de se passer d'un client « intelligent », capable d'assurer la sécurité sans demander un mot de passe à chaque intervention de l'utilisateur, et à même d'acquérir des données de manière largement autonome. Un framework purement web2, à supposer qu'il soit exécutable sans problèmes sur un navigateur de PDA, n'offre pas de possibilités comparables à celles d'un client natif.

- Un portage de l'application sur un autre OS est envisageable, puisque l'infrastructure n'est qu'une application : la réécrire sur un autre environnement ne pose pas de problèmes insolubles. En revanche, il est vrai que la maintenance s'en trouverait quelque peu compliquée. Il est dans la nature du serveur OSMOSYS de gérer de manière homogène un parc de dispositifs mobile hétérogène, ce qui tend à relativiser cet inconvénient, bien que le développement double reste un inconvénient à ne pas négliger.

- Le serveur SAVER peut parfaitement abriter des applications WEB / WEB2 traditionnelles : si pour une application donnée il s'avérait nécessaire d'envisager un déploiement dépassant le cadre d'un service spécialisé (comme le CIU) pour toucher éventuellement l'ensemble du CHUV, voire des unités externes, sur des PDA non spécialisés, on pourrait parfaitement utiliser l'infrastructure serveur de SAVER pour déployer des applications WEB sur PDA, exécutables sur un navigateur classique pour smartphones. Il faut toutefois être conscient du fait que ce type d'applications ne saurait

prétendre au même niveau de sécurité que les applications natives, ni à un confort d'utilisation comparable. En particulier, ces applications devront correspondre au modèle WEB, (URL / question / réponse) ou éventuellement messagerie (e-mail / réponse) qui est notoirement inadapté aux possibilités d'entrées-sorties des terminaux de petites dimensions, mais qui présente en revanche l'avantage d'être applicable sur tous les PDA disposant d'un navigateur et d'un client de messagerie.

- Le concept à la base de SAVER est le même que celui qui a présidé à la conception de OSMOSYS : il s'agit de terminaux d'acquisition et de présentation **dédiés** à une certaine catégorie d'applications et d'utilisateurs évoluant dans un environnement mobile sécurisé (ici, le CIU). Le serveur offre un support de sécurisation, d'authentification et de maintenance à ces terminaux et applications dédiées. Il s'agit d'un serveur JEE (Java Enterprise Edition), il respecte en ce sens les conditions d'utilisation de tout serveur de ce type (tomcat, jBoss, Glassfish), et de plus assure le respect des spécifications OSMOSYS pour les terminaux adéquats. Pour des terminaux ne correspondant pas aux prérequis OSMOSYS, le serveur se comporte comme un serveur JEE sécurisé normal. En particulier, il supporte les applications web ou web2 comme tout serveur de ce type, avec tous les mécanismes que l'on peut désirer introduire à ce niveau.

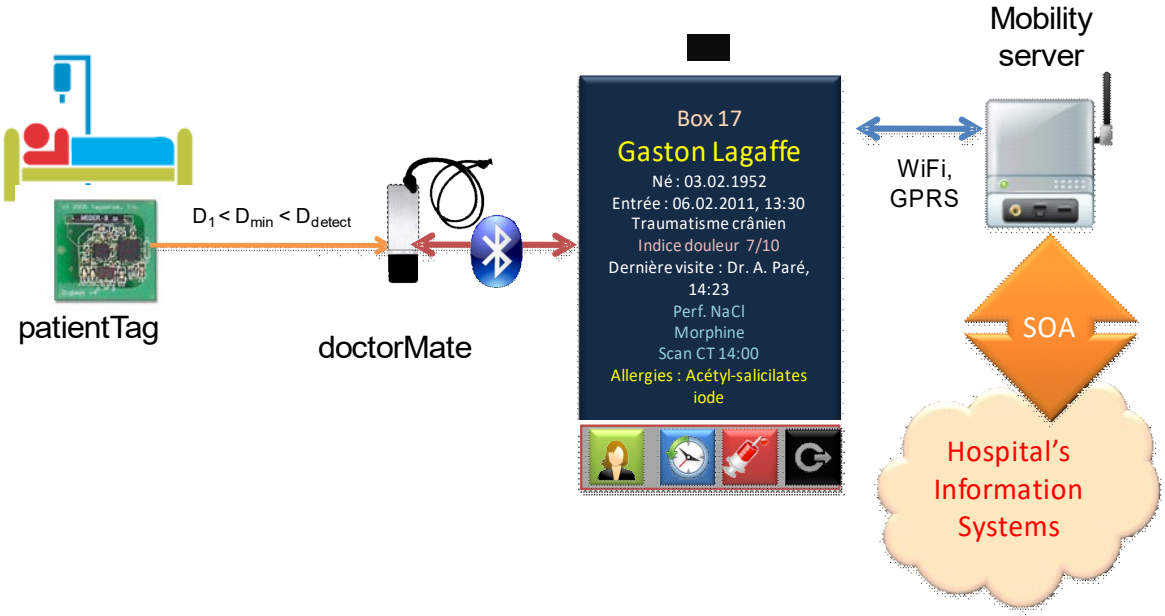
- L'utilisation d'applications WEB sur le serveur abritant la fonctionnalité de SAVER a été d'emblée prévue : la maintenance du service de SAVER est basé sur l'utilisation d'applications WEB. Les questionnaires structurés de IMINET constituent aussi une application WEB qu'il devrait être particulièrement aisé à transporter sur des terminaux mobiles de n'importe quel type, à commencer par Android.

SAVER , tel que vu de l'utilisateur, se compose de trois composants distincts :

- Un smartphone, ou une tablette, qui permet au personnel soignant de consulter des données, et aussi d'utiliser le smartphone tout à fait normalement. Selon la politique d'investissement et d'utilisation mise en œuvre, on pourrait même imaginer installer la fonctionnalité SAVER -sans restrictions de sécurité ou de confidentialité- sur des téléphones privés de divers modèles à choisir parmi un ensemble de constructeurs supportés (par exemple, modèles Android à partir de 2.3, iPhone OS4, WM7).

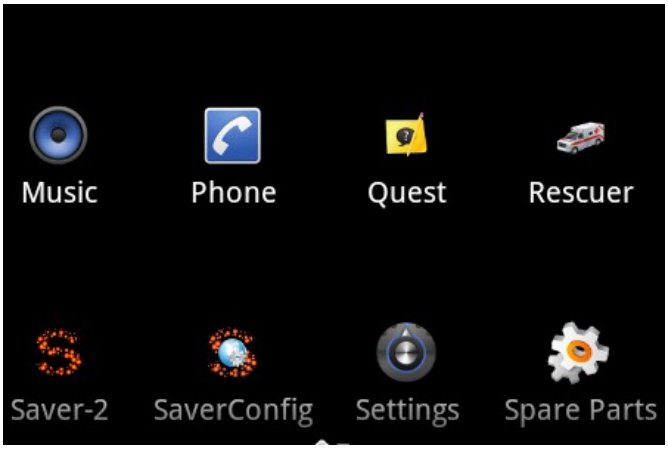
- Un dispositif annexe qui va identifier le personnel soignant (tag radio-fréquences actif, appelé DoctorMate), et qui permet de communiquer avec les balises actives dont seront équipés les patients.

- Des balises actives (PatientTag) qui sont remises au patient (bracelet, ou association avec le lit, etc...) qui permettent une identification implicite du patient.


Mobility server
Box 17
Gaston Lagaffe
Né : 03.02.1952
Entrée : 06.02.2011, 13:30
Traumatisme crânien
Indice douleur 7/10
Dernière visite : Dr. A. Paré, 14:23
Perf. NaCl
Morphine
Scan CT 14:00
Allergies : Acétyl-salicilates iode
WiFi, GPRS
SOA
Hospital's Information Systems
$D_1 < D_{min} < D_{detect}$
patientTag
doctorMate

CHAPITRE 4 Utilisation de SAVER

En pratique, une personne soignante (prenons l'exemple de l'infirmière-chef Nicole Martin qui prend son service) démarre Saver en tapotant sur l'icône associée de son smartphone (ou de celui qui lui est fourni par son employeur) à son arrivée sur son lieu de travail:



SAVER va chercher à s'assurer du nom et de la fonction de la personne, et va lui présenter un dialogue de login/mot de passe tout à fait conventionnel :

Le programme se lance sur l'activité Saver. Celle-ci se charge d'initialiser l'application et d'afficher l'écran de login (qui retourne les valeurs à Saver). L'application défini l'identifiant et le mot de passe du Caregiver et démarre un enrôlement ou un login dans une tâche parallèle (l'opération prend un certain temps). Si la procédure n'engendre pas d'exception, l'utilisateur est identifié et son nom est ajouté à la liste des utilisateurs Saver. Le programme essaie ensuite de se connecter au compagnon Bluetooth puis envoie une demande d'informations sur le médecin au serveur. Le médecin est ensuite initialisé grâce à ces informations. Le programme va ensuite récupérer les informations initiales de l'application (première synchronisation) puis afficher le dashboard.

4.1 Login

A l'affichage de la fenêtre de login, le programme cherche combien d'utilisateur il connait déjà. La liste des utilisateurs (sans leur mot de passe évidemment) est inscrite dans un fichier en clair dans la mémoire interne d'Android. S'il ne connait pas d'utilisateurs, il affiche la fenêtre d'enrôlement. S'il en connait au moins un, il affiche la fenêtre de login avec une liste déroulante pour choisir l'utilisateur. L'utilisateur a accès à un bouton lui permettant d'enrôler une nouvelle personne. Pour ajouter une nouvelle personne, il est impératif d'avoir son login (sans le @integration.sense.com), son mot de passe et son PIN d'enrôlement. Un utilisateur qui

s'identifie avec un mauvais mot de passe est supprimé de la liste et doit s'enrôler à nouveau. Les trois champs de l'enrôlement sont l'identifiant (sans le @integration.sense.com, fixé dans SaverConfiguration.java). Le deuxième est le mot de passe et le dernièr le PIN d'enrôlement fournit par SENSE. Le mot de passe de l'utilisateur doctor1 et doctor2 est le même et il est défini comme valeur par défaut du champ mot de passe de l'enrôlement pour faciliter le développement.

Une authentification réussie mène à un dialogue de confirmation, qui indique aussi si l'environnement de l'utilisateur a pu être correctement initialisé.

4.2 Dashboard

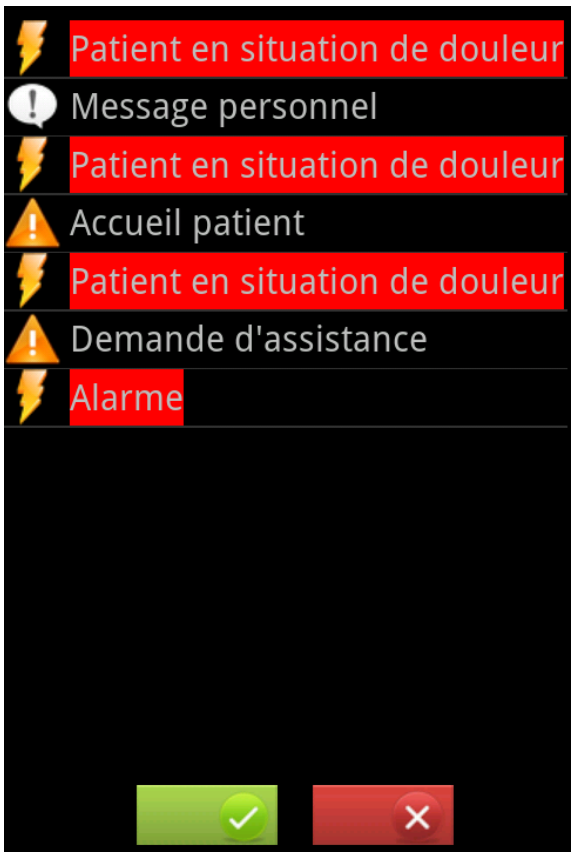
L'utilisateur parvient dés lors dans le *dashboard*, le centre de contrôle de son application SAVER, où se trouvent regroupées les diverses commandes qu'il peut utiliser pour interagir avec le système. Dans l'exemple ci-dessous, c'est l'utilisateur Nicole Martin qui s'est identifiée correctement au système, son identité est rappelée dans la bannière au-dessus du dashboard.

Les divers boutons correspondent à des actions spécifiques qui sont énumérées ci-après :

Annonces. Les annonces représentent un "whiteboard", des annonces d'intérêt général qui sont à disposition de tous. Le contenu de ces annonces est dépendent de l'institution. Il peut s'agir d'annonces spécifiques au service des urgences, d'annonces génériques à tout l'hôpital, voire des deux.

Les annonces peuvent être importées d'un système pré-existant, ou générées à partir d'un dialogue adéquat sur le serveur (JSP, par exemple). Il est également possible de réutiliser l'infrastructure des alarmes (voire point suivant)

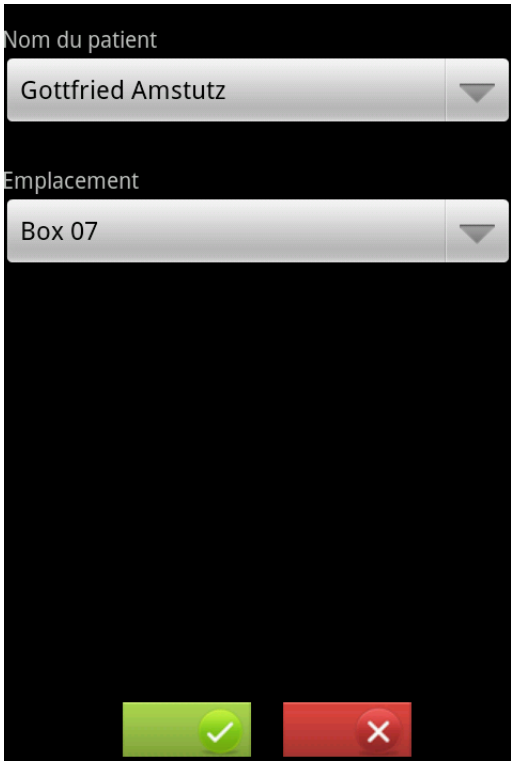
Messages et alarmes. Les messages et alarmes constituent des annonces asynchrones spécifiques au service des urgences et à son fonctionnement thérapeutique. Les annonces peuvent être générales, ou ne s'adresser qu'à un nombre restreint de personnes, voire à un individu



4.3 Sélection d'un patient

La sélection du patient peut se faire de deux manières pas forcément exclusives. Une sélection implicite, qui choisit d'ouvrir le dossier du patient qui se trouve à proximité du soignant (cas d'une sélection normale), ou un patient sélectionné librement parmi les patients qui se trouvent dans le périmètre du service des urgences.

Dans le deuxième cas, le soignant va se trouver confronté à une liste double :

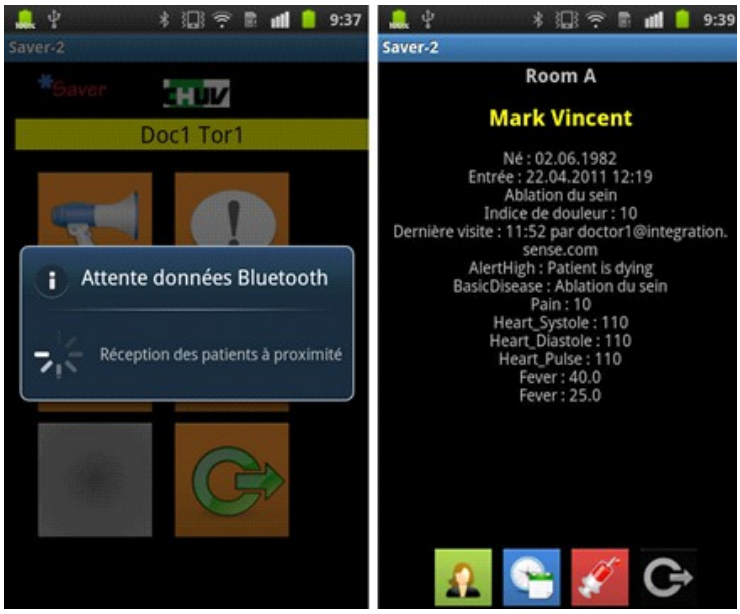


La première liste énumère les patients, la seconde énumère leurs localisations. Il est donc possible de sélectionner un patient aussi bien par sa localisation (box L) ou par son nom.

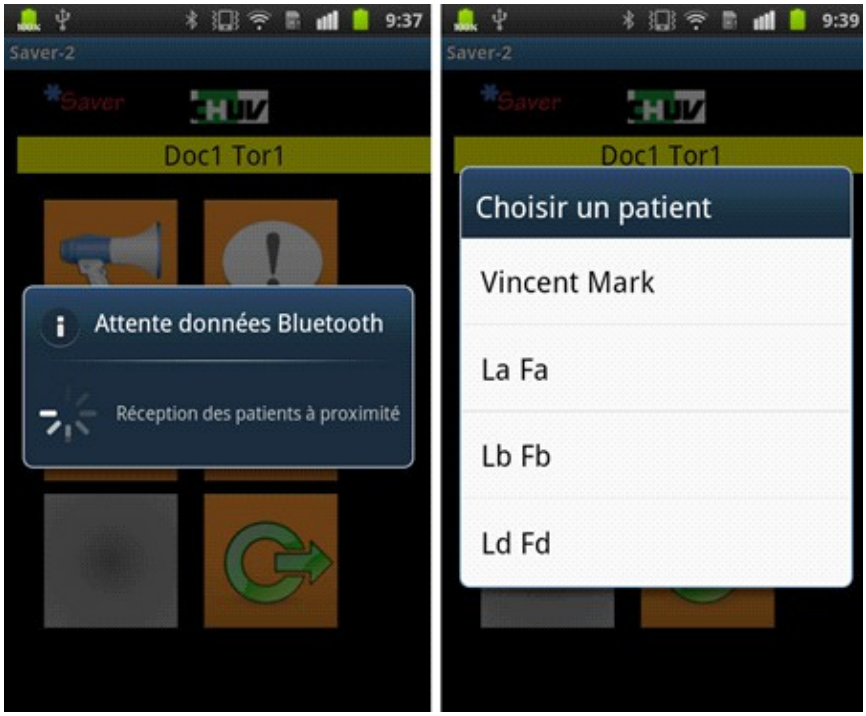
Dans le premier cas, il n'y a normalement qu'un seul patient concerné, celui qui se trouve à proximité. Il se peut toutefois que le système d'identification ne parvienne pas à déterminer l'identité du patient de manière univoque ; dans ce cas, il va identifier tous les patients à proximité, et proposer une liste restreinte à ces derniers, au fonctionnement identique au cas de figure numéro 2.

4.4

Si un seul patient est trouvé, l'application va directement ouvrir la fenêtre de détails de celui-ci.



Si plusieurs patients sont détectés, une liste des patients à proximité est affichée. Le clic sur l'un de ces patients va ouvrir sa fenêtre de détail.

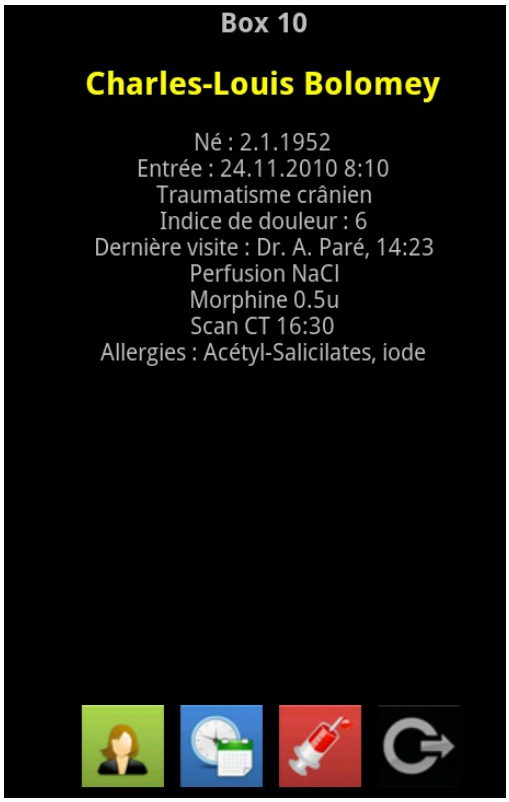


Exit. Ce bouton permet de quitter l'application SAVER.

Il existe un bouton grisé, non assigné pour l'instant, qui n'a pas d'effet en l'état, et qui a pour seule justification la symétrie du dashboard.

4.5 Consultation et introduction de données patient

Après avoir sélectionné un patient (implicitement ou explicitement), le soignant tombe dans un dialogue secondaire qui est spécifique à ce patient.



La page introductive rappelle quelques informations essentielles du patient, et résume éventuellement les données immédiates (par exemple un scan CT à subir, les allergies, etc...). Dans le dialogue patient, quatre boutons se retrouvent systématiquement dans les écrans de sélection, qui permettent d'accéder à quatre domaines d'accès aux informations concernant le patient.

Cette icône correspond à la page principale d'informations patients (voir ci-dessus).

Permet l'affichage de l'historique des soins. L'activation de ce bouton mène à l'affichage de la liste ordonnée chronologiquement des soins et des interventions effectuées en relation avec ce patient.

Une description plus précise de l'intervention peut être affichée dans une boîte de dialogue lors de l'activation d'un message.

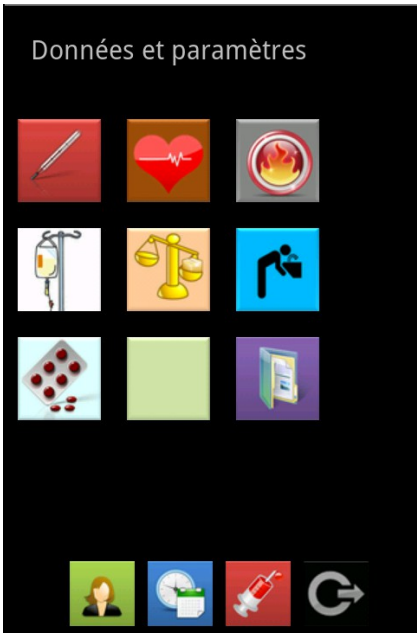
La page d'introduction de données est accessible via ce bouton ; son activation conduit à une fenêtre permettant la sélection de diverses actions. Pour l'instant, ces actions sont définies de manièpre très provisoire, et uniquement à titre d'illustration ; certains dialogues ne sont pas implémentés, ou implémentés de manière incomplète ou imprécise.

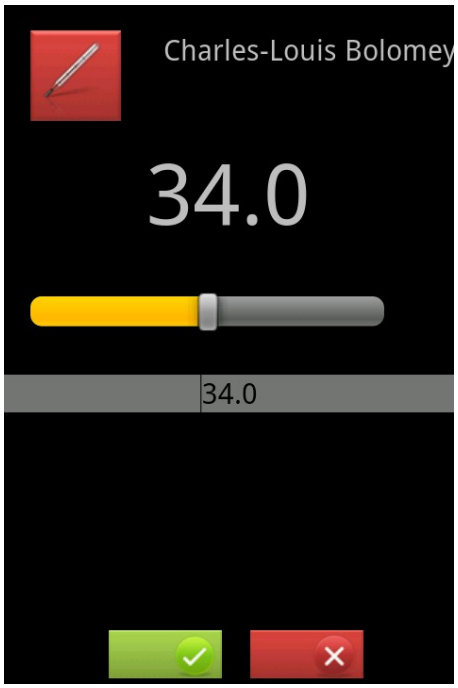
D'une manière générale, le fonctionnement de SAVER est basé sur l'asynchronisme. Un soignant qui introduit une donnée dans le système ne doit de ce fait pas s'attendre à ce que cette information soit immédiatement prise en compte par le système : la mise à jour peut prendre un certain temps. Ainsi, deux soignants (appelons-les A et B) qui compareraient les données d'un même patient sur l'application SAVER en utilisant leurs terminaux respectifs pourraient constater des incohérences. Si A vient d'introduire un seuil de douleur à 6/10, par exemple, alors que précédemment il était à 3/10, il est probable que pendant quelques dizaines de secondes, l'affichage sur le terminal de B reste à 3/10 pour ce même patient, simplement parce que la notification n'a pas encore eu le temps matériel pour être propagée à tous les soignants, ainsi qu'au système backend (Gyroflux, dans notre cas).

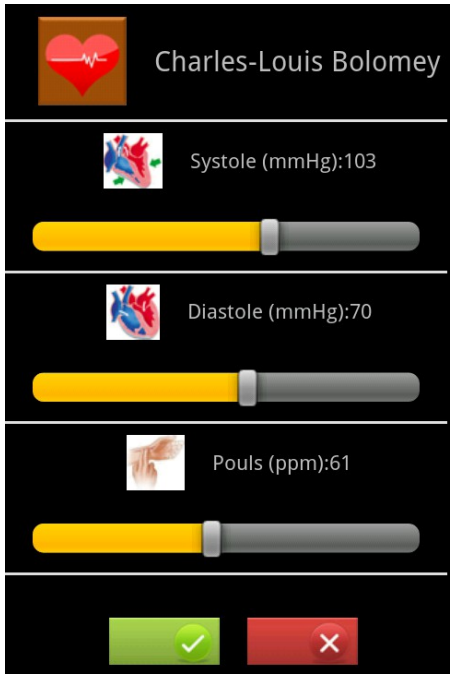
Normalement, ce mode de fonctionnement ne devrait pas constituer une gêne, mais permet d'améliorer notablement la réactivité du système ; en revanche, dans certains cas bien particuliers,

il peut apparaître des incohérences temporaires sur des terminaux différents[2].

Les images suivantes donnent quelques exemples d'introduction de données possibles .



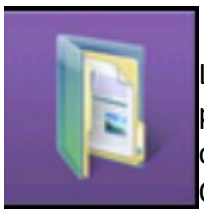




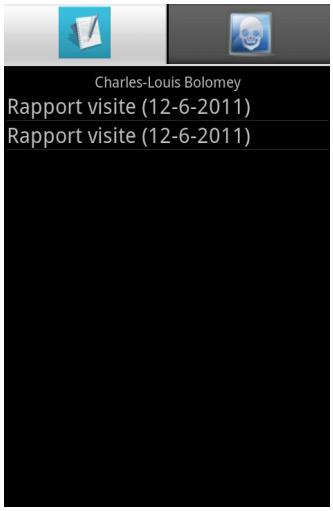
Il est également possible de consulter des données du patient par le biais de cette vue.

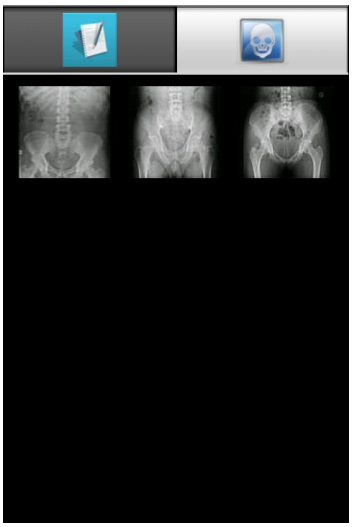
2 *Même avec un protocole synchrone, ces incohérences pourraient se faire jour, car les protocoles de transport utilisés ne permettent pour l'heure pas le multicast ; il y a donc forcément un délai entre la mise à jour des différents terminaux mobiles, et ce délai est d'autant plus long que les terminaux sont nombreux.*

4.6 Consultation de données du patient



Le bouton ci-contre livre accès à un extrait choisi de données tirées du dossier du patient. Les critères de choix de données à présenter dépendent bien évidemment d'une démarche médicale, et nous n'en discuterons pas plus avant pour l'instant. On a choisi ici de proposer l'accès à deux types de documents pour l'exemple.





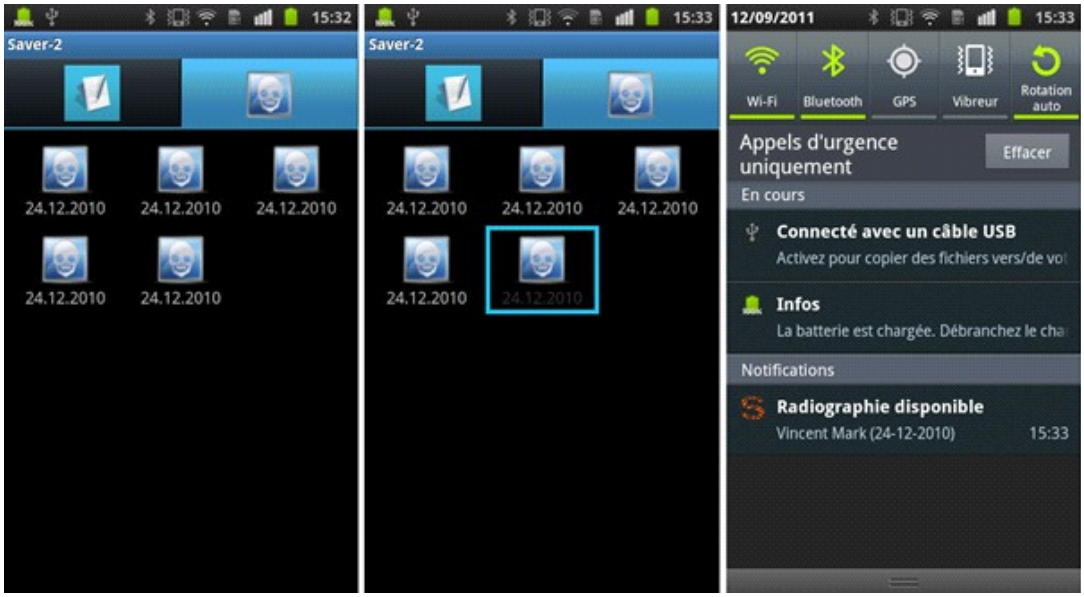
Les documents de type "rapport" sont tous types de documents pouvant être affichés dans un format PDF (un logiciel de lecture de fichiers PDF doit exister sur le smartphone). En principe, ce format est suffisamment générique pour s'accommoder même de notes manuscrites au besoin.

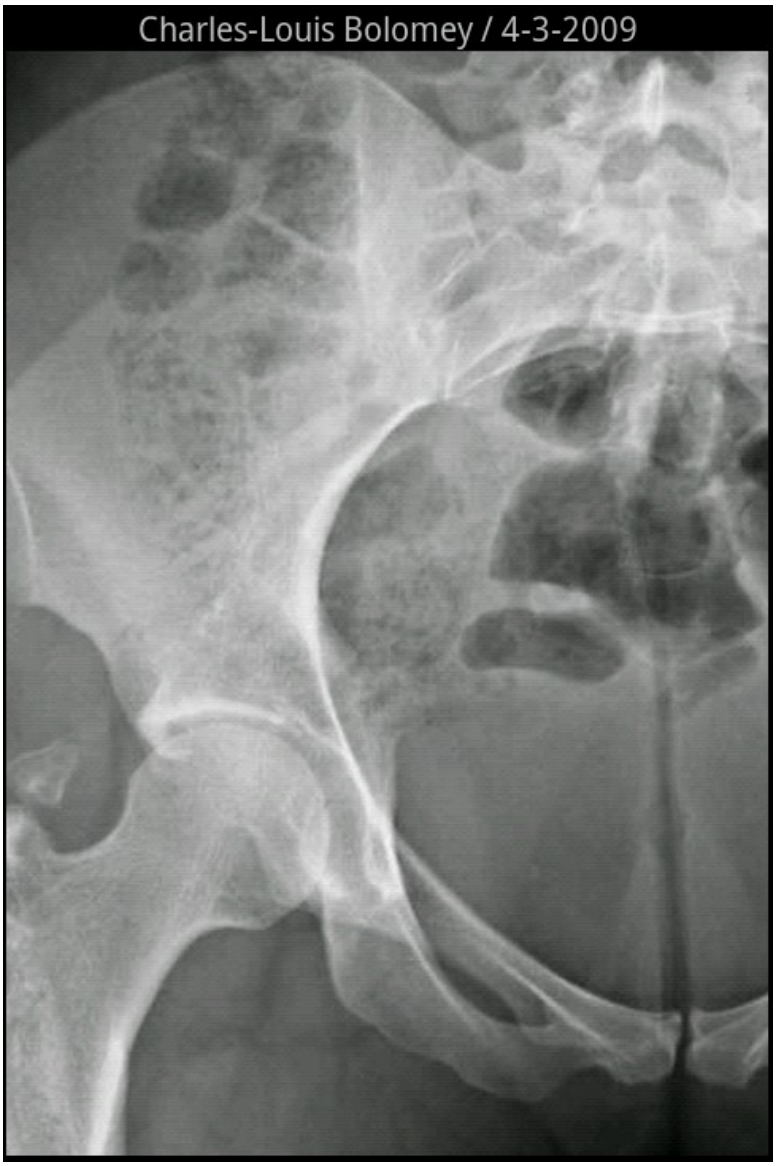
4.7 Radiographies

En fonction du réseau et de poids de la radiographie, une image peut être plus ou moins longue à télécharger. Il est donc malvenu de bloquer l'utilisateur durant le téléchargement de celle-ci.

Lorsque l'utilisateur choisi une radiographie, un message lui apparaît pour lui signaler qu'elle est en téléchargement (dans une tache parallèle). Une fois ce téléchargement terminé, une notification permettant d'ouvrir la radiographie apparaît dans la zone de notifications d'Android.

Pour l'instant, seul un format JPEG est supporté : il n'est pas prévu de développer une visionneuse d'images en format DICOM, excessivement lourdes et inutilkes sur un écran de smartphone ou de tablette.





4.8 Accès direct aux appareils de mesure

Sur ce principe, rien ne s'oppose à la lecture directe des données depuis des appareils de mesure, pour autant que ces derniers proposent une interface et une connectivité utilisables. Cette option n'a pour l'instant pas été développée, essentiellement pour des raisons de disponibilité de ces appareils. L'accès par Bluetooth ou IP depuis le smartphone du soignant ne devrait poser aucun problème particulier ; en revanche, l'accès par USB est moins évident, les smartphones et les tablettes ne permettant généralement pas de jouer le rôle de master dans une relation USB.

CHAPITRE 5 Pairage du PatientTag et du DoctorMate

Le problème fondamental est l'identification du malade près duquel se trouve le soignant, et réciproquement, du soignant qui va se préoccuper de ce malade. En principe, il y a de nombreuses manières d'aborder ce genre de problème. Cela peut se faire simplement, en demandant au soignant d'introduire cette identification d'une manière ou d'une autre.

5.1 L'identification optique

Ainsi, on pourrait demander, dans un système particulièrement simple, que le médecin, lorsqu'il s'approche d'un lit d'un malade, introduise un numéro (le numéro du lit, ou du box dans lequel se trouve le patient). On pourrait aussi munir le patient d'un code-barre, ou plutôt d'un code QR, que le soignant pourrait lire avec son smartphone. Ces solutions sont simples, et ont l'avantage pour le développeur de ne pas demander de contrôles d'identité trop conséquents, puisque c'est le soignant qui prend en charge cette identification. Si erreur il y a , elle peut être attribuée à celui qui a introduit la mauvaise information, donc au soignant. Une manière simple d'éviter un problème d'identification mutuelle épineux et critique.

Pour séduisante que soient ces alternatives, du moins du point de vue du développeur, elles ont toutefois l'inconvénient de demander au soignant un acte qu'il doit impérativement effectuer avant même de pouvoir commencer son véritable travail de soignant. Il n'est pas certain qu'il se donne la peine de faire ce geste si le lit en question contient un patient qu'il connaît déjà. On introduit donc potentiellement un risque de ne pas renseigner les données du système de manière aussi complète qu'on le désirerait. Et puis, avouons qu'il n'est pas forcément très sympathique

pour un patient probablement anxieux, lorsque le médecin s'approche de lui, de le voir commencer par tapoter son smartphone, ou pire encore, promener le même smartphone près de son poignet pour lire un code optique. Si de plus le patient a mis son poignet sous les couvertures, ou que le code est placé sur l'autre poignet, cela induit des complications : il faudrait alors mettre le code sur le lit, et ne pas oublier de déplacer le code avec le patient. Des objections similaires peuvent être faites à l'introduction d'un numéro de lit , indications qui requièrent des mises à jour de la base de données lorsque l'on change le patient de place.

5.2 Identification par la position

Identifier un malade par sa position est possible : certains systèmes WiFi de positionnement parviennent à localiser un récepteur à un ou deux mètres près en bonnes conditions, ce qui semble à priori suffisant : lorsque la balise médecin et la balise patient se trouvent proches l'une de l'autre, on réalise le pairage. Mais là aussi, le changement de lit du malade (ou le déplacement de son lit) doit impérativement être renseigné dans le cadre du serveur de mobilité; qui va assurer cette tâche pratiquement en temps réel ?

Il existe d'autres facteurs d'imprévu dans le cadre de cette solution; la configuration en boxes de certaines parties du service des urgences au CHUV peut influer sur la précision du positionnement. Le positionnement par WiFi est probablement un peu « limite » en ce qui concerne la précision; et un positionnement par différence de temps de vol, arbitrairement précis, requiert une infrastructure tout de même relativement lourde pour sa mise en œuvre.

5.3 Pairage mutuel de balises

La solution que nous préconisons dans ce cadre est l'utilisation de balises radiofréquence actives. Les balises passives demandent une grande quantité d'énergie à la lecture; prélever cette énergie sur un lecteur mobile (smartphone du soignant) semble irréaliste. Par ailleurs, les impulsions de lecture pourraient entrer en conflit avec des appareils de mesure médicaux, comme des ECG, par exemple (problème de pollution électromagnétique).

Une balise active à basse énergie peut durer plusieurs mois sur une simple pile bouton, et l'énergie radio-électrique émise est pratiquement négligeable. Une telle balise (qui peut être un simple transmetteur périodique, dans sa version la plus élémentaire) est lisible avec peu d'énergie également par un dispositif accouplé au smartphone, et ceci à partir d'une distance de deux ou trois mètres. Le lecteur peut utiliser l'indication RSSI (Received Signal Strength Indicator) pour évaluer l'intensité du signal reçu.

Ce principe a l'avantage d'être largement transparent pour le soignant : on détecte le pairage patient-soignant sans intervention réelle du médecin. Il est aussi transparent pour le patient, qui n'a pas à « montrer patte blanche » (cas des codes barre ou QR) au soignant.

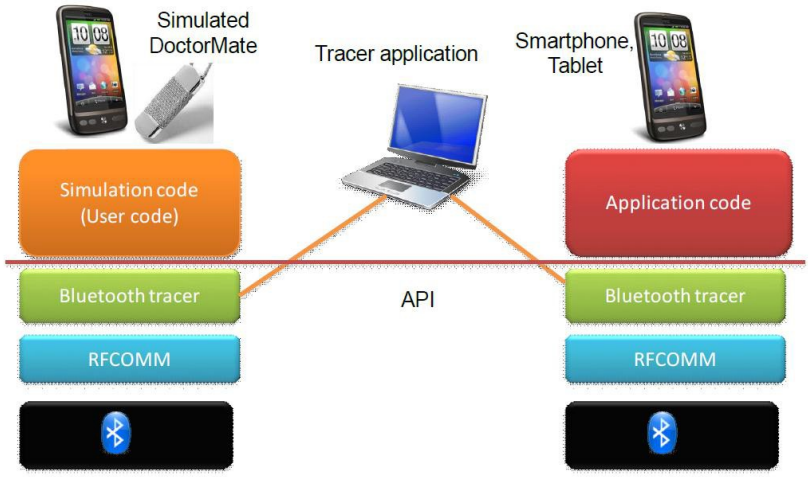
Il y a en revanche quelques inconnues qu'il s'agit encore de lever à la lumière de l'expérience pratique telle que vécue par le personnel hospitalier :

- Certains cas ne permettront probablement pas un pairage univoque. Il faudra donc, au moment où le praticien désire renseigner le système, recourir à un dialogue lui demandant de préciser le pairage.

- L'utilisation de l'indication RSSI requiert quelque précaution. Tout système ne dispose pas de cette indication, et quand elle est disponible, elle ne donne en principe pas d'indication de distance, mais d'indication du niveau de signal reçu. Cette indication dépend de la distance, mais aussi des conditions respectives de propagation, ainsi que du niveau de puissance de la pile de l'émetteur. Le pairage doit donc être validé par une algorithmique adéquate, vraisemblablement au niveau du serveur.

5.4 Utilitaire de développement pour Bluetooth

Dans le cadre du développement de SAVER, il a été développé un utilitaire permettant un dépannage de protocoles d'application développés sur la base de Bluetooth. Cet utilitaire permet le traçage en temps réel sur un serveur de traçage résident sur un PC équipé d'une machine virtuelle Java 1.6.



Deux éléments dans cette architecture sont à définir dans le cadre de ce projet :

1) Le code « **Bluetooth Tracer** » présent dans chaque application Android

2) L'application présente sur le serveur qui récolte les différents logs envoyés par les Smartphones Android. Cette application porte le nom de « **Bluetooth Tracer Server** ».

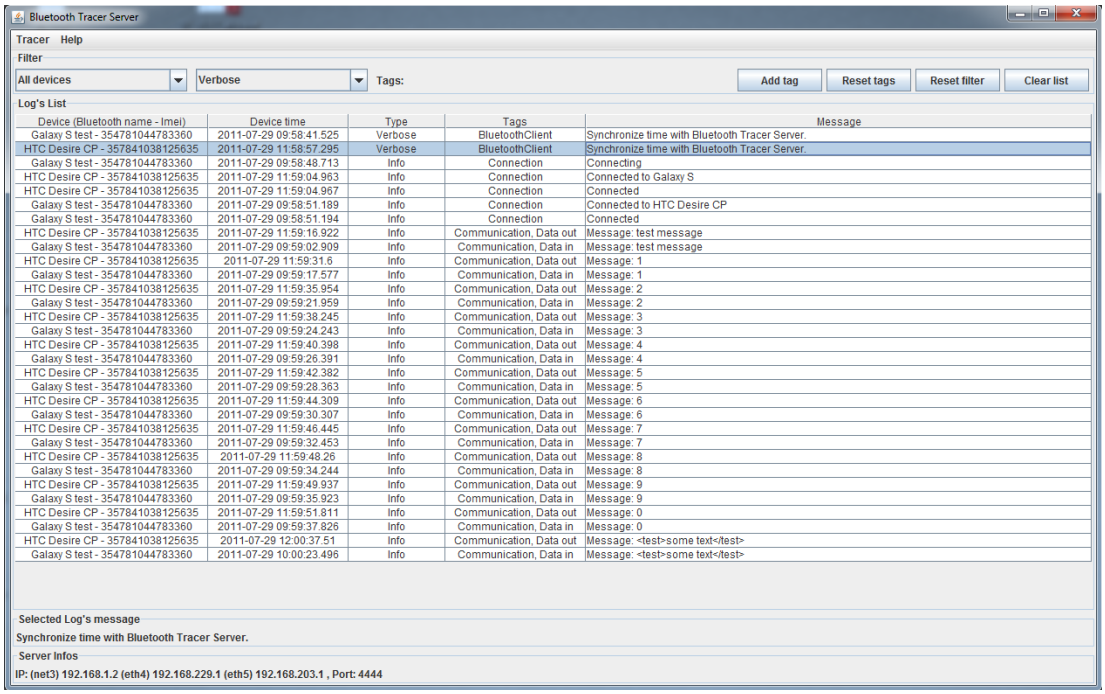
Un manuel d'utilisation est fourni en annexe. Ce document traite des aspects techniques liés à l'application du point 2.

IMPORTANT : Plus d'informations sur le fonctionnement des Log peut être trouvée sur les URLs suivantes :

- http://developer.android.com/reference/android/util/Log.html

- http://developer.android.com/guide/developing/debugging/debugging-log.html

Ce programme permet de récolter les différents logs envoyés par les applications Android contenant le code « Bluetooth Tracer » présenté au chapitre précédent. Le fichier « Bluetooth Tracer Server.jar » contient le programme et les librairies dont il a besoin. Il suffit de l'exécuter pour ouvrir l'application.



Nous pouvons distinguer cinq parties distinctes qui ont chacune une fonction différente :

1) Menu de l'application permettant d'accéder au paramétrage du serveur ainsi qu'à la sauvegarde des logs.

2) Composants graphiques permettant de gérer le filtre appliqué sur la liste des logs affichés dans le point suivant.

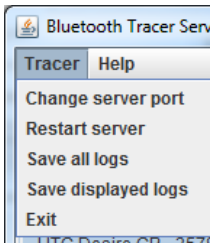
3) Liste de logs reçus par le serveur et étant inclus dans le paramétrage du filtre

4) Affichage du message du log sélectionné dans la liste

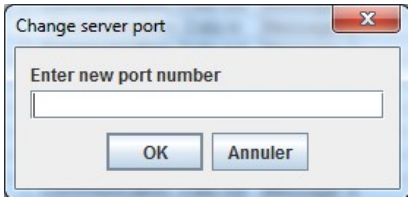
5) Affichage des informations du serveur

Voici en détail les possibilités offertes par ces différentes parties :

5.4.1 Menu de l'application



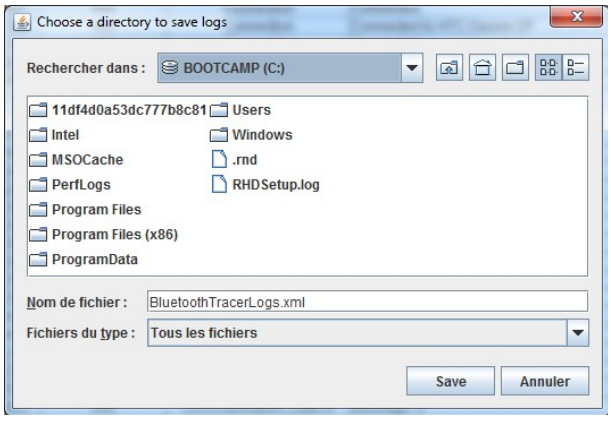
Ce serveur écoute par défaut sur le port 4444 en TCP. Pour changer le port, il suffit de cliquer sur le sous-menu «Change server port ». Une nouvelle fenêtre va apparaître demandant de saisir le nouveau port :



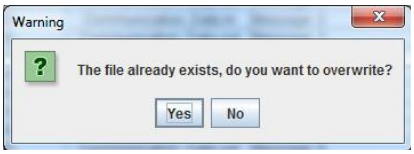
Si le port n'est pas correct, le programme ne le prendra pas en compte (min = 1, max = 65535).

Vous avez la possibilité de redémarrer le serveur en cliquant sur le sous-menu « Restart server ». Le serveur redémarrera en écoutant sur le dernier port entré dans le programme (le port affiché dans les informations du serveur, point 5).

Les sous-menus « Save all logs » ainsi que « Save displayed logs » servent à sauver les logs dans un fichier au format XML. Le premier sous-menu permet de sauvegarder la totalité des logs reçus par le serveur. Le second sous-menu permet de sauvegarder seulement les logs affichés dans la liste présente dans la partie 3 (donc seulement les logs filtrés). Une fois un des deux sous-menus cliqués, une fenêtre apparaîtra demandant de sélectionner le répertoire et le fichier de destination :



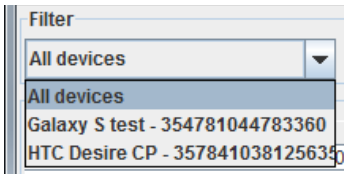
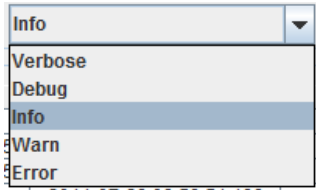
Si le fichier existe déjà, le message suivant sera affiché :

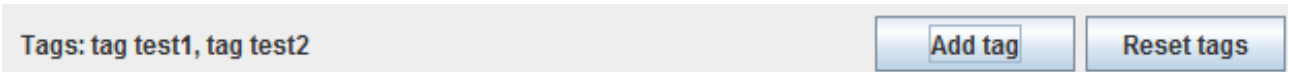


En cliquant sur « No », aucun log ne sera sauvegardé.

5.4.2 Filtre

Le filtre est composé de quatre parties :

	Possibilité de filtrer les logs par appareil.
	Possibilité de filtrer les logs par type. Pour plus d'informations, se référer aux adresses URL données dans les descriptifs de log.



Possibilité de filtrer les logs avec des tags. Les deux boutons permettent de 1) Ajouter un tag 2) Effacer tous les tags. ATTENTION : Ne pas utiliser les caractères '<' et '>'

	Ces deux boutons permettent respectivement de réinitialiser le filtre (affiche tous les logs) et d'effacer tous les logs reçus jusqu'à présent par le serveur.

5.4.3 Liste des logs

Device (Bluetooth name - Imei)	Device time	Type	Tags	Message
Galaxy S test - 354781044783360	2011-07-29 09:58:41.525	Verbose	BluetoothClient	Synchronize time with Bluetooth Tracer Server.
HTC Desire CP - 357841038125635	2011-07-29 11:58:57.295	Verbose	BluetoothClient	Synchronize time with Bluetooth Tracer Server.
Galaxy S test - 354781044783360	2011-07-29 09:58:48.713	Info	Connection	Connecting
HTC Desire CP - 357841038125635	2011-07-29 11:59:04.963	Info	Connection	Connected to Galaxy S
HTC Desire CP - 357841038125635	2011-07-29 11:59:04.967	Info	Connection	Connected

Nous pouvons remarquer que chaque log possède 5 champs dans la liste :

1) L'appareil qui l'a envoyé (nom et IMEI)

2) La date et l'heure de sa création sur l'appareil qui l'a envoyé

3) Le type

4) Les tags associés

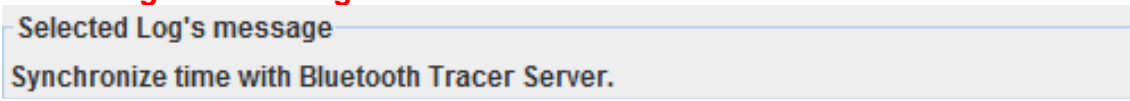
5) Le message (peut contenir du code XML)

Remarques :

- Le message peut être consulté dans sa totalité et plus clairement dans la partie suivante.

- L'ordre d'affichage des différents logs n'est pas correct à 100% car les appareils ne sont pas synchronisés via un même serveur NTP (remarque valable seulement lorsque les logs de plusieurs appareils sont affichés)

5.4.4 Affichage du message



Permet d'afficher complètement un message assez long qui ne passerait pas dans la liste des logs. Pour afficher le message d'un log en particulier, il suffit de le sélectionner dans la liste.

5.4.5 Informations sur le serveur

Affiche les différentes adresses IP en fonction des interfaces actives (seulement pour les réseaux en local) ainsi que le port d'écoute. Par défaut le port est 4444, néanmoins il peut être changé via le menu de l'application.

5.5 Développement du traceur

Ce programme a été réalisé en Java avec l'IDE Eclipse Indigo (Build id: 20110615-0604). Dans le but de proposer une interface pour la consultation des logs à l'utilisateur, la librairie Swing a été utilisée. De plus, un modèle MVC a été utilisé pour séparer la vue du modèle de données.

Le code est complètement commenté en annexe. Voici une description des classes et de leurs utilités :

- **ch.iict.bluetooth.tracer**

  - **Launcher**

Lancement de l'application, crée le contrôleur et initialise quelques modèles

  - **Controller**

Contrôleur de l'application, gère les interactions entre la vue et les modèles de données, gère les abonnements aux événements (changements dans la vue ou dans les modèles), lance le serveur TCP

- **TCPServer**

Gère le socket du serveur TCP/IP

- **ClientServiceThread**

Gère chaque client connecté au serveur (échange des données)

- **ch.iict.bluetooth.tracer.model**

- **Device**

Gestion des informations d'un appareil (nom, IMEI) qui a envoyé des tags au serveur

- **Filter**

Gestion d'un filtre (par appareil, par tag(s), type de log)

- **ServerInfos**

Gestion des informations sur le serveur (IP et port). Par défaut, le serveur est lancé sur le port 4444 mais l'utilisateur peut changer cette valeur via le menu.

- **TracerLog**

Gestions d'un log (appareil qui l'a envoyé, message, type, etc.)

- **ModelComboBoxDevices**

Modèle de données lié à la liste des appareils qui ont envoyé des logs au programme.

- **ModelLogJTable**

Modèle de données lié à la liste de logs qui est affichée au milieu de la fenêtre principale.

- **NewDeviceEvent**

Classe qui gère les nouveaux événements concernant l'ajout d'un appareil. Gère les notifications aux différents abonnés.

- **NewDeviceListener (Interface)**

Listener abonné aux événements de type NewDeviceEvent

- **ResetDeviceFilterEvent**

Classe qui gère les nouveaux événements concernant le reset de la liste déroulante contenant les appareilles utilisée pour filtrer les logs. Gère les notifications aux différents abonnés.

- **ResetDeviceFilterListener**

Listener abonné aux événements de type ResetDeviceFilterEvent

- **SelectedDeviceChangedEvent**

Classe qui gère les nouveaux événements concernant le changement du filtre des appareils (utilisateur qui sélectionne un appareil en particulier dans la liste déroulante). Gère les notifications aux différents abonnés.

- **SelectedDeviceChangedListener**

Listener abonné aux événements de type SelectedDeviceChangedEvent

- **TagAddedEvent**

Classe qui gère les nouveaux événements concernant l'ajout d'un tag pour le filtre. Gère les notifications aux différents abonnés.

- **TagAddedListener**

Listener abonné aux événements de type TagAddedEvent

- **ServerInfosChangedEvent**

Classe qui gère les nouveaux événements concernant le changement des infos (IP, port) du serveur. Gère les notifications aux différents abonnés.

- **ServerInfosListener**

Listener abonné aux événements de type ServerInfosChangedEvent

- **TypeChangedEvent**

Classe qui gère les nouveaux événements concernant le changement du type de logs filtrés. Gère les notifications aux différents abonnés.

- **TypeChangedListener**

Listener abonné aux événements de type TypeChangedEvent

- **ch.iict.bluetooth.tracer.view**

- **MenuItemAboutAction**

Classe qui gère l'action du bouton « About » présent dans le menu de l'application (Affiche une popup avec quelques informations sur l'application).

- **MenuItemExitAction**

Classe qui gère l'action du bouton « Exit » présent dans le menu de l'application (quitte l'application).

- **MenuITemSaveLogsAction**

Classe qui gère l'action des deux boutons de sauvegarde de logs présents dans le menu de l'application (Crée un fichier sur le disque dur avec les logs).

- **MenuItemServerRestartAction**

Classe qui gère l'action du bouton « Restart server » présent dans le menu de l'application (Relance le serveur TCP/IP).

- **MenuItemServerSettingsAction**

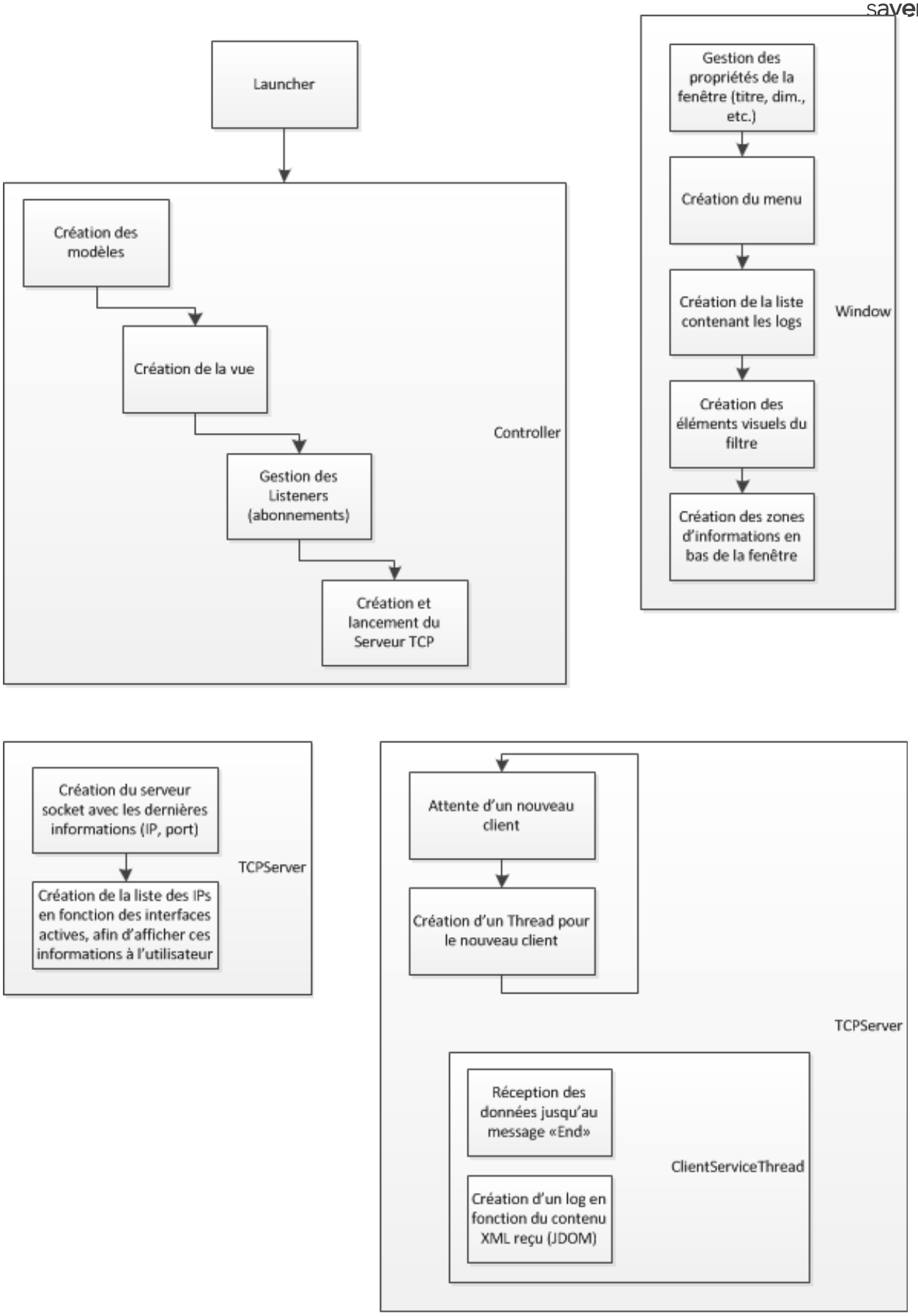
Classe qui gère l'action du bouton « Change server port » présent dans le menu de l'application (Ouvre une popup qui demande à l'utilisateur de saisir un nouveau numéro de port, change les informations du serveur et redémarre le serveur).

- **Window**

Cette classe gère pratiquement toute la partie visuelle du programme (fenêtre, éléments graphiques tels que listes, boutons, etc.).

Toutes les informations sont contenues dans les commentaires du code et dans la Javadoc qui se trouve en annexe.

Voici un diagramme qui représente les différentes tâches faites par le programme ainsi que leur séquence:


Launcher
Création des modèles
Création de la vue
Gestion des Listeners (abonnements)
Création et lancement du Serveur TCP
Controller
Gestion des propriétés de la fenêtre (titre, dim., etc.)
Création du menu
Création de la liste contenant les logs
Création des éléments visuels du filtre
Création des zones d'informations en bas de la fenêtre
Window
Création du serveur socket avec les dernières informations (IP, port)
Création de la liste des IPs en fonction des interfaces actives, afin d'afficher ces informations à l'utilisateur
TCPServer
Attente d'un nouveau client
Création d'un Thread pour le nouveau client
TCPServer
Réception des données jusqu'au message «End»
Création d'un log en fonction du contenu XML reçu (JDOM)
ClientServiceThread

5.5.1 Logs

Les logs qui sont reçus par le serveur sont contenus dans un code XML. Voici la structure du XML :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bluetoothTracer id="…" imei="…">
<log>
<creationTimeStamp>…</creationTimeStamp>
<tags>
<tag>…</tag>
</tags>
<type>…</type>
<message>…</message>
</log>
</bluetoothTracer>
```

Un interpréteur DOM est utilisé pour récupérer les informations et créer un log (avec la classe TracerLog) depuis un code XML reçu.

5.5.2 Appareil (Classe Device)

Un appareil est composé d'un nom (dans la plupart des cas le nom Bluetooth) ainsi que d'un identifiant IMEI. L'utilisateur pourra filtrer les logs en fonction de l'appareil qu'il désire. A chaque fois que le serveur reçoit un log, il regarder si l'appareil qui l'a envoyé est déjà présent dans la liste des appareils pouvant être filtrés.

5.5.3 Serveur TCP/IP

Le serveur TCP/IP est automatiquement lancé au démarrage de l'application et utilise plusieurs threads pour gérer plusieurs connexions simultanées. Par défaut, le port utilisé est le numéro 4444 mais l'utilisateur a la possibilité de changer cette valeur.

Remarque : La classe qui gère les informations du serveur (ServerInfos) contient une adresse IP ainsi qu'un port.

5.5.4 Tests

Plusieurs tests ont été effectués avec deux appareils différents :

- HTC Desire

- Samsung Galaxy S

Les tests ont été tout simplement réalisés avec le code d'exemple (disponible en annexe) contenant le code Android du Bluetooth Tracer (Chat Bluetooth). De plus, du code XML a été envoyé en tant que **message (texte du log)** pour tester l'interpréteur présent dans le code de ce serveur.

Tous les tests effectués ont été concluants. Cependant vu le temps utilisé pour coder cette application (2 semaines), les tests n'ont pas pu être approfondis.

5.5.5 Améliorations

L'application peut être enrichie pour gérer le contenu du message plus en détail. Par exemple, si le codeur envoie des logs au Bluetooth Tracer Server en les formatant de manière prédéfinie, il pourrait traiter de manière personnalisée le contenu du message (affichage personnalisé du message en fonction du type, etc.).

5.5.6 Conclusion

Bluetooth Tracer Server est un programme simple et intuitif qui permet d'afficher de manière centralisée des logs provenant de plusieurs appareils différents, ceci dans le but de débuguer ou de tracer une certaine activité.

Il est actuellement difficile de coder sur le protocole Bluetooth en utilisant les outils mis à disposition car l'émulateur ne gère pas le Bluetooth. De plus, deux appareils ne peuvent pas être connectés et débugués en même temps (sauf en utilisant des ressources supplémentaires). Bluetooth Tracer Server est donc un outil qui apporte une solution à ce problème.

5.6 Protocole entre le DoctorMate et le smartphone

Le protocole de communication entre le DoctorMate et le PatientTag ne fait pas l'objet d'une spécification dans le cadre de ce document.

Entre le DoctorMate et le smartphone, le protocole est basé sur le service RFCOMM de Bluetooth. Rappelons que RFCOMM est un équivalent sans fil de RS-232, du moins dans ses spécifications.

La liaison est une liaison maître-esclave : le smartphone interroge le DoctorMate, et ce dernier lui répond par la liste des identificateurs de PatientTags détectés à proximité, accompagnés d'un indicateur d'intensité de signal reçu. Le DoctorMate et le smartphone sont supposés appairés préalablement. Cette condition d'appariement est contrôlée et vérifiée par SAVER au moment de l'initiation de la session (login).

Les identificateurs de PatientTags sont codés sur 32 bit ; seuls K bits sont significatifs, les bits (de poids fort, 32-K MSB) non significatifs sont mis à zéro, l'identificateur est unique dans une zone d'implantation de SAVER. L'indication RSSI est codée sur 8 bits. Une réponse du DoctorMate à une demande d'exploration de l'environnement se compose donc pour l'essentiel à N paquets de 40 bit (32 bits pour l'identification du PatientTag et 8 bit pour l'indication RSSI correspondante). Les N paquets ne sont pas ordonnés de manière particulière : il appartient au smartphone de décider de la plausibilité des indications données par le DoctorMate pour déterminer quel est le PatientTage effectivement le plus proche. Pour mémoire, le RSSI du chip radio est codé sur 8 bits avec une résolution de **0.5dB**, donc une dynamique équivalente de **127dB**. Dans le monde RF la dynamique effective est de **80dB**, cet indicateur baisse de **6 dB** quand on double la distance.

Ce protocole fait l'objet d'un développement spécifique sur du matériel dédié, et aussi d'un développement en simulation sur un smartphone Android. Ce dernier développement inclut une librairie de traçage très performante, utilisable comme traceur générique de protocoles basés sur

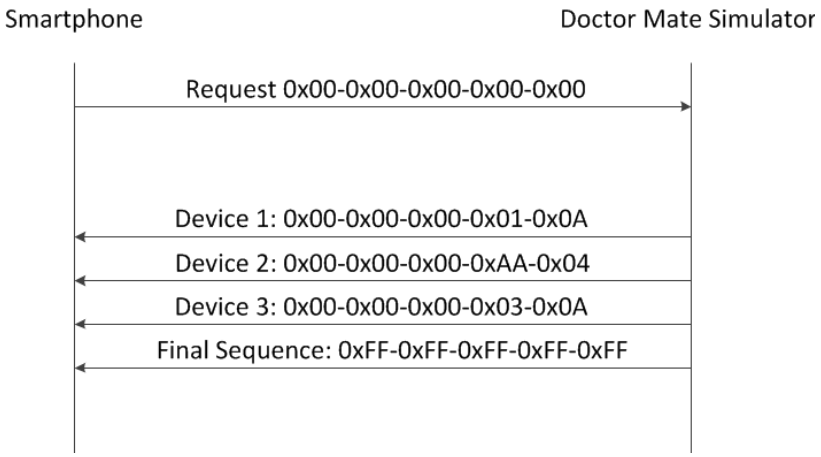
Bluetooth, profil RFCOMM.

5.6.1 Transmission de l'état de la batterie du Patient Tag

Le patient Tag transmet en permanence l'état de charge de la batterie. Ceci est réalisé dans le postambule du protocole de transmission. Le dernier identificateur transmis est effectivement "0" (32 bits nuls). Le RSSI correspondant, en principe non significatif en tant que tel, contient une indication de charge de la batterie, codée sur 8 bit.

5.7 Tâche du simulateur

Ce programme permet de simuler le Doctor Mate sur la plateforme Android. Lorsqu'un smartphone Android pairé envoie (via Bluetooth) une séquence avec 40 bits à 0, notre Doctor Mate lui répond avec une liste de tag ids à proximité ainsi que leurs RSSI respectifs. En fin de liste, une séquence comprenant 40 bits à 1 est envoyée. Voici un schéma représentant ces échanges:



Dans ce schéma les données échangées sont en hexadécimal. Chaque échange contient 40 bits = 5 bytes. Ces 5 bytes ont un sens bien précis:

- 4 premiers bytes: tag id

- Dernier byte: RSSI (Received Signal Strength Indication)

Dans cet exemple, une liste de trois tags sont envoyés au Smartphone. Ce simulateur peut envoyer de 0 à 5 tags en fonction du paramétrage fait par l'utilisateur. Dans le cas où aucun tag n'est envoyé dans la liste, la séquence finale comportant 40 bits à 1 est quand même envoyée au Smartphone.

Les cinq tags sont prédéfinis dans le code de l'application, voici les bits qui les composent:

- Tag 1

  - Tag id: 0x00, 0x00, 0x00, 0x01 (0000 0001)

  - RSSI: 0x0A (0000 1010)

- Tag 2

  - Tag id: 0x00, 0x00, 0x00, 0xAA (1010 1010)
  - RSSI: 0x04 (0000 0100)

- Tag 3

  - Tag id: 0x00, 0x01 (0000 0001), 0x02 (0000 0010), 0x03 (0000 0011)
  - RSSI: 0x0A (0000 1010) [3]

Tag 4

  - Tag id: 0x01 (0000 0001), 0x02 (0000 0010), 0x03 (0000 0011), 0x04 (0000 0100)
  - RSSI: 0x0F (0000 1111)

- Tag 5

  - Tag id: 0x0A (0000 1010), 0x0B (0000 1011), 0x0C (0000 1100), 0x0D (0000 1101)
  - RSSI: 0xFF (1111 1111)

5.8 Programme Android

5.8.1 Fonctionnement

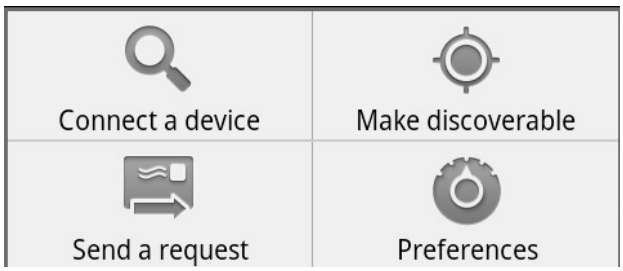
Pour faire fonctionner le programme Doctor Mate sur Android, il suffit d'exécuter le fichier DoctorMate.apk fourni en annexe de ce document sur le Smartphone. Une fois l'application installée, il suffit de cliquer sur la bonne icône dans la liste des programmes:

Une fois l'application lancée, un message d'avertissement est affiché si le Bluetooth n'est pas activé :

3 *Les RSSI des tags 1 et 3 sont les mêmes. Cette situation peut s'avérer utile pour les tests.*

En appuyant sur « oui », l'utilisateur se retrouve dans l'activité principale de l'application :

La partie de l'écran en noir va servir à afficher les données reçues et envoyées par le Doctor Mate. La barre de titre contient sur la partie de droite une indication sur l'état de la connexion Bluetooth. En cliquant sur le bouton « menu », l'utilisateur aura accès aux éléments suivants :



Connect a device

Affiche la liste des appareils Bluetooth déjà pairés et détectés à proximité.

- **Make discoverable**

Permet de rendre le Smartphone détectable via Bluetooth pendant 300 secondes afin d'établir un pairage via un autre Smartphone.

- **Send a request**

Envoie la séquence de 40 bits à 0 afin de demander la liste des tags à proximité du Doctor Mate. **Cette option est présente dans ce programme pour tester son bon fonctionnement.** En utilisant deux Smartphones avec cette application, on peut tester si tout fonctionne correctement.

- **Preferences**

Permet d'accéder aux préférences de l'application qui sont décrites dans la suite de ce document.

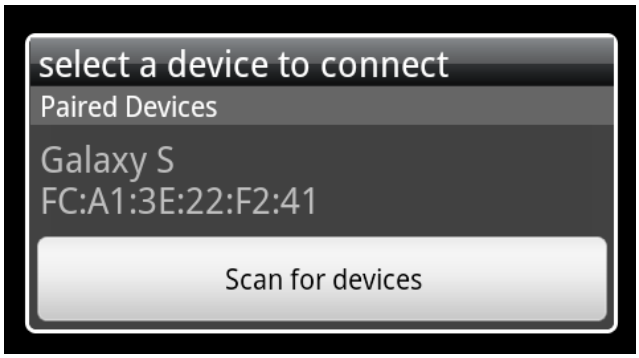
Pour que l'échange via Bluetooth soit possible, il faut être connecté à un autre appareil. Pour cela, il y

a deux possibilités :

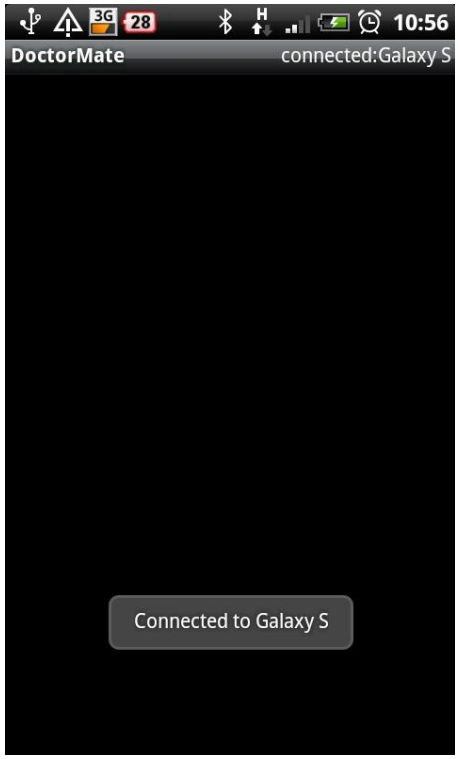
1) Démarrer la connexion depuis le Smartphone distant

2) Démarrer la connexion depuis le Doctor Mate

Pour la deuxième solution, il suffit d'appuyer sur le menu « Connect a device » (affiche la liste des appareils pairés à proximité) et de cliquer sur l'appareil désiré :

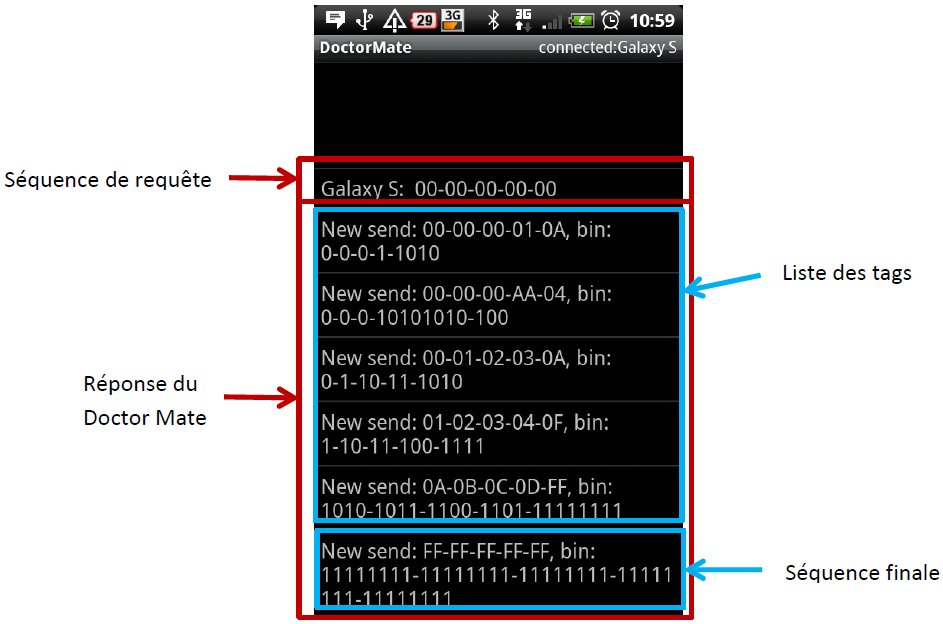


Une fois que la connexion est établie, un petit message est affiché sur l'écran :

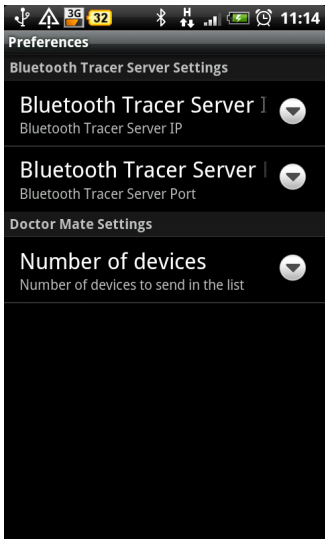


Nous pouvons aussi remarquer que sur la partie en haut à droite de l'écran, l'état de la connexion Bluetooth a changé.

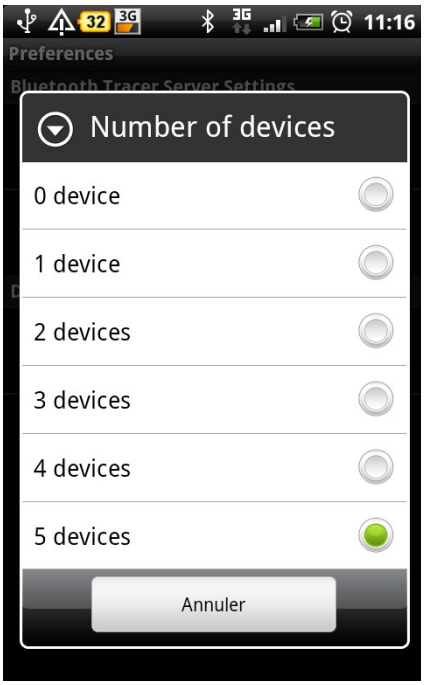
Maintenant que la connexion est établie, il ne reste plus qu'à attendre une requête pour que le Doctor Mate réponde avec la liste des tags et la séquence finale :



Le fonctionnement de l'application est donc très simple. **Les tags envoyés sont prédéfinis et fixés dans le code de l'application.** Néanmoins, l'utilisateur peut changer le nombre de tags qu'il veut inclure dans la liste qui est envoyée en réponse aux requêtes. Pour cela, il lui suffit d'aller dans les préférences de l'application et de changer le dernier paramètre :



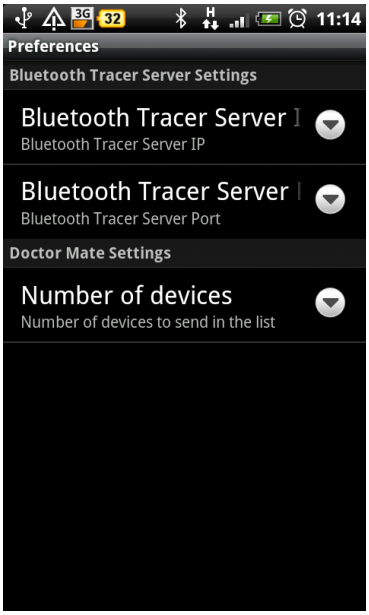
Une liste sera affichée permettant de choisit entre 0 et 5 tags :



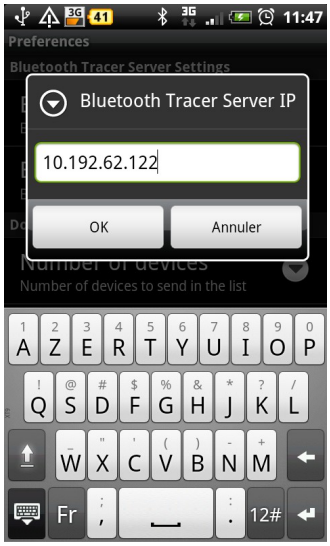
Toutes les réponses aux requêtes envoyées après la validation de cette préférence tiendront compte du nouveau nombre de tags à inclure dans la liste.

5.8.2 Lien avec le projet BlueSim

Dans ce programme, pour pourvoir débuguer et mieux suivre les divers échanges Bluetooth, un lien a été fait avec le Bluetooth Tracer Server du projet BlueSim. Tous les logs générés par le programme sont donc enregistrés sur la carte SD (dans le répertoire « bluetoothTracer ») et envoyés au Bluetooth Tracer Server. Pour pouvoir modifier l'IP et le port du serveur, il suffit d'aller dans les préférences du programme :



Le premier menu permet de changer l'adresse IP :



Le deuxième menu permet de changer le port :

Les logs prendront en compte les nouveaux paramètres dès leurs modifications. **Toute modification des paramètres restera en mémoire même si l'application est fermée.**

Une vidéo a été réalisée pour mieux illustrer l'utilisation de ces outils.

CHAPITRE 6 Balises patient, compagnon soignant

Ce chapitre constitue une « *tentative data sheet* » pour le couplage balise patient- balise médecin – smartphone du soignant. En temps que spécification exploratoire, elle est bien sûr sujette à critiques, et les propositions d'améliorations, voire les alternatives, sont les bienvenues.

6.1 Balise patient (patientTag)

On peut imaginer la balise patient comme un simple système transmetteur (TX pur). Pour que le soignant soit informé en temps utile de l'identité de la balise (du patient) de laquelle il s'approche, on peut se baser sur les chiffres estimés suivants :

- La balise doit être identifiée dans un délai de quelques secondes entre l'approche du soignant et son identification par le système. Le temps de s'approcher, de dire bonjour et de constater *de visu* l'état global du patient. Si la balise émet un code d'identification avec une périodicité de l'ordre de trois secondes, il ne devrait en résulter aucune gêne notable pour le soignant.

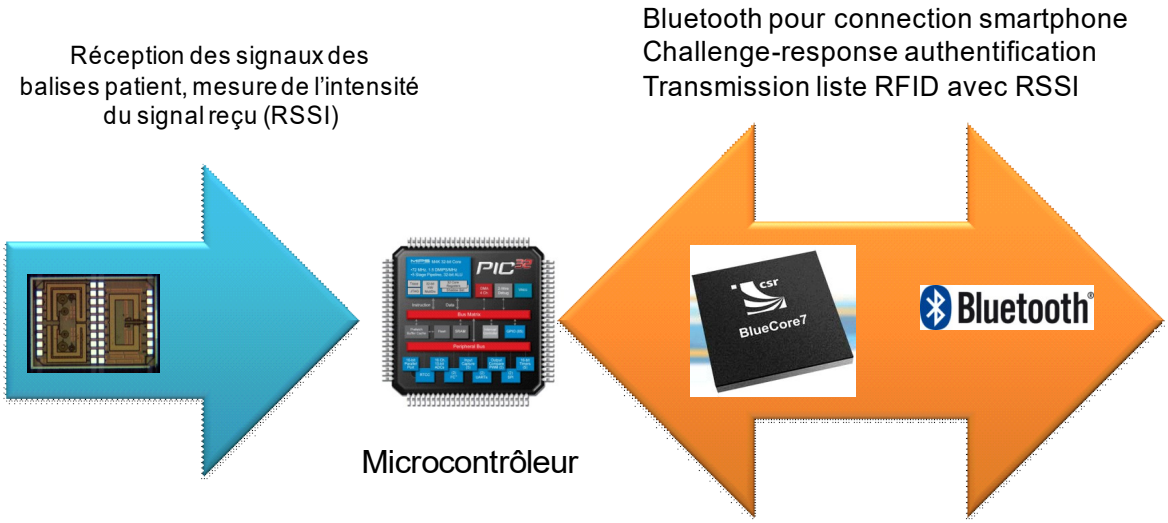
- La balise émet un code d'identification unique. Un service des urgences peut abriter quelques dizaines de malades, mais il serait probablement dommage de se limiter à ce nombre, étant donné que cette infrastructure est certainement réutilisable dans des contextes différents. A priori, si la balise peut supporter une dizaine de milliers de codes, il semble que l'on puisse « voir venir » pour de futures extensions. Si l'on ajoute un bit d'indication d'état de la pile, deux bytes seraient donc suffisants.

- La distance à laquelle la balise peut être reçue doit correspondre à une proximité raisonnable du soignant et du patient. Le compromis n'est pas facile à déterminer, mais dans le cas du projet SAVER, le CHUV est organisé en boxes, et les patients sont de fait relativement éloignés l'un de l'autre. Une distance de détection de trois mètres environ semble raisonnable. Une application dans laquelle les balises patients sont plus proches les unes des autres pourrait requérir de modifier ce chiffre, par exemple en diminuant sensiblement la puissance d'émission.

6.2 Balise soignant (doctorMate)

La balise soignant est chargée de deux tâches bien distinctes :

- Recevoir les divers signaux en provenance des balises patient suffisamment proches pour être détectées, et en garder une liste à jour en permanence, avec la puissance de réception constatée (RSSI, Received Signal Strength Indication)



- Rester à l'écoute du smartphone qui va périodiquement demander la liste des balises détectées à proximité. En principe, le smartphone possède déjà les données essentielles des patients en cache; leur mise à jour s'effectue en background, ce qui rend une interrogation périodique Bluetooth (consommatrice d'énergie) inutile.

- Alternativement, on pourrait connecter le smartphone par USB plutôt que par Bluetooth; ceci résoudrait aussi en partie le problème d'alimentation de la balise soignant. Il y a néanmoins quelques difficultés inhérentes à cette connexion; ainsi, la plupart des smartphones n'ont pas le profil « contrôleur USB », ce qui pose un sérieux problème d'implémentation. Bluetooth n'est de loin pas une garantie de compatibilité, mais au moins les problèmes mécaniques sont-ils résolus par défaut, et les divers smartphones possèdent-ils généralement un profil leur permettant de contrôler un micro-réseau (Exception iPhone).

- Implémenter un protocole de type « challenge-response » pour authentifier

mutuellement la balise soignant et le smartphone vis-à-vis du serveur.

- Inscrire dans le protocole de communication la transmission automatique de l'état de l'alimentation des deux balises.

CHAPITRE 7 Protocole de communication

Le mobile communique avec le serveur à l'aide d'un protocole d'application basé sur XML. Le protocole se fonde sur le paradigme *request-response* des protocoles de la famille HTTP.

Le protocole de communication correspond pour l'essentiel à une version quelque peu allégée du protocole utilisé par l'infrastructure OSMOSYS (http://useraware.iict.ch/fr/mobilite/osmosys/index.html), et qui est commercialisé par la société sysmosoft sa (http://www.sysmosoft.com) dans le cadre du produit SENSE. Dans la suite de ce chapitre, nous allons volontairement passer sous silence (en particulier dans les graphiques explicatifs) la division des tâches entre OSMOSYS/SENSE et SAVER pour ne pas alourdir notre propos. Nous reviendrons sur cette dualité lorsque nous nous intéresserons au serveur et à sa structure de connecteurs vers l'extérieur.

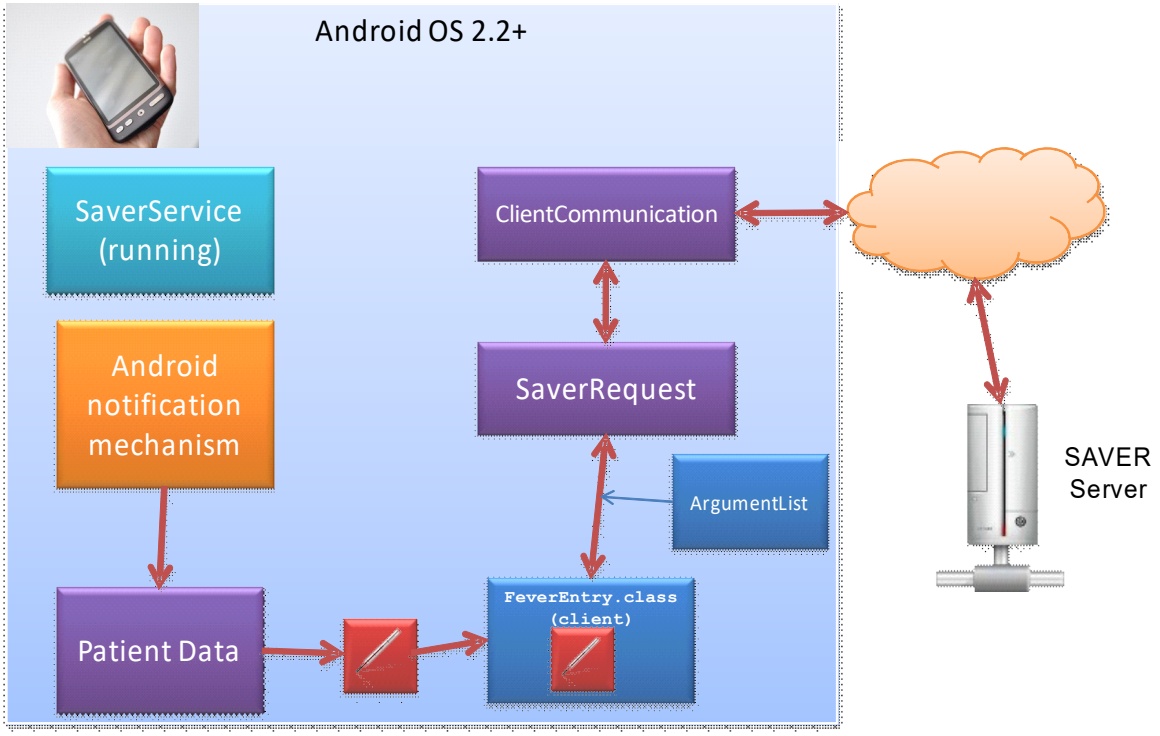
La communication proprement dite est implémentée à l'aide d'une API dédiée. La classe Java `ClientCommunication.java` permet à l'infrastructure SAVER de communiquer du client au serveur. Les divers modules composant l'application SAVER communiqueront en utilisant un paquetage spécifique à SAVER, qui permet de cacher les détails d'implémentation du protocole de communication, en particulier l'interfaçage avec SENSE. En principe, les modules applicatifs n'ont pas à se soucier de XML ou de SENSE, et se contentent d'échanger des couples nom/valeur entre module client et le pendant serveur.

Le protocole de communication peut gérer plus d'un module applicatif dans le cadre d'une seule et même requête / réponse, l'infrastructure ayant des besoins de communication en propre qui peuvent être avantageusement combinés dans une même requête, respectivement annexés à une réponse par le serveur lorsqu'il veut implémenter une annonce « spontanée ».

Les architectures côté client et côté serveur sont très semblables; les principales nuances

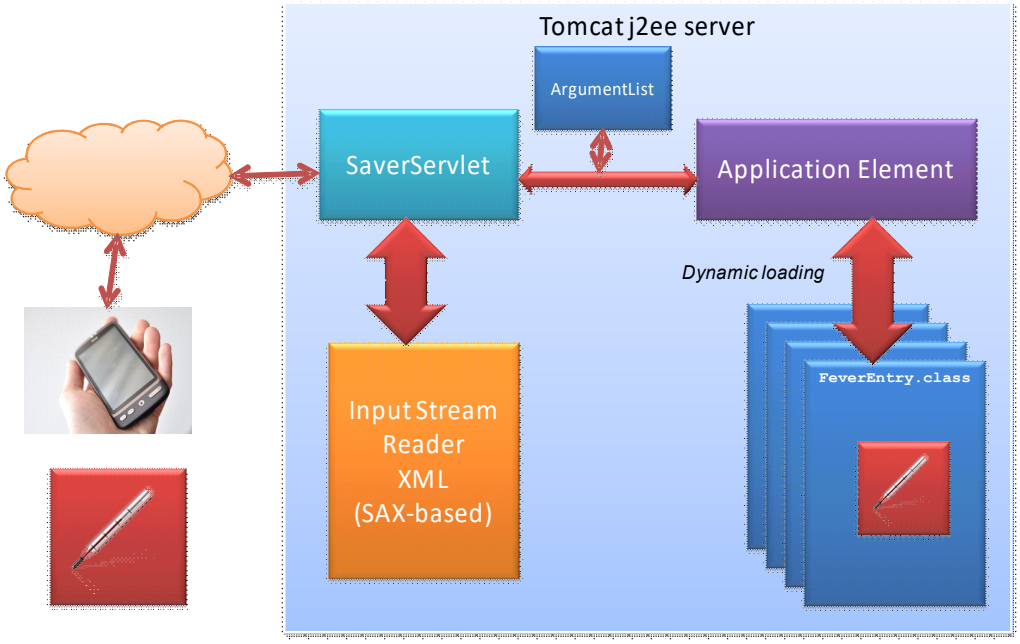
(hormis les dissymétries du protocole d'application) sont les suivantes :

- Le client n'émet en principe qu'une seule requête à la fois. Une requête du client ne concerne qu'une et une seule application. Cette restriction pourrait être abolie dans une future version, car elle ne dépend d'aucune limite technique : il se trouve que dans la version actuelle du client, cette possibilité n'a pas encore trouvé d'utilisation.
- La réponse du serveur peut concerner plusieurs éléments applicatifs dans le client.
- Le client et le serveur sont aussi tributaires de l'infrastructure SENSE ; celle-ci n'est pas explicitée plus avant dans les exemples ci-dessous. Sa présence est considérée comme implicite dans le texte qui suit. En particulier les paramètres d'authentification sont pris en charge par SENSE.



Cette asymétrie provient du mécanisme de communication utilisé par le serveur pour implémenter des communications asynchrones. On a voulu éviter – ou à tout le moins limiter au maximum- les connexions permanentes, sachant que certains systèmes de messagerie en abusent déjà plus que de raison (Apple iPhone); il est donc nécessaire de demander au client de se connecter périodiquement. A cet effet, un service de watchdog est prévu sur le client; de plus, on peut profiter des notifications volontaires effectuées par les utilisateurs pour communiquer en retour des informations de nature asynchrone. On a représenté ici le service SAVER sous forme de servlet, mais toute autre forme de réalisation est également possible.

Le service de notifications de SAVER peut être au besoin chaîné à une messagerie Web, voire à l'échange de SMS.



Au niveau applicatif, le principe est relativement simple : la communication s'établit entre entités homologues. Du côté du client, un `ClientApplicationElement` (interface Java) implémente la gestion d'un type de données sur le smartphone. Il possède un homologue appelé `ApplicationElement`, sur le serveur. Entre les deux sont échangées des OPDU (Object Protocol Data Unit), elles-mêmes convoyées, avec éventuellement d'autres unités OPDU, dans un conteneur appelé APDU.

On se référera aussi, pour les aspects programmatiques de la suggestion d'implémentation (non complètement testée) aux commentaires figurant dans l'en-tête de la classe `ApduManagement.java`.

7.1 Application Protocol Data Unit (APDU)

La communication utilise la couche de transport SENSE ; une unité de communication est appelée une APDU, et elle se compose des éléments suivants :

- Un en-tête, lui-même composé du préambule xml habituel, et porteur du tag <apdu>. A ce tag peuvent être associés des attributs globaux pour la PDU (localisation, conditions exceptionnelles, par exemple)
- Un corps formé de plusieurs éléments OPDU (voir plus loin)
- Un postambule (actuellement le simple tag </apdu>)

7.2 Object Protocol Data Unit (OPDU)

Une OPDU correspond à la communication d'un objet client avec son homologue serveur et vice-versa. Ainsi, prenons l'exemple du dialogue de début de session avec un patient:

l'objet implémentant le dialogue choix de patient met sur pied une OPDU contenant l'identification du tag repéré et indique que pour lui, il s'agit d'un début de session. (La résolution des conflits de tags patients est réalisée directement par le client). Cette OPDU aurait la structure suivante :

```
<opdu className="ch.iict.saver.server.patientSession">
     <param name="patientTag" value="12349876" />
     <param name="sessionBegin" value="true" />
</opdu>
```

L'attribut className indique au protocole l'activité à qui il faudra passer les paramètres patientTag et sessionBegin. SENSE est responsable de fournir l'authentification du soignant aussi bien au côté client qu'au côté serveur.

La classe patientSession peut être instanciée dynamiquement par la servlet réceptionnant cette OPDU. Cette classe pourra alors analyser les paramètres donnés et répondre avec une nouvelle OPDU ayant la forme suivante (toujours un exemple) :

```
<opdu className="ch.iict.saver.client.patientSessionBegin" />
     <param name="patientId" value="bbolomey" />
...etc...
</opdu>
```

Noter que les données des patients étant en principe pré-chargées (attribut *prefetch*, voir stockage) sur le smartphone, il n'y a que peu de données supplémentaires à transmettre. Le protocole pourrait ici , le cas échéant, ne transmettre que les modifications ayant eu lieu depuis la dernière session entre ce soignant et ce patient.

Une APDU peut contenir plusieurs OPDU ; une OPDU n'est pas nécessairement liée à une réponse de la part de l'extrémité éloignée. Ainsi, le service de notification d'annonces est asynchrone . Il transmet les alarmes à toutes les personnes concernées, mais il n'attend pas de réponse.

Un exemple d'APDU plus complet pourrait ainsi être :

```
<?xml version="1.0" encoding="utf-8"?>
<apdu>
```

```
    <opdu className="ch.iict.saver.server.documentQuery">
            <param                                          name="handler"
value="ch.iict.saver.client.RadiographicImageHandler" />
            <param name="patientTag" value="12349876" />
            <param name="docId" value="345678" />
    </opdu>
    <opdu className="ch.iict.saver.server.messageAck">
            <param name="msgId" value="234" />
            <param name="ack" value="true" />
    </opdu>
</apdu>
```

La réponse pourrait ressembler à :

```
<?xml version="1.0" encoding="utf-8"?>
<apdu>
    <opdu className="ch.iict.saver.client.documentQuery">
            <param                                          name="handler"
value="ch.iict.saver.client.RadiographicImageHandler" />
            <param name="patientTag" value="12349876" />
            <param name="docId" value="345678" />
            <param                    name="docUrl"                    value=
"http://saver/temp/data/radioPoumon.jpg" />
    </opdu>
    <opdu className="ch.iict.saver.client.messageCenter">
            <param name="msgId" value="235" />
            <param name="type" value="personnel" />
            <param name="title" value="Priere de passer au desk" />
    </opdu>
    <opdu className="ch.iict.saver.client.messageCenter">
            <param name="msgId" value="238" />
            <param name="type" value="indication" />
            <param name="title" value="Fermeture cafètéria" />
    </opdu>
    <opdu className="ch.iict.saver.client.messageCenter">
            <param name="msgId" value="239" />
            <param name="type" value="urgent" />
            <param name="title" value="Douleur excessive" />
    </opdu>
</apdu>
```

Une demande synchrone (qui attend une réponse) est toujours envoyée immédiatement par le client : donc une APDU est éventuellement crée pour cette OPDU particulière. Le terme synchrone est en réalité impropre, car il n'y a pas de comportement réellement synchrone ; mais comme on espère une réponse, cette OPDU sera envoyée immédiatement. En revanche, des OPDU asynchrones ne sont pas forcément envoyées immédiatement, mais mises en attente jusqu'à ce que :

- Une OPDU requérant une réponse doive être envoyée : les OPDUs asynchrones sont ajoutées à l'APDU convoyant l'OPDU requérant une réponse.
- Le service de watchdog du client décide qu'il faut générer une demande spontanée, éventuellement en créant une OPDU vide (de contrôle)
- Le nombre d'OPDU en attente dépasse une certaine limite.
- La stratégie momentanée de gestion des sessions implique que les envois sont immédiats (non délayés)

Le serveur n'a pas de moyen de forcer une transmission : il doit attendre au plus la génération d'une APDU de contrôle par le service de watchdog du client SAVER.

Il n'est possible d'envoyer qu'une OPDU requérant une réponse par APDU : ceci est dû au principe d'émission choisi. En conséquence, il faut éviter la transmission d'OPDU synchrones inutiles, car elles alourdissent le protocole et pénalisent la transmission.

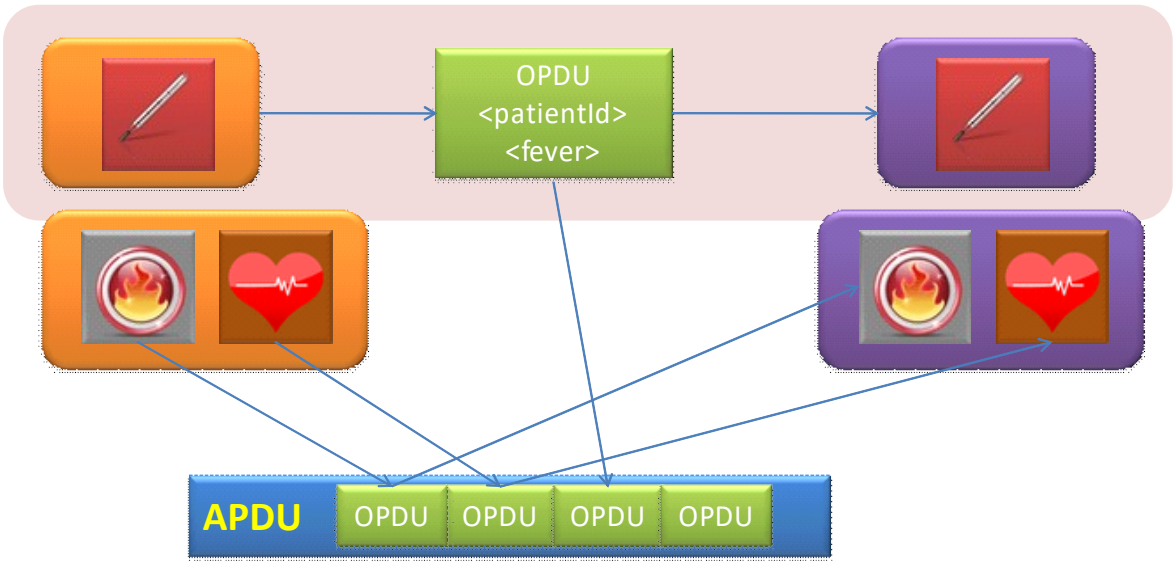
Prenons l'exemple de la prise de température du patient. Le dialogue sur le client (à gauche sur l'image ci-dessous, encadré en orange) est représenté par la classe FeverEntry.java, qui représente aussi une activité au sens Android du terme. Lorsque le soignant quittance une introduction de température, cette activité va transmettre les paramètres "identité du patient" et "température" vers le serveur de mobilité :

```
ok.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // The Fever class has still to be defined on the server :

        BasicOpdu bop = new BasicOpdu("ch.iict.saver.server.Fever");
        bop.addParam("patientId", FeverEntry.this.patientId);

        bop.addParam("fever", tv.getText());
        ApduManagement apdu = ApduManagement.getInstance();

        apdu.sendAsynchronousApdu(bop);
```

```
            act.finish();
        }

    });
```

On notera ici la mise sur pied d'une OPDU **asynchrone** pour la transmission, et l'invocation de la classe partenaire `ch.iict.saver.server.Fever` sur le serveur. Il est très important d'utiliser pour ce genre de requête un protocole asynchrone, afin de ne pas faire attendre le soignant sur un résultat qui n'est pas forcément significatif pour lui. Cette transmission asynchrone peut ne pas s'exécuter immédiatement : le moment va dépendre de la stratégie d'envoi par le gestionnaire d'APDU (`ch.iict.tota.framework.comm.ApduManagement`). Cette classe devrait en principe être à instanciation unique (singleton). On pourrait aussi imaginer un service Android, mais ce n'est probablement pas nécessaire de recourir à un service séparé.

Ainsi, il est possible, selon cette stratégie, que la situation suivante se présente au moment de l'envoi de l'APDU : le soignant a eu le temps, dans l'intervalle, d'acquérir des données concernant les paramètres de base du coeur, et une indication de ressenti de douleur.



SMARTPHONE (client) SERVER

Le composant serveur (ch.iict.saver.server.Fever) va, de son côté, mettre à jour les données du patient avec cette nouvelle information (parallèlement, des composants homologues mettront à jour l'information pour le coeur et la douleur). Mais ce n'est pas suffisant : les données du patient sont pré-chargées sur les mobiles des soignants. Il faut donc mettre à jour l'information dans l'ensemble des smartphones connectés (y compris celui d'où a été initiée la transaction de mise à jour).

Le principe de cette mise à jour est identique : l'objet qui s'est chargé de l'information sur le serveur (`ch.iict.saver.server.Fever`) va à son tour générer une OPDU, adressée à un service de mise à jour d'informations patient sur le client (par exemple

`ch.iict.saver.service.updatePatient`), et comprenant :

- Identification patient
- Nom du soignant, date et heure
- Action entreprise (Mesure de la température)
- Les informations additionnelles nécessaires pour permettre au client la génération d'un texte en clair dans l'historique des soins du patient.

Le client a, de surcroît, la possibilité d'indiquer un paramètre supplémentaire optionnel qui lui permet de spécifier un gestionnaire pour une éventuelle réponse. Cette possibilité est surtout utilisée dans le cas d'OPDU qui attendent une réponse:

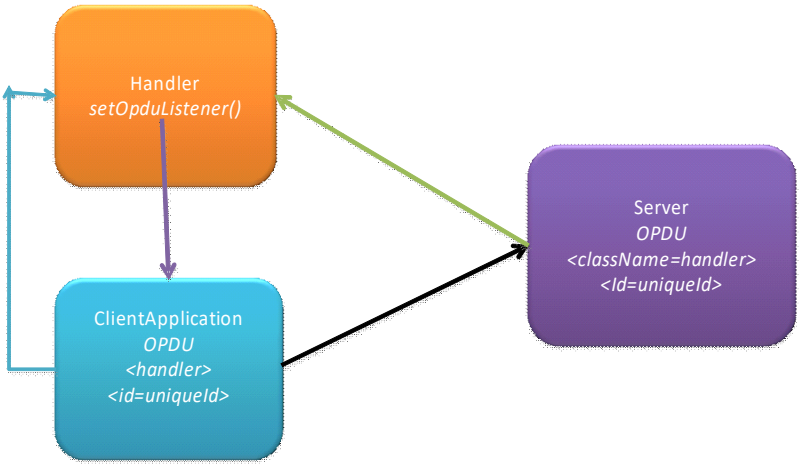
```
<param name="handler" value="ch.iict.saver.client.MyHandler" />
```

La spécification de ce paramètre aura pour effet :

- Le serveur va renvoyer une éventuelle réponse au gestionnaire défini par "handler"

Pour les recherches de documents, ou plus généralement pour l'accès à des informations qui ne sont pas préalablement téléchargées sur le mobile, l'accès se fait forcément de manière synchrone, avec une attente à la clé pour le soignant. Mais du point de vue du protocole, requête et réponse restent non liées l'une à l'autre, et tout se passe en réalisté de manière asynchrone, l'utilisateur recevant au besoin une notification par le service de notification Android.

Une telle OPDU (que nous appellerons synchrone bien que le terme soit impropre) se génère de manière similaire à une OPDU asynchrone, mais le requérant définit un gestionnaire de réponse en spécifiant les deux paramètres ci-dessus, qui auront pour effet de retourner le résultat à ce gestionnaire. Ce dernier sera implémenté sous forme d'un service générique, qui sera en mesure d'invoquer une méthode du client ayant généré l'opdu sur la base d'un `EventListener` lorsqu'il reçoit la réponse.



Les gestionnaires de réponses (OpduHandler) sont stockés dans un répertoire pour éviter une instanciation dynamique incessante des gestionnaires fréquemment utilisés. Voire la classe

`OpduHandlerManager` pour une possible implémentation.

7.3 Modification d'informations

Lors d'une tentative de modification d'information comme la température, la douleur ou les informations cardiaques, un OPDU est généré puis mis en attente sur la liste des d'OPDU sortants. Cette OPDU est généré uniquement si la valeur a été modifiée. Cette liste est envoyée après avoir atteint un certain nombre d'éléments (configuré dans SaverConfiguration.java) ou au prochain envoi d'un OPDU actif. Dans le pire des cas, l'OPDU sera transmis avec un délai maximal égal à l'intervalle de synchronisation (puisque que la synchronisation émet des OPDU actifs, l'OPDU en attente sera joint à celui-ci).

Voici un exemple d'OPDU permettant de fixer la douleur d'un patient :

```
<opdu className="ch.iict.saver.server.PatientPainSet">
      <param name="patientId" value="2" />
      <param name="pain" value="4" />
      <param name="type" value="lazy" />
      <param name="username" value="doctor1@integration.sense.com" />
</opdu>
```

7.4 Synchronisation

Le client demande à chaque synchronisation la liste des patients. Lors de la première synchronisation, il demande également la liste des alertes et des entrées du whiteboard. Lors des synchronisations suivantes, les alertes et les entrées du whiteboard ne sont pas demandées car les changements s'y rapportant (add – delete de la table « syncentry ») arriveront en réponse à n'importe quelle autre demande en tant que lazy OPDU.

Ajout d'une annonce au whiteboard du client :

```
<?xml version="1.0" encoding="utf-8"?>
<apdu>
      <opdu className="ch.iict.tota.NewsStore">
            <param name="summary" value="Souper du personnel" />
            <param name="content" value="Repas gratuit ce soir" />
            <param name="importance" value="1" />
            <param name="id" value="1500" />
            <param name="action"" value="add" />
      </opdu>
```

```
</apdu>
```

Suppression d'une annonce du whiteboard du client :

```
<?xml version="1.0" encoding="utf-8"?>
<apdu>
      <opdu className="ch.iict.tota.NewsStore">
            <param name="id" value="1500" />
            <param name="action"" value="delete" />
      </opdu>
</apdu>
```

Les messages pour les alertes sont identiques mis à part pour le className de l'OPDU qui sera « ch.iict.tota.MessageStore ».

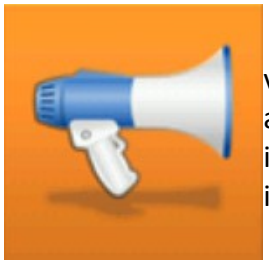
7.5 Annonces spontanées du serveur, et de sa périphérie

Lorsque le serveur veut communiquer à un, plusieurs ou tous les terminaux une information de manière spontanée, il devoir préparer cette information, et attendre que le ou les terminaux envoient une APDU pour pouvoir insérer dans la réponse une OPDU contenant l'information à transmettre. Ce délai n'est en principe pas critique, SAVER ne convoyant pas d'informations de caractère "life-critical", et que par ailleurs, les clients testent très régulièrement si des informations doivent être transmises ou non, pour ne rien dire des annonces faites spontanément par les clients.

CHAPITRE 8 Système de messages

Le système de message est étroitement lié au protocole défini au chapitre précédent. Deux mécanismes très voisins utilisent le même principe de communication :

Les **alarmes** ont un caractère médical, et relèvent de la logique de fonctionnement du service. C'est par ce canal que parviennent les indications relatives aux patients, ou au service des urgences, aux smartphones des soignants. Les indications et alarmes en provenance de Gyroflux ou du serveur SAVER parviennent aux soignants par ce canal.



Les indications du "*whiteboard*" (**annonces**) ont une structure très voisine, du point de vue informatique, des précédentes. Ce sont également des annonces, mais qui ne sont pas directement liées au service des urgences. Ces informations sont liées au fonctionnement général de l'hôpital, à des indications de service, à la publication d'*events*, carnets roses, etc...

Même si les structures de ces deux éléments de messagerie sont quasi identiques, plusieurs différences existent dans le comportement du système et des utilisateurs relativement à ces éléments :

- Une annonce n'a pas à être quittancée : elle se contente d'être affichée. Sa disparition n'est pas liée à un comportement de la part du personnel soignant, mais à une décision de la part de l'opérateur ayant le contrôle sur cette annonce.

- Par contraste, une alarme doit être quittancée, et cette opération peut être arbitrairement complexe :

  - Une annonce individuelle (***Nicole, peux-tu passer au desk, s'il te plaît?***) peut être quittancée par la personne à laquelle elle est destinée, ou par la personne ayant émis l'annonce (généralement l'opérateur de la console serveur).

  - Une **alarme** concernant un patient (***Le patient Cyprien Bolomey, box L, est en situation de douleur 6/10 depuis 25 minutes***) peut être quittancée par un soignant qui va s'occuper de monsieur Bolomey. La quittance doit être confirmée ensuite par l'ouverture d'une session avec le patient (pairage du DoctorMate de ce soignant et du PatientTag du patient correspondant) sans quoi l'alarme sera répétée après un certain délai.

- Les deux éléments participent à des éléments de dialogue (Activity) différents, n'ayant pas de relation entre eux. Les deux activités sont clairement séparées sur le dashboard, et devraient aussi présenter un "look and feel" suffisamment différent pour que les nuances de manipulation de l'information soient évidentes.

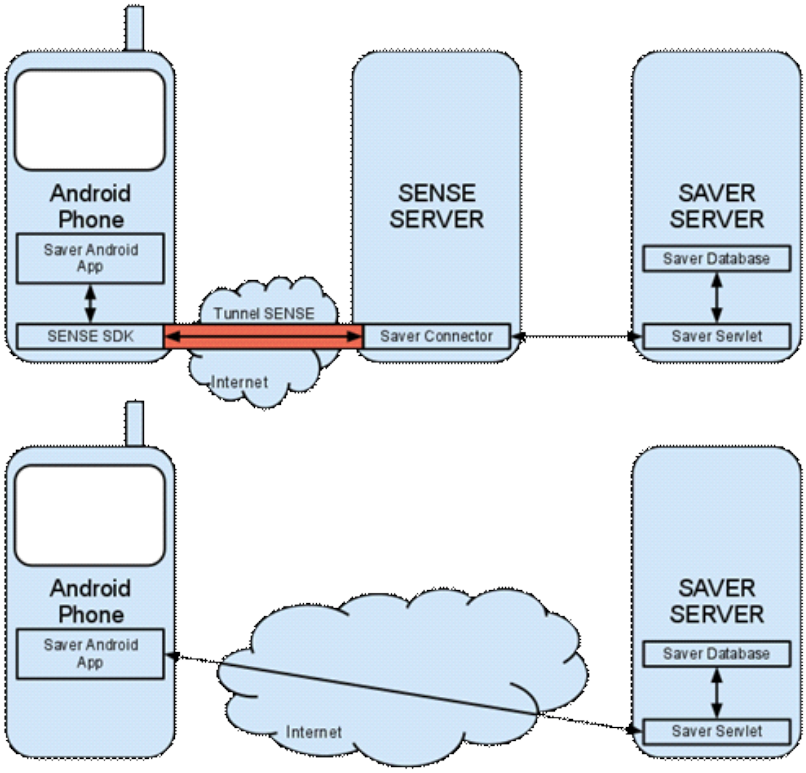
Les deux éléments sont en principe stockés dans la mémoire du smartphone, de manière à être immédiatement disponibles. **La quittance d'une alarme par un utilisateur n'a pas pour effet sa disparition du smartphone du soignant ayant effectué cette quittance**. Seul le serveur peut prendre l'initiative de faire disparaître une alarme, sur un terminal bien défini, ou sur tous les terminaux. La quittance va générer une OPDU asynchrone vers le serveur, et c'est ce dernier qui choisira de faire disparaître l'alarme des smartphones concernés par une OPDU appropriée. Cette disparition peut, selon la logique du serveur, être différée jusqu'à ce qu'un autre élément vienne confirmer la disparition effective de la cause de l'alarme en question.

Les alarmes peuvent avoir pour origine une opération de dialogue manuelle sur la console de l'opérateur, une décision appartenant à la logique de fonctionnement du service des urgences, ou un événement externe comme par exemple une annonce venant de Gyroflux.

CHAPITRE 9 Architecture serveur

L'architecture du serveur est étroitement liée à celle de SENSE pour l'infrastructure de transport et d'authentification. La flexibilité de SAVER repose sur la notion de connecteur logiciel, déjà utilisé par SENSE dans le cadre de l'accès au réseau sécurisé d'entreprise.

Il est possible d'utiliser le mode sécurisé (mode SENSE) ou non sécurisé (UNSECURE). La différence de structure est illustrée par la figure ci-après.



En mode standard, la communication entre le client et la servlet SAVER est protégée par le protocole SENSE. L'application Android envoie donc ses données non pas directement à la servlet SAVER, mais à SENSE. Ce dernier se charge ensuite de les transmettre de façon sécurisée à son serveur (et au connecteur SAVER). Le connecteur SAVER transmet la requête qu'il a reçue à la Servlet SAVER en http. Le Serveur SENSE agit donc comme un proxy. La réponse de la servlet suit le même principe.

L'authentification de l'utilisateur est gérée par SENSE. SENSE reçoit de la part de l'application Android un identifiant, un mot de passe et éventuellement un code d'enrôlement. L'application Android reçoit en retour une exception si l'authentification n'a pas eu lieu. L'application Android n'a pas besoin d'ajouter des paramètres d'identifications dans sa communication à SAVER puisque le serveur SENSE (via le connecteur) va s'en charger. En effet, celui-ci va parcourir chaque paquet (OPDU) qu'il reçoit, ajouter l'identifiant de l'utilisateur[4], et transmettre ce paquet complété au serveur SAVER. La communication entre le serveur SENSE et le serveur SAVER est considérée comme sûre. Les paquets ne peuvent ni être lu ni être altérés.

Dans le cas d'une connexion non sécurisée, il est nécessaire de simuler (avant d'envoyer les paquets sur le client ou juste après leur réception sur le serveur) l'ajout du paramètre « username » (qui est normalement ajouté par le connecteur SAVER (sur le serveur SENSE).

4En mode connexion directe, le client ajoute cette information puisque les données ne passent pas par le connecteur

9.1 Protocole

9.1.1 Protocole général

La communication est basée sur une connexion http à laquelle on ajoute du code XML en tant que paramètre POST. Ce code XML a pour balise racine un <apdu> (conteneur) qui peut contenir plusieurs <opdu> (paquets). Voici un exemple de ce code :

```
<?xml version="1.0" encoding="utf-8"?>
<apdu hardwareid="XXXXXXXXXXXXXXXXXXXX">
	<opdu className="ch.iict.saver.server.PatientTemperatureSet">
		<param name="patientId" value="2" />
		<param name="temperature" value="35" />
			<param name="type" value="lazy" />
		<param name="username" value="doctor1@integration.sense.com" />

	</opdu>
	<opdu className="ch.iict.saver.server.PatientPainSet">
		<param name="patientId" value="2" />
		<param name="pain" value="4" />
		<param name="type" value="lazy" />
		<param name="username" value="doctor1@integration.sense.com" />
	</opdu>
	<opdu className="ch.iict.saver.server.PatientHeartSet">
		<param name="patientId" value="2" />
		<param name="systole" value="7" />
		<param name="diastole" value="23" />
		<param name="pulse" value="91" />
		<param name="type" value="lazy" />
		<param name="username" value="doctor1@integration.sense.com" />
	</opdu>
</apdu>
```

Ce code (sortant du connecteur puisqu'il y a les username) contient les mises à jour des données médicales du patient portant l'id 2. On remarque le 1er OPDU est destiné à la partie « température » du serveur grâce à son argument « className ».

Un OPDU contient trois catégories paramètres : l'id du patient, une ou plusieurs valeurs

spécifiques et des valeurs de « contrôle » comme le type d'OPDU (s'il a été envoyé en tant que LAZY OPDU ou ACTIVE[5]). **Etant donné ce paramètre, il ne doit pas y avoir de paramètre nommé « type » dans les valeurs spécifiques. On utilisera, par exemple, « alerttype » pour informer le type d'alerte.**

On remarque dans le 3ème OPDU que celui-ci transmet trois paramètres (valeurs) en plus de l'id du patient et du type de message.

Un autre type de valeurs de contrôle que peut contenir un OPDU est le paramètre « handler ». Celui-ci informe le serveur qu'il y a une réponse à cette requête et la valeur permet de savoir quelle classe cliente devrait s'en occuper. A noter que dans le cas d'un OPDU actif, cette valeur est nécessaire mais ne sera utilisée que par le serveur pour savoir qu'il doit fournir une réponse. Le client quant à lui retournera la réponse à la fonction appelante (return).

```
<?xml version="1.0" encoding="utf-8"?>
<apdu hardwareid="XXXXXXXXXXXXXXXXXXXX">
    <opdu className="ch.iict.saver.server.PatientDetailsGet">
        <param name="handler" value="XXXXXX" />
        <param name="patientId" value="2" />
        <param name="type" value="active" />
    </opdu>
</apdu>
```

Dans le cas du code ci-dessus, le serveur constate qu'il y a un « handler », il sait donc qu'il y a une réponse à fournir. Il va donc créer un OPDU de réponse avec pour « className » XXXXXX. Le client recevant cet OPDU constatera que c'est un OPDU actif et il sera donc transmis par le « return » de la fonction.

9.1.2 Synchronisation alertes / whiteboard

Les alertes et le whiteboard ne sont pas récupérées intégralement lors des synchronisations pour éviter de surcharger le réseau. Une table « SyncEntry » a donc été ajoutée à la base de données pour gérer ce cas.

Champ	Description

5Une fonction émettant un OPDU actif attendra une réponse. La réponse à l'OPDU actif sera donc retournée (return) par la fonction d'envoi alors que OPDU lazy seront traités mais non renvoyés à la fonction appelante. Exemple : Je mets en attente (lazy) une demande de synchronisation (ce cas de figure n'existe pas dans l'état actuel du programme, la synchronisation étant active). Cette demande n'est pas transmise immédiatement. Je demande ensuite les détails de l'utilisateur (active) dans une fonction getDetail(). Le serveur va répondre avec les détails de l'utilisateur (active) que le client retournera (return) à la fonction appelante getDetail() et les informations de synchronisation (lazy) que le client traitera mais sans les retourner à getDetail().

ID	Identifiant de l'entrée de synchronisation
CAREGIVER_ID	Identifiant du soignant
ALERT_ID	Identifiant de l'alerte (null dans le cas d'un élément du whiteboard)
WHITEBOARDENTRY_ID	Identifiant de l'élément du whiteboard (null dans le cas d'une alerte)
ADDED	L'élément a-t-il déjà été envoyé au client ?
TODELETE	L'élément doit-il être supprimé du client ?

Lors de l'ajout d'une alerte ou annonce au système, une entrée est ajoutée à la table Alert ou WhiteboardEntry et une entrée **pour chaque** soignant est ajoutée dans cette table SyncEntry. Celle-ci aura la valeur ADDED à false et la valeur TODELETE à false également.

Lors de la suppression d'une alerte ou annonce, l'entrée est modifiée dans la table Alert ou WhiteboardEntry en passant le champ ACTIVE de true à false. Dans la table SyncEntry, l'entrée voit son ADDED passer à true (pour qu'elle ne soit plus retransmise) et TODELETE à true (pour envoyer l'ordre d'effacement) **pour tous** les soignants[6].

Voici une requête pour tester la synchronisation des alertes (afin de ne pas avoir à chercher l'id de l'alerte pour les deux requêtes suivantes, elle n'est pas ajoutée (INSERT) mais juste passée d'inactive à active) :

```
UPDATE alert SET ACTIVE = 1 WHERE ID = 1;
INSERT INTO syncentry (TODELETE, ADDED, ALERT_ID, CAREGIVER_ID) VALUES (0,0,1,2);
INSERT INTO syncentry (TODELETE, ADDED, ALERT_ID, CAREGIVER_ID) VALUES (0,0,1,3);
```

Lors de la première synchronisation d'une session utilisateur, le client va demander l'intégralité des alertes et des annonces. Le serveur va lui retourner toutes les alertes et annonces marquées actives dans leur table respective et passer tous les éléments de l'utilisateur dans la table SyncEntry à ADDED true (puisqu'il aura tout reçu). Si un élément était marqué TODELETE true, il est supprimé de la table SyncEntry (si il est marqué TODELETE true, il est obligatoirement marqué ACTIVE false et ne sera donc pas récupéré par cette synchronisation. On peut donc effacer l'élément de SyncEntry sans transmettre l'ordre au client).

Les synchronisations suivantes des alertes et des annonces sont faites en réponse à n'importe quelle demande de la part du client. Dans ce cas, le serveur parcourt toutes les entrées

6Partie importante pour la réalisation d'une interface de gestion des alertes et annonces

de SyncEntry pour l'utilisateur concerné. Pour chacune d'elle, deux cas sont intéressants :

- ADDED = false : L'entrée est passée à ADDED true et l'alerte ou l'annonce est transmise au client avec son détail.

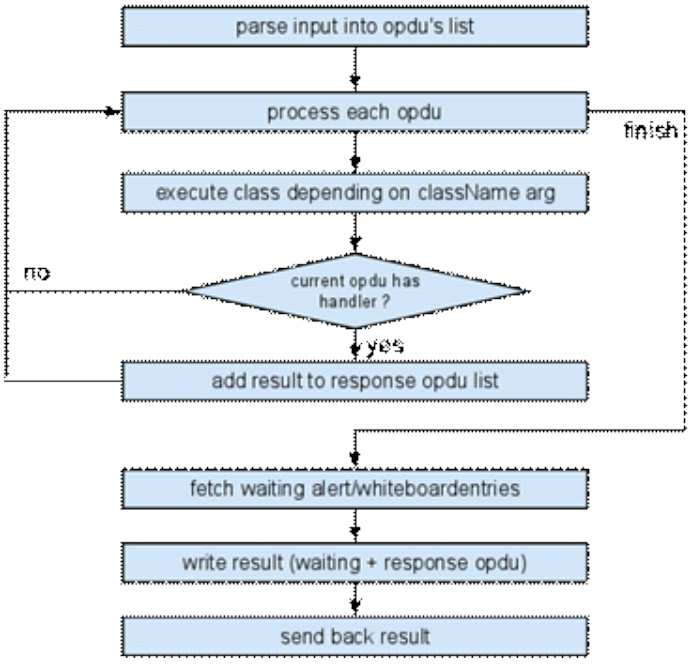
- TOREMOVE = true : L'entrée est supprimée de la table SyncEntry et un OPDU de suppression de l'alerte/annonce est transmis au client.

Lors d'une réponse d'un client **à une alerte uniquement**, si la réponse est positive, les éléments de SyncEntry sont marqués à TODELETE true pour cette alerte pour tous les médecins. Si la réponse est négative, l'élément de SyncEntry est marqué à TODELETE uniquement pour le médecin ayant émis cette réponse. Le serveur effectuant la synchronisation des alertes/annonces **après** le traitement de l'action (voir schéma fonctionnement serveur), le médecin recevra immédiatement en réponse à cette action un OPDU de suppression de l'alerte qu'il vient de valider.

Ce principe de synchronisation partielle peut également être appliqué aux autres éléments tels que les patients ou leur historique mais ne serait pas idéal. Il serait peut-être judicieux, dans ce cas, d'envisager la conservation dans la base de données des dates d'ajout, modification, suppression des éléments afin qu'un utilisateur puisse obtenir les changements depuis sa dernière demande.

9.1.3 Serveur

Le fonctionnement de la servlet serveur est décrit par l'image suivante :



Le serveur commence par recevoir une requête en POST, dont il lit le paramètre « request ». Ce paramètre POST contient le code XML de la requête à exécuter. Le serveur poursuit son travail en « parsant » le code XML pour en tirer une liste d'OPDU.

La servlet appelle ensuite pour chaque OPDU la méthode « processOpdu » afin de traiter les données entrantes. ProcessOpdu appelle la classe adaptée grâce à l'argument className de OPDU[7]. Si celle-ci retourne des résultats, elle possède une fonction getResults() retournant une liste de BasicOpdu. Ces résultats sont ajoutés à la liste des OPDUs à renvoyer au client.

Le programme continue en récupérant les alertes à ajouter / supprimer du client et en ajoutant ces éléments à la liste des réponses. Une réponse suivant le même protocole que la requête est transmise au client en parcourant la liste des OPDU à renvoyer.

9.2 Maven

9.2.1 Goal

```
clean install
```

9.2.2 Configuration

Dans les dépendances de Maven, il est nécessaire pour la partie servlet SAVER d'ajouter les paquets suivants :

```
<dependency>
    <groupId>commons-httpclient</groupId>
    <artifactId>commons-httpclient</artifactId>
    <version>3.1-rc1</version>
</dependency>
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-io</artifactId>
    <version>1.3.2</version>
</dependency>
<dependency>
    <groupId>servlets.com</groupId>
    <artifactId>cos</artifactId>
    <version>05Nov2002</version>
</dependency>
```

7Cette partie a été présentée dans les notes de développement comme partie à améliorer

9.3 Connecteur

Le connecteur est lancé sur le serveur SENSE pour recevoir et traiter les requêtes du client. Le SDK Android SENSE fournit une entrée du tunnel et la sortie est gérée par le connecteur. Celui se charge donc de recevoir les requêtes, y ajouter le nom de l'utilisateur (puisque c'est SENSE qui gère l'identification de l'utilisateur) et la retransmet au serveur SAVER. Il est doté de deux fonctions, une pour le transfert de requête SAVER (comme présenté plus haut) et une pour du transfert de fichier http. Il agit grossièrement comme un proxy qui sert d'intermédiaire entre le client et le serveur et qui « marque » les OPDU qui passent du client au serveur avec l'identifiant de l'utilisateur. Voici le fonctionnement du transfert de requêtes SAVER :

Figure 3 : Fonctionnement général connecteur

Quant au fonctionnement du transfert de ressources, celui-ci reçoit simplement l'adresse de la ressource à transfert, la télécharge et la transmet au client.

9.4 Maven

9.4.1 Goal

```
clean install
```

9.4.2 Configuration

Les dépendances suivantes sont nécessaires pour la compilation du connecteur :

```
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpclient</artifactId>
  <version>4.1.1</version>
  <scope>compile</scope>
</dependency>
```

9.5

CHAPITRE 10 Connexion à Gyroflux

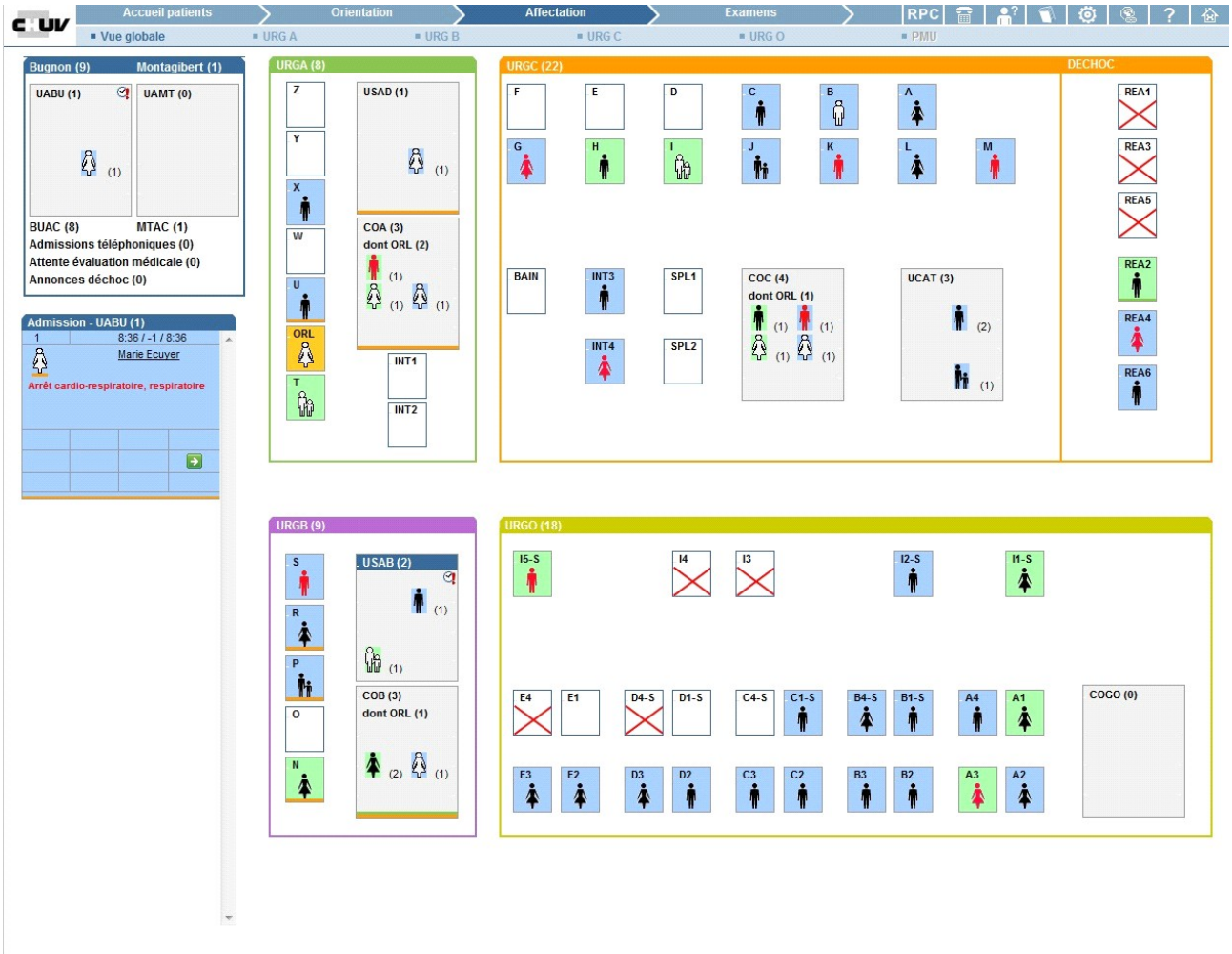
La connexion de SAVER au système d'informations des urgences du CHUV (application GYROFLUX) éait prévue à l'origine à l'aide de services WEB. Gyroflux étant une application Web, cela semblait pertinent, et reste probablement d'actualité. La liaison avec Gyroflux est réalisée par le biais de connecteurs dédiés, ainsi que décrit au chapitre précédent. SAVER n'est de ce fait que peu touché par l'application Gyroflux, puisque tout le travail d'adaptation va être effectué par le connecteur. SAVER n'est concerné que lorsqu'il s'agit d'implémenter une fonctionnalité qui n'est pas encore présente dans le logiciel. Ainsi, transmettre une alarme de Gyroflux à SAVER ne concerne que le connecteur. En revanche, vouloir restituer la carte du service des urgences du CHUV telle qu'elle existe dans Gyroflux sur les terminaux mobiles par le biais de SAVER nécessite l'introduction d'une nouvelle fonctionnalité.

Dans le cadre du présent projet, et essentiellement pour des raisons d'obtention des autorisations nécessaires, la connexion au service Gyroflux de production n'a pas été autorisé. En revanche, nous avons pu utiliser une installation de développement de Gyroflux (gyroflux-dev) qui a permis de vérifier de manière exhaustive la fonctionnalité de SAVER en conjonction avec Gyroflux.

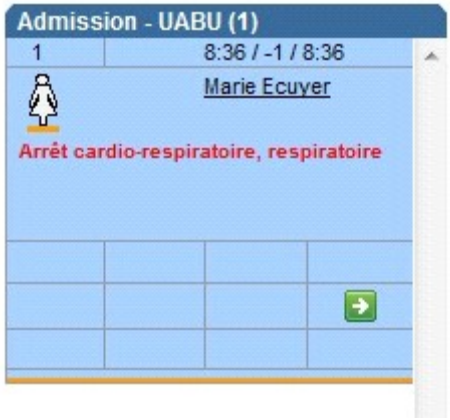
Gyroflux constitue une application de contrôle de flux, par opposition à l'infrastructure SOARIAN utilisée par le CHUV, qui constitue pour sa part un système de gestion de soins, ou plus précisément un système de gestion de données liées à la santé. SOARIAN ne prévoit pas de fonctionnalités de gestion de flux, c'est pourquoi l'application Gyroflux reste pertinente en dépit de l'acquisition du logiciel SOARIAN.

A terme, SAVER devrait également pouvoir se connecter à SOARIAN ; mais il s'agit là d'un détail de réalisation, plus conditionné par des restrictions sécuritaires et politiques que techniques.

Gyroflux se présente comme une application Web, dont l'écran principal a la configuration suivante :



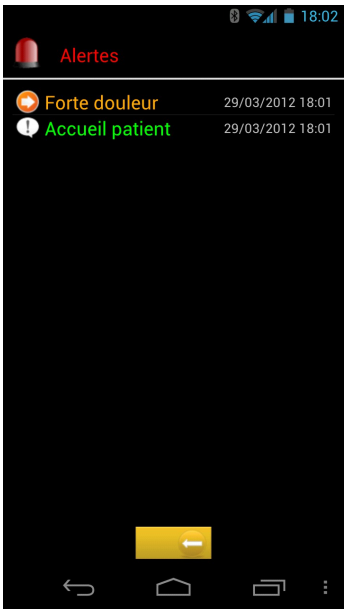
On a une représentation grossièrement topographique du service des urgences, depuis les patients en attente (ne haut à gauche), jusqu'aux patients déjà installés dans des boxes de soins. Dés l'admission, le patient est pris en charge par Gyroflux, et son état est traduit sur l'écran de visualisation ; ainsi, dans l'écran ci-dessus, le patient Marie Ecuyer est indiquée en alarme critique (arrêt cardio-respiratoire), alors même qu'elle n'est qu'en phase d'admission.



Le serveur de mobilité SAVER se connecte à l'application Gyroflux comme un utilisateur tout à fait normal, et navigue dans l'application à l'aide des hyperliens mis à disposition. Il analyse le code HTML et peut ainsi déceler les situations d'alarme et en évaluer le caractère critique. Si Gyroflux émet une alerte grave (indiquant qu'un patient a un besoin de soins urgent), cette alerte grave va être également interprétée par le serveur de mobilité de SAVER, et répercutée sur tous les terminaux mobiles du service (dans la mesure où le profil du soignant est pertinent).

Cette répercussion va se traduire par une vibration de l'appareil mobile, incitant ainsi le soignant à interrompre son activité en cours (dans la mesure du possible), et à consulter les alarmes.

Il pourra ensuite naviguer dans la liste des alarmes pour décider de l'intervention qu'il va effectuer.



On peut bien sûr également renseigner Gyroflux de la même manière ; et c'est probablement ce qu'il convient de faire dans un premier temps. Mais il faut être conscient que dans une optique de généralisation de l'usage de SAVER en milieu hospitalier, cette manière de faire est sans doute insuffisante, Gyroflux étant un outil spécifique à l'environnement des urgences. La notion de connecteur mentionnée plus haut prend ici tout son sens, les outils informatiques des divers établissements hospitaliers ne se caractérisant pas forcément par une normalisation poussée... Ainsi, l'application SAVER pourrait rester largement indépendante des divers sites où on souhaiterait l'installer, et seul les connecteurs au niveau du serveur de mobilité seraient touchés par une adaptation à un autre environnement hospitalier.

10.1 Conclusion

Le bilan de la connexion avec Gyroflux a démontré si besoin était l'extrême difficulté qu'il peut y avoir à tenter d'utiliser une infrastructure informatique fermée et sécurisée comme celle d'un grand établissement hospitalier universitaire. Obtenir l'autorisation d'installer un équipement "étranger" dans le réseau demande beaucoup de patience et de tact, en dépit de relations excellentes avec les responsables, tous convaincus de l'utilité du projet et du produit. Connecter ce même équipement à des unités en production relève de la plus pure science-fiction, quels que soient les appuis et l'extrême bonne volonté des personnes responsables (que l'équipe de développement de SAVER ne remerciera jamais assez!).

Par chance, le CHUV disposait d'une installation de développement du logiciel Gyroflux que nous avons pu utiliser pour nos essais, et nous avons ainsi pu valider le fonctionnement de SAVER dans une chaîne susceptible de passer un jour en production ; il est probable que pour déployer un tel logiciel dans un environnement comme celui du CHUV, hormis les considérations politiques que cela pourra impliquer, il faudra non seulement l'appui, mais aussi et surtout l'implication directe du

CHUV comme partenaire de développement et financier.

CHAPITRE 11 Système de stockage client

Le système de stockage côté client divise les ensembles de données en "*DataSets*", le client SAVER accède aux données en adressant un *dataset* (*voir aussi* `ch.iict.tota.framework.DataSet` *et classes dérivées*). Ce concept a le défaut d'une certaine lourdeur, mais dans le cas général, il permet de cataloguer les différents *datasets* en leur conférant des propriétés intéressantes. Un aspect intéressant dans cette optique est bien sûr la sécurité d'accès : on peut conférer à différents *datasets* des restrictions d'accès différentes ; mais dans le cas de SAVER, certains *datasets* sont définis avec un attribut de *prefetching*. Ainsi, les coordonnées des patients présents dans la salle des urgences sont téléchargées sur le mobile en début de session, et systématiquement tenues à jour tant que la session perdure. Cette mise à jour est transparente à l'application : elle se contente d'accéder aux données comme si elles étaient locales.

De manière similaire, un *dataset* peut être défini comme "externe". Il n'en reste pas moins accessible normalement par l'application, avec un éventuel délai dû à la transmission ; mais cette transmission reste transparente, du point de vue du code, à l'application. Si dans une implémentation donnée, on décide de changer le mode d'accès à ces données, l'application n'est pas touchée.

Un dataset a aussi une certaine durée de vie ; cette propriété n'est probablement pas pertinente dans le cadre de SAVER, mais permet à l'infrastructure SAFE d'implémenter des propriétés de sécurisation de données intéressantes.

La possibilité d'échanger des OPDU de manière transparente est bien évidemment utilisée pour implémenter les précautions de cohérence et de mise à jour automatique des données. Plusieurs autres paramètres reçoivent l'attribut "prefetch" : d'une manière générale, cela permet de grandement améliorer la réactivité du système. La décision de mettre un attribut "prefetch" à

un *dataset* ou non dépend de l'utilisation et du volume de ce dataset. Ainsi, une radiographie, volumineuse, plus rarement examinée, et susceptible d'être étudiée pendant un laps de temps relativement long ne sera probablement pas munie de cet attribut, alors que les imagettes (*thumbnails*) indexant les radio pourraient l'être, éventuellement.

CHAPITRE 12 Le cas pré-hospitalier

Les ambulanciers, sanitaires et secouristes doivent en principe transmettre des dossiers de prises en charge lorsqu'ils amènent un patient aux urgences. Or, ce dossier pourrait aisément être rendu sous forme électronique, établi pendant le transfert du patient, et simplement intégré au produit SAVER dés que les formulaires indispensables sont renseignés.

Pour diverses raisons pratiques, il ne semble pas très judicieux de permettre un accès en ligne par le personnel dit « pré-hospitalier ».

- Il faudrait munir les personnes concernées de smartphones, et s'assurer qu'ils ont toujours la bonne version du logiciel permettant de se connecter. Ceci n'est pas forcément un problème insoluble, une plate-forme comme Osmosys permet de garantir ceci.
- Il faudrait gérer ces accès externes de manière à garantir la sécurité de l'accès que l'on ouvre ainsi vers l'extérieur. Problème organisationnel, pas forcément soluble.
- On peut imaginer que le personnel pré-hospitalier puisse être en relation avec plus d'un hôpital, et que ces hôpitaux disposent de systèmes pas forcément compatibles...
- Authentifier les personnel pré-hospitalier constituerait probablement une tâche irréaliste pour le service informatique, d'autant que les secouristes ne sont pas affilés à un hôpital, mais à un organisme externe (REGA

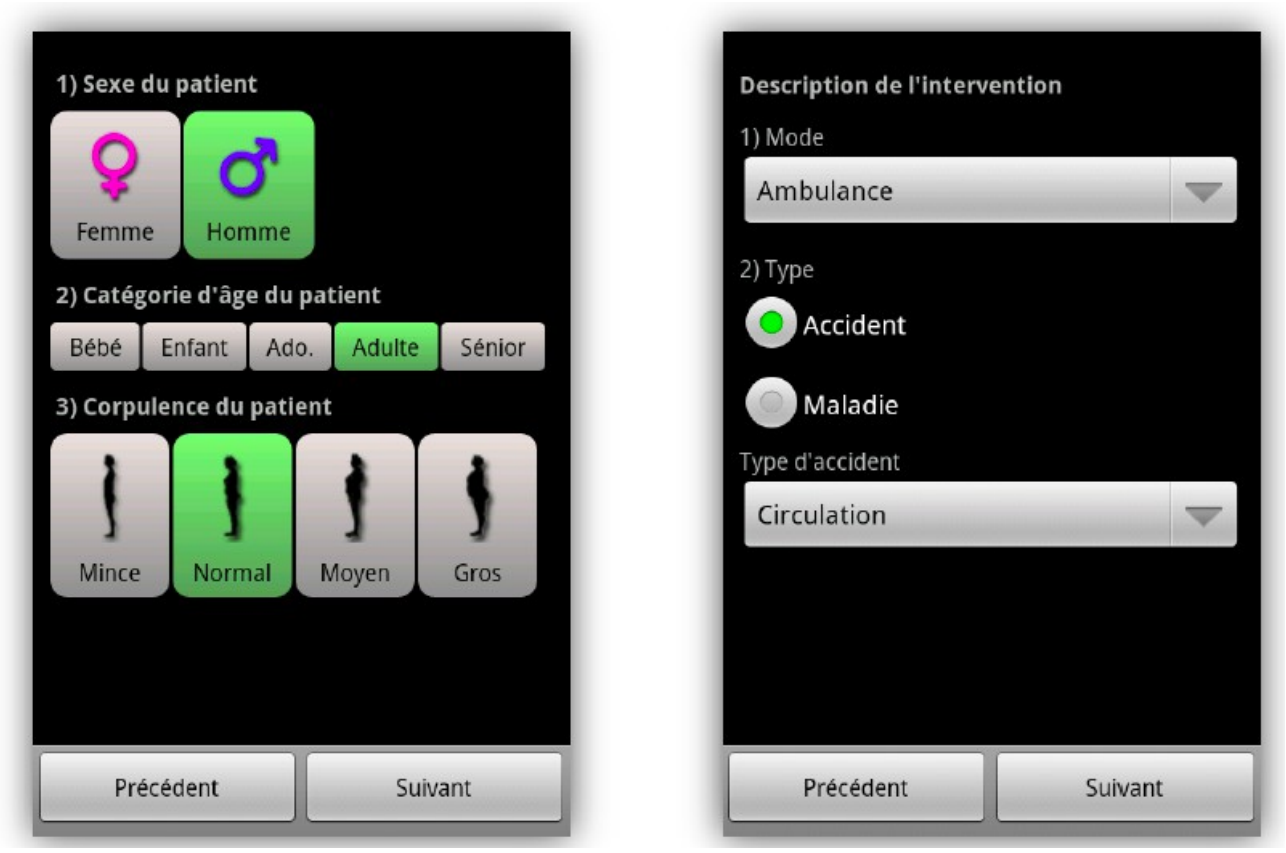
On a choisi de ce fait de ne pas intégrer le personnel pré-hospitalier à SAVER; mais pour être en mesure de néanmoins intégrer les données en provenance des ambulanciers, samaritains ou autres secouristes, on se propose d'implémenter la procédure suivante :

Un serveur séparé est mis sur pied, qui collecte les informations en provenance de toutes les sources pouvant amener des patients en situation de détresse au service des urgences. Ceci peut inclure aussi bien des secouristes, que des personnes d'un service samaritain, la police, ou encore les pompiers. Ce serveur distribue ensuite les cas d'urgence concernés en direction des hôpitaux concernés. Un hôpital ne "voit" donc pas le secouriste, mais uniquement le service de collecte des secours. Si un hôpital est contraint de refuser une urgence pour raisons de manque de place, par exemple, le cas est immédiatement géré par le service de collecte qui va rediriger l'intervention vers un autre hôpital.

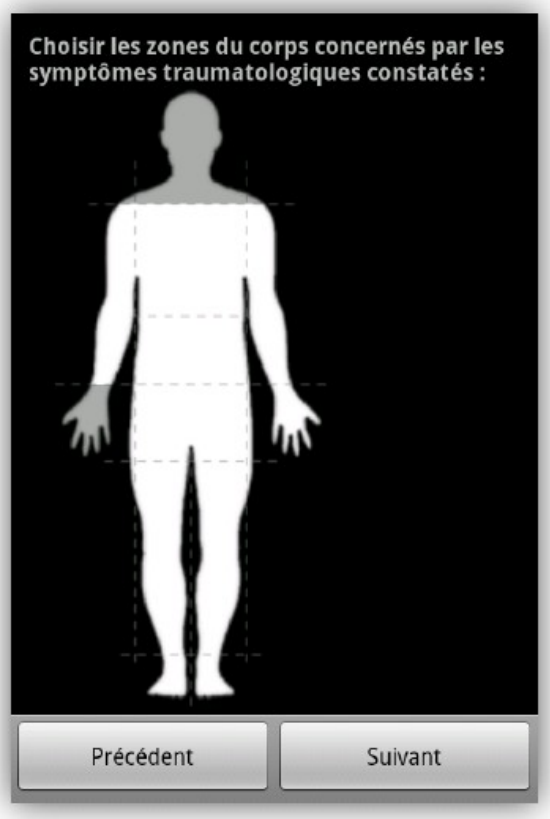
Un prototype d'un tel service a été mis sur pied dans le cadre d'un travail de diplôme du CPNV, ainsi que divers travaux de semestre du cours IFC2 à la HEIG-VD. Les documents concernant ces travaux sont disponibles sur demande auprès de l'auteur du présent document.

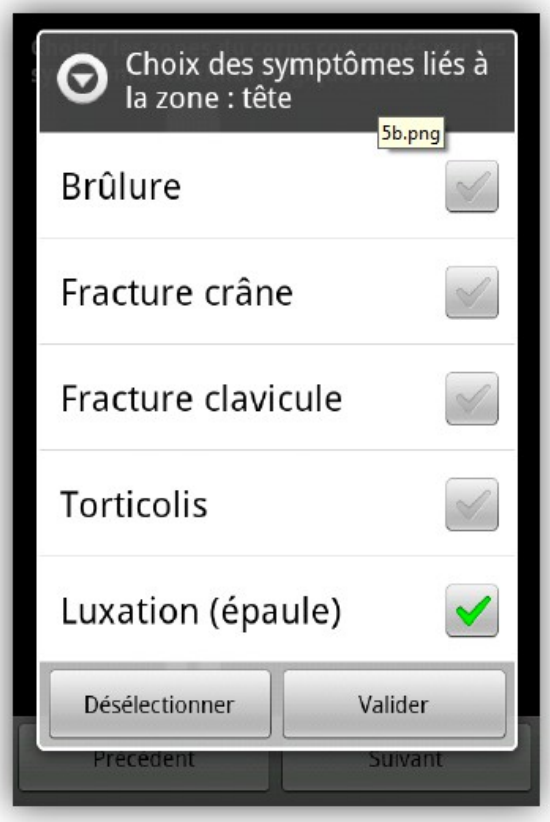
Les saisies d'écran suivantes montrent un exemple d'entrées possibles par le sauveteur sur un téléphone multifonctions sous Android 2.2.

12.1 Entrée des paramètres du patient



12.2 Documentation de la blessure





12.3 Type de symptômes

12.4 Le cas des données auxiliaires d'intervention

Dans le cadre de projets d'étudiants, il a été mis sur pied un "*Proof of Concept*" (POC) sur un client Android, ainsi qu'un serveur PHP permettant de collecter les annonces d'interventions. Ces deux composants ne sont mis à disposition qu'à titre d'exemple de ce qui peut être fait.

A ce titre, le POC permet de transmettre (associés au rapport d'intervention) des photos de l'intervention, des commentaires parlés, et même de petites séquences vidéo. Potentiellement, le concept permet d'envisager l'envoi d'autres types de données, comme des ECG mesurés *in situ* par des appareils portables équipés du protocole Bluetooth.

Ce POC n'est de loin pas à considérer comme un produit utilisable ; il ne s'agit que d'un modèle de démonstration permettant la critique et la vérification d'hypothèses ou d'options.

L'avantage déterminant de ce mode d'acquisition est qu'il permet la saisie de données hors ligne (en cas d'absence de disponibilité de connexion, par exemple), et la transmission des informations dés qu'une connexion devient possible.

CHAPITRE 13 Application patient

Le patient peut aussi bénéficier de l'infrastructure de SAVER ; lors de son entrée à l'hôpital, il peut télécharger une petite application sur son smartphone, et cette application va lui permettre d'entrer en contact avec SAVER, et partant, avec toutes données qui peuvent lui être utiles dans le périmètre de l'hôpital.

Cette application permettra aussi de se connecter sur les services propres de l'hôpital, et il pourra ainsi obtenir aisément les renseignements nécessaires (heures d'ouverture cafétéria, salle télé, cultes et messes, autres évènements sociaux).

13.1 Login

Pour simplifier au maximum l'utilisation, l'application ne comporte pas de login explicite et est immédiatement utilisable. Lors de la première utilisation, il est nécessaire d'introduire le numéro de séjour, et ceci peut se faire de plusieurs manières selon l'établissement où l'on se trouve :

13.1.1 Introduction explicite

A l'aide d'un menu de configuration, il est possible d'introduire le numéro de séjour UF (ou ce qui en tient lieu dans l'établissement concerné). Cette introduction peut êtr5e effectuée par le personnel soignant.

13.1.2 Introduction par code-barre

Dans certains cas, le patient peut être identifié par un code-barres ; là encore, le personnel soignant peut être d'un secours efficace, cette introduction n'étant pas forcément évidente pour tout un chacun.

13.1.3 Introduction par balise NFC

Pour les hôpitaux disposant d'une identification par balise NFC, il suffira de scanner la balise – bracelet pour réaliser l'identification.

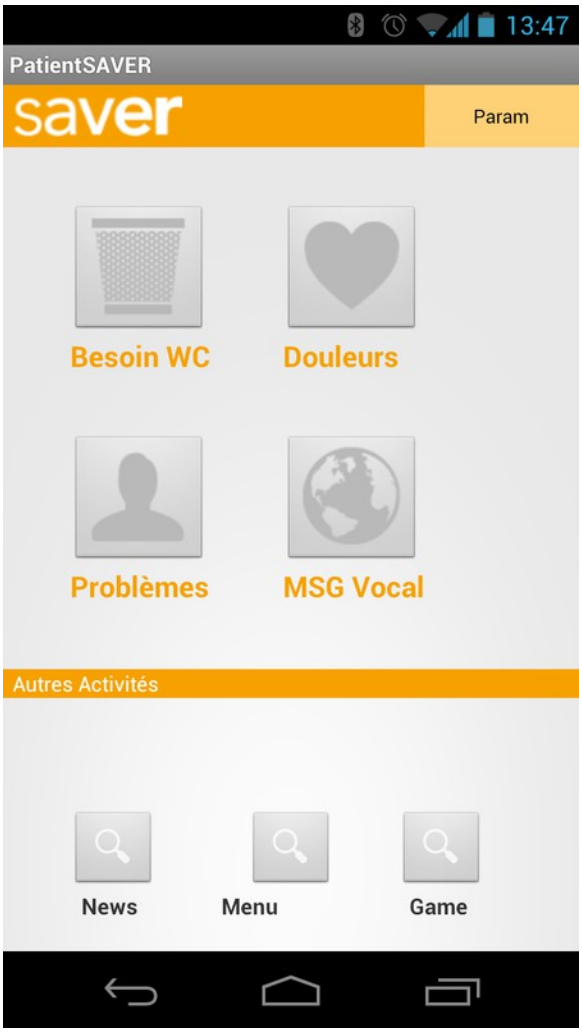
L'identification peut demander une confirmation par le personnel soignant ou administratif ; cette identification pourra également servir lors de la connexion au réseau WiFi public de certains établissements.

13.2 Fonctionnalités

Hormis l'accès simplifié au site web de l'hôpital, pour acquérir des renseignements d'ordre généraux sur le fonctionnement logistique de l'établissement, l'application client permet aussi d'implémenter une sonnette d'appel plus sophistiquée que son homologue traditionnel. L'écran principal permet ainsi de spécifier directement la raison pour laquelle on désire l'assistance du personnel soignant :

- Toilettes
- Besoin (soif, faim)
- Douleur, incident
- Problème thérapeutique (p. ex. perfusion terminée)
- Autres...

Cette spécification permet de mieux organiser le service dans certains cas, en évitant éventuellement des allées et venues intempestives.



Dans le même ordre d'idées, on pourrait imaginer que le patient puisse dans une certaine mesure renseigner le système de santé sur son propre état ; mais dans l'état actuel du projet, ceci dépasse le cadre de SAVER ; il est envisagé en revanche d'étudier plus avant le cas dans la suite de ce projet.

CHAPITRE 14 Utilisation de SAVER pour l'acquisition de données TARMED

La saisie en décentralisé de l'activité médicale et para médicale sur le plan tarifaire est un véritable problème dans le milieu hospitalier en particulier et de surcroît qui n'intéresse que très peu de personnes ; en revanche, il a un impact direct sur les revenus du service producteur, de l'institution générale et sur le revenu des médecins cadres payés partiellement comme salariés avec un complément basé sur l'acte produit.

En fait la facturation dépasse le système de tarification tarmed car elle repose sur de multiples structures tarifaires (entre 15 et 25 selon les établissements) associées à une multitudes de conventions en référence avec les partenaires payeurs cantonaux, fédéraux et internationaux, principalement pour le CHUV.

Tarmed est un tarif national composé d'environ 4800 références avec une structure hiérarchique bien définie (position principale et secondaire) à respecter. Dans ces références on trouve des positions en temps (ex. par période de 5 minutes) et forfaitaire (ex. le petit examen est de 15 minutes). Ces 2 positions par exemple, peuvent se cumuler sous certaines conditions tarifaires. De là le minutage ne devrait pas dépasser la durée réelle de la consultation. Les 600 catalogues de prestations utilisés dans le cadre du CHUV utilisent environ 3900 de ces références.

Il semble possible de saisir de manière simple les données TARMED depuis un smartphone. En principe, il n'y a rien qui s'oppose à la saisie de ces données, si ce n'est l'inconfort de devoir attribuer des codes d'intervention sur un écran de smartphone ou une tablette.

Le personnel soignant, pour sa part, effectue des actes médicaux (qu'il soit infirmier-ère ou

médecin) qui vont être traduits en une série de codes TARMED par la suite. Dés lors, il semble pertinent de se concentrer sur la saisie de ces actes médicaux plutôt que sur des codes plus ou moins ésotériques ; d'ailleurs, la méthode actuelle de saisie par formulaires papier correspond à cette idée. On se retrouve donc confronté au problème de définir des formulaires (que nous appellerons plutôt **catalogues de prestations** dans la suite de ce texte, orientés vers un domaine d'activité, pédiatrie, cardiologie, infirmier-ère, etc...) auxquels on peut ensuite associer diverses règles de conversion permettant de traduire un "acte" en une suite de codes arbitraire, dépendant d'unités de temps à prédéfinir ou à reconvertir en fonction des besoins..

Il a été d'emblée question d'acquérir, dans le cadre de SAVER, les données de temps de session actives des soignants pour en déduire des données TARMED, actuellement peu ou pas renseignées dans le cadre du service des urgences. Le temps de session n'est hélas pas suffisant pour recueillir les données nécessaires à la tarification, il faut encore saisir des codes (par exemple TARMED) pour catégoriser les interventions.

14.1 Cahier des charges pour un développement "POC" (Proof Of Concept"

L'approche a été pensée dans un cadre en principe indépendant de SAVER, mais de telle manière que le résultat puisse être intégré aisément, le cas échéant. Le cahier des charges ne reflète en conséquence pas de filiation particulière au projet SAVER.

L'architecture de la solution se présentera essentiellement sous la forme de deux composants, un client mobile et un serveur. L'essentiel de la réalisation va se concentrer sur le client mobile et le protocole de communication entre client et serveur.

14.1.1 Le client mobile

- Le **client mobile est associé au soignant** par une procédure d'authentification. Cette dernière peut être simplement une procédure classique de compte / mot de passe, ou une procédure plus sophistiquée, telle qu'utilisée par exemple dans le cadre du projet SAVER. Lors de la procédure d'authentification le soignant devrait identifier (explicitement ou implicitement) pour quel service il travaille, car un prestataire peut être amené à travailler pour plusieurs services. Le fait d'identifier le service pourrait amener à restreindre l'accès aux patients rattachés à un service donné et de donner accès uniquement aux catalogues dédiés à une activité particulière.

- Le **client mobile est en mesure d'identifier le patient**. Cette identification peut être effectuée de diverses façons. Il est possible d'introduire dans le formulaire d'acquisition les coordonnées du patient "à la main", ou de lire un code-barres tel qu'il est utilisé actuellement. Il est également possible de lire une balise NFC que le patient porterait au poignet en lieu et place du bracelet qu'on lui confère actuellement. On peut s'inspirer du mode utilisé dans le cadre de SAVER, où l'identification est pratiquement implicite, dés que le soignant se rapproche suffisamment du patient. Enfin, il peut aussi être possible de proposer une liste de patients en fonction du service dans lequel travaille le soignant, et de

permettre la sélection par le biais d'une liste déroulante. Dans la situation actuelle, l'étiquette reste le support unique pour identifier le patient. L'étiquette comporte le nom et prénom date de naissance et le n° de séjour UF (lieu de la responsabilité médicale) et le n° IPP. Le séjour UF est sous forme de code à barre. L'étiquette est en principe pour les cas d'hospitalisation au lit du malade alors que pour les cas ambulatoires l'étiquette est sur le dossier du patient ou le cardex. L'accès à l'identification du patient par balise ou code à la lecture du code barre semblent des solutions alternatives intéressantes.

- Le **client mobile dispose d'un ensemble de catalogues de prestations** (inspirés des actuels formulaires papier utilisés) qui sont adaptés au profil du soignant en question. Ces catalogues sont maintenus à jour de manière automatique, transparente pour le soignant. Une nouvelle version est automatiquement téléchargée depuis le serveur si elle devient disponible et publiée. Si le client mobile change de possesseur, les catalogues seront automatiquement adaptés lors de la procédure d'authentification du nouveau possesseur. Pour une meilleure répartition, les catalogues de prestations devraient être décomposés de la manière suivante : **Médicaments**, **Matériels** et **gestes médicaux ou infirmiers**. En effet vu les multiples mises à jours effectuées, -on pense ici plus particulièrement aux médicaments- il semble judicieux de répartir les catalogues par typologies de prestations.

- Le **client mobile acquiert par lui-même les paramètres de contexte,** dans la mesure du possible. Parmi ces paramètres, date et heure, ainsi que données d'identification du soignant et du patient (avec les restrictions que le mode d'acquisition de ces données peut comporter), du service utilisé (implicite dans le choix du catalogue rempli). Il peut ainsi indiquer dans quel environnement il va travailler. Prenons l'exemple du physiothérapeute qui selon un programme établi va le matin effectuer les différentes visites pour des patients hospitalisés dans le bâtiment du CHUV et l'après midi il sera à l'hôpital orthopédique pour dispenser des soins ambulatoires selon un file active dédiée.

- Les **dialogues** proposés par le client mobile sont très **simples**. En principe, il est possible de remplir un catalogue en n'utilisant qu'une seule main (la main "gauche" de préférence, idéalement la main droite pour un gaucher). En pratique, cette condition devrait être remplie dans tous les cas où une entrée alphanumérique n'est pas absolument indispensable (ainsi, une durée d'intervention doit être remplie par une liste déroulante, un compteur à incrémentation fixe ou un rouleau de sélection). Les dialogues d'acquisition ne se préoccupent en principe **aucunement** de codes TARMED, mais de **prestations**. Les prestataires ne connaissent pour ainsi dire pas les libellés Tarmed ainsi que la structure tarifaire. Sur les catalogues utilisés actuellement (papier) ce sont des libellés formatés dans le but d'apporter des précisions sur l'interprétation.

- La **transmission des résultats d'acquisition vers le serveur se fait dés que cela devient possible**. Le client peut continuer à effectuer des acquisitions même s'il n'est plus relié au serveur ; les résultats seront stockés localement et transmis vers le serveur dés que la transmission deviendra possible. Le client mobile maintient localement un protocole précis des acquisitions retransmises. Les acquisitions sont stockées dans le poste mobile pendant un temps T (paramètre de configuration), indépendamment du succès ou de l'échec de cette transmission. En cas d'échec, le transmission sera répétée dans la mesure où l'erreur

peut être attribuée à un problème de communication.

14.1.2 Le serveur

Le serveur se préoccupe de gérer l'ensemble des dispositifs d'acquisition mobiles, ainsi que les divers catalogues contenant les actes médicaux prédéfinis. Son rôle ne consiste pas à remplacer des bases de données ou des logiciels existants, ni à implémenter tout ou partie de la facturation. En revanche, l'un de ses rôles est de se porter garant de la validité des catalogues, et de la cohérence des prestations documentées. Une saisie peut comporter des incohérences (valeurs trop élevées, incohérentes tarifaires, prestation par cohérence avec le sexe du patient, etc.). Ces erreurs seront protocolées, et classées par types ; dans la mesure du possible, on cherchera à résoudre les cas les plus simples de manière automatique ; mais lorsque la solution s'avère impossible par la simple utilisation d'un moteur de règles, ou d'une algorithmique à but comparable, ces erreurs seront redirigées vers des postes de traitement d'erreurs spécialisés dans la catégorie d'erreur incriminée.

- Le **serveur propose une infrastructure simple et facilement utilisable pour générer les catalogues** qui seront utilisés par les postes mobiles. La génération de catalogues est effectuée par des personnes ayant un rôle dédié à cette mise en œuvre. La génération peut se faire "en ligne" (depuis un navigateur), ou par le biais d'un programme dédié (voir par exemple l'utilitaire EditQuest[8] utilisé dans le cas des projets iminet et plus récemment Quest ou DCS). Le développement de cette infrastructure ne sera probablement pas priorisé dans un premier temps.

- Le **serveur permet de valider un questionnaire généré**, et ainsi de le **publier** à l'intention des terminaux mobiles. Dans cette optique, il est capable de gérer les diverses versions des catalogues, de publier de nouveaux questionnaires, et d'attribuer chacun de ces questionnaires à un ensemble de rôles donné (par exemple, infirmier-ère, médecin spécialiste en..., etc...). On peut s'inspirer (mais pas forcément imiter!) de ce qui a été fait dans le cas du serveur MediQuest[9] dans le cadre de cette gestion.

- Le **serveur est en mesure de gérer les postes mobiles** ainsi que leurs **utilisateurs**. Ceci implique que l'authentification de l'utilisateur associe le mobile et l'utilisateur, et charge au besoin les catalogues nécessaires dans la mémoire du poste mobile pour qu'il soient immédiatement disponibles. Il n'est en principe pas prévu de s'échanger des postes mobiles entre utilisateurs sans une nouvelle procédure d'authentification ; une telle possibilité serait toutefois envisageable dans le cas du projet SAVER, grâce au dispositif du DoctorMate.

- Le **serveur met à jour les catalogues par un protocole "Over The Air"** ou OTA. De même, le poste mobile transmet les résultats de l'acquisition par un protocole OTA également. Les deux protocoles sont des protocoles de type texte (basé XML). Les protocoles sont

8 *EditQuest est un utilitaire de génération de questionnaires développé à la HEIG-VD dans le cadre du projet iminet (voir aussi http://useraware.iict.ch). Cet utilitaire a été "recyclé" dans le cadre du projet Quest.*

9 *MediQuest (dénomination non définitive) est le nom du serveur du projet Quest, développement réalisé à la HEIG-VD en collaboration avec la Polyclinique Médicale Universitaire (Dr Nicolas Senn) et permettant l'acquisition en situation de grande mobilité de données par questionnaires dans le cadre de la médecine de première instance. L'implémentation actuelle (septembre 2011) est à considérer comme fonctionnelle, mais non déployable.*

sécurisés par une liaison SSL (HTTPS-TLS). Dans l'état actuel, il n'est pas prévu de mettre à jour l'application du client mobile par un protocole OTA, même si cela ne pose pas de problème technique particulier[10].

- Le **serveur comptabilise les actes documentés par les soignants**. En cas d'absence d'actes en provenance d'un soignant authentifié, il va constater une situation d'anomalie, et la documenter comme telle.

- Le **serveur convertit les résultats d'acquisition en une liste de codes TARMED** selon les règles définies lors de la constitution du catalogue. Les résultats d'acquisition sont des actes médicaux, et le serveur va leur substituer une liste de codes définis lors de la mise en place du catalogue. **Note** : il est probablement plus pertinent de n'effectuer cette conversion qu'au moment où une exportation est requise (voir paragraphe suivant), et de ne stocker que des actes médicaux en base de données.

- Le serveur permet, sur demande d'un système externe (par exemple service de facturation), une **exportation des codes TARMED générés en format XML**. Ces résultats ne constituent pas des données de facturation, mais devraient potentiellement pouvoir être convertis en données de facturation par le logiciel adéquat.

14.2 Protocole de communication

Le protocole de communication doit remplir un certain nombre de conditions dans la situation particulière où se trouve le personnel soignant.

- Le protocole doit être en mesure de supporter le travail de mise à jour des catalogues de prestations, et ceci dans les cas suivants :
  - Mise à disposition d'une nouvelle version d'un catalogue de prestations existant.
  - Mise à disposition d'un nouveau catalogue de prestations
  - Changement du possesseur du dispositif mobile, le nouveau possesseur a une fonction différente du précédent
  - Le possesseur n'a pas changé, mais il travaille momentanément dans un autre service, où il fournit d'autres prestations.

- Le protocole doit permettre au client mobile de transmettre les renseignements de prestations vers le serveur. Cette transmission doit être sécurisée (TLS semble suffisant), et doit être confirmée correcte. Les cas d'échecs de transmission doivent être correctement documentés. Au besoin, le protocole supporte la retransmission automatique.

- Le protocole permet la gestion de la transmission aussi en l'absence de liaison. Dans ce cas,

10 *Dans un environnement de production, cette problématique doit être résolue, avec les problèmes de sécurité qu'elle implique.*

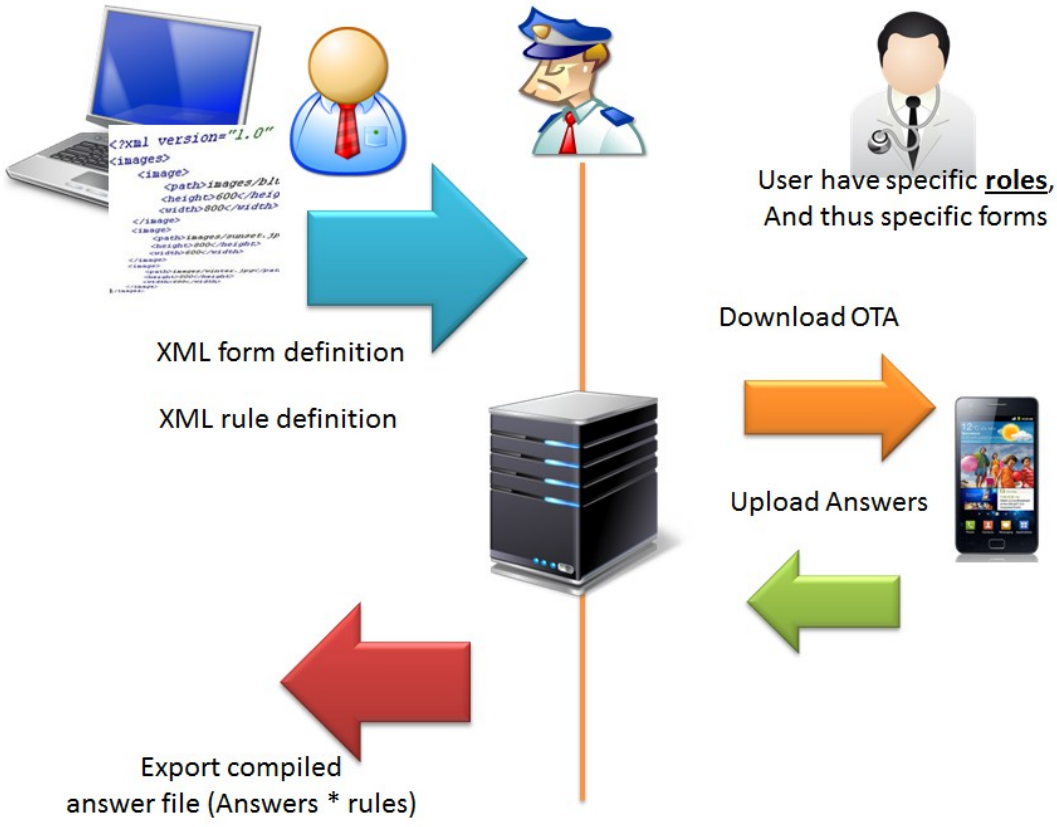
l'interface d'accès au protocole va simplement délayer la transmission jusqu'à un moment où cette dernière devient possible. Le délai maximum tolérable est un paramètre de configuration du protocole. Cette fonctionnalité est transparente à l'application.

- La configuration des paramètres du protocole se fait uniquement au niveau du serveur. Seule exception : l'adresse du serveur elle-même qui doit être configurée sur le poste mobile. On peut imaginer en revanche une manière particulièrement élégante d'implémenter la procédure d'installation qui permettrait de personnaliser la distribution si bien que le paramètre de configuration en question soit implicitement stocké sur le smartphone à la mise en service.

- Le protocole ne supporte pas l'installation OTA. Ce mécanisme est laissé à l'implémentation du smartphone sur lequel l'application est installée, ainsi qu'à la politique de sécurité qu'entend mener l'hôpital.

14.3 Objectifs à court terme

Divers travaux d'étudiants vont être concernés par cette problématique dans le courant de l'automne 2011, dont l'objectif premier est de démontrer la faisabilité et la pertinence d'une telle solution d'acquisition, de manière à ce que les principaux intéressés soient en mesure de l'évaluer et la critiquer, et que les résultats de cette évaluation puissent induire le développement ultérieur d'un produit plus élaboré.



Dans cette optique, les composantes importantes sont :

- La possibilité de documenter des prestations de manière confortable et intuitive sur le poste mobile, en partant d'un catalogue de prestations codé -provisoirement à la main- en XML, et résidant sur le smartphone (carte SD, par exemple).
- La possibilité d'acquérir les données environnementales, ainsi que les données du soignant et du patient pour les adjoindre au catalogue de prestations effectuées. Les données du patient vont poser de multiples interrogations : il serait avantageux de prévoir d'emblée des entrées multimodales pour ce genre d'informations :
  - Bar code (scanner de code-barres, peu pratique, mais évident)
  - Puce NFC (solution pour le futur?)
  - Entrée explicite clavier (le serveur peut ici proposer une aide précieuse, en fournissant une liste des patients dans le service considéré, et en permettant le choix dans une liste déroulante)
  - La solution SAVER en détection de proximité

  - Entrée vocale

- La possibilité de transmettre les prestations documentées de manière naturelle et transparente pour l'utilisateur vers un serveur de collecte

- La possibilité de procéder à une extraction de ces prestations depuis le serveur. Il n'est pas évident que la conversion en codes tarmed lors de l'extraction soit à prioriser absolument dans cette démarche.

14.4 Pistes pour la classification des catalogues

Ce chapitre entend énumérer quelques manières de réduire autant que faire se peut le nombre de catalogues de prestations pour proposer à un utilisateur donné un choix raisonnablement pertinent pour les activités qu'il remplit à un instant donné. En même temps, il veut proposer quelques pistes permettant de mettre sur pied une interface utilisateur aussi homogène et simple que possible.

Les interfaces utilisateur sont en principe toujours bâties sur un modèle unique, et les réactions du système sont largement homogènes. Un choix est présenté sous forme de liste, chaque élément de la liste est composé d'un libellé[11], et d'un bouton sur lequel figure une icône symbolisant l'élément en question. Chaque libellé utilise une couleur particulière de fond. Une action sur le libellé entraîne la sélection de la catégorie ou de l'action désignée par ce libellé. Une action sur le bouton accompagnant le libellé entraîne l'affichage d'un texte explicatif explicitant la signification du libellé en question.

Une activation du bouton amène à l'écran un texte explicatif qui pourra être quittancé après lecture.

- Les textes explicatifs doivent-ils être proposés en plusieurs langues, même si l'interface utilisateur est uniformisé sur la langue "standard" de l'hôpital ? (On peut imaginer proposer le texte dans la langue "standard" par défaut, mais de proposer un ou deux petits drapeaux pour choisir l'anglais ou l'allemand, le cas échéant).

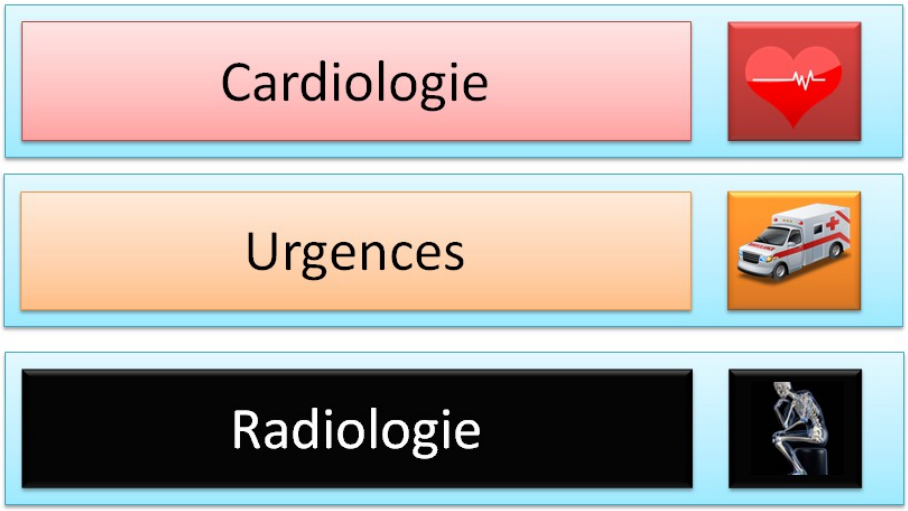
Ces interfaces utilisateur ne sont en aucun cas figés sur le client. Ils sont définis dynamiquement par le serveur, qui peut ajouter ou retrancher des catégories en tout temps ; seule la logique de fonctionnement de l'interface utilisateur est figée dans le terminal mobile, pas les

11 *Les libellés figurant dans le présent document en guise d'exemples sont bien sûr fantaisiste.*

catégories de choix proposés.

14.4.1 Classification par services

Lorsque le soignant s'identifie (implicitement ou explicitement), le système est en mesure de savoir quels sont les services avec lesquels il est susceptible de travailler. Il va en conséquence demander au soignant dans quel service (parmi ceux auxquels il est susceptible d'être rattaché) il doit être rattaché (on pourrait imaginer de consulter son emploi du temps, mais c'est probablement trop restrictif du point de vue de la flexibilité).



Ce renseignement permet de limiter l'éventail de catalogues de prestations au seul sous-ensemble nécessaire dans le service considéré.

- Est-il prudent de prévoir une "échappatoire", qui permettrait d'aller quérir un catalogue non prévu par la logique de répartition des catalogues entre les divers services ?
- Y a-t-il des catégories de personnel qui sont susceptibles de travailler dans plusieurs services à la fois (simultanément!) ?

14.4.2 Classification par responsabilité du soignant

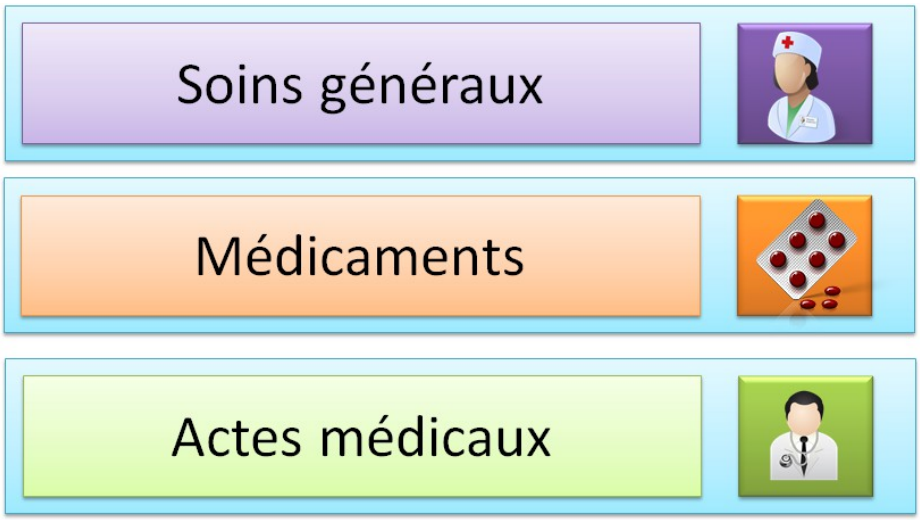
Cette classification est liée à l'identification du soignant et ne requiert en principe pas d'introduction supplémentaire de la part du soignant. Un médecin a des responsabilités différentes de celles d'un infirmier-ère, donc les catalogues de prestations sont différents.

- Est-il prudent de prévoir une "échappatoire", qui permettrait d'aller quérir un catalogue non prévu par la logique de répartition des catalogues entre les diverses catégories de soignants ?
- Les responsabilités sont-elles figées ? Par exemple, une infirmière-chef peut-elle fonctionner comme aide-infirmière au besoin (ce qui impliquerait un cumul des catalogues de prestations, je suppose) ?
- Dans certains cas, un acte thérapeutique peut-il être partagé ? Deux infirmiers qui

changent ensemble le lit d'un patient ne vont pas faire la même quantité de travail que s'ils avaient été seuls. Comment fait-on la distinction, en particulier si l'on songe au point précédent.

14.4.3 Classification par catégorie de prestations

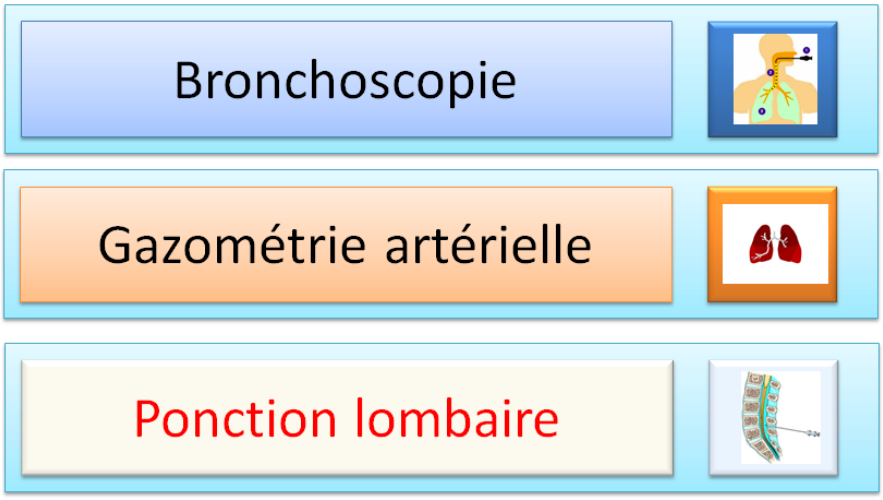
Pour limiter encore davantage l'éventail des catalogues de prestations, on peut subdiviser encore l'ensemble en catégories de prestations, dans la mesure où cette subdivision est pertinente eu égard au rôle du soignant et au nombre de prestations possible.



- Après toutes ces divisions, à quel sous-ensemble de prestations sera confronté l'utilisateur ? En particulier, quel sera le volume de catalogues parmi lequel il aura à choisir ?

14.5 Les catalogues de prestations

La présentation des catalogues de prestations proprement dits correspond à la police générale de l'interface utilisateur ; là encore, comme pour toutes les listes précédentes, les contenus des listes sont définis au niveau du serveur, donc aisément ajustables.



Après le choix de la prestation peuvent intervenir l'introduction de paramètres auxiliaires dépendant du type de prestation (nombre d'unités de temps, intervention gauche-droite, etc...).

14.6 Pistes pour l'identification du patient

On l'a vu précédemment, il y a plusieurs pistes pour permettre une identification du patient, à commencer par celle qui a été implémentée dans le cadre du projet SAVER, basée sur des balises actives (PatientTag et DoctorMate). Mais dans le cadre de tarmed, l'utilisation potentielle du système dépasse de loin le cadre du service des urgences, et peut aussi sortir du cadre simplement hospitalier pour une utilisation dans des cliniques, des cabinets médicaux de groupe, ou des établissements médico-sociaux.

Il faut donc garantir à la fois la compatibilité avec SAVER (ne serait-ce que pour la cohérence du projet), et la généricité de l'authentification du patient, en gardant à l'esprit qu'il s'agit d'une opération complexe, et sujette à erreur de manipulation. On part également du principe que l'organe d'entrée-sortie le plus communément utilisé va rester le smartphone ou une tablette de petite dimension, en gardant à l'esprit que le plus grand confort d'utilisation de la tablette se paie par le transport d'un outil supplémentaire (le téléphone reste indispensable), et que sa manipulation occupe généralement les deux mains. De plus, une tablette de 10 pouces ne tient pas dans une poche de blouse (7 pouces semble le maximum) ; donc il est nécessaire de la déposer quelque part dés que l'on désire se livrer à un acte thérapeutique quelconque, d'où risque d'oubli, de perte, ou de chute.

Idéalement, le renseignement du nom du patient[12] devrait pouvoir être effectué de manière multimodale. Les modes suivants de renseignement sont suggérés ici, et devraient pouvoir être utilisés indifféremment, selon la situation. Il va de soi que dans tous les cas, le smartphone possède en local une liste partielle ou complète des patients en séjour dans le service courant, et peut de ce fait compléter intelligemment les entrées (fonction d'auto-complétion, déjà présente dans la fonctionnalité de SAVER) :

14.6.1 Entrée explicite

C'est le mode d'introduction le plus évident. Un clavier (réel ou virtuel) permet d'introduire en toutes lettres le patronyme du patient, ou une identification numérique. Cette introduction peut être grandement facilitée par un serveur qui connaît les personnes hospitalisées dans un service donné, et peut ainsi les proposer sous forme de liste au soignant (dans la mesure où cette liste ne devient pas trop longue, bien sûr!). Si l'on dispose de la localisation par WiFi, on peut aussi imaginer restreindre la liste aux personnes présentes dans une salle donnée ; mais cette restriction est probablement peu pertinente : comment administrer un soin (remplacement de perfusion, prise de tension, etc...) à une personne qui s'est déplacée pour visionner un match à la télévision ?

12 *Le "nom" du patient est formé de plusieurs paramètres, qui peuvent varier en fonction des hôpitaux. Dans le cadre du CHUV, l'identification du patient est formée du nom et du prénom, de la date de naissance, du n° de séjour UF (lieu de la responsabilité médicale) et le n° IPP (Identifiant Permanent du Patient). Le séjour UF est actuellement sous forme de code à barre. Dans le cadre de ce document, le terme "identification" désigne un objet permettant univoquement de déterminer l'identité d'un patient, et par extension, tous les paramètres nécessaires à son identification au sens médical et/ou juridique du terme.*

14.6.2 Entrée par code-barres

L'introduction de codes-barres peut être intéressante dans la mesure où ces codes sont déjà largement utilisés sur les plaquettes d'identification des patients (numéro de séjour UF). En revanche, un code-barre n'est pas très aisé à lire. Il faut que le patient (éventuellement endormi, voire inconscient) tende son poignet, viser correctement avec l'appareil photo du smartphone, ou sortir un lecteur spécialisé de sa poche, etc...

Ces arguments négatifs nous font quelque peu douter de la pertinence de baser la reconnaissance sur cette lecture optique. C'est par ailleurs pour une raison similaire que ce mode d'identification n'avait pas été retenu dans le cadre du projet SAVER.

14.6.3 Entrée par balise NFC

La technologie NFC (Near Field Communications, lecture à quelques dizaines de millimètres de distance) cumule de nombreux avantages : faible consommation, intégration du lecteur dans de nombreux terminaux, absence d'alimentation de la balise, coût négligeable, lecture explicite... Beaucoup d'indicateurs montrent que cette technologie a de fortes chances de s'imposer dans de nombreux environnements, d'autant que les balises peuvent être conditionnées dans de très nombreux supports (cartes de visite, porte-clés, badges, bracelets de poignet, monture de lit, textiles,...) ce qui les rend bien sûr très faciles à intégrer dans un cadre existant. L'absence d'alimentation et l'utilisation de lecteurs capables de communiquer évite aussi tout frais d'installation de câbles ou de répéteurs, rendant ainsi le déploiement aisé à réaliser et rapide à mettre en œuvre.

Le faible coût de cette technologie peut amener un grand hôpital à décider son déploiement dans les bracelets d'identification des patients si les avantages sont suffisamment conséquents ; et l'identification du patient dans le contexte de la facturation des prestations (tarmed, donc) pourrait être un argument de poids dans cette optique.

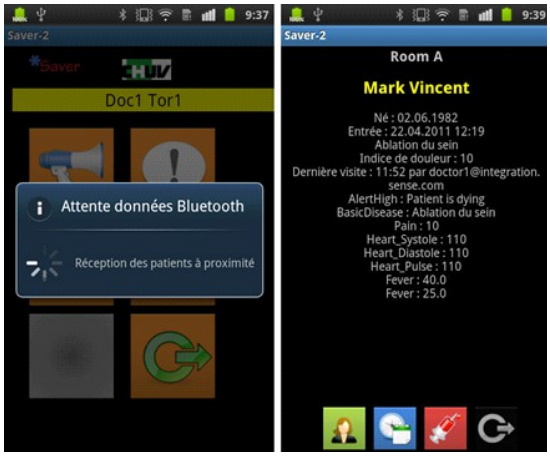
Les balises NFC sont recyclables ; en admettant que l'on conditionne la balise du patient sur un bracelet en plastique résistant à la température, il est possible de le récupérer après usage, de le désinfecter à relativement haute température (> 100 degrés), puis de le réinscrire (par exemple en stockant le numéro de séjour dans le bracelet NFC).

Ce même critère de coût modéré peut aussi amener d'autres acteurs, plus modestes du point de vue de la dimension (EMS, hôpitaux régionaux) à adopter cette technologie pour la saisie de prestations.

L'un des inconvénients est la lecture par balayage à quelques millimètres de la balise : cela demande une action volontaire de la part du soignant, et un minimum de coopération de la part

du patient ; encore que dans le cas d'un patient à autonomie très réduite, on peut "dupliquer" la balise sur le lit ou la chaise (par exemple) où il repose, grâce au coût très faible et à l'infrastructure inexistante.

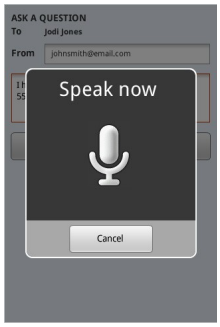
14.6.4 Entrée par balises DoctorMate/ClientTag (méthode SAVER)



SAVER identifie les patients par un jeu de balises radio-fréquences actives (PatientTag et DoctorMate). Les balises communiquent avec le terminal mobile par Bluetooth (voir aussi chapitres 5 et 6).

Cette identification est normalement implicite (contrairement à la technologie NFC, également basée sur des radiofréquences, mais qui impose un geste délibéré du soignant pour fonctionner), et semble bien adaptée à une utilisation dans le cadre d'un service des urgences ; mais elle peut éventuellement sembler relativement luxueuse pour être déployée dans un hôpital entier, dans un EMS ou dans un cabinet médical de groupe. En l'état, et en attendant les tests de déploiement, c'est potentiellement la plus efficace des méthodes d'identification, puisqu'elle va aussi permettre de mesurer le temps de session (temps pendant lequel le soignant est en présence du patient), et rendre ainsi possible le contrôle de la cohérence des renseignements fournis par le soignant (documentation d'une intervention de 3 unités de 5 minutes, alors que le temps de session n'est que de dix minutes, par exemple).

14.6.5 Entrée vocale



L'entrée vocale est de plus en plus performante ; mais son taux de succès n'est actuellement pas suffisant pour une utilisation professionnelle ; en revanche, elle pourrait seconder efficacement l'entrée explicite en permettant de définir des raccourcis vocaux. S'il est vrai qu'identifier un patronyme, de surcroît étranger, prononcé par une personne peu au fait de la prononciation adéquate en l'occurrence, et analysé par un logiciel qui tente désespérément d'appliquer des règles de prononciation françaises à un patronyme finnois (au hasard ; tchèque, mongol ou swahili peuvent aussi servir d'exemples) peut causer quelques confusions, épeler les lettres initiales du nom du patient peut permettre de réduire rapidement l'ensemble de noms proposés dans une liste déroulante proposée par le terminal mobile. Cette méthode d'introduction est très peu coûteuse, et relativement aisée à implémenter sur les smartphones modernes ; en revanche, il est nécessaire de la coupler à l'introduction explicite par le biais d'un algorithme de tri et de pré-sélection.

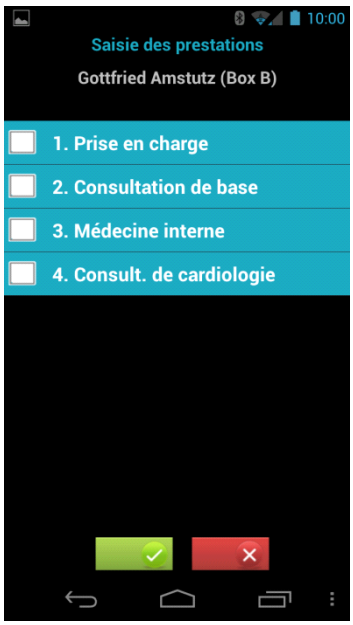
14.7 Conclusion

L'utilisation de smartphones ou à la rigueur de tablettes de petite dimension semble constituer un choix intéressant pour la documentation de prestations de manière décentralisée et

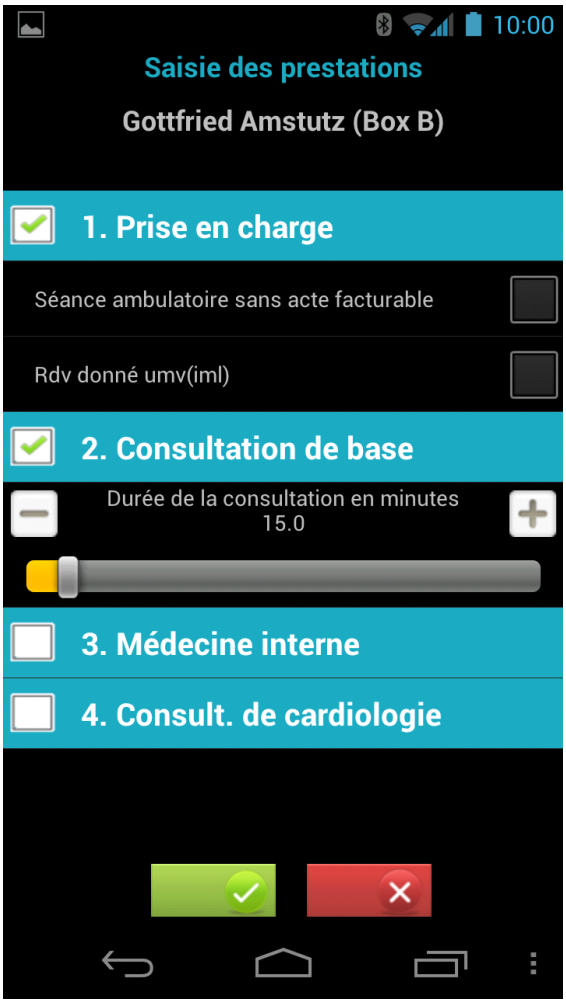
quasi en temps réel. Dans le cas le plus général, tout employé est probablement muni d'un téléphone : le terminal d'introduction ne constitue donc pas un bagage supplémentaire, dans la mesure où l'on peut remplacer le simple "Natel" de l'employé-e par un smartphone adéquat (Nominatif ? Banalisé ? Plusieurs avantages à utiliser un terminal nominatif...). Ce genre de terminal peut être déployé dans les sites les plus divers : Établissements médicaux-sociaux, cabinets médicaux de groupe, médecine de première instance, cliniques et hôpitaux universitaires.

Reste la problématique de la confection des catalogues ; nous proposons l'introduction d'éditeurs XML spécialisés (validation par un schéma adéquat d'un éditeur du commerce genre XMLSpy, ou éventuellement développement d'un outil dédié) sur un poste de type ordinateur personnel standard, puis exportation du catalogue validé sur un serveur de distribution, où on pourra éventuellement le valider par une autorité compétente (détection d'erreurs et d'incohérences, décision de l'instant d'introduction le plus judicieux pour une nouvelle version, ...), puis téléchargement transparent des catalogues vers les terminaux mobiles en fonction des responsabilités et de l'emploi du temps du possesseur du smartphone en question. La logique de fonctionnement correspond en gros à celle de MediQuest, qui est actuellement en tests à la Policlinique Médicale Universitaire de Lausanne.

Un service "Proof of Concept" a été introduit à titre expérimental dans le cadre de SAVER, pour faire la démonstration que la saisie de prestations en temps réel était une possibilité pertinente et réaliste, et que l'utilisation de smartphones pouvait constituer un vecteur approprié :



Selon le mode d'identification du patient, et le mode d'identification mutuelle patient-soignant, la saisie de la durée de la prestation peut être effectuée de manière partiellement automatique (mode d'authentification SAVER), ou explicite. Les deux options ne sont par ailleurs pas mutuellement exclusives.


10:00
Saisie des prestations
Gottfried Amstutz (Box B)
1. Prise en charge
Séance ambulatoire sans acte facturable
Rdv donné umv(iml)
2. Consultation de base
Durée de la consultation en minutes
15.0
3. Médecine interne
4. Consult. de cardiologie

CHAPITRE 15 Les organismes participant au projet

15.1 CHUV

Le CHUV est l'un des 5 hôpitaux universitaires suisses.

Grâce à sa collaboration avec la Faculté de biologie et médecine de l'Université de Lausanne, le CHUV joue un rôle de pointe d'envergure européenne dans les domaines des soins médicaux, de la recherche médicale et de la formation.

En quelques points :

- Le CHUV est un centre hospitalier d'envergure européenne. Il est l'un des cinq hôpitaux universitaires suisses, avec Genève, Berne, Bâle et Zurich.
- Il assure des soins dans tous les domaines de la médecine: des affections somatiques aux maladies psychiatriques, dans les disciplines médicales et chirurgicales, qu'il s'agisse de l'ambulatoire ou de l'hospitalier.
- Il comprend douze départements cliniques, médico-techniques et académiques, et un EMS psychogériatrique à Gimel.
- Il est étroitement lié à la Faculté de biologie et de médecine (FBM) de l'Université de Lausanne (UNIL) afin d'assurer la formation pré-graduée, post-graduée et continue des médecins.

- Il collabore avec les autres institutions universitaires lémaniques (UNIL, EPFL, ISREC, Institut Ludwig), les Hôpitaux universitaires de Genève et d'autres hôpitaux, établissements de soins ou institutions (Fédération des hôpitaux vaudois, Société vaudoise de médecine).

Présentation en quelques chiffres (2010) :

- Budget : un milliard 300 millions francs suisses
- Collaborateurs : 10'335 collaborateurs et collaboratrices
- Nombre de lits exploités : 1'428
- Nombre de patients hospitalisés : 44'285
- Journées d'hospitalisation : 509'097
- Taux d'occupation moyen des lits : 93.8%
- Nombre d'urgences traitées : 35'821
- Nombre d'interventions : 26'146 interventions ambulatoires et hospitalières
- Appels téléphoniques : 2'160 appels téléphoniques gérés chaque jour ouvrable
- Nombre de repas : 1'800 repas servis aux patient-e-s par jour et 2'500 aux collaborateurs
- Formation : Nombre de diplômé-e-s en médecine: 113 (2008)
- Nombre de naissances : 2'757 naissances (2'603 en 2009, 2'411 en 2008)

Source : http://www.chuv.ch

15.2 HEIG-VD / IICT

15.2.1 Un campus

La Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (HEIG-VD) offre à Yverdon-les-Bains huit filières de formation HES dans les domaines de l'ingénierie et de l'économie d'entreprise. Avec ses 1'500 étudiant-e-s, elle est la plus grande école partenaire de la Haute Ecole Spécialisée de Suisse Occidentale (HES-SO). La HEIG-VD est installée sur un grand campus urbain, dans trois bâtiments proches les uns des autres, dans la ville d'Yverdon-les-Bains: Route de Cheseaux, Centre St-Roch et Centre Y-Parc (plan). Elle est ainsi intégrée au tissu économique et aux entreprises.

15.2.2 Une formation scientifique orientée vers la pratique

A la HEIG-VD, les connaissances théoriques sont continuellement en relation avec leur mise en application. L'enseignement, dispensé tant par des enseignant-e-s chercheur-e-s que par des praticien-ne-s, amène progressivement l'étudiant-e à une approche globale et transversale. Il-elle

passe ainsi du statut d'exécutant-e à celui d'ingénieur-e ou d'économiste apte à créer, concevoir, innover, piloter, encadrer ou même fonder sa propre entreprise.

15.2.3 Une proximité reconnue avec le milieu économique

La HEIG-VD accorde, dans l'esprit de la politique des HES, une importance particulière au transfert de connaissances entre l'Ecole et les milieux économiques. Ainsi, étudiant-e-s, dirigeant-e-s et collaborateur-rice-s d'entreprises sont amenés à partager leurs compétences de manière concrète au travers de projets communs.

Grâce à ces projets, les professeur-e-s maintiennent également un contact étroit avec la pratique et enrichissent leur enseignement d'illustrations et de cas concrets.

Les ingénieur-e-s et les économistes des instituts et laboratoires de la HEIG-VD s'impliquent activement dans l'économie et l'industrie par le développement de produits nouveaux, de solutions innovantes et de pratiques de gestion.

Les étudiant-e-s sont aussi associés à ces recherches par les projets de semestre et surtout par les travaux pratiques de diplôme dont les thèmes sont en majorité proposés par des entreprises. La formation ainsi acquise permet d'aborder la vie professionnelle avec de solides atouts.

15.2.4 Une ouverture de nombreux débouchés

Les technologies sont en perpétuel développement. Pour maîtriser cette réalité mouvante, la Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (HEIG-VD) analyse en permanence les dernières évolutions et adapte régulièrement ses plans d'études. L'objectif est clair : offrir une formation de haute qualité et parfaitement en phase avec les besoins du marché. L'accueil très positif réservé aux Bachelor HES par les employeurs en est la preuve.

15.2.5 L'institut IICT

L'Institut ICT des technologies de l'information et de la communication de la Haute École d'Ingénierie et de Gestion du Canton de Vaud (HEIG-VD) a été crée en 2000. Son but est d'accomplir une des trois mission des hautes écoles spécialisées conformément à l'article 414 de la LHES qui est la recherche appliquée (Ra&D) et la prestation de service facilitant ainsi le transfert technologique auprès du tissus économique et industriel régional, avec comme ambition la création de nouvelles entreprises.

L'IICT regroupe actuellement plus de 50 collaborateurs qui sous l'égide de plus de 20 professeurs et responsables de recherches contribuent à la réalisation de nombreux projets et programmes de formation scientifiques et techniques.

L'IICT fait partie des réseaux de compétences régionaux et nationaux comme le RCSO-TIC de la HES-SO et ICT-Net. IICT contribue à l'essor des unités de recherches des Master de l'HEIG-VD MRU dans le domaine TIC et Biotechnologies appliquées à la santé.

Source : http://www.heig-vd.ch, http://www.iict.ch

15.3 HESAV

La Haute Ecole de Santé Vaud est l'une des sept écoles HES du Canton de Vaud qui regroupent près de 4'750 étudiant-e-s et sont rattachées au Département vaudois de la formation, de la jeunesse et de la culture.

Créée le 1er octobre 2002, pour répondre aux exigences de la loi fédérale sur les Hautes écoles spécialisées (LHES), elle est née de la réunion de quatre écoles cantonales : infirmier-ère-s et sages-femmes de Chantepierre, physiothérapeutes et technicien-ne-s en radiologie médicale. La plus ancienne, l'école des sages-femmes, date de 1803, la plus récente, celle de technique en radiologie médicale, de 1987. Ainsi, ces écoles réunies apportent à la jeunesse de HESAV la richesse de leur histoire, leur culture et leur expérience.

La Haute Ecole de Santé Vaud, HESAV, offre quatre filières de formation Bachelor HES :

- Soins infirmiers
- Physiothérapie
- Sage-femme
- Technique en radiologie médicale.

Des offres de formation continue et postgrade si vous êtes professionnel-le de la santé.

L'occasion de côtoyer plus de 800 étudiantes et étudiants en formation initiale et donc de se former sur le plus grand site des professions de la santé de la Haute école spécialisée de Suisse occidentale (HES-SO).

L'opportunité d'étudier au cœur de la Suisse romande, dans l'enceinte du Centre hospitalier universitaire vaudois (CHUV), en étroite relation avec la Faculté de médecine de l'UNIL, et de bénéficier de l'apport de professeur-e-s, médecins et chercheur-e-s réputés.

L'opportunité de participer à des échanges d'étudiant-e-s Erasmus et au programme d'Université d'été ouvert dès l'été 2009. Ces premiers échanges ont été organisés avec des facultés de sciences infirmières de Californie (USA). D'autres projets sont d'ores et déjà en négociation.

La perspective de pouvoir suivre une filière universitaire de Master et Doctorat en sciences infirmières ouverte par l'Institut universitaire de formation et de recherche en soins (IUFRS) située au sein du CHUV.

De plus, HESAV collabore avec des partenaires romands, nationaux et internationaux, des milieux de la pratique, de la recherche et de la formation. Elle participe au développement intellectuel, scientifique et économique de la ville de Lausanne, du canton de Vaud et de la Suisse romande.

Source : http://www.hesav.ch

CHAPITRE 16 Intervenants dans le cadre du projet

16.1 Spécification, direction du projet

16.1.1 Service des urgences CHUV

Olivier Hügli (olivier.hugli@chuv.ch) , direction médicale, expertise

16.1.2 HESAV

Régis Le Coultre (Regis.LECOULTRE@hesav.ch) , support logistique, relations

16.1.3 HEIG-VD / IICT

Markus Jaton (markus.jaton@heig-vd.ch), direction technique, réalisation

16.2 Développement

16.2.1 Coordination

Markus Jaton (markus.jaton@heig-vd.ch)

16.2.2 Serveur

Mark Vincent (mark.vincent@sysmosoft.com)

Carlo Criniti (carlo@criniti.name)

Fabien Dutoit (fabien.dutoit@heig-vd.ch)

16.2.3 Client (terminal portable)

Markus Jaton (markus.jaton@heig-vd.ch)

Carlo Criniti (carlo@criniti.name)

Claudio Pimpao (claudio.pimpao@crako.net)

Mark Vincent (mark.vincent@sysmosoft.com)

Fabien Dutoit (fabien.dutoit@heig-vd.ch)

16.2.4 DoctorMate

Dominique Bovey (dominique.bovey@heig-vd.ch)

Christophe Donzelot (christophe.donzelot@heig-vd.ch)

Claudio Pimpao (claudio.pimpao@crako.net)

16.2.5 Patient Tag

Dominique Bovey (dominique.bovey@heig-vd.ch)

Christophe Donzelot (christophe.donzelot@heig-vd.ch)

16.3 Implantation sur site

16.3.1 Local de démonstration (rue du Bugnon 19)

Régis Le Coultre (Regis.LECOULTRE@hesav.ch)

Christophe Greppin (christophe.greppin@heig-vd.ch)

16.3.2 Soutien du service informatique du CHUV

Stéphane Misteli (stephane.misteli@chuv.ch)

Philippe Noth (philippe.noth@chuv.ch)

Pierre-François Regamey (pierre-francois.regamey@chuv.ch)

Zandi Rad Behrouz (Behrouz.Zandi-Rad@chuv.ch)

Coordonnées de l'auteur

Markus Jaton

professeur HEIG-VD

institut IICT/useraware

Avenue des Sports 20

CH-1400 **Yverdon-les-Bains**

Tél. : +4124.5576292

Mobile : +4179.2105545

mailto: markus.jaton@heig-vd.ch

Web site : http://useraware.iict.ch