



COLBRANT Audrey
RICM POLYTECH'GRENOBLE
Rapport de stage 3^{ième} année.

Projet Tuberculose
Une architecture de substitution pour monitorer la tuberculose

Tome Principal
et
Annexes

2008/2009

Rapport de stage

RESUME EN FRANCAIS :

Le projet Tuberculose (projet TB) est né de la volonté de 4 acteurs : l'hôpital tibétain Delek (Inde), le Gouvernement en exil tibétain (Inde), l'Association Italienne pour la Solidarité aux Personnes (Italie) et l'Université Johns Hopkins (Amérique) afin d'assurer une lutte plus accrue contre le fléau que représente la tuberculose.

L'association à but non lucratif Tibtec (Dharamsala – Inde) intervient dans ce contexte afin d'apporter une solution au monitoring de la tuberculose par les nouvelles technologies. Cette solution sera destinée aux hôpitaux tibétains en Inde dans un premier temps et diffusée par la suite à plus large public, mais qui plus est sera Open Source.

Jusqu'alors, chaque hôpital tenait de manière séparée les registres sur ses patients. L'idée du projet TB est d'apporter une solution centralisée des données afin tout d'abord de pouvoir mieux combattre la tuberculose grâce à un système pointu de statistiques pour en avoir une compréhension accrue, et dans un second temps de subvenir aux problèmes récurrents de coupures d'électricité et pannes de réseaux internet en Inde.

Il a également été demandé d'offrir une plateforme d'information et de gestion des données patients non seulement pour le corps médical, mais également pour les patients afin de connaître l'emploi du temps d'un médecin se trouvant à l'autre bout de l'Inde.

Enfin, un système de rappels pour la prise de médicaments et/ou de visite chez le médecin des patients a été sollicité.

Afin de répondre aux besoins de cette demande, il a été décidé de mettre en place un site web représentant la plateforme d'information où la gestion des dossiers patients est accessible de manière protégée.

De plus, un serveur tournant sur technologie mobile, donc à faible consommation d'électricité, se charge de recueillir les mises à jour des dossiers patients en cas de coupure de courant et des réseaux internet.

Les données patients quant à elles sont stockées sur un serveur distant se trouvant dans les pays de l'Ouest pour une sécurité vis-à-vis des coupures de courant.

D'un point de vue technique, le site a été codé sur l'approche REST grâce à un savant mariage des technologies Sling (pointeur de ressources), JSP (Communication avec le serveur) et AJAX.

Le serveur SMS est installé sur un téléphone Android dernière génération sur lequel tourne une version de RapidAndroid (version adaptée de RapidSms à Android) modifiée afin de subvenir aux besoins du projet TB. Elle permet via un système de formulaires de mettre à jour/créer/supprimer des données. RapidAndroid reçoit des SMS contenant un préfixe et une suite de champs. Il les analyse, si le format est correct, il transmet le SMS sous forme de requête HTTP au serveur distant puis envoie la réponse adéquate au téléphone initialement émetteur du SMS. Si le format est incorrect, il le signale à ce dernier pour demander une réémission correcte.

Rapport de stage

Le site ainsi que le serveur SMS pourront mettre à jour/supprimer/créer des données dans le repository via un gateway sur lequel circulent des requêtes HTTP POST/GET sur framework REST. Le serveur SMS intercepte les requêtes HTTP POST/GET via des handlers qui traitent de manière adéquate chaque requête. On peut en effet spécifier un handler spécifique pour une requête donnée, si ce n'est pas le cas, c'est le handler par défaut de Sling qui s'en charge.

Un serveur vocal devra également être mis en place afin de répondre aux besoins des personnes aveugles. Ce point n'était pas à implémenter dans la version prototypaire sur laquelle j'ai travaillé.

Enfin, le service de rappels aux patients sera géré grâce au service Google Agenda qui permet de programmer des envois de rappels sur un portable à une date et heure précise. Google Agenda Service enverra les rappels sur le serveur SMS Android qui se chargera de parser et reformatter les rappels pour les envoyer au patient visé par ledit rappel.

Rapport de stage

THANKS

I would like to sincerely thank my two tutors, M. Lemordant Jacques and M. Dorjee Phuntsok. The first one for having given me the opportunity to work in TCV structure, it is very hard for spontaneous volunteers to have this chance. Moreover, it was very interesting to work with him, first of all because of his good mood and all his good advices, secondly I have learnt a lot of new exotic and interesting technologies. The second one for his joviality, his open minded, his sense of the joke and his helpful advices.

Thanks to them, the atmosphere during the internship was wonderful. It was a real pleasure to work on this project, among the Tibetan community in exile.

I would like to thanks all the Tibetan community with whom I have lived for four months. I have learnt the culture, the traditions and the real meaning of the word "Respect". I have discovered that "Peace" is not only a concept in my mind but can be in the heart of an entire community.

Thanks to the jovial and nice people of the TCV Sponsorship Office for their teaching of Tibetan language and for their good mood.

Thanks to the trainees at the Mother's Training Centre (TCV) for having integrated me in their daily life, and for teaching me everything that is needed to understand Tibetan culture.

Finally, I want to thanks nurses of the Delek Hospital for their patience during meetings for answering to all my questions.

This travel was more than a simple internship and improvement in my engineer life on a technical point of view, it was a lesson of well being, a human lesson and I can only encourage more students to move abroad for their internship.

Rapport de stage

TABLE OF CONTENTS

I. Presentation of the enterprise

1. Description of the working context
2. Description of the enterprise
 - 2.1 L'INRIA
 - 2.2 Tibtec
3. Description of the subject

II. Study of the subject

1. The existing system
 - 1.1 Presentation of the existing system
 - 1.2 Analyze of the existing system
 - 1.3 The needs
- 2 The future system
 - 2.1 Principles
 - 2.2 Global architecture
 - 2.3 Description of solutions
 - 2.3.1 Overall principles of solutions
 - 2.3.1.1 How it works
 - 2.3.1.2 The power of Sling
 - 2.3.1.3 RapidAndroid description
 - 2.3.2 Description of solutions
 - 2.3.2.1 Interacting with the system through the website
 - 2.3.2.2 Interacting with the system thanks to SMS
 - 2.3.2.3 Reminders from Google Agenda

III. Discussing the project's progress

1. Organization of the work
2. Description of the work left
 - 2.1 Security
 - 2.2 Data of patients
 - 2.3 Management of SMS
3. The vagaries of a project
 - 3.1 Dealing with vagaries of a project
 - 3.2 Solutions given for the project and problems in regards of this solution
 - 3.2.1 Google Agenda reminders format
 - 3.2.2 Security
 - 3.3 Consequences on the organization

IV. Outcome

1. Outcome on new knowledge and experiences acquired thanks to the internship
 - 1.1 Difficulties encountered
 - 1.2 What I have liked
2. Discussing about the future

Glossary

Abstract

Appendixes of figures and sources

Technical appendixes

Let's roaming in Dharamsala

Rapport de stage

TABLE OF FIGURES

TITLE	PAGE
Figure 1. Tibetan Schools in India, Nepal and Bhutan	29
Figure 2. GSM Inde Népal et Bhutan	30
Figure 3. Projet TB – Architecture	31
Figure 4. Projet TB – Prototype architecture	32
Figure 5. Repository structure	33
Figure 6. Add Hospital GUI	34
Figure 7. Add Hospital Structure Repository	35
Figure 8. Add Hospital Properties	36
Figure 9. Patient Properties	37
Figure 10. Architecture part between Android phone and Google Agenda Service	38
Figure 11. RapidAndroid welcoming screen	39
Figure 12. RapidAndroid forms list	40
Figure 13. RapidAndroid form example	41
Figure 14. Google Agenda	42
Figure 15. Details of the reminder in the Google Agenda	43
Figure 16. Reception on the phone of the reminder	44
Figure 17. RapidAndroid reception of reminder on patient cell phone	45
Figure 18. Get_sputum test	46

Rapport de stage

TABLE OF SOURCES

TITLE	PAGE
add.jsp: form for adding a hospital in the system	47
AddHospitalHandler.java: handler of the previous form	49
AddPatientHandler.java: handler for adding a patient in the database	51
identVerif.jsp: verification of the user identity	54
FormSmsStorage.java: sending a SMS from the Android Server to the Remote Repository	55
forms.json: forms definition in JSON	57
fieldtypes.json: field types definition in JSON	58
fields.json: fields definition in JSON	60
ManageReminders.java: manage reminders in the Google Agenda Service	62
places.js: scripts AJAX mixed to Sling, used on the website	63

Rapport de stage

TABLE OF TECHNICAL APPENDIXES

TITLE	PAGE
Use case diagram	68
Class diagram	69
Sequence diagram: add new TB for a patient	70
Sequence diagram: ask for the schedule of a doctor	71
Sequence diagram: receive a reminder	72

Rapport de stage

I. PRESENTATION OF THE ENTERPRISE

1. Description of the working context

This training course of ending of studies was in two parts: the first part was in Grenoble (France) for 2 months in WAM team in the INRIA under the responsibility of M. Jacques LEMORDANT, teacher and researcher, and the second part was in Dharamsala (India) in the non-profit startup TibTec under the responsibility of M. Phuntsok DORJEE, CEO of TibTec.

The project is a result of an effort of cooperation between TibTec and the INRIA and take place in the Web4Dev domain, and more specifically in web-mobiles communication systems adapted to suit the needs of rural areas.

2. Description of the enterprise

2.1 L'INRIA

The first part of my internship has taken place in the INRIA, research laboratory, in WAM team. My supervisor, M. Lemordant has helped me to prepare the subject of the internship, learning technologies that have been necessary for developing the TB Project tool.

We have often had debriefings on how the project goes with him, even when I was in India I was updating him about what I done, what I planned and the steps of the evolution of the project.

2.2 Tibtec

Tibtec is a nonprofit technology center based in Dharamsala that helps the Tibetan community in harnessing modern technologies to help the community. The project of my internship is a part of its work.

The office is in the TCV (Tibetan Children's Village). TCV is an educational place for Tibetan children's in exile in India. TCV works for the care and education of the Tibetan refugee children. So it has given me the opportunity to discover Tibetan culture and traditions. On the figure 1 (see appendixes), we can see the different cities where are based Tibetan schools all around the India.

(See Let's roaming in Dharamsala)

Tibtec has taken in the past a trainee coming from Communication Networks and Multimedia named Aurélien PERSONNAZ that has worked on wireless mesh technology.

3. Description of the subject

The Tibetan Delek Hospital list all the patients affected by tuberculosis in India. The project of monitoring the tuberculosis in India was born 1 year ago thanks to four actors: the Tibetan Delek Hospital (Gangchen Kyishong – India), DoH (Department of Health, Tibetan Government in Exile), AISPO (Italian Association for Solidarity Of Persons), and the Johns Hopkins University (USA). The project named “TB project” (for TuBerculosis project) is made by Tibtec, in order to help the Tibetan community in India in monitoring the tuberculosis. This project begins from scratch, so everything has to be done.

Rapport de stage

First, we have worked with the Tibetan Delek Hospital, our principal contact amongst the afore mentioned actors, and then it will be spread in India in DOT (Direct Observation Treatment) centers for having a global knowledge of all the patients and be able to make statistics on TB in order to understand it better, and fight better against it.

In India, there are a lot of Tibetan refugees in exile. Tuberculosis is an illness that spread itself rapidly, killing one person every 18 seconds. Tibetan population in India is widely spread out in India in settlements which functions under the administration of the Tibetan Government in Exile based in Dharamsala. All of the settlements in India have clinics and hospitals of the DoH.

Sizeable portion of the Tibetan community lives in Dharamsala and since this is also the where Tibetan Government in exile and TCV school is located. The role of Tibetan Technology Centre comes in here.

II. STUDY OF THE SUBJECT

1. The existing system

1.1 Presentation of the existing system

At the moment, all data are centralized in the Tibetan Delek Hospital where Dr. Tsetan is the CMO (Chief Medical Officer) and the doctor specialist and a reference in Tuberculosis. There are also lot of DOT centers spread all around India and data are spread too. Moreover, just a little part of data is registered on computer, otherwise, everything is written in books.

When a patient comes at the hospital, and the diagnostic is tuberculosis:
First time "Tuberculosis" patients are referred to as "New & Standard TB Patient".
Second time "Tuberculosis" patients are referred to as "Relapse TB patient".
Third Time "Tuberculosis" patients are referred to as "Extra Pulmonary TB patient"
Fourth Time "Tuberculosis" patients are referred to as "MDR (Multi Drug Resistant) patient"
After MDR the patients are categorized as "XDR (Extensively Drug Resistant) TB patient".

1.2 Analyze of the existing system

This system, as in all the old papers systems, has worked very well until now, but it shows weaknesses. First of all, it is obviously difficult to access to a specific data (data on a patient) and looking for a particular data among all the folders is time consuming. Due to the above problem and non-centralization of records on patients, tracking patient's movement which is necessary for containing a plague like TB is hampered.

Also when a patient moves from a DOT to another, transferring data is through postal service which is unsecure, unreliable and time consuming.

Moreover, if a fire or a calamity occurs, all data are lost forever. Thanks to a computing system, the data backups and restoration can be done automatically with least human intervention. Finally, Dr. Tsetan's (CMO and Dr. in charge of MDR TB and XDR TB) is also the personal physician to His Holiness the XIV Dalai Lama and is in His Holiness' entourage when He travels abroad or within India. Hence, it is difficult to inform the patients who come from all over India of Dr. Tsetan's schedule at the Delek Hospital.

1.3 The needs

I have had several meetings with nurses of the Delek hospital in order to understand the real needs, and to adapt the future system to the "reality on the ground". It appears that there is a serious problem in understanding how this illness comes from, what are the main reasons that helps to its development. Having data centralized in a database will allow making statistics that will be very useful in order to understand and fight against TB in a better way and eventually in containing the spread of the disease.

For an example, doctors and nurses need to know how many patients a year contract TB, whether there's increase/decrease in numbers, which group of people are more susceptible to have TB and why, what about recoveries/deaths ?

Rapport de stage

Moreover, monitoring TB for refugees means studying this disease for people that come from different places. It means understanding if people that come from Tibet are more touched by TB, if there are differences of the immunity system between people that came from Tibet and those that have lived all their life in India.

Finally, focusing on TB's categories & drugs, medical core need to know which kind of TB category is the more wide spread, for each category which drug is the more used for each category and which give more responsiveness. Of those responses they will be able to understand the reasons.

Statistics will help in answering all the above questions and help in making good decision to make good prevention campaign by educating people about the disease and what is better to do and to avoid to curb the spread of TB.

That is why, centralization of data and creating a connectivity between the Delek Hospital and DOT centers with a cheap technology is crucial in recording, sharing and transferring information of "transferred out" patients between the various DOT centers.

When we say "centralization", it means providing a unique and user friendly platform of information for everybody and a possibility to have correct and updated data, updated status and state of each and every patient. The platform should also have some extra and entertaining features to attract visitors. Moreover, in India there are a lot of long power cuts during which doctors and nurses still work, so we need to take care about this important fact.

Finally, it is asked to provide a way for every kind of patients to know when a doctor (Dr Tsetan at the moment) is available at his office, a sort of calendar. This system must suits the needs of the patients that don't know how to read. For convenience of non-regular patients (those who have to come once all the three/six/more months at the hospital, not daily), it is useful to provide a way to remind them to come at the hospital on a specific day for collecting medicines or for a routine check up by a doctor.

An important point is that technologies that will be chosen need to consume less electricity and which can be run on backup batteries considering the erratic power situation in India. And obviously, this tool must be user friendly so that it can be easily understood by a non-technical person.

2 The future system

2.1 Principles

We need to provide to the Delek Hospital and DOT centers a system based on low cost and low power technologies. To maintain connectivity between DOT centers and the Delek Hospital and give some information on the schedule of a doctor to patients, we have chosen the SMS technology because of its availability everywhere in India. We have chosen to normalize data in order to collect them thanks to a system of forms.

In order to provide a platform of information, we have chosen to build a website, easy way to centralize and offer statistics visualization for everybody and offer a way to manage data of patients for the doctors. So the website has to provide an authentication system in order to give access to doctors to the management of data concerning patients.

Rapport de stage

In India, we have a problem of available technologies. As we have seen on the picture that shows the spread of Tibetan schools in India, the Tibetan population is spread mostly in the North, the South and the East side of India. We need a technology available in all those places. As we can see on the figure 2 (see appendixes), the GSM coverage matches with those regions. That's why we have chosen the mobile technology as key-technology used in TB Project. In case of power cuts, a system of SMS form allows to manage and update the system. In addition to updating the system, the reminder system for non regular patient will be based on SMS layer too.

Finally, we will provide a VOIP system in order to make the system usable by persons that don't know how to read or write.

See *use case diagram* in technical appendixes.

2.2 Global architecture

To collect and to process SMS, we need a SMS Server. It will be based on the last Android technology (Android 1.5, HTC Magic G2). It will collect forms, analyze them and transfer them if the format is good to a repository. Android phone is an open source Operating System and have low power consumption so it can work many hours without electricity thanks to its battery (ideal solution for power cut problems).

The repository will probably be on a computer outside India (France, USA, ...) in order to be safe of power cut. The repository will be managed thanks to Sling web-application framework and Jackrabbit JCR.

Android and the remote repository will communicate thanks to HTTP POST/GET.

The VOIP Server will be based on Asterisk technology which is used by Tibtec since many years.

Our server will handle all requests send by SMS or VOIP (one server for each) and transfer them to the repository.

See the figure 3 in appendixes for the Architecture of the system.

N.B.: For the rest of the report, we will only focus on the prototype solutions. The two differences between the final solution and the final version are on one hand the Asterisk Server that I haven't studied, first because of a lack of time, and secondly because it is not available at the moment, and on the other hand statistics on the website.

2.3 Description of solutions

2.3.1 OVERALL PRINCIPLES OF SOLUTIONS

2.3.1.1 How it works

We have two repositories.

One is based on Sling and CRX while the other one is based on a modified version of RapidAndroid.

The second one runs on an Android phone and handles all requests sent by phones, analyzes them and transfers them to the Remote Repository. But it also collect the reminders sent by the Google Agenda server, parse them and send them to the targeted patient.

Rapport de stage

The Remote Repository will store data on patients, but also processing requests sent from the website and the Android phone. It has no direct interaction with phones. Data in the repository are stored hierarchically, CRX allows us to see them under tree form while thanks to WebDav, we can see them as the well known folders/file system of an operating system. See the figure 5 for visualizing the structure in CRX. See *class diagram* in technical appendixes to visualize the formal structure.

For retrieving and inserting data in the repository from the website or the Android Server, we use mostly Sling (see section bellow).

Our system can be used by all terminals that can send SMS, which means, all the GSM proposed on the market (and others more advanced devices, but we find them two rarely here to quote them).

2.3.1.2 The power of Sling

Sling abstracts requests of database layer. We manipulate content as we manipulate a tree thanks to the REST approach. Moreover, Sling works on a HTTP POST/GET form system, offering different options that allow managing content easily (create/modify/delete/move and so on).

Our SMS Service runs on Android phone, so less powerful than a computer. That's why it is necessary to avoid all useless operations.

Thanks to Sling, instead of the heavy well know way of request (SELECT ... FROM ... WHERE...), just specify a simple input in the POST/GET form (in the code a simple argument in the request, in only one line) and no more.

```
// Prepare parameters
NameValuePair[] tabParam = new NameValuePair[SIZE];
// Set the argument resourceType: script that will handle the default resource
visualization
tabParam[0]= new NameValuePair("sling:resourceType", "tb_monitoring/patients");
```

We can easily in the same way delete a node, move it, etc... replacing "sling:resourceType" by the name of the adequate operation (":delete", ":move", ...).

2.3.1.3 RapidAndroid description

RapidAndroid is the implementation of RapidSms for Android. RapidSms is a SMS Service that handles SMS, analyzes them in order to determine if they correspond to a know form thanks to a keyword (prefix). If it is a know form, it parses the SMS, saves them in the SD Card of Android and answer to the original sender. If it is an unknown one, it asks for a reemission to the sender.

RapidSMS is a system that has proved itself in a lot of humanitarian project for data collection and mass scale monitoring (in Malawi, in Uganda, ...). That's why we have chosen to base our tool on it.

2.3.2 DESCRIPTION OF SOLUTIONS

2.3.2.1 Interacting with the system through the website

The website is mostly made of forms in order to retrieve or update resources. To explain how it works, we will take the example of adding a hospital in the system.

The client side of the website is a mix between JSP, AJAX, Sling and HTML.

Rapport de stage

HTML is the backbone of pages, through which are included pieces of JSP and Javascript scripts (see *places.js* in sources appendixes) in order to retrieve content pointed thanks to Sling URL from the repository and process the answer.

A form looks like the figure number 6 “Add Hospital GUI” in appendixes.

Once we submit the form, it can be processed by the `SlingDefaultHandler` or we can specify a specific handler. In our case, we always submit to a specific handler on the server side thanks to a Sling operation that we give in the form as a hidden parameter:

```
<input name=":operation" type="hidden" value="tb_monitoring_add_hospital" />
```

(in sources appendixes you can find files from which comes this piece of code : *add.jsp*, this is the form for adding an hospital)

The handler targeted by the property intercepts the request and processes it. First we analyze parameters, verifying their validity. Then, if everything is ok, we add one to the hospital counter (for the id), this operation is protected thanks to a mutex. Finally, we add the node and its structure in the repository.

(in sources appendixes you can find the handler : *AddHospitalHandler.java*)

As we can see on the figures 7 and 8 in appendixes, the node have been added with the properties submitted in the fields.

Here, the example is simple, but the structure can be more complex. An example, those for patients has to suit the fact that a patient can have several times TB, move from one place to another, etc...

To see the structure for a patient data see figure 9. The code that handles the creation of a patient is in sources appendix *AddPatientHandler.java*.

From the website, in the same way, we can add a patient in the system, update information concerning this patient (add a new sputum analysis, add a new X-Ray analysis, add a new weight [needed because when somebody has TB, add a new place when the patient move from one place to another, changes medicines given, he loses weight, we need to monitor that], add a new user typically a nurse or a doctor or an admin, ...)

See *sequence diagram “add a new TB”* in technical appendixes to visualize the database.

Speaking about security, on each JSP page that needs it, the line

```
<%@ include file="/apps/tb_monitoring/verifIdent.jsp" %>
```

protected it from a evil-minded access that is not logged on the system to access pages having the knowledge of their URI.

See the code in source appendix for *identVerif.jsp*.

2.3.2.2 Interacting with the system thanks to SMS

As said previously, in India connections are uncertain and power cuts happen often, we need to provide a system where those two variables are unneeded. Obviously, a SMS is restricted in space available and because it takes time to write an SMS, we can't do the same operations with this system and the previous one (website).

When a SMS arrives on the Android phone, the SMS is handled by RapidAndroid that makes a powerful analyze of requests in order to accept or reject them. In this way, it constitutes a strong protection for the system because it is needed to know the exact format expected by

Rapport de stage

RapidAndroid and except this application that undelivered information on this format, no other application can give information on it.

If the format is inadequate, it rejects the request, answering to the transmitter phone that the format is not good.

If the format is correct (means that it has a known prefix and good fields), RapidAndroid processes it, and transfers the request under a HTTP POST form to the Remote Repository (see method *sendToRepository* in sources appendixes *FormSmsStorage.java*). The Remote Repository processes the request in the same way described in the previous part (it is also a form) and answers to the Android Server that transfers the answer to the transmitter phone.

This system will be restricted to the following operations:

(see the welcoming screen on Figure 11, and the forms list on Figure 12 in figures appendixes)

Legend:

- > *what user sends*
- <- *what user receives*

Obtaining the results of previous sputum analysis of a patient identified by the number

-> **get_sputum** number_patient

<- Sputum for NUMBER_PATIENT: DATE_SPUTUM RESULTS_SPUTUM

Adding new results of sputum of a patient identified by the number number_patient:

-> **put_sputum** number_patient "date_analysis" "results_analysis"

<- New sputum added for NUMBER_PATIENT

X-Ray analysis:

number_patient:

-> **get_xray** number_patient

<- X-Ray for NUMBER_PATIENT: DATE_XRAY RESULTS_XRAY

Adding new results of x-ray of a patient identified by the number number_patient:

-> **put_xray** number_patient "date_analysis" "results_analysis"

<- New sputum added for NUMBER_PATIENT

Adding a new treatment outcome of a patient identified by the number number_patient :

-> **put_outcome** number_patient "outcome"

<- New outcome well added for NUMBER_PATIENT

Moving of a patient from one place to another:

-> **move** number_patient number_hosp "date_move" "reason_of_moving"

<- Moving of NUMBER_PATIENT registered in the system

Moreover, this action sends an email to the new hospital with a referral letter containing information on patient.

Updating DOT treatment of a patient identified by the number number_patient for monitoring if a patient has taken medicines or not:

-> **put_dot** number_patient

<- DOT well added for NUMBER_PATIENT

Obtaining the schedule of a doctor/nurse/CHW identified by the number number_doctor:

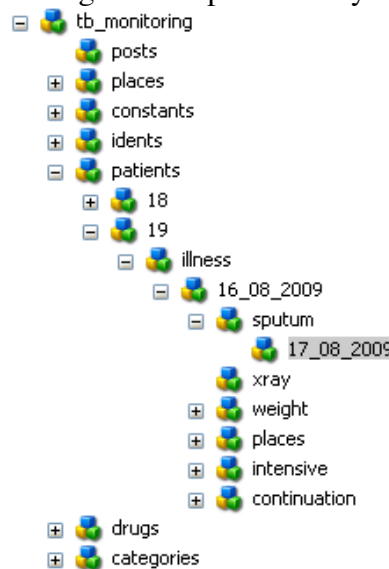
Rapport de stage

-> **schedule** number_doctor

<- NAME_DOCTOR is available from DATE_BEGINNING to DATE_ENDING

See an example of implementation of a form on Figure 13 in figures appendixes, and the JSON definition of those forms and their fields, and fields types definition in sources appendixes. Also see *sequence diagram “ask the schedule of a doctor”* in technical appendixes.

Assuming that in the Remote Repository we have a patient identified by the number 19, having done a sputum analysis with those properties:



Name »	Type	Value
.	nt:unstructured	
date_sputum	String	17/08/2009
sputum_results	String	results positive

You can find an example of “get_sputum” on Figure 18 with all steps described.

For implementing this part, RapidAndroid was powerful but not enough suitable for our needs. That’s why some modifications were necessary.

The first modification was to redirect the parsed SMS through an internet gateway toward the Remote Repository; initially the SMS was stored on the SDcard of the Android phone. However, we still register forms structure in order to give the possibility to RapidAndroid to decide of the validity of a SMS when it receives one.

The second modification was about field’s types available. Fields of a form could be Integer, Long, and some others types, but the type String wasn’t implemented yet. For facilitating the use of TB Project tool, I have added it and modified some others.

Moreover there were some bugs to fix. See the section “Vagaries of a project”.

On top of that, all those modifications can easily be made on the basic version of RapidSMS that is cross-platform. It means that our server can easily be cross-platforms, those on which RapidSMS runs and by extension, we can plan to collaborate with the Unicef that uses it. Moreover, this tool is also open source, which means it is safe of spy and the system will be in constant improvement thanks to the open source community, but also that everybody can use it adapting it easily to its needs.

Rapport de stage

In conclusion, we have transformed the RapidAndroid Server in a real Micro Intelligent Mobile Web Gateway for SMS based on a Sling client, because of its capacity to take decision on SMS.

2.3.2.3 Reminders from Google Agenda

Google Agenda API allows managing events in an electronic calendar accessible via Internet. It is possible to program reminders by SMS, email or popup. For our system, only the first one is interesting.

The Google Agenda SMS reminders system works like that:
First, in Google Agenda management panel, we configure the number phone on which reminders will be sent.

Then we define an event manually or programmatically and a reminder for this event with parameters (how many minutes/hours/days/weeks before the beginning of the event). (see in sources appendixes *ManageReminders.java* that explains how to add an event programmatically and setting a reminder for this event)

Finally, we wait for the reminder.

Example :

We define the reminder: “Come at the Delek Hospital to check doctor” the 25 august 2009 between 4.30 PM and 5.30 PM.

Then we specify we want to have a reminder 30minutes before the beginning of the event and we registered the event.

The 25 august 2009, at 4.00 PM, we receive this SMS:

Reminder: Come at the Delek Hospital to check doctor @ Tue 25 Aug 10:20 – 11:20 (TB Project)

It means that SMS sent by GoogleAgenda have, at the moment, this format:

Reminder: *text_reminder @ Day_letters_abbreviation day_number month_letters_abbreviation hour_beginning:minutes_beginning - hour_ending:minutes_ending (Name_agenda_of_the_event)*

To mix up this system and Android Server SMS system, we add a Server part in the architecture, see figure 10.

The idea is that Android phone handles reminders sent by the Google Agenda Service, analyzes them, processes them, and transfers them to the targeted patient.

We needed to find a suitable format to register the SMS sent by the Google Agenda Reminders Service, and then they can be analyzed and transferred by RapidAndroid to the patient’s phone concerned by the reminder. This format is the following:

Reminder: *text_reminder @ Day_letters_abbreviation day_number month_letters_abbreviation time_beginning - time_ending (Name_agenda_of_the_event)*

The *text_reminder* is the part on which we can interact because we insert it in the Google Agenda. It has to be no longer than 57 characters (otherwise the message is cut with “...”) and will be divided in two parts:

text_reminder: number_phone “reminder text”

Rapport de stage

Where `number_phone` is the patient's number phone (14 characters including country code) where the reminder have to be sent, and "`reminder_text`" (43 characters maximum) is for example "please come in Delek Hospital the 25/09/2009"

It means we need to modify a little bit formats of field types that the SMS Server Android handles.

The form defined in RapidAndroid will be:

Reminder *number string word word number word ratio word ratio comment*

At the moment, because I haven't yet a real Android phone, I have worked on the Android emulator and my own phone.

It means that I have had to simulate the two steps:

a. Reception of the reminder on a the Android Server

I have registered my own number phone in the Google Agenda, so reminders are sent on it. The process is the same as on an Android phone except that I can't make run RapidAndroid on my phone.

See figure 14 in appendixes for visualizing the Google Agenda and the reminder registered, and the figure 15 for reminder details. I have asked to be informed 3 minutes before the beginning of the event. On figure 16, you can see the SMS received on my phone.

b. From the Android Server to the patient phone

Then, the Android Server have to analyze the SMS reminder received and to send it to the targeted phone thanks to the method `sendToMobile` in `FormSmsStorage.java`.(see sources appendixes)

Running two Android emulators: one for the Server with the number 5556, the other one for the patient's phone with the number 5554, I have made manually the emission of the Google Agenda reminder on the Android Server.

You can find on the figure 17 in figures appendixes the message sent to the patient's phone.

See also *sequence diagram "receive a reminder"* in technical appendixes.

III. DISCUSSING THE PROJECT'S PROGRESS

1. Organization of the work

First of all, I have learnt how to use technologies needed in the project thanks to a specialist in this domain, M. Lemordant Jacques, emeritus teacher in new technologies at Polytech'Grenoble. It was during the first month of the internship were we have decided which technology will be the best among all available. I have mostly work on RapidAndroid and the CRX Repository for understanding in which way we can use them for TB project, and learnt REST framework, Sling and XPath technologies.

It was the first steps on the communication between the Android Server and GSM.

Then, in June, I have been in India. I have improved the communication between the Android Server and phones and developed the gateway from the Android Server and the repository. During this phase of the project, some bugs on RapidAndroid have appeared with the need to fix them. Moreover, I have had some new features (type String and deletion of SMS). In the same time, I have had exchanges of emails and discussions with actors involved in the project in order to define the needs. I have defined the specifications document, the global architecture and the architecture for the prototype (see Figure 4. Projet TB – Prototype architecture).

In July and August, while continuing to have some meetings with nurses of the Delek hospital to understand how they proceed in the treatment of TB (I have not a medical formation, so it was completely new for me), I have made a first version of the website in ESP that I have later transformed in JSP in order to suit the needs of the application. In the same time, I have defined a first version of the database in RelaxNG after having learnt this language.

As one goes along, I have created forms and corresponding handlers.

Then, in order to retrieve data in a simple way from the repository to the Android Server, I have created a pipeline XSLT in order to retrieve nodes in a XML format. Once on the phone, just need to parse them thanks to the integrated parser of Android and extract information, format them, and send them to users (others phones in client side). I have made some tests for sending a complete node from the Remote Repository to a phone after an XPath request.

After this step, I needed to define SMS Format that the Android Server manages. Then I needed to implement the system of the specific answer from the Remote Repository to the client's phones after a POST.

Finally, I have worked on the choice of the SMS reminder service and its implementation.

All those different steps were livened up by written documents on one hand for having formal documents for TB project, and on the other hand to explain my choices and how the project goes to my tutors.

Rapport de stage

2. Description of the work left

2.1 Security

Speaking about security, there are a lot of things to do. I haven't done them because not urgent for a prototype that is just a demonstration version. But for a system that manipulates personal data on patients, protecting it is more than needed.

First of all, we need to encrypt data while transferring on the Internet from the Server Side to the Client Side: from the repository to Android, from Android to the repository, from the repository to the website and from the website to the repository.

At the moment, I haven't had enough time to go in depth about this point.

In the same idea, we have to encrypt password of users in the database for avoiding problems with malicious people.

I have made a system of login, but at the moment no unlog system is available. I have considered that it wasn't a priority in the prototype version.

Finally, this Remote Repository one can be in private domain area if necessary for ensuring more protection.

2.2 Data of patients

We needed to be able to monitor if a patient take medicine as it was prescribed by nurses or doctors. I haven't had time to implement this part.

Same thing for the management of scheduler for a doctor on the repository server side.

Finally, I haven't had time to integrate the Google Agenda bundle to the server, so obviously, the GUI part on the website isn't implemented yet.

2.3 Management of SMS

If I haven't had time to implement the management of scheduler for medical people, the part of sending of a SMS from the patient to know when a doctor is available or not isn't obviously implemented yet.

Then, when a patient move from one place to another, the system has to send an email to the new hospital in charge of him/her, including a referral letter and information on patient illness. This part too isn't implemented yet.

3. The vagaries of a project

3.1 Dealing with vagaries of a project

The prototype was supposed to be finished at the end of my internship. But due to some problem, the project is progressing slowly and not as fast as was expected in the beginning.

The first one was the wait for the Delek's nurses answers. In the beginning of the project, when I have had finished to work on technologies and need to go ahead, defining the database, I have written a first email with questions to nurses. Waiting for an answer, one or two weeks later, I have sent another email for reminder. Only one week after that, I have had an answer. At this step of the project, I was too shy to assert my position and asking them to hurry up. I have understood later on, that they have their own work, but waiting too much put my internship in jeopardy. That is why, on the precious advices of M. Dorjee, I have been going down to Delek Hospital to meet with the nurses each I needed a precise answers. Emails were definitively unsuitable for defining needs because it allows a period of waiting for an answer that can be very long, and if precisions are needed, it takes more time. Moreover, going directly for meeting people reinforce relationships between Tibtec and actors in TB Project – that is very useful for the trust in the project, allows discussing deeper on problems, and on a human point of view is much better, especially for a project were the “client” is the direct user of the software.

On the other hand, we have had to face a problem of bugs in RapidAndroid. This application was used by many others big humanitarian projects, so we could supposed it to be safe of bugs. When I have had too much difficulty to fix bugs by myself, I have asked to M. MYUNG Daniel, the RapidAndroid creator, and he usually has answered promptly, giving me a suitable fix.

For example, there was a problem during deletion of SMS once they were parsed and handled. Sometimes RapidAndroid tried to delete the SMS before processing it and some problems occurred. I have fixed it delaying the deletion.

The second problem was when RapidAndroid recognize the prefix of a form. It tried to parse the message for extract fields. But if the type of the parameters in the message didn't match with the type of fields in the form, or if the number of parameters in the message wasn't the same as the number of fields for the form, an error occurred. I have fixed it too by making more precise the parsing of the SMS.

The third problem was on field's name. It was impossible to have two fields with the same name, even for two different forms. Removing a UNIQUE constraint in the database definition has fixed the problem.

The last problem was about forms prefix spelling, some verifications missed in RapidAndroid that began to create content about a form allowing characters such as semi colon, but when trying to register it in the database, impossible because those characters are not allowed. I have fixed it adding adequate verifications in RapidAndroid.

3.2 Solutions given for the project and problems in regards of this solution

3.2.1 GOOGLE AGENDA REMINDERS FORMAT

Rapport de stage

Choosing to externalize reminders management include that the system depends on the stability of the external system. This is the problem with Google Agenda reminders. At the moment, SMS sent by GoogleAgenda have this format:

Reminder: *text_reminder @ Day_letters_abbreviation day_number month_letters_abbreviation hour_beginning:minutes_beginning - hour_ending:minutes_ending (Name_agenda_of_the_event)*

Two choices to handle this kind of SMS and process them on RapidAndroid:

* Defining and implementing one new type in RapidAndroid that can contain everything and have to begin by @ and end by)

So the message will have the form name: **Reminder:** and two fields text_reminder and @blablabla)

* Modifying a little bit the fields form format in RapidAndroid to handle reminders of a format like that:

Reminder: *text_reminder @ Day_letters_abbreviation day_number month_letters_abbreviation time_beginning - time_ending (Name_agenda_of_the_event)*

Even if the second option is less simple to implement, if Google decides to change the format of reminders, even just a little bit, we have more chances that the Android Server still handle SMS in the first case because it is less specific than the second one.

But the first option is less elegant than the second. This last one allows us to retrieve dates and time of the event pointed by the reminder. So no need to put them inside the text_reminder field, it avoids redundancy.

Another solution consists in imagining a system in which the system will send an email to the administrator if it encounters an error or a log system where all the SMS are stored to be investigated later on.

But here, if the prefix changes, it is impossible to know. It means that collecting all the requests rejected by the Android Server is needed. First problem, this solution engender the hiring of a qualified person to analyze the error logs. The second problem depends on the first one, having a specialized person for that means having money for paying him (except if it is a voluntary work), so finding sponsors.

3.2.2 SECURITY

The number phone of the Android phone is a security flaw. If the phone suffers a DDOS attack (flooding by SMS in this case) with SMS with the same format than Google Agenda reminders, the Android Server will send as many SMS as reminders received. From the cost point of view (the part between the Android Server and phones is not free), it is bad. We can imagine some systems where we analyze how many times we received from a same number in a short period reminders and block the number.

3.3 CONSEQUENCES ON THE ORGANIZATION

All the vagaries that have occurred have caused delays in finished the prototype of the project. Sometimes, I have had to wait two weeks to continue a part. But lucky me, the TB project is a huge project, and there was often something to do on another part.

Finally, an exotic behavior of India is power cuts and problems with networks. They are impromptu and happen often. Sometimes, especially in the first part of the project (while it was needed for me to make a lot of internet research in order to learn technologies, and discussing with

Rapport de stage

different actors involved in the project) and the last part (while implementing and testing the communication with the Google Agenda Reminders Service), it has been very difficult to work. The solution in this case consists of forgetting the busy Western style and following the Tibetan rhythm: wait, take a tea and discuss with your neighbors.

IV. OUTCOME

1. Outcome on new knowledge and experiences acquired thanks to the internship

1.1 Difficulties encountered

The first month of the internship was quiet difficult because of technologies to use. I come from RICM option Network and the beginning of the training course was mostly oriented Multimedia in the way to think, in the technologies to learn and use. But also because of the big lack of documentation on Sling that is a new technology and the few people that talk about it don't explain it in the same way... quiet confusing for a beginner. Moreover, some parts of the official website on Sling are not updated giving some wrongs information (for example those on how to create our own POST handler that is the core of our application) or is incomplete. Finally, Sling mixed up with Javascript has not the same syntax than with JSP. Here also, the lack of documentation was difficult and sometimes it was more "guessing" what the solution is because of the impossibility to find somebody that have had the same need as me before (once again because Sling is a young technology).

Slowly I began to be used to this way to work and now I think I can tell that I have tamed those technologies (or they have tamed me, whatever).

The second difficult point was the power and network cuts that have happened often and can sometimes go on several hours.

Finally, understanding the real needs of clients/users of the system while I was studying them was maybe the more uneasy point. First of all, because of the difference of language: they spoke medical language, I tried to express my questions in non-computing language. The typical example here was to try to express them the essentials information to retrieve on SMS, which one was useful, which one was not.

1.2 What I have liked

Even after all those difficult moments that often lead to headaches, it was very interesting and exciting. First of all because of the diversity of domains I have had to work on. I like having a lot of different things to work on (the GSM platform, two kinds of repositories, the website, ...). Secondly because of liberty and autonomy that was given to me on this project. My tutors have given me the subject and a huge freedom, letting me lead the project by myself, answering when I have had questions, otherwise they have given me free hand. It was exciting to "change my skin", having every minute a different role: project manager, programmer, process owner, ... I felt fulfilled to have been considered no more as a student but as a real engineer, having the complete trust of my tutors.

I have liked very much working on this more than any other interesting projects. On a technical and human point of view, it was complete and exciting. Moreover, this is a useful project, not a capitalist one to create needs and make more and more money. So it was a real pleasure to work on this kind of project, a real pleasure to come each morning to work and to have headaches to fix problems.

Finally, the atmosphere and the way to work in Tibetan community were amazing and more than a simple experience. Before leaving France, my French tutor told me "if you go 2 months in

Rapport de stage

this place, you change for a long time...you go for more than that, it changes a life”. And I can’t agree more.

On top of that, I have had the amazing opportunity to learn basics of Tibetan language (that is a real headache) and to improve my English.

Looking at this experience whatever the point of view, I can say that it was more than a simple internship.

2. Discussing about the future

I have worked here as volunteer, and has said before, it was more than a real pleasure to work on this project. That’s why I will still work on it until the opportunity will be given to me.

My teaching courses are now finished, and I feel fulfilled in the basics needed for a new engineer in the labour market world.

Let’s begin the “active life”.

Rapport de stage

GLOSSARY

ANR: Application Non Responding. An ANR occurs in Android when the application takes too much time to process an operation.

CHO: Chief Medical Officer

CHW: Community Health Worker, nurses and doctors in training

DOT: Direct Observation Treatment, monitoring if a patient takes medicines regularly or not in a center of thanks to a dedicated person certified “of trust”

ESP: Ecma Script Page

JSP: Java Server Page

REST: Representational State Transfer. Term invented by Roy Fielding in 2000, it designed a stateless architectural style where the URI allows the direct access to a resource thanks to HTTP basics functions (PUT, DELETE, POST, GET) using hypermedia standards as HTML or XML for the navigation between resources.

TB: Tuberculosis

TCV: Tibetan Children’s Village, it is an educational place for the education and care of the Tibetan refugee children in India.

Rapport de stage

ABSTRACT

French

Ce rapport de stage fait état des 24 semaines de travail effectué dans le cadre du stage de fin de cursus Réseaux Informatiques et Communication Multimédia de Polytech'Grenoble. Il s'est déroulé pour sa première partie en France à l'INRIA et pour la seconde en Inde pour l'organisation à but non lucratif Tibtec afin de réaliser un logiciel de monitoring de la tuberculose grâce à des technologies bas coût.

Il consistait à la mise en place d'une solution centralisée des données pour pouvoir mieux comprendre et combattre la tuberculose grâce à un système de statistiques en prenant en compte les problèmes récurrents en Inde. Il a également été demandé d'offrir une plateforme d'information et de gestion des données patients pour le corps médical, ainsi que les patients afin de connaître l'emploi du temps d'un médecin se trouvant à l'autre bout de l'Inde. Enfin, un système de rappels pour la prise de médicaments et/ou de visite chez le médecin des patients a été sollicité.

La solution consistait tout d'abord en la mise en place d'un serveur distant stockant les données patients et utiles à l'application, accessibles via un gateway transportant des requêtes HTTP POST/GET répondant au framework REST traitées par des handlers spécifiques. Dans un second temps, elle consistait en le développement d'accessieurs Sling à ce serveur distant, à savoir un serveur d'analyse de SMS tournant sur la dernière version d'Android ainsi qu'un site web. Enfin, un service de rappels aux patients est géré grâce au service Google Agenda qui envoie des rappels sur un portable à une date et heure précise au serveur Android qui les transférera aux patients.

Mots-clés : Tuberculose, Sling, JSP, Serveur distant, RapidAndroid, Google Agenda reminders, HTTP POST handlers, REST, AJAX, formulaires

English

This report describes the 24 weeks of training course for the ending of engineering school in Communication Networks and Multimedia at Polytech'Grenoble. It has taken place for the first part in France at the INRIA, and for the second part in the non lucrative organization Tibtec in order to implement a software of tuberculosis monitoring thanks to low cost technologies.

It was needed to put in place a centralized solution in order to understand and fight with more efficiency against tuberculosis thanks to a statistics system, taking care about recurrent problems in India. It was also asked to provide an information platform and data management platform for medical staff and patients in order to be able to know the schedule of a doctor. Finally, a reminders system for taking drugs and/or coming to visit doctors was sought.

The solution consisted in the implementation of a remote repository that stores data on patients and useful for the application, accessible via a gateway carry on HTTP POST/GET requests following REST framework and handheld specifically. But also in the implementation of Sling assessors to this remote repository: a server for analyzing SMS running on the last Android technology and a website. Finally, the reminders service for patient is managed thanks to the Google Agenda reminders service that send reminders on phones at a special date and hour to the Android Server that transfers them to patients.

Keywords: Tuberculosis, Sling, JSP, Remote Repository, RapidAndroid, Google Agenda reminders, HTTP POST handlers, REST, AJAX, forms

APPENDIXES OF FIGURES AND CODE

1 Figures



Figure 1. Tibetan Schools in India, Nepal and Bhutan
http://www.tcewf.org/schools/school_map/index.html

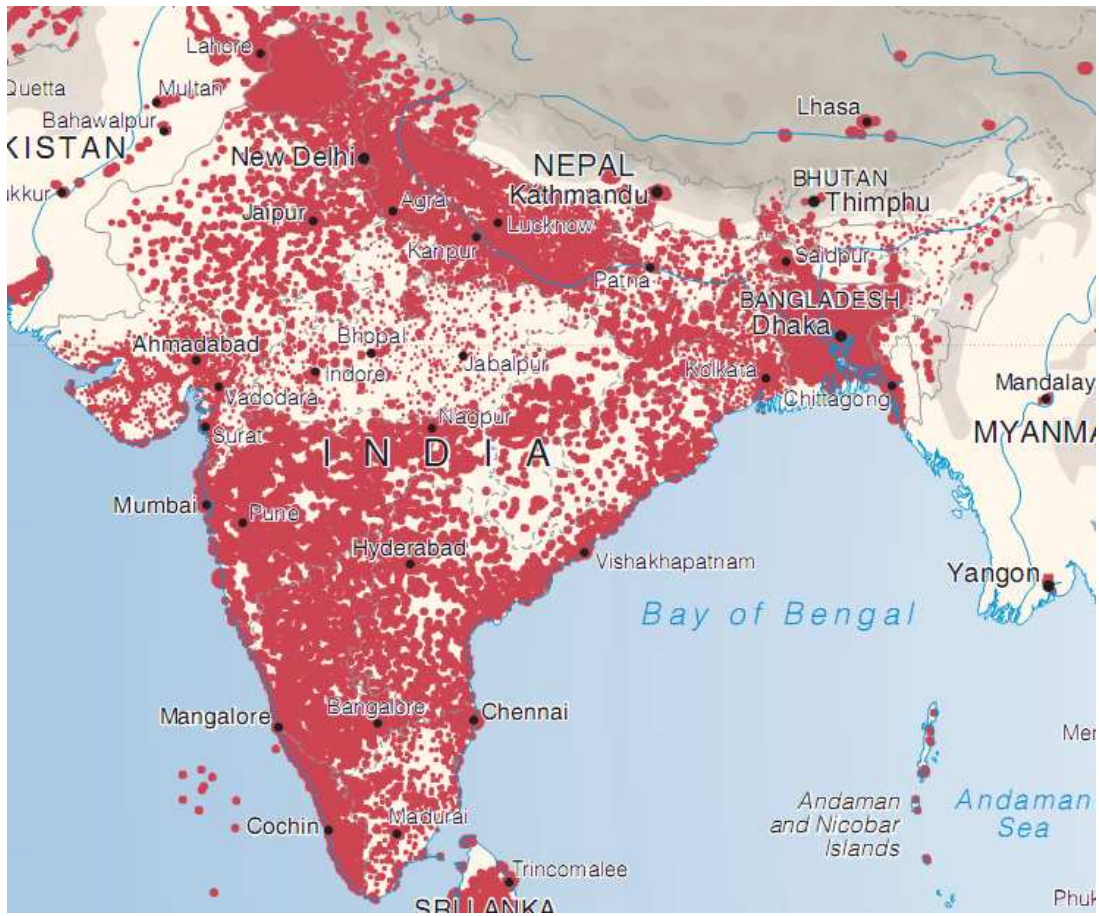


Figure 2. GSM Inde Népal et Bhutan

In red : places where GSM signal is available

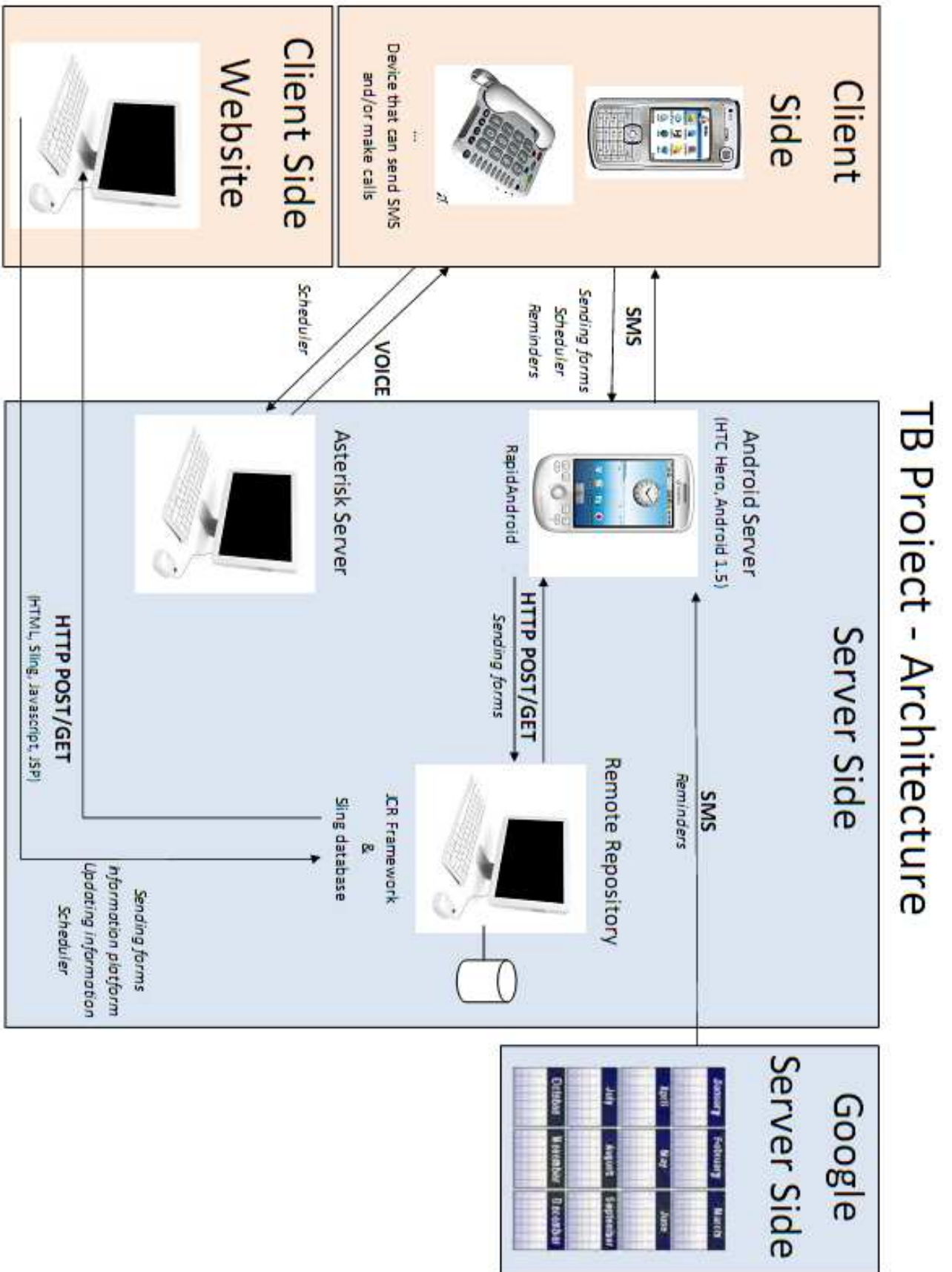


Figure 3. Projct TB - Architecture

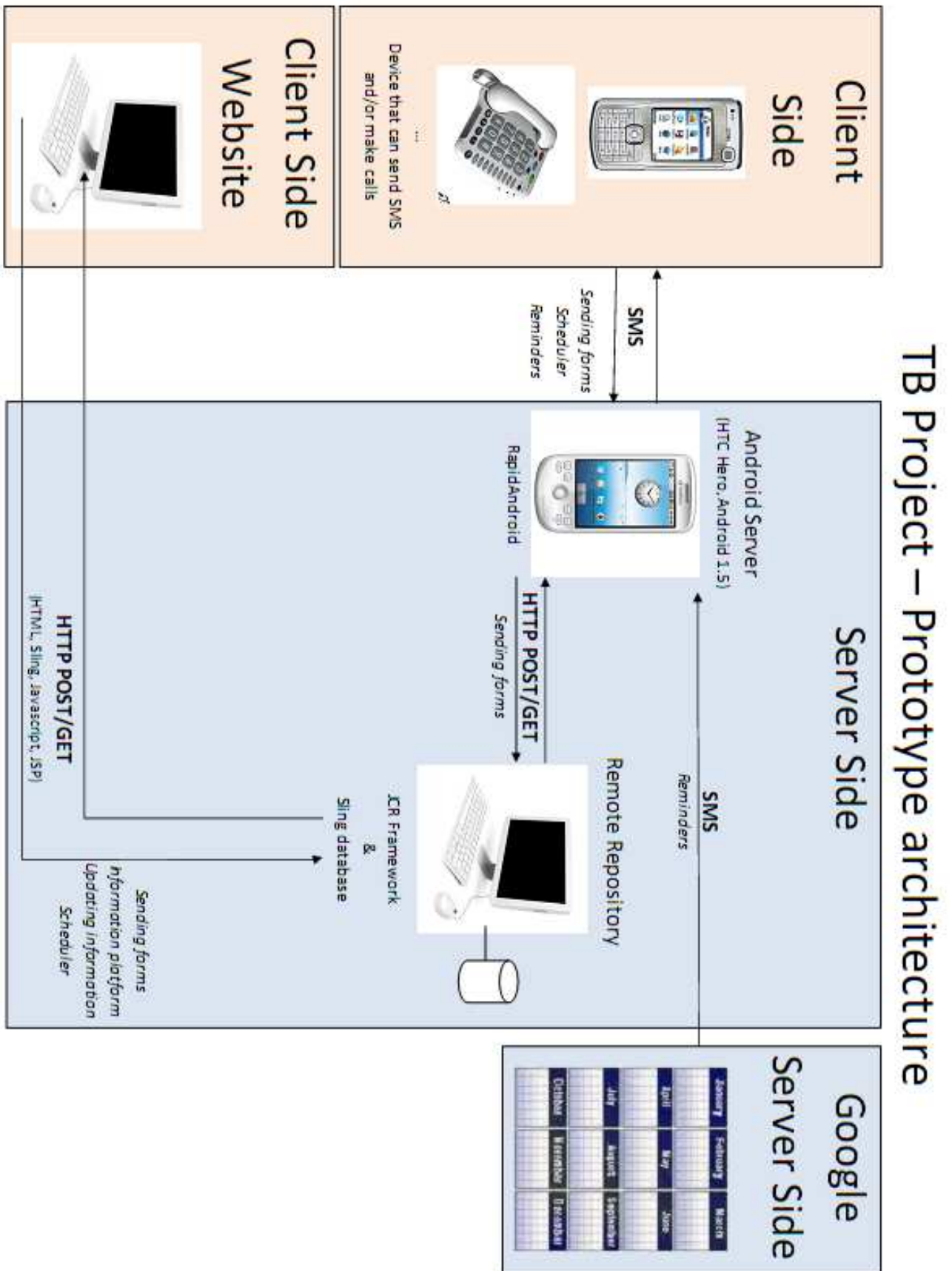


Figure 4. Projet TB – Prototype architecture

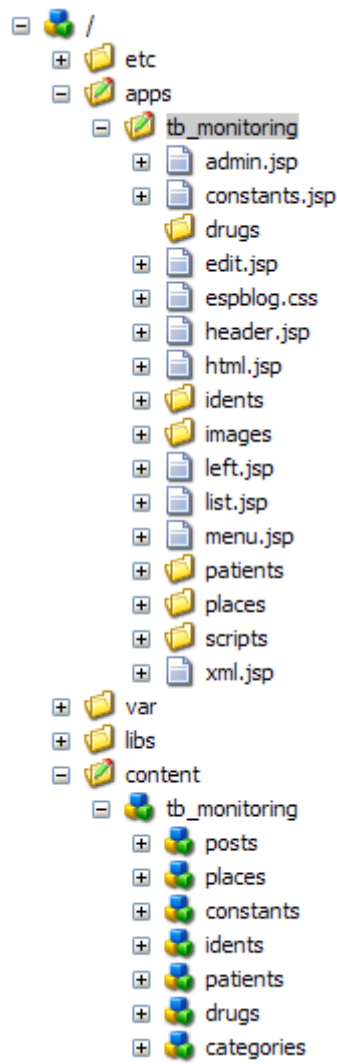



Figure 5. Repository structure

http://localhost:7402/content/tb_monitoring/places.add

Google Mes favoris Traduire Envoyer à word reference



Home
Login
Add Tuberculosis Post
RSS feed
Search

Add an hospital

Name * Delek Hospital

Email * dele@sacharnet.com

Number phone * 0091189222053

Address

Choose the country: *
India

States: *
Himachal Pradesh

Districts: *
Kangra

City * Dharamsala

Gangchen Kyishong

Road *

Post

* : obligatory fields

Figure 6. Add Hospital GUI

Rapport de stage

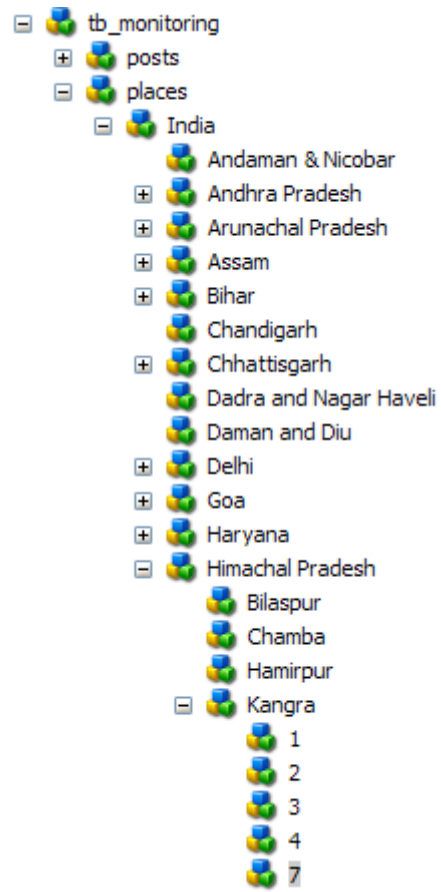



Figure 7. Add Hospital Structure Repository

Name >>	Type	Value
.	nt:unstructured	
city	String	Dharamsala
country	String	India
created	String	
district	String	Kangra
email	String	dele@sacharnet.com
name	String	Delek Hospital
number_phone	String	0091189222053
road	String	Gangchen Kyishong
sling:resourceType	String	tb_monitoring/places
state	String	Himachal Pradesh

Node	
Name	7
Path	/content/tb_monitoring/places/India/Himachal Pradesh/Kangra/7
UUID	N/A (node not referenceable)
Depth	7
# of child nodes	0
Is New	false
Is Modified	false
Is Locked	false
Is CheckedOut	true

Primary Node Type  nt:unstructured

Mixin Node Types

Orderable Child Nodes true

Primary Item Name *(none)*

Figure 8. Add Hospital Properties

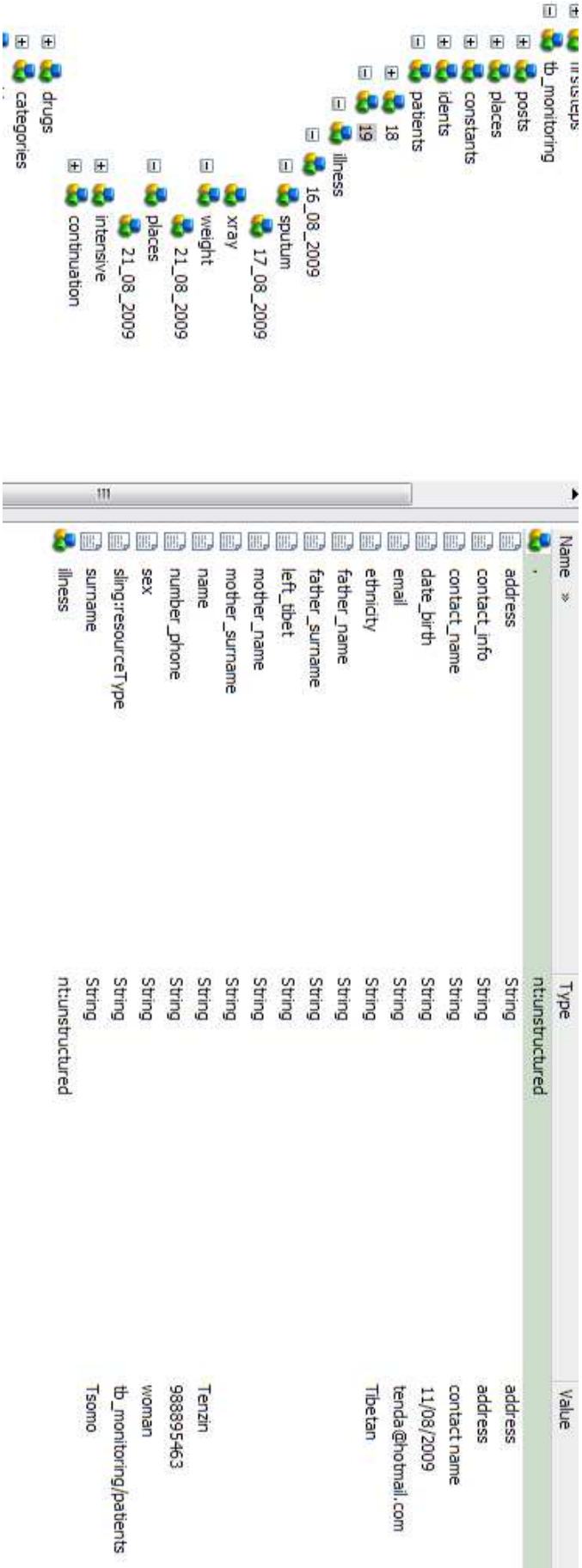


Figure 9. Patient Properties

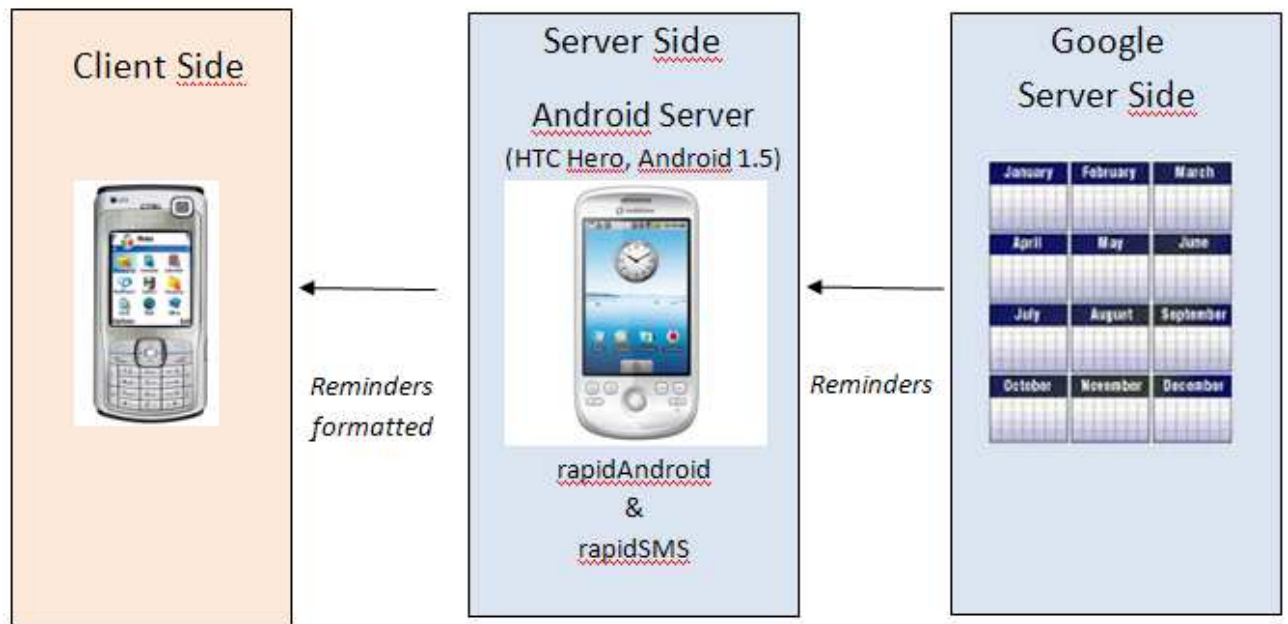


Figure 10. Architecture part between Android phone and Google Agenda Service



Figure 11. RapidAndroid welcoming screen

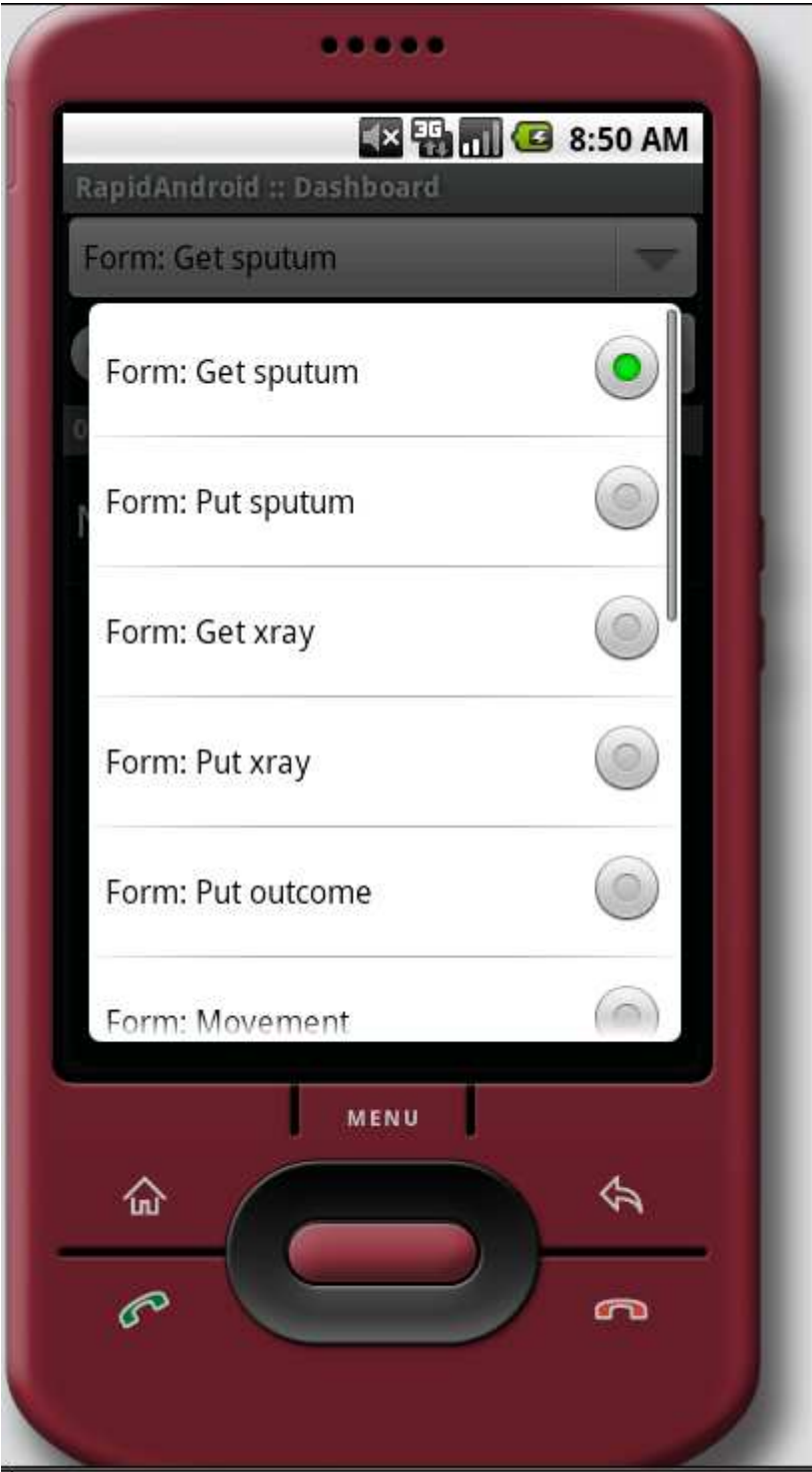


Figure 12. RapidAndroid forms list

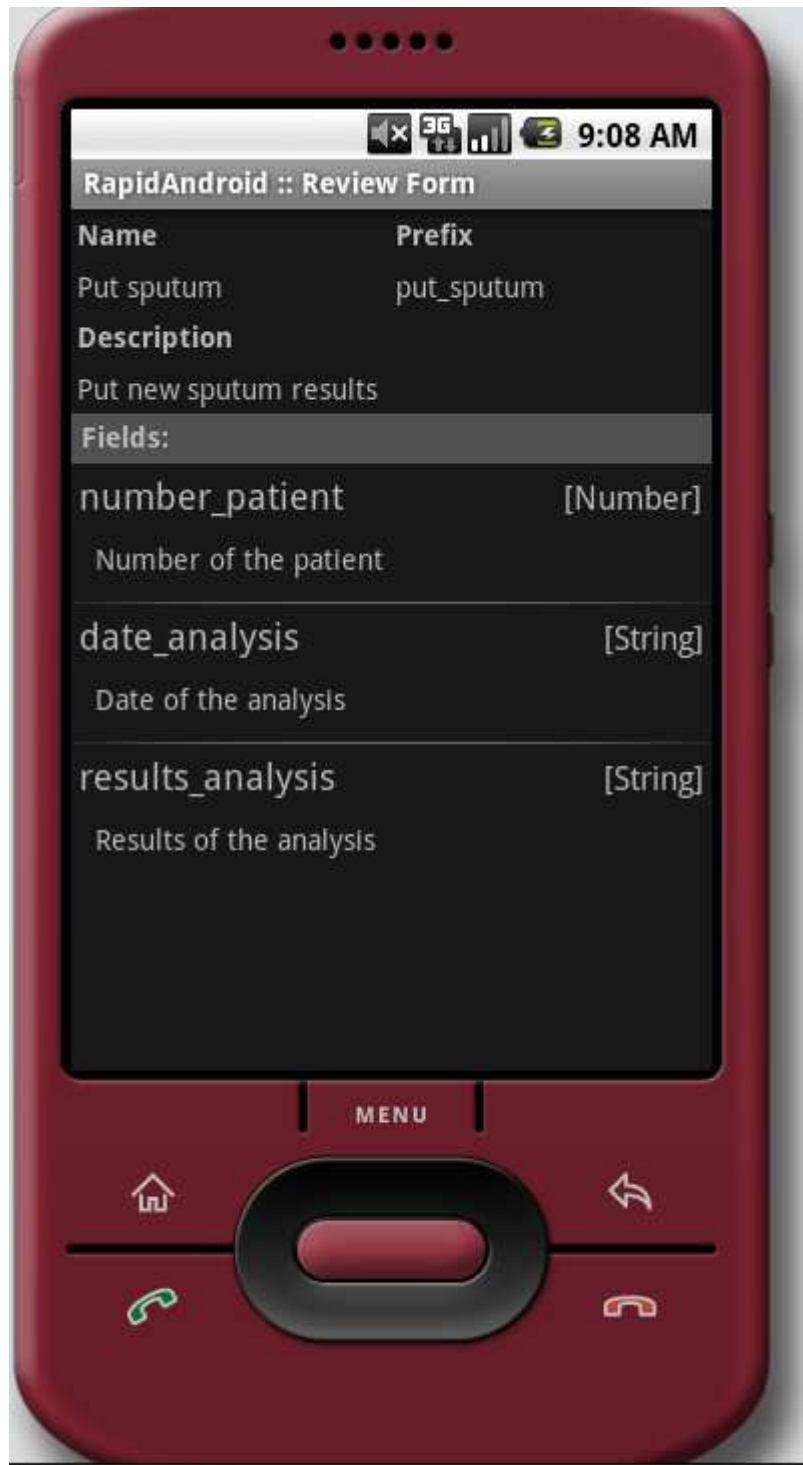


Figure 13. RapidAndroid form example

[Gmail](#) [Calendar](#) [Documents](#) [Photos](#) [Reader](#) [Sites](#) [Web](#) [more](#) ▼

Google calendar

[Create Event](#)
[Quick Add](#)
[Tasks](#)

« **September 2009** »

M	T	W	T	F	S	S
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

▼ My calendars
[aourey.coubran@uq](#) ▼
TB_Project ▼
[Settings](#) [Create](#)

▼ Other calendars

Today **31 Aug – 6 Sep 2009** [Refresh](#)

Mon 31/8	Tue 1/9	Wed 2/9
16:00		
17:00	17:25 – 19:25 5554 "Come at the Delek Hospital to be checked by the doctor"	
18:00		
19:00		
20:00		
21:00		
22:00		

Sync Google Calendar with your iPhone. [Learn more](#)

Figure 14. Google Agenda

The screenshot shows the details of a reminder in Google Agenda. The event is titled "B554 'Come at the Delek Hosp. to be checked by the doctor'". The date is 1/9/2009, from 17:30 to 19:30. The location is "TB_Project". The event is set to "Does not repeat". The description field is empty. The "Options" section includes a reminder set for 3 minutes via SMS, and the event is set to "Default" privacy. There are also links for "Add a reminder", "Remove", "Email organiser", "Add guests", "modify event", "invite others", and "see guest list".

What B554 "Come at the Delek Hosp. to be checked by the doctor"

When 1/9/2009 17:30 to 19:30 1/9/2009 All day

Repeats: Does not repeat

Where

Calendar TB_Project

Description

Guests [Email organiser](#)
 [Add guests](#)

Guests can modify event
 invite others
 see guest list

Options

Reminder SMS 3 minutes [Remove](#)
[Add a reminder](#)

Show me as Available Busy

Privacy **This event is:** Default
 Private
 Public

[Learn about private v. public events](#)
[Publish this event](#)

Figure 15. Details of the reminder in the Google Agenda

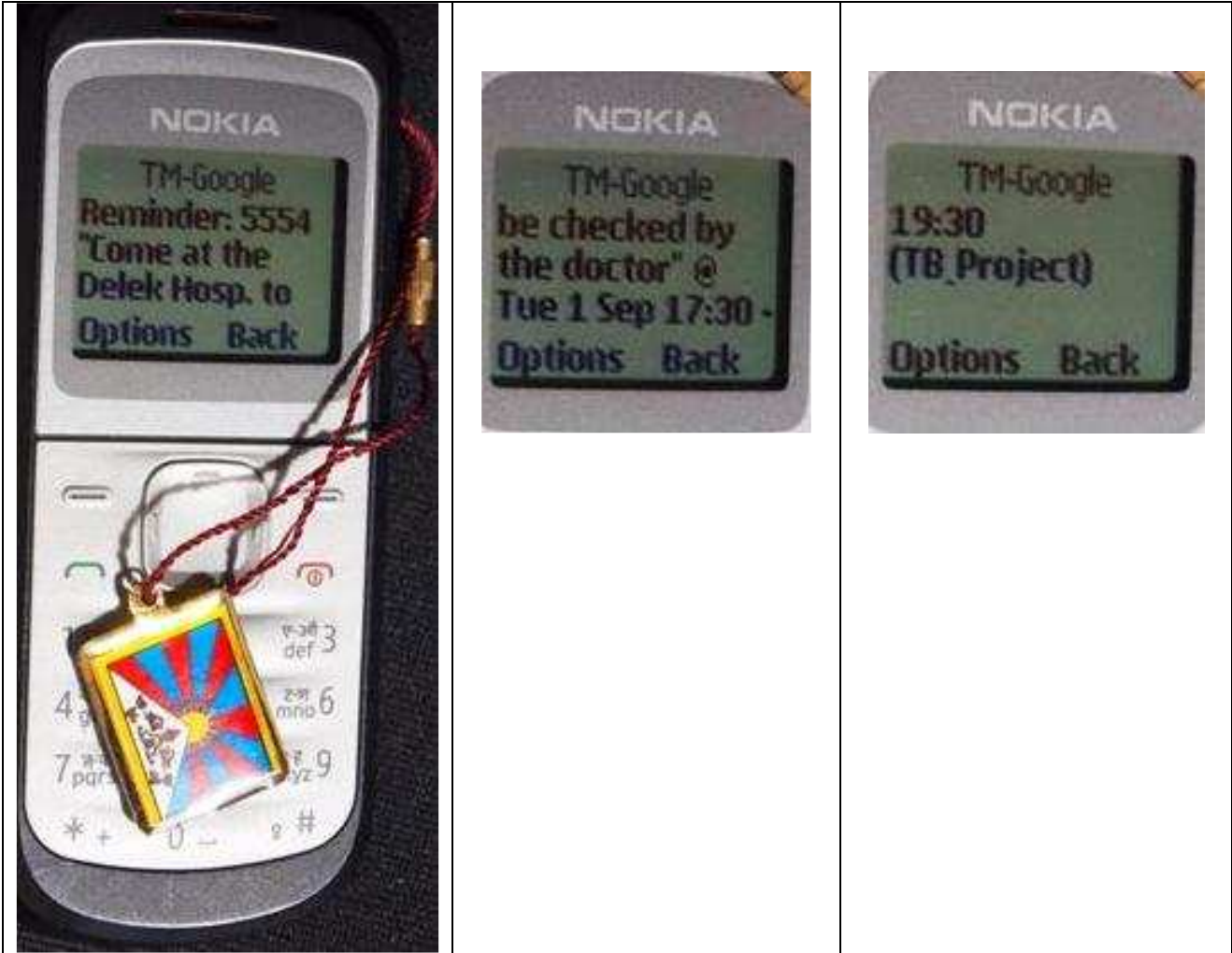


Figure 16. Reception on the phone of the reminder

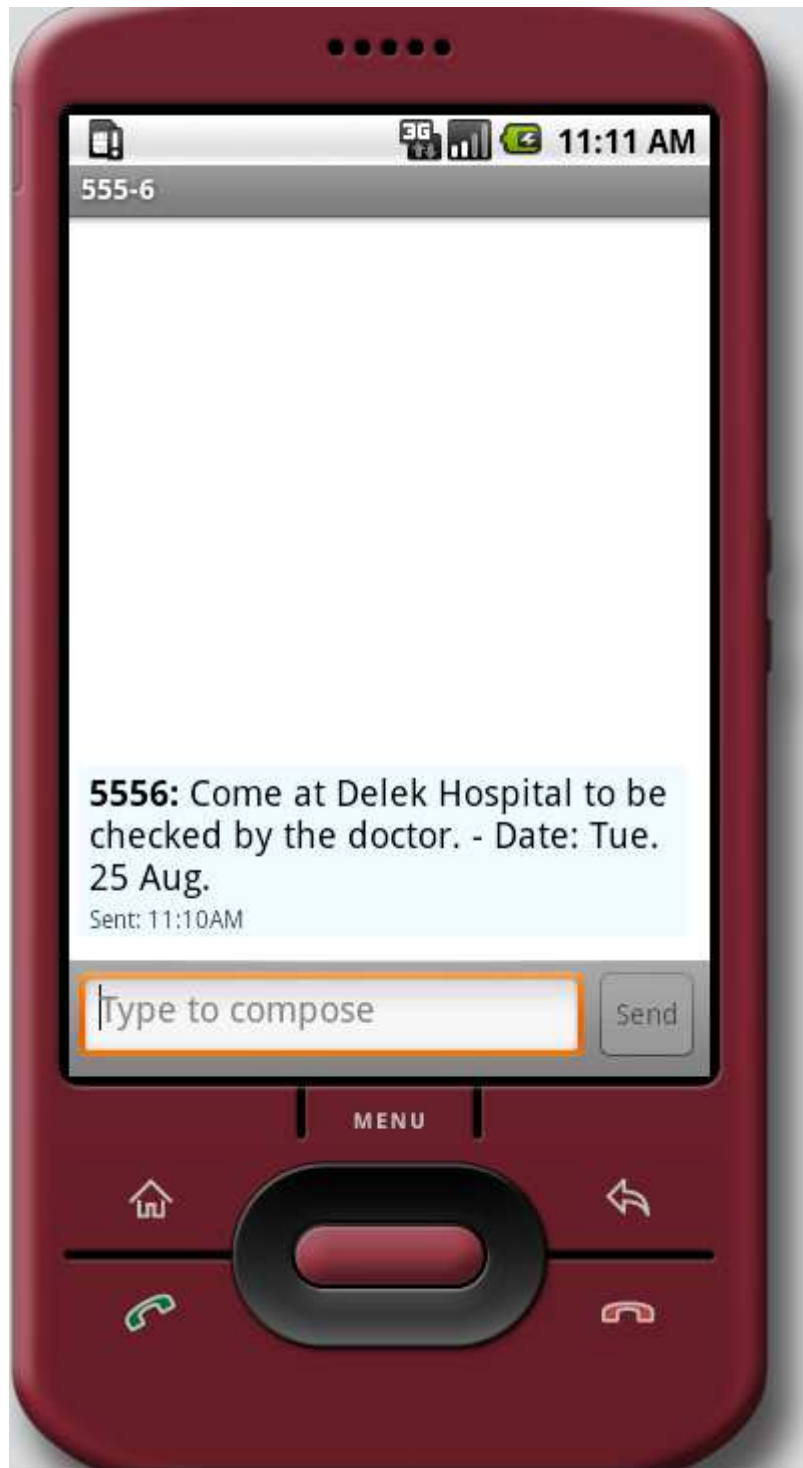
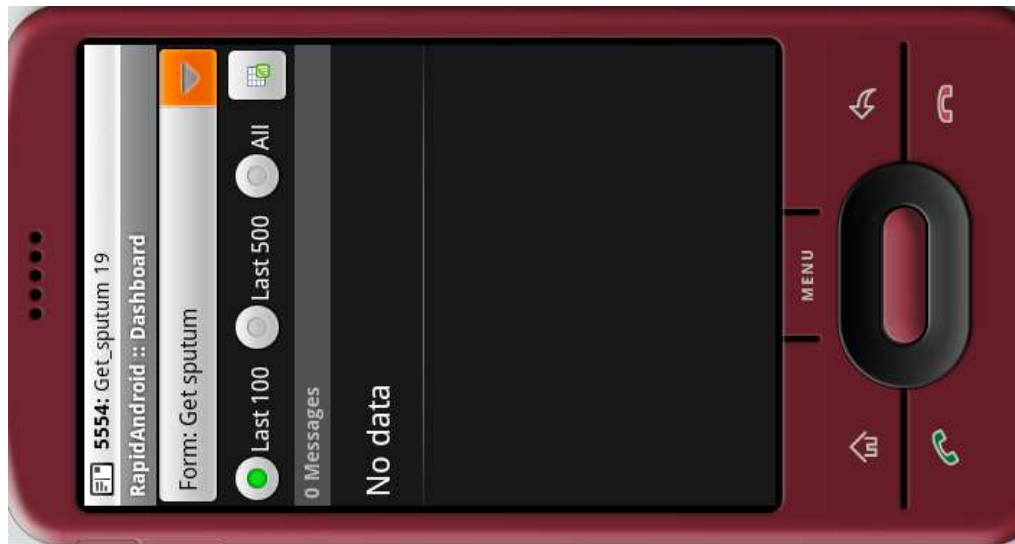


Figure 17. RapidAndroid reception of reminder on patient cell phone

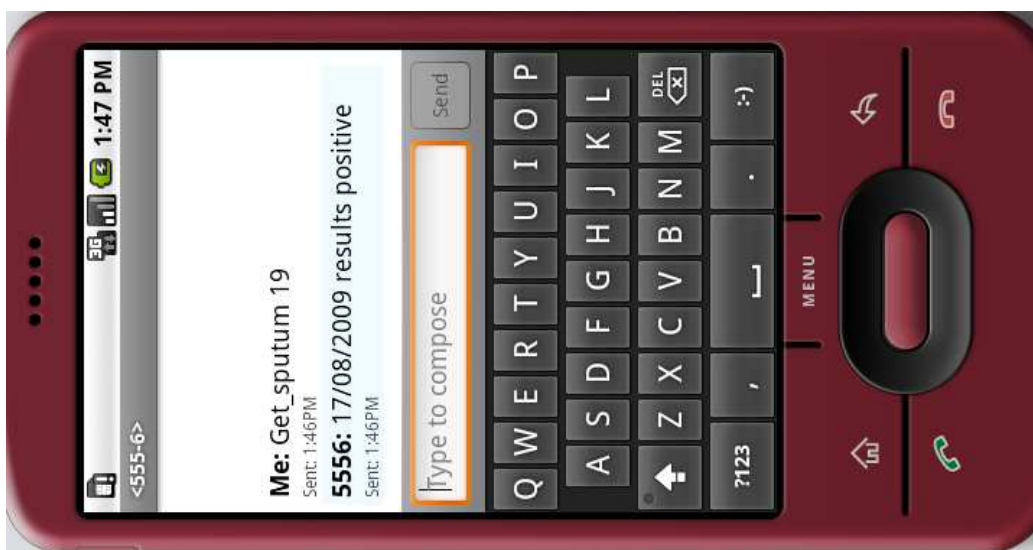
Rapport de stage



Step1: a nurse sends get_sputum with the number of the patient in order to obtain sputum analysis



Step2: reception and processing of the SMS on the RapidAndroid Server



Step3: the nurse receive the response with sputum results

Figure 18. Get_sputum test

add.jsp: form for adding a hospital in the system

```
<%@page import="javax.servlet.http.HttpSession"%>
<%@page import="javax.jcr.Session"%>
<%@page import="javax.jcr.Node"%>
<%@page import="javax.jcr.NodeIterator"%>
<%@page import="java.util.*"%>

<%@taglib prefix="slings" uri="http://slings.apache.org/taglibs/slings/1.0"%>
<slings:defineObjects/>

<html>
<head>

    <script type="text/JavaScript" SRC="/apps/tb_monitoring/scripts/places.js"></script>
    <script type="text/JavaScript" src="/system/slings.js"></script>
</head>

<body>
<%@ include file="/apps/tb_monitoring/verifIdent.jsp" %>
<%@ include file="/apps/tb_monitoring/header.jsp" %>
<%@ include file="/apps/tb_monitoring/left.jsp" %>

<% pageTitle = "Add an hospital"; %>

<div class="main">
    <h1><%= pageTitle %></h1>

    <form class="hform" method="POST" action="/content/tb_monitoring/places/*"
    enctype="multipart/form-data">
        <p><label>Name <span class="obligatory">*</span></label>
        <input name="name" type="text" size="80" value=""></p>

        <p><label>Email <span class="obligatory">*</span></label>
        <input name="email" type="text" size="80" value=""></p>

        <p><label>Number phone <span class="obligatory">*</span></label>
        <input name="number_phone" type="text" size="80" value=""></p>

        <h2><label>Address</label></h2>
        Choose the country: <span class="obligatory">*</span> <br/>
        <select name="country" id="country" onChange="changeStates(this.value);">
            <option value="null">- Choose the country - </option>
        </select>
        <%
            NodeIterator ni = currentNode.getNodes();
            Node nc;
            while (ni.hasNext()){
```

Rapport de stage

```
                nc=(Node) ni.next();
            %>
            <option value="<%= nc.getName() %>" ><%= nc.getName() %></option>
            <% } %>
        </select>
        <br/>

        <span id="blocStates"></span><br />
        <span id="blocDistricts"></span><br />

        <p><label>City <span class="obligatory">*</span></label>
        <input name="city" type="text" size="80" value=""></p>

        <p><label>Road <span class="obligatory">*</span></label>
        <TEXTAREA rows="2" cols="35" name="road"></TEXTAREA>

        <input name="sling:resourceType" type="hidden" value="tb_monitoring/places"/>

        <input type="hidden" name="created"/>
        <input name=":redirect" type="hidden" value="/content/tb_monitoring/posts.list.html"/>
        <input name=":operation" type="hidden" value="tb_monitoring_add_hospital"/>

        <br/><br/>
        <input type="submit" value="Post" class="button">
    </form>
    <p><span class="obligatory">*</span> : obligatory fields</p>
    <script>Sling.wizard();</script>
</div>
</body>
</html>
```


Rapport de stage

AddHospitalHandler.java: handler of the previous form

```
/**
 * @scr.component metatype="no" immediate="true"
 * @scr.service interface="org.apache.sling.servlets.post.SlingPostOperation"
 * @scr.property name="sling.post.operation" value="tb_monitoring_add_hospital"
 * @author Audrey COLBRANT
 */
public class AddHospitalHandler extends AbstractSlingPostOperation {

    @Override
    protected void doRun(SlingHttpServletRequest request, HtmlResponse response,
        List<Modification> changes) throws RepositoryException {
        System.out.println("*****");
        System.out.println("BEGINNING OF ADD HOSPITAL HANDLER");

        HttpSession sessionUser = request.getSession();
        Session session = request.getResourceResolver().adaptTo(Session.class);
        sessionUser.setAttribute("addHosp", null);

        if (session == null) {
            throw new RepositoryException("JCR Session not found");
        }

        Node root = session.getRootNode();

        // retrieve content of the form : request.getParameterMap() returns a
        Map(String, String[])
        Map paramMap=request.getParameterMap();
        boolean paramOk=true;
        // Analyse all parameters
        for (Iterator i = paramMap.keySet().iterator() ; i.hasNext() ; ){
            String key = (String) i.next();
            // skip sling operations (:redirect, :operation, created, ...)
            if(key.charAt(0)!=':' && !key.equals("created")) {
                if(paramMap.get(key)!=null && !(((String[])
paramMap.get(key))[0]).trim().equals("") && paramMap.get(key)!="null") {
                    System.out.println("key = " + key + " value = " +
((String[]) paramMap.get(key))[0]);
                }else{
                    System.out.println("Problem with parameter = " + key + "
value = " + ((String[]) paramMap.get(key))[0]);
                    paramOk=false;
                    sessionUser.setAttribute("addHosp", false);
                    throw new RepositoryException(key+" was not submitted");
                }
            }
        }

        if(paramOk) {
            // request is good, we can save data in repository
            sessionUser.setAttribute("addHosp", true);

            // Add 1 to the counter of hospitals, mutual exclusion made thanks
            to a mutex
            Node hosp =
            root.getNode("content/tb_monitoring/constants/hospitals");
            Long counter;
            synchronized(this) {
                counter=hosp.getProperty("count").getLong();
                counter++;
            }
        }
    }
}
```

Rapport de stage

```
        hosp.setProperty("count",counter);
    }

    // Retrieve content targeted
    Node target =
root.getNode("content/tb_monitoring/places"+"/"+request.getParameter("country")+
"/"+request.getParameter("state")+"/"+request.getParameter("district"));

    // create node and fill properties without those specific to sling
(:operation, :redirect, ...)
    Node hospital = target.addNode(Long.toString(counter));
    for (Iterator i = paramMap.keySet().iterator() ; i.hasNext() ; ){
        String key = (String) i.next();
        if(key.charAt(0)!=':') {
            hospital.setProperty(key,((String[])
paramMap.get(key))[0]);
            System.out.println("Add property = " + key + " value = "
+ ((String[]) paramMap.get(key))[0]);
        }else{
            System.out.println("Non added property = " + key + "
value = " + ((String[]) paramMap.get(key))[0]);
        }
    }

    // Node has to be referenceable for Medicin
}

System.out.println("END OF ADD HOSPITAL HANDLER");
System.out.println("*****");
}
}
```

Rapport de stage

AddPatientHandler.java: handler for adding a patient in the database

```
/**
 * @scr.component metatype="no" immediate="true"
 * @scr.service interface="org.apache.sling.servlets.post.SlingPostOperation"
 * @scr.property name="sling.post.operation" value="tb_monitoring_add_patient"
 * @author Audrey COLBRANT
 */
public class AddPatientHandler extends AbstractSlingPostOperation {

    @Override
    protected void doRun(SlingHttpServletRequest request, HtmlResponse response,
        List<Modification> changes) throws RepositoryException {
        System.out.println("*****");
        System.out.println("BEGINNING OF ADD PATIENT HANDLER");

        HttpSession sessionUser = request.getSession();
        Session session = request.getResourceResolver().adaptTo(Session.class);
        sessionUser.setAttribute("addPatient", false);

        if (session == null) {
            throw new RepositoryException("JCR Session not found");
        }

        [... verification on validity of fields ...]

        sessionUser.setAttribute("addPatient", true);

        Node root = session.getRootNode();

        // Add 1 to the counter of hospitals, mutual exclusion made thanks to a
        mutex
        Node hosp = root.getNode("content/tb_monitoring/constants/patients");
        Long counter;
        synchronized(this) {
            counter=hosp.getProperty("count").getLong();
            counter++;
            hosp.setProperty("count",counter);
        }

        // Retrieve content
        Node target = root.getNode("content/tb_monitoring/patients");
        Node patient = target.addNode(Long.toString(counter));
        patient.setProperty("name",name);
        patient.setProperty("surname",surname);
        patient.setProperty("address",address);
        patient.setProperty("date_birth",date_birth);
        patient.setProperty("sex",sex);
        patient.setProperty("email",email);
        patient.setProperty("number_phone",number_phone);
        patient.setProperty("ethnicity",ethnicity);
        patient.setProperty("left_tibet",left_tibet);
        patient.setProperty("mother_name",mother_name);
        patient.setProperty("mother_surname",mother_surname);
        patient.setProperty("father_name",father_name);
        patient.setProperty("father_surname",father_surname);
        patient.setProperty("contact_name",contact_name);
        patient.setProperty("contact_info",contact_info);

        // node illness of patient created
        Node illness = patient.addNode("illness");
```

Rapport de stage

```
// node time of patient created
String tmpName = date_beginIntensive.replace( '/', '_' );
Node time = illness.addNode(tmpName);
time.setProperty("occupation", occupation);
time.setProperty("height", height);
time.setProperty("category", category);
time.setProperty("date_beginIntensive", date_beginIntensive);
time.setProperty("date_endIntBeginCont", date_endIntBeginCont);
time.setProperty("date_endContinuation", date_endContinuation);
time.setProperty("exposure", exposure);
time.setProperty("additionnal_information", additionnal_information);
time.setProperty("type", type);
time.setProperty("hiv", hiv);
time.setProperty("date_hiv", date_hiv);
time.setProperty("date_hivTx", date_hivTx);
time.setProperty("site", site);
time.setProperty("site_precision", site_precision);

Node intensive=time.addNode("intensive");
Node drugs_int=null;
for(int i=0;i<drugs_intensive.length;i++){
    if(drugs_intensive[i]!=null){
        drugs_int=intensive.addNode(i+"");
        drugs_int.setProperty("path", drugs_intensive[i]);
    }
    drugs_int.setProperty("quantity", (request.getParameter("intensive_"+drugs_
intensive[i])!=null)?":request.getParameter("intensive_"+drugs_intensive[i]);
}

Node continuation=time.addNode("continuation");
Node drugs_cont=null;
for(int i=0;i<drugs_continuation.length;i++){
    if(drugs_continuation[i]!=null){
        drugs_cont=continuation.addNode(i+"");
        drugs_cont.setProperty("path", drugs_continuation[i]);
    }
    drugs_cont.setProperty("quantity", (request.getParameter("continuation_"+dr
ugs_continuation[i])!=null)?":request.getParameter("continuation_"+drugs_contin
uation[i]);
}

// node sputum of patient created
Node sputums = time.addNode("sputum");
Node sputum = sputums.addNode(date_sputum.replace( '/', '_' ));
sputum.setProperty("date_sputum", date_sputum);
sputum.setProperty("sputum_results", sputum_results);

// node xray of patient created
Node xrays = time.addNode("xray");
if (!(date_xray == null || date_xray.length() == 0)) {
    Node xray = xrays.addNode(date_xray.replace( '/', '_' ));
    xray.setProperty("date_xray", date_xray);
    xray.setProperty("xray_results", xray_results);
}

SimpleDateFormat formatter = new SimpleDateFormat ("dd/MM/yyyy");
String currentDate = formatter.format(new Date());

// node sputum of patient created
```

Rapport de stage

```
Node weights = time.addNode("weight");
Node weight_n = weights.addNode(currentDate.replace( '/', '_' ));
weight_n.setProperty("date_weight",currentDate);
weight_n.setProperty("weight",weight);

// node places of patient created
Node places = time.addNode("places");
// new place added
Node place = places.addNode(currentDate.replace( '/', '_' ));
// retrieve the hospital
Node hospital=root.getNode(hospitalRef);
place.setProperty("hospitalRef",hospitalRef);
place.setProperty("doctorRef",doctorRef);
place.setProperty("city",hospital.getProperty("city").getString());
place.setProperty("date_arrival",currentDate);

patient.setProperty("sling:resourceType","tb_monitoring/patients");

System.out.println("END OF ADD PATIENT HANDLER");
System.out.println("*****");
}
}
```

Rapport de stage

identVerif.jsp: verification of the user identity

```
<%
// verification of the user identity
// if a user try to access a page without being logged, he is redirected to the
posts list page

    if(session.getAttribute("stateLog")==null) {
%>
        <jsp:forward page="/content/tb_monitoring/posts.list.html"/>
<%
    }
%>
```

Rapport de stage

FormSmsStorage.java: sending a SMS from the Android Server to the Remote Repository

```
public class FormSmsStorage extends Activity{

private SmsManager sms = SmsManager.getDefault();

public FormSmsStorage() {
    super();
}

public void sendForm(Form form, Vector<IParseResult> param,String from) {
    if(form.getPrefix().toLowerCase().equals("reminder")){
        // google reminders case, specific processing
        sendToMobile(form, param);
    }else{
        // otherwise we send the message to the remote repository
        sendToRepository(form, param,from);
    }
}

private void sendToRepository(Form form, Vector<IParseResult> param, String
from) {
    System.out.println("beginning of send form to repository");

    HttpClient client = new HttpClient();
    client.getHostConfiguration().setHost(ServerConstants.LOGON_SITE,
ServerConstants.LOGON_PORT, "http");

    // set the url for the post method
    PostMethod authpost = new PostMethod("/content/tb_monitoring");

    // Prepare parameters
   NameValuePair[] tabParam = new NameValuePair[param.size()+2]; // +2 =
slings:resourceType and operation of the object we send
    Field[] formFields = form.getFields();
    // set the resourceType : script that will handle the display of the
resource
    tabParam[0]= new NameValuePair("slings:resourceType",
"tb_monitoring/"+form.getFormName());
    // set the title as the number
    tabParam[1]= new
NameValuePair(":operation","tb_monitoring_"+form.getPrefix());
    // set the parameters of the form
    for(int i=0;i<param.size();i++) {
        tabParam[i+2]=new
NameValuePair(formFields[i].getName(),param.get(i).getParsedToken());
    }

    // set the request body
    authpost.setRequestBody(tabParam);

    // Provide custom retry handler is necessary
    authpost.getParams().setParameter(HttpMethodParams.RETRY_HANDLER, new
DefaultHttpClientRetryHandler(3, false));

    try {
        // send the query
        int statusCode = client.executeMethod(authpost);
        if (statusCode != HttpStatus.SC_OK) {
```

Rapport de stage

```
        System.err.println("Method failed: " +
authpost.getStatusLine());
    }

    // Read the response body.
    byte[] responseBody = authpost.getResponseBody();
    sms.sendTextMessage(from, null, split(split(new
String(responseBody), "<div id=\"Message\">")[1], "</div>")[0], null, null);
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    System.out.println("Login form post: " +
authpost.getStatusLine().toString());
    System.out.println("Status code: " +authpost.getStatusCode()+" , texte :
"+ authpost.getStatusText()+" , line : "+ authpost.getStatusLine());

    // release any connection resources used by the method
    authpost.releaseConnection();

    System.out.println("ending of send form to repository");
}

private void sendToMobile(Form form, Vector<IParseResult> param) {
    // google reminders case, specific processing
    //param.get(0) = number phone of the patient
    //param.get(1) = purpose of the reminder
    //param.get(3) = Day_letters_abbreviation
    //param.get(4) = day_number
    //param.get(5) = month_letters_abbreviation
    String body_sms=param.get(1).getParsedToken()+" - Date:
"+param.get(3).getParsedToken()+" "+param.get(4).getParsedToken()+"
"+param.get(5).getParsedToken();

    sms.sendTextMessage(param.get(0).getParsedToken().toString(), null,
body_sms, null, null);
}

/*
 * Split the string given in first parameter with the second parameter
 * Returns the string splitted with the separator
 */
public String[] split(String string, String separator) {
    int start = 0;
    int index = string.indexOf(separator);
    List list = new ArrayList();

    while (index >= 0) {
        list.add(string.substring(start, index));
        start = index + separator.length();
        index = string.indexOf(separator, start);
    }
    list.add(string.substring(start));
    return (String[]) list.toArray(new String[list.size()]);
}
}
```


Rapport de stage

forms.json : forms definition in JSON

```
[{"pk": 1, "model": "rapidandroid.form", "fields": {"parsemethod": "simpleregex", "prefix": "get_sputum", "description": "Get sputum results", "formname": "Get sputum"}}, {"pk": 2, "model": "rapidandroid.form", "fields": {"parsemethod": "simpleregex", "prefix": "put_sputum", "description": "Put new sputum results", "formname": "Put sputum"}}, {"pk": 3, "model": "rapidandroid.form", "fields": {"parsemethod": "simpleregex", "prefix": "get_xray", "description": "Get xray results", "formname": "Get xray"}}, {"pk": 4, "model": "rapidandroid.form", "fields": {"parsemethod": "simpleregex", "prefix": "put_xray", "description": "Put new xray results", "formname": "Put xray"}}, {"pk": 5, "model": "rapidandroid.form", "fields": {"parsemethod": "simpleregex", "prefix": "put_outcome", "description": "Put new outcome", "formname": "Put outcome"}}, {"pk": 6, "model": "rapidandroid.form", "fields": {"parsemethod": "simpleregex", "prefix": "move", "description": "Movement of a patient", "formname": "Movement"}}, {"pk": 7, "model": "rapidandroid.form", "fields": {"parsemethod": "simpleregex", "prefix": "put_dot", "description": "Put new DOT observation", "formname": "Put DOT"}}, {"pk": 8, "model": "rapidandroid.form", "fields": {"parsemethod": "simpleregex", "prefix": "schedule", "description": "Get schedule of a doctorPut new sputum results", "formname": "Get schedule"}}, {"pk": 9, "model": "rapidandroid.form", "fields": {"parsemethod": "simpleregex", "prefix": "reminder", "description": "Google Agenda reminders handler", "formname": "Google Reminders"}]}
```

Rapport de stage

fieldtypes.json : field types definition in JSON

```
[{
  "pk": 1,
  "model": "rapidandroid.fieldtype",
  "fields": {
    "datatype": "word",
    "regex": "^[A-Za-z0-9|\\-|\\@|\\.]+(\\$|\\s)",
    "name": "Word"
  }
}, {
  "pk": 2,
  "model": "rapidandroid.fieldtype",
  "fields": {
    "datatype": "number",
    "regex": "^(\\d+)(\\$|\\s)",
    "name": "Number"
  }
}, {
  "pk": 3,
  "model": "rapidandroid.fieldtype",
  "fields": {
    "datatype": "float",
    "regex": "^(\\d+|\\d+\\.\\d+)(\\s*(kg|kilo|kilos))(\\$|\\s)",
    "name": "Weight"
  }
}, {
  "pk": 4,
  "model": "rapidandroid.fieldtype",
  "fields": {
    "datatype": "integer",
    "regex": "^(\\d+)(\\s*(cm|m|meter|meters))(\\$|\\s)",
    "name": "Height"
  }
}, {
  "pk": 5,
  "model": "rapidandroid.fieldtype",
  "fields": {
    "datatype": "float",
    "regex": "^(\\d+\\.\\d+|\\d+\\/\\d+|\\d+\\s*%|\\d+\\s*pct|\\d+\\.\\d+)",
    "name": "Ratio"
  }
}, {
  "pk": 6, "model":
  "rapidandroid.fieldtype",
  "fields": {
    "datatype": "integer",
    "regex": "^(\\d+)(\\s*(cm|m))(\\$|\\s)",
    "name": "Length"
  }
}, {
  "pk": 7,
  "model": "rapidandroid.fieldtype",
  "fields": {
    "datatype": "boolean",
    "regex": "^(t|f|true|false|y|no|yes|n|n0)(\\s|$)",
    "name": "Yes/No"
  }
}, {
  "pk": 8,
```

Rapport de stage

```
"model": "rapidandroid.fieldtype",
"fields": {
  "datatype": "string",
  "regex": "^(\")(.[^\"])+\"(\s|$)",
  "name": "String"
}
},{
  "pk": 9,
  "model": "rapidandroid.fieldtype",
  "fields": {
    "datatype": "comment",
    "regex": "^(\(\)(.[^\(|\)]+)(\(\))(\s|$)",
    "name": "Comment"
  }
}]
```

Rapport de stage

fields.json : fields definition in JSON

```
[{"pk": 1, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Number of the patient", "name":
"number_patient", "form": 1, "sequence": 1}},
{"pk": 2, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Number of the patient", "name":
"number_patient", "form": 2, "sequence": 1}},
{"pk": 3, "model": "rapidandroid.field",
"fields": {"fieldtype": 8, "prompt": "Date of the analysis", "name":
"date_sputum", "form": 2, "sequence": 2}},
{"pk": 4, "model": "rapidandroid.field",
"fields": {"fieldtype": 8, "prompt": "Results of the analysis", "name":
"sputum_results", "form": 2, "sequence": 3}},
{"pk": 5, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Number of the patient", "name":
"number_patient", "form": 3, "sequence": 1}},
{"pk": 6, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Number of the patient", "name":
"number_patient", "form": 4, "sequence": 1}},
{"pk": 7, "model": "rapidandroid.field",
"fields": {"fieldtype": 8, "prompt": "Date of the analysis", "name":
"date_xray", "form": 4, "sequence": 2}},
{"pk": 8, "model": "rapidandroid.field",
"fields": {"fieldtype": 8, "prompt": "Results of the analysis", "name":
"xray_results", "form": 4, "sequence": 3}},
{"pk": 9, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Number of the patient", "name":
"number_patient", "form": 5, "sequence": 1}},
{"pk": 10, "model": "rapidandroid.field",
"fields": {"fieldtype": 8, "prompt": "Outcome results", "name": "outcome",
"form": 5, "sequence": 2}},
{"pk": 11, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Number of the patient", "name":
"number_patient", "form": 6, "sequence": 1}},
{"pk": 12, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Number of the new hospital", "name":
"number_hospital", "form": 6, "sequence": 2}},
{"pk": 13, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Number of the new doctor", "name":
"number_doctor", "form": 6, "sequence": 3}},
{"pk": 14, "model": "rapidandroid.field",
"fields": {"fieldtype": 8, "prompt": "Date of arrival", "name": "date_arrival",
"form": 6, "sequence": 4}},
{"pk": 15, "model": "rapidandroid.field",
"fields": {"fieldtype": 8, "prompt": "Reason of moving", "name":
"reason_of_moving", "form": 6, "sequence": 5}},
{"pk": 16, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Number of the patient", "name":
"number_patient", "form": 7, "sequence": 1}},
{"pk": 17, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Number of the doctor", "name":
"number_doctor", "form": 8, "sequence": 1}},
{"pk": 18, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Phone number of the patient", "name":
"number_phone", "form": 9, "sequence": 1}},
{"pk": 19, "model": "rapidandroid.field",
"fields": {"fieldtype": 8, "prompt": "Purpose of the reminder", "name":
"content", "form": 9, "sequence": 2}},
{"pk": 20, "model": "rapidandroid.field",
```

Rapport de stage

```
"fields": {"fieldtype": 1, "prompt": "Fixed @", "name": "at", "form": 9,
"sequence": 3}},
{"pk": 21, "model": "rapidandroid.field",
"fields": {"fieldtype": 1, "prompt": "Day in letters of the event", "name":
"day_letter", "form": 9, "sequence": 4}},
{"pk": 22, "model": "rapidandroid.field",
"fields": {"fieldtype": 2, "prompt": "Day number of the event", "name":
"day_number", "form": 9, "sequence": 5}},
{"pk": 23, "model": "rapidandroid.field",
"fields": {"fieldtype": 1, "prompt": "Month of the event", "name": "month",
"form": 9, "sequence": 6}},
{"pk": 24, "model": "rapidandroid.field",
"fields": {"fieldtype": 5, "prompt": "Beginning of the event", "name": "begin",
"form": 9, "sequence": 7}},
{"pk": 25, "model": "rapidandroid.field",
"fields": {"fieldtype": 1, "prompt": "Fixed -", "name": "tiret", "form": 9,
"sequence": 8}},
{"pk": 26, "model": "rapidandroid.field",
"fields": {"fieldtype": 5, "prompt": "Ending of the event", "name": "end",
"form": 9, "sequence": 9}},
{"pk": 27, "model": "rapidandroid.field",
"fields": {"fieldtype": 9, "prompt": "Name of the agenda", "name": "agenda",
"form": 9, "sequence": 10}}]
```

Rapport de stage

ManageReminders.java: manage reminders in the Google Agenda Service

```
public class ManageReminders {  
  
public void addReminder(String agenda, String title, String content, String  
startDate, String endDate, int delay){  
    CalendarService myService = new CalendarService("exampleCo-exampleApp-1");  
    try {  
        // connexion to google agenda account  
        myService.setUserCredentials(email_address, password);  
  
        URL postUrl = new URL(agenda);  
        CalendarEventEntry myEntry = new CalendarEventEntry();  
        myEntry.setTitle(new PlainTextConstruct(title));  
        myEntry.setContent(new PlainTextConstruct(content));  
  
        // set starting and ending time for the event  
        DateTime startTime = DateTime.parseDateTime(startDate);  
        DateTime endTime = DateTime.parseDateTime(endDate);  
        When eventTimes = new When();  
        eventTimes.setStartTime(startTime);  
        eventTimes.setEndTime(endTime);  
        myEntry.addTime(eventTimes);  
  
        // insert in the agenda the new event  
        CalendarEventEntry insertedEntry = myService.insert(postUrl,  
myEntry);  
  
        // configuration on the reminder  
        Method methodType = Method.SMS;  
        Reminder reminder = new Reminder();  
        reminder.setMinutes(delay);  
        reminder.setMethod(methodType);  
  
        // add the reminder to the event previously inserted  
        insertedEntry.getReminder().add(reminder);  
        insertedEntry.update();  
  
    } catch (AuthenticationException e) {  
        e.printStackTrace();  
    } catch (MalformedURLException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } catch (ServiceException e) {  
        e.printStackTrace();  
    }  
    }  
}
```

Rapport de stage

places.js: scripts AJAX mixed to Sling used on the website

```
/**
 * ALL THE FOLLOWING SCRIPTS ALLOW TO CHARGE
 * DYNAMICALLY ELEMENTS IN PAGES OF THE WEBSITE
 */

/* Load countries from the id given in parameters in the bloc blocCountries,
otherwise reinitialize blocDistricts and blocStates */
function loadCountries(idr) {
if(idr=="medical"){
    var countriesList = Sling.getContent("/content/tb_monitoring/places/", 2);
    var form_s = 'Countries: <span class="obligatory">*</span> <br/>';

    form_s += '<select name="country" id="country"
onChange="changeStates(this.value);">';
    form_s += ' <option value="null">-- Choose the country --</option>';
    for(var i in countriesList) {
        if(countriesList[i].name){
            form_s += ' <option value="'+countriesList[i].name +' ">'+
countriesList[i].name +'</option>';
        }
    }
    form_s += '</select>';
}else{
    form_s = "";
    document.getElementById("blocDistricts").innerHTML = "";
    document.getElementById("blocStates").innerHTML = "";
}

    document.getElementById("blocCountries").innerHTML = form_s;
}

/* Load states from the id of the country given in parameters in the bloc
blocStates, otherwise reinitialize blocDistricts */
function changeStates(idr) {
if(idr != "null"){
    var statesList = Sling.getContent("/content/tb_monitoring/places/"+idr,
2);
    var form_s = 'States: <span class="obligatory">*</span> <br/>';

    form_s += '<select name="state" id="state"
onChange="changeDistricts(this.value);">';
    form_s += ' <option value="null">-- Choose the state --</option>';
    for(var i in statesList) {
        if(statesList[i].name){
            form_s += ' <option value="'+statesList[i].name +' ">'+
statesList[i].name +'</option>';
        }
    }
    form_s += '</select>';
}else{
    form_s = "";
    document.getElementById("blocDistricts").innerHTML = "";
}

    document.getElementById("blocStates").innerHTML = form_s;
}

/* Load districts from the id of the states given in parameters in the bloc
blocDistricts */
```

Rapport de stage

```
function changeDistricts(idr) {
if(idr != "null"){
    var districtsList =
Sling.getContent("/content/tb_monitoring/places/"+document.getElementById("count
ry").value+"/"+idr, 2);
    var form_d = 'Districts: <span class="obligatory">*</span> <br/>';
    if(document.getElementById("blocHospitals")){
        form_d += '<select name="district" id="district"
onChange="changeHospitals(this.value);">';
    }else{
        form_d += '<select name="district" id="district">';
    }
    form_d += ' <option value="null">-- Choose the district --</option>';
    for(var j in districtsList) {
        if(districtsList[j].name){
            form_d += ' <option value="'+districtsList[j].name +' ">'+
districtsList[j].name +'</option>';
        }
    }
    form_d += '</select>';
}else{
    form_d = "";
}

document.getElementById("blocDistricts").innerHTML = form_d;
}

/* Load hospitals from the id of the districts given in parameters in the bloc
blocHospitals */
function changeHospitals(idr) {
if(idr != "null"){
    var
path="content/tb_monitoring/places/"+document.getElementById("country").value+"/
"+document.getElementById("state").value+"/"+idr;
    var hospitalsList = Sling.getContent("/"+path, 2);
    var form_d = 'Hospitals: <span class="obligatory">*</span> <br/>';

    if(document.getElementById("blocDoctors")){
        form_d += '<select name="hospitalRef" id="hospitalRef"
onChange="changeDoctors(this.value);">';
    }else{
        form_d += '<select name="hospitalRef" id="hospitalRef">';
    }

    form_d += ' <option value="null">-- Choose the hospital --</option>';
    var i=0;
    for(var j in hospitalsList) {
        if(hospitalsList[j].name){
            form_d += ' <option value="'+path+"/"+j +' ">'+
hospitalsList[j].name +'</option>';
        }
    }
    form_d += '</select>';
}else{
    form_d = "";
}

document.getElementById("blocHospitals").innerHTML = form_d;
}
```


Rapport de stage

```
/* Load doctors from the id of the hospitals given in parameters in the bloc
blocDoctors */
function changeDoctors(idr) {
if(idr != "null"){
    var path="content/tb_monitoring/idents/medical";
    var doctorsList = Sling.getContent("/"+path, 2);
    var form_d = 'Doctors/Community Health Workers: <span
class="obligatory">*</span> <br/>';

    form_d += '<select name="doctorRef" id="doctorRef">';
    form_d += ' <option value="null">-- Choose the doctor --</option>';
    for(var j in doctorsList) {
        if(doctorsList[j].hospitalRef==idr){
            form_d += ' <option value="'+path+"/"+j +'>'+
doctorsList[j].name +'</option>';
        }
    }
    form_d += '</select>';

}
}

}else{
    form_d = "";
}

document.getElementById("blocDoctors").innerHTML = form_d;
}

/* if the patient is born in tibet, display an input field for indicating the
year of left tibet, otherwise delete it */
function bornTibet() {
if (document.getElementsByName("born_in")[0].checked == true) {
    var form_d = 'Year left Tibet: <span class="obligatory">*</span> <input
type="text" name="left_tibet">';
}
}
}

document.getElementById("blocBornIn").innerHTML = form_d;
}

/* display an input field for adding specification after a field */
function givePrecision(idr,inputName,blocName) {
if (idr == "other") {
    var form_d = 'Specify: <span class="obligatory">*</span> <input
type="text" name="'+inputName+'>';
}
}
}

document.getElementById(blocName).innerHTML = form_d;
}

/* display an input field for adding specification after site field */
function givePrecisionSite(idr) {
if (idr == "Both" || idr=="Extrapulmonary") {
    var form_d = 'Specify: <span class="obligatory">*</span> <input
type="text" name="site_precision">';
}
}
}

document.getElementById("blocSiteOther").innerHTML = form_d;
```

Rapport de stage

```
}

/* display drugs in link with the id of TB given in parameter */
function changeDrugs(idr) {
var form_d = '';
if(idr != "null"){
    var drugsList =
Sling.getContent("/content/tb_monitoring/categories/"+idr+"/drugs", 2);
    form_d += '<table>';
        form_d += '<tr>';
            form_d += '<td>Intensive phase</td>';
            form_d += '<td>Continuation phase</td>';
        form_d += '</tr>';

        for(var j in drugsList) {
            var drug = Sling.getContent("/"+drugsList[j], 2);
            form_d += '<tr>';
                form_d += '<td><input type="checkbox"
name="drugs_intensive" value="'+drugsList[j]+' " >'+drug.abbreviation+',
quantity: <input type="text" name="intensive_'+drugsList[j]+' " value="" > (X per
week)<br/></td>';
                    form_d += '<td><input type="checkbox"
name="drugs_continuation" value="'+drugsList[j]+' " >'+drug.abbreviation+',
quantity: <input type="text" name="continuation_'+drugsList[j]+' " value="" > (X
per week)<br/></td>';
                form_d += '</tr>';
            }
        form_d += '</table>';
    }else{
        form_d = "";
    }
document.getElementById("blocDrugs").innerHTML = form_d;
}

/* display drugs in link with the id of TB given in parameter */
function changeCheckDrugs(idr,datesNode) {
var form_d = '';
if(idr != "null"){
    var drugsList =
Sling.getContent("/content/tb_monitoring/categories/"+idr+"/drugs", 2);
    var drugsIntList = Sling.getContent(datesNode+"/intensive", 2);
    var drugsContList = Sling.getContent(datesNode+"/continuation", 2);
    form_d += '<table>';
        form_d += '<tr>';
            form_d += '<td>Intensive phase</td>';
            form_d += '<td>Continuation phase</td>';
        form_d += '</tr>';

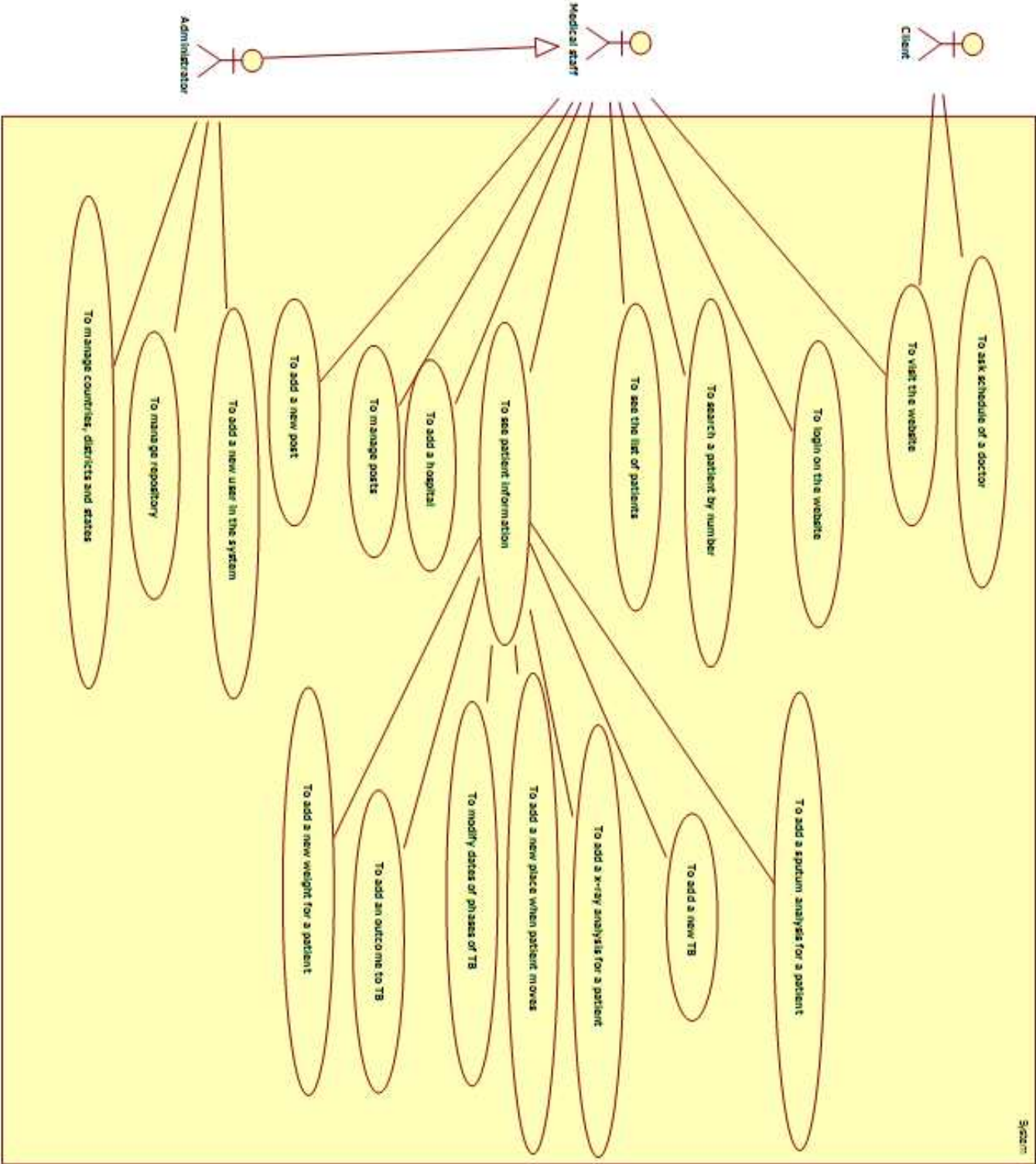
        for(var j in drugsList) {
            form_d += '<tr>';
            var drug = Sling.getContent("/"+drugsList[j], 2);
            var found=false;
            var quantity="";
            for(var k in drugsIntList){
                if(drugsIntList[k].path==drugsList[j]){
                    found=true;
                    quantity=drugsIntList[k].quantity;
                }
            }
            if(found){
```

Rapport de stage

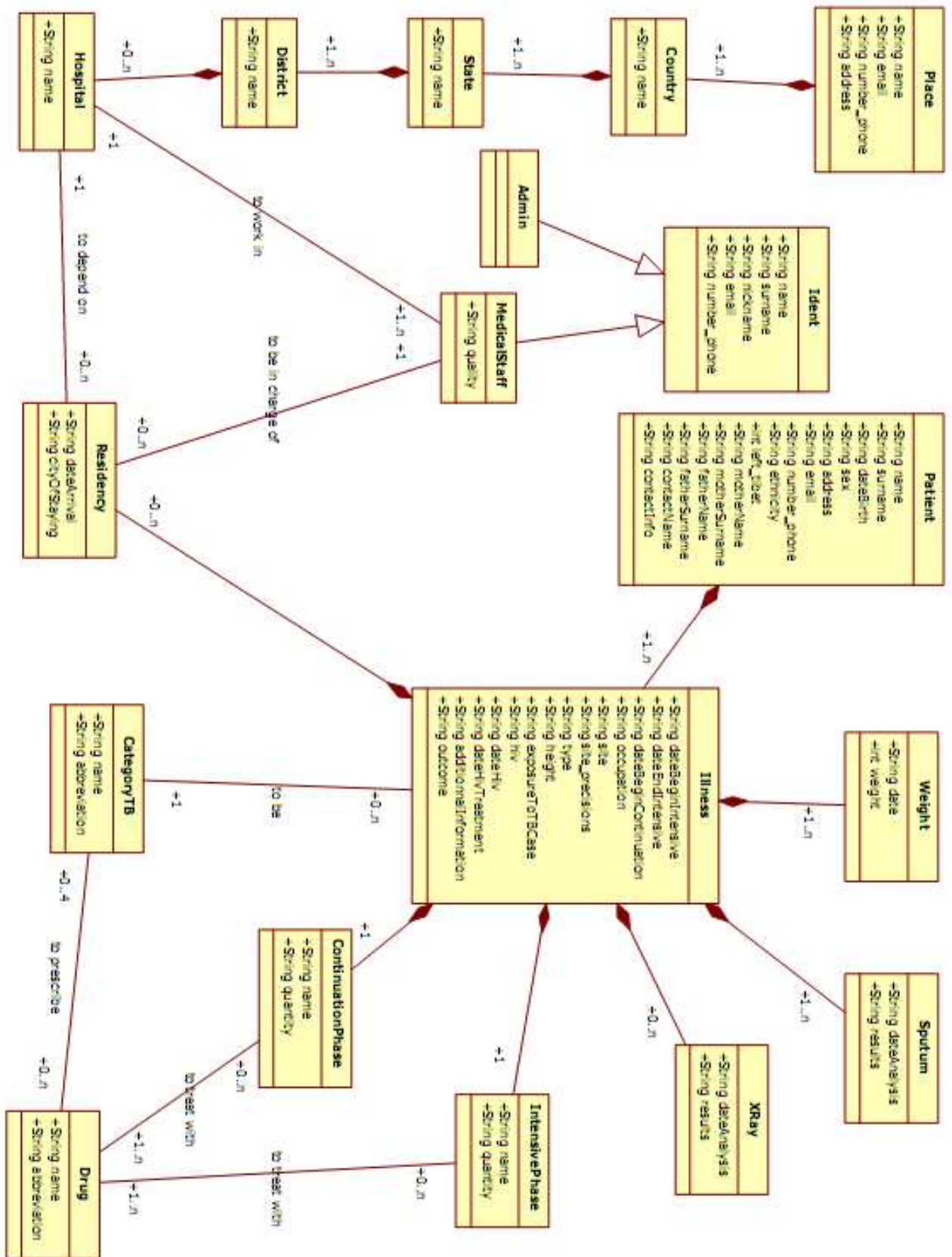
```
        form_d += '<td><input type="checkbox"
name="drugs_intensive" value="'+drugsList[j]+' " checked>'+drug.abbreviation+',
quantity: <input type="text" name="intensive_'+drugsList[j]+' "
value="'+quantity+' " > (X per week)<br/></td>';
        found=false;
    }else{
        form_d += '<td><input type="checkbox"
name="drugs_intensive" value="'+drugsList[j]+' " >'+drug.abbreviation+',
quantity: <input type="text" name="intensive_'+drugsList[j]+' " value="" > (X per
week)<br/></td>';
    }

    for(var k in drugsContList){
        if(drugsContList[k].path==drugsList[j]){
            found=true;
            quantity=drugsContList[k].quantity;
        }
    }
    if(found){
        form_d += '<td><input type="checkbox"
name="drugs_continuation" value="'+drugsList[j]+' "
checked>'+drug.abbreviation+', quantity: <input type="text"
name="continuation_'+drugsList[j]+' " value="'+quantity+' " > (X per
week)<br/></td>';
        found=false;
    }else{
        form_d += '<td><input type="checkbox"
name="drugs_continuation" value="'+drugsList[j]+' " >'+drug.abbreviation+',
quantity: <input type="text" name="continuation_'+drugsList[j]+' " value="" > (X
per week)<br/></td>';
    }
    form_d += '</tr>';
}
form_d += '</table>';
}else{
    form_d = "";
}
document.getElementById("blocDrugs").innerHTML = form_d;
}
```

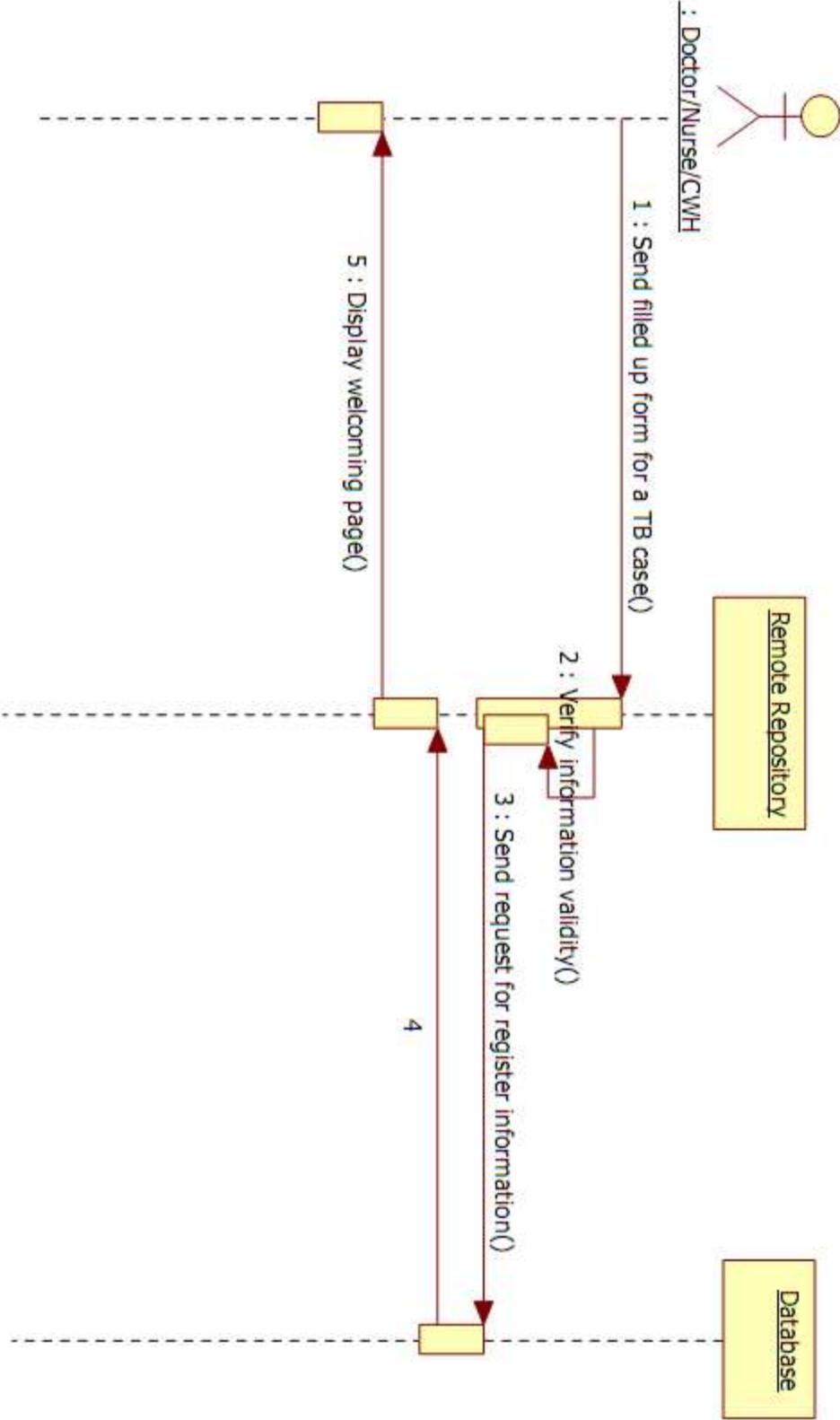
TECHNICAL APPENDIXES



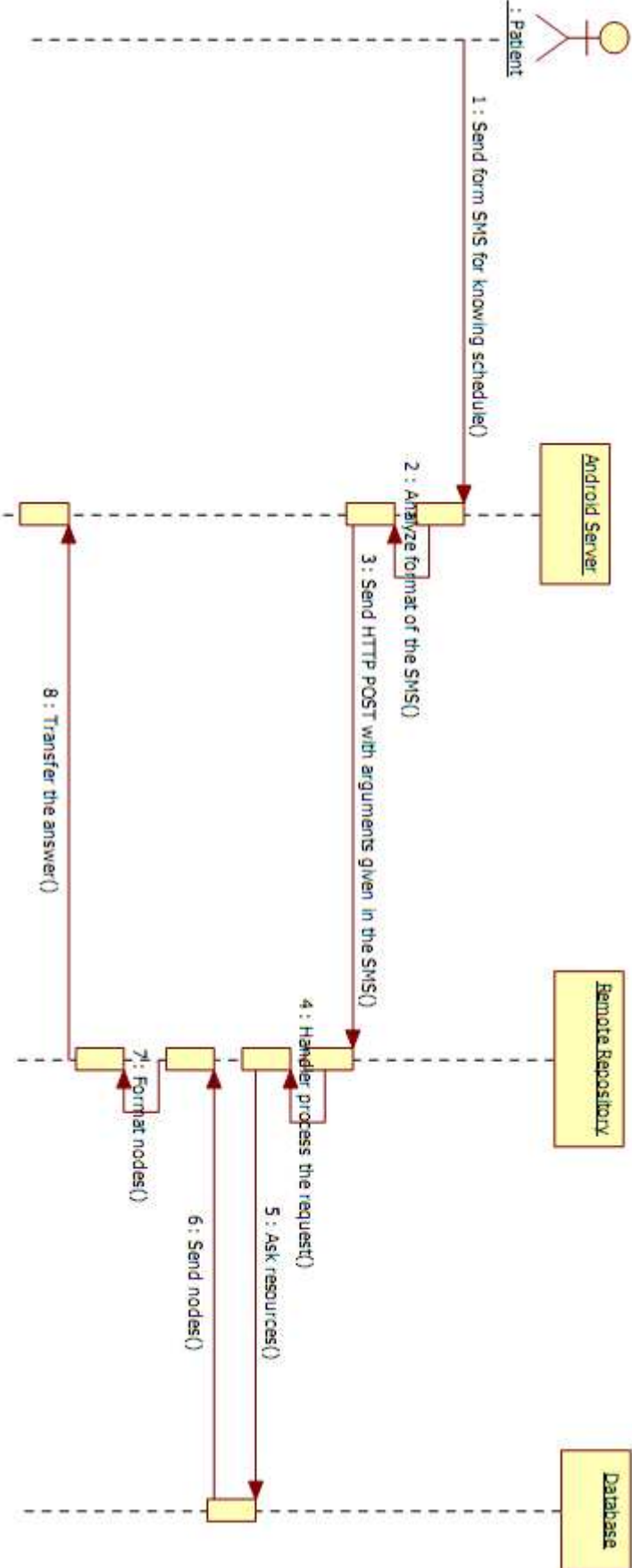
Use case diagram



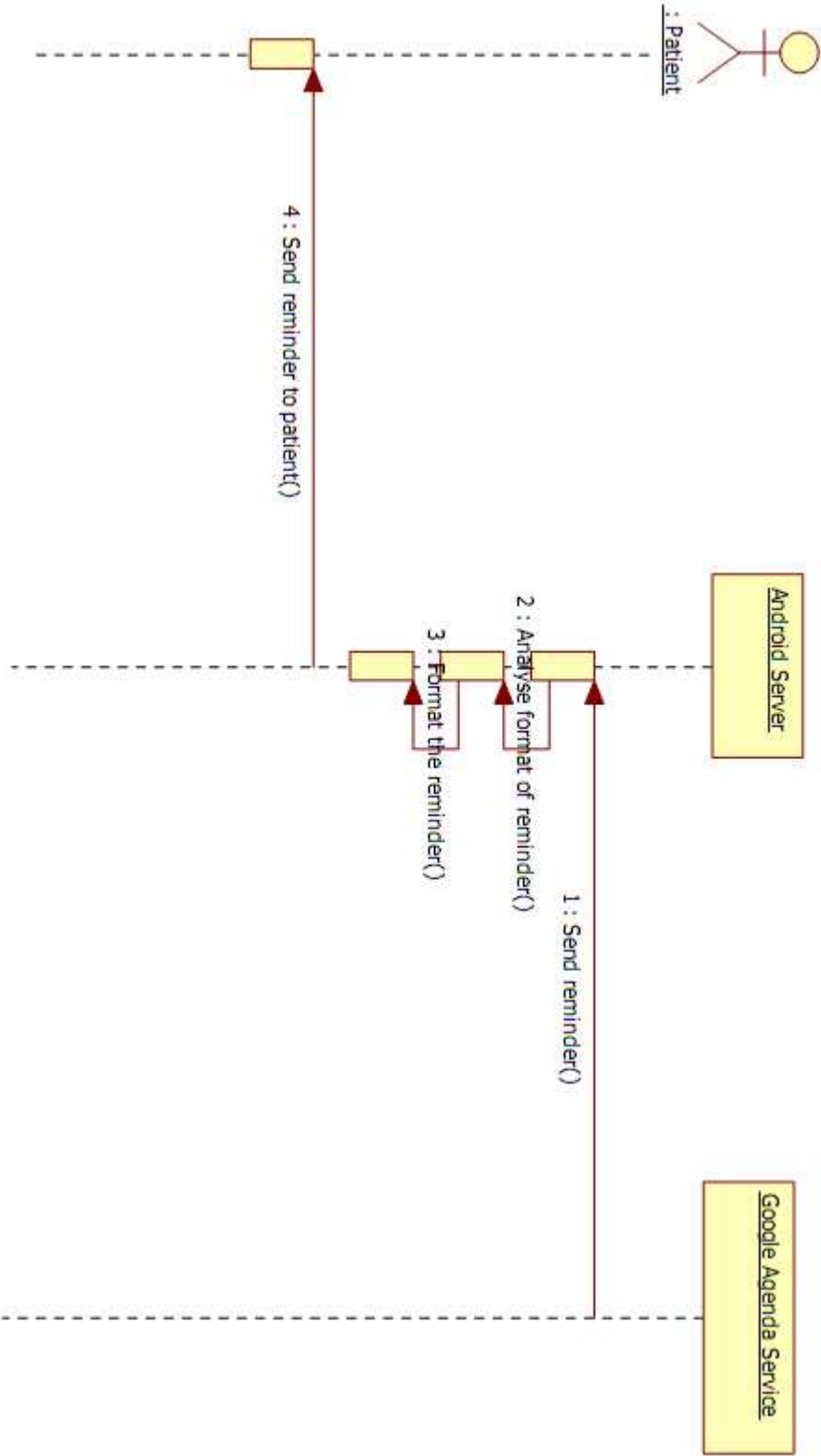
Class diagram



Sequence diagram: add a new TB for a patient



Sequence diagram: ask for the schedule of a doctor



Sequence diagram: receive a reminder

LET'S ROAMING IN DHARAMSALA

Dharamsala this is...



... buildings piled on the mountain side ...



... a bouddhist monastery ...



... a city at the rhythm of the Om Mani Padme Hum mantra, turning wheels while reciting ...



... but this is also the Tibetan Children Village (my office was on the right) ...



... finally, and maybe the more important here, Dharamsala, this is a peaceful fight for freedom...
... Never Give Up ...

Rapport de stage

Période du stage : 08/04/09 au 08/09/09

Etudiant (nom et prénom) : COLBRANT Audrey

Année d'étude dans le département : RICM3

Entreprise : INRIA

Adresse complète :

655 Avenue de l'Europe
38334 Saint Ismier Cedex
FRANCE

Téléphone (standard) : +33 4 76 61 52 00

Télécopie +33 4 76 61 52 52

Entreprise : Tibtec

Adresse complète :

TCV School - Head office
Upper Dharamsala - H.P. 179216
INDIA

Téléphone (standard) : +91 1892 221685

Télécopie +91 1892 221 670

Responsable administratif (nom et fonction) :

Mr. COSNARD Michel (PDG)

Téléphone : +33 4 76 61 52 00

Télécopie +33 4 76 61 52 52

Mél : michel.cosnard@inrialpes.fr

Responsable administratif (nom et fonction) :

Mr. DORJEE Phuntsok (CEO)

Téléphone : +91 1892 221685

Télécopie +91 1892 221 670

Mél : phuntsok@tcv.org.in

Maître de stage (nom et fonction) :

Mr. LEMORDANT Jacques (enseignant chercheur)

Téléphone : +33 4 76 61 54 27

Télécopie +33 4 76 61 52 52

Mél : jacques.lemordant@inrialpes.fr

Maître de stage (nom et fonction) :

Mr. DORJEE Phuntsok (CEO)

Téléphone : +91 1892 221685

Télécopie +91 1892 221 670

Mél : phuntsok@tcv.org.in

Tuteur enseignant (nom et fonction) : Mr. LEMORDANT Jacques (enseignant chercheur)

Téléphone : +33 4 76 61 54 27

Mél : jacques.lemordant@inrialpes.fr

Télécopie :

Titre : Projet Tuberculose - Une architecture de substitution pour monitorer la tuberculose

Résumé :

Ce rapport de stage fait état des 24 semaines de travail effectué dans le cadre du stage de fin de cursus Réseaux Informatiques et Communication Multimédia de Polytech'Grenoble.

Il s'est déroulé pour sa première partie en France à l'INRIA et pour la seconde en Inde pour l'organisation à but non lucratif Tibtec afin de réaliser un logiciel de monitoring de la tuberculose grâce à des technologies bas coût.

Il consistait à la mise en place d'une solution centralisée des données pour pouvoir mieux combattre la tuberculose grâce à un système pointu de statistiques pour en avoir une compréhension accrue tout en prenant en compte les problèmes récurrents en Inde. Il a également été demandé d'offrir une plateforme d'information et de gestion des données patients pour le corps médical, ainsi que les patients afin de connaître l'emploi du temps d'un médecin se trouvant à l'autre bout de l'Inde. Enfin, un système de rappels pour la prise de médicaments et/ou de visite chez le médecin des patients a été sollicité.

La solution consistait tout d'abord en la mise en place d'un serveur distant stockant les données patients et utiles à l'application, accessibles via un gateway transportant des requêtes HTTP POST/GET répondant au framework REST traitées par des handlers spécifiques.

Dans un second temps, elle consistait en le développement d'accesseurs à ce serveur distant, à savoir un serveur d'analyse de SMS tournant sur la dernière version d'Android ainsi qu'un site web.

Enfin, un service de rappels aux patients est géré grâce au service Google Agenda qui envoie des rappels sur un portable à une date et heure précise au serveur Android qui les transférera aux patients.