TrackMe

Requirments Analysis and Description Document
Desing Document

Goals

- **G3.1:** Allow third parties to access information of specific individuals (through an identifier).
- **G3.2**: Allow third parties to access anonymized information of groups of individuals.
- **G3.3**: Allow third parties to subscribe to new information of a specific individual and to receive it.
- **G4.1**: Guarantee elderly users to receive an immediate assistance by an ambulance in case of high risk disease.

Domain Assumptions

- **D5-D6-D7:** the parameters periodically received by users' devices and the provided personal information and clinical data are assumed to be correct.
- **D8-D9-D10-D11:** users' devices support the mobile application, the GPS technology, sensors to retrieve health parameters and they can communicate through the internet.
- **D12-D13:** the Ambulance Dispatching System is always available and, in case of emergency, all relevant data are correctly reported to it.

Requirements

- **R9-R10-R11:** the system must allow third parties to search users through their SSN and to send them access request. It also must allow users to accept/reject requests.

- **R12-R13-R14:** the system must allow third parties to perform sampling searches, but it must accept and anonymise only those that returrn at least 1000 results.

- **R15-R16:** if a user accepts a requests, the system must allow the third party to access user's data and it must notify the third party whenever new data are available.

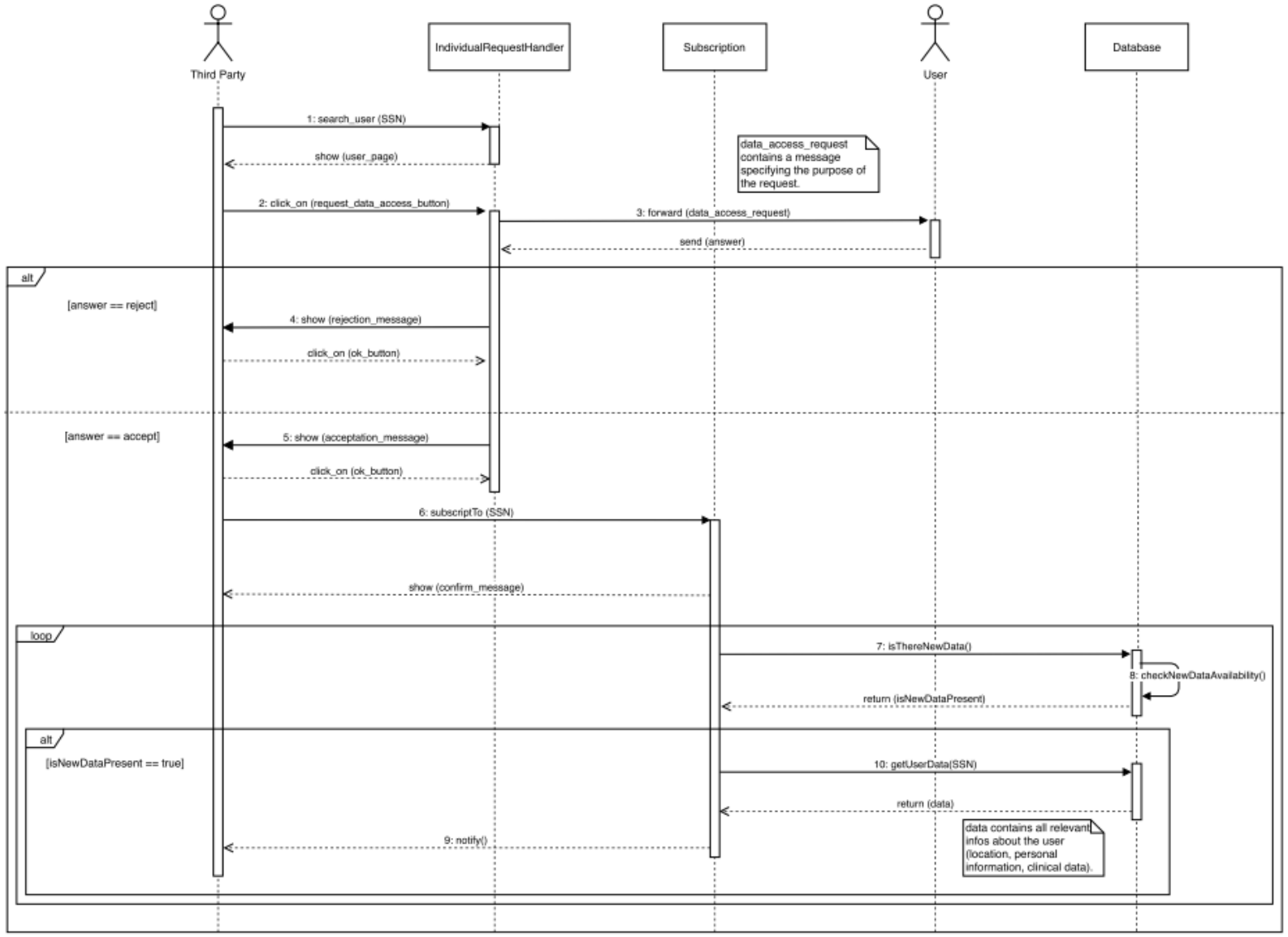
- **R17-R18-R19:** the AutomatedSOS service is automatically activated during the registration process for elderly users and the system must notify the Ambulance Dispatching System as soon as possible when it is needed, sending all relevant data.

Relevant Use Case

NAME	Accept Request
ACTOR	User
GOALS	[G3.1]
ENTRY CONDITION	The User has received an access request from a Third Party
EVENTS FLOW	1. The User receives a request from a Third Party 2. The User clicks on "Accept" button
EXIT CONDITIONS	The User has successfully accepted the request and the Third Party can access to his/her data
EXCEPTIONS	None

NAME	Subscription to User's Data
ACTOR	Third Party
GOALS	[G3.3]
ENTRY CONDITION	The User has accepted Third Party's access request
EVENTS FLOW	1. Third party clicks on "Subscribe to User's new data" button 2. Third party confirms the subscription
EXIT CONDITIONS	The Third Party has successfully subscripted to new User's data
EXCEPTIONS	None

Sequence Diagram



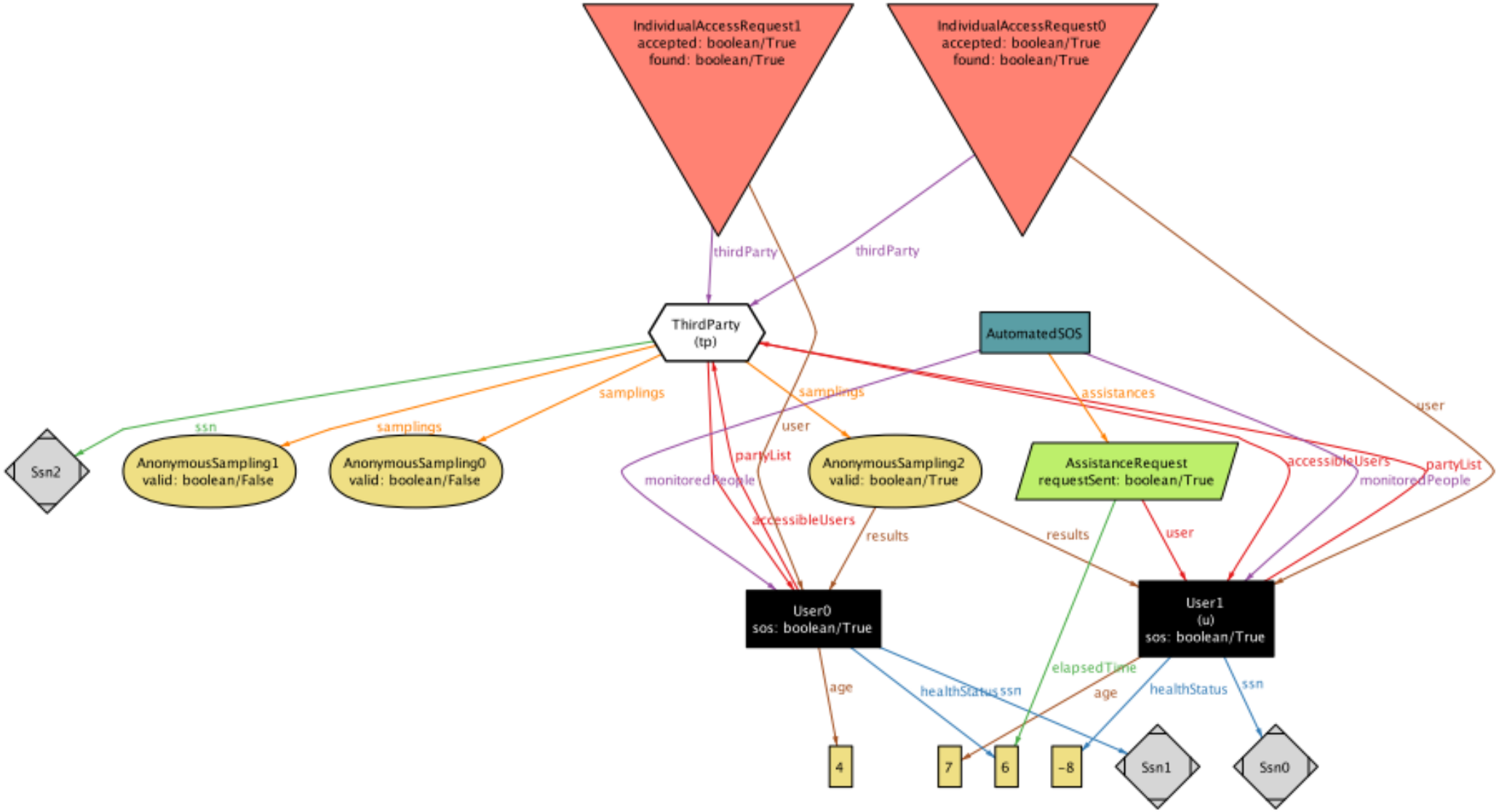
Alloy Model

- All the users that receive an automated assistance are necessarily subscribed to the AutomatedSOS service. Furthermore, users cannot receive multiple assistances at the same time and an automatic assistance is generated whenever his/her health status gets below the critical threshold (here represented by 0). All the users with an age grater or equal to 60 (here represented by 6) are automatically registered to AutomatedSOS, but this doesn't preclude younger users to subscribe to the service too. If an automated assistance request is generated, then it must be sent to the Ambulance System within 5 seconds.

- All the third parties in order to have access to some users' data, have to send a request to the specific user, that can be either accepted or rejected. Only if the user accepts the request, then the third party can access all his/her data. If the access is granted, then the third party has the possibility to subscribe to user's new data (but it is not necessary).

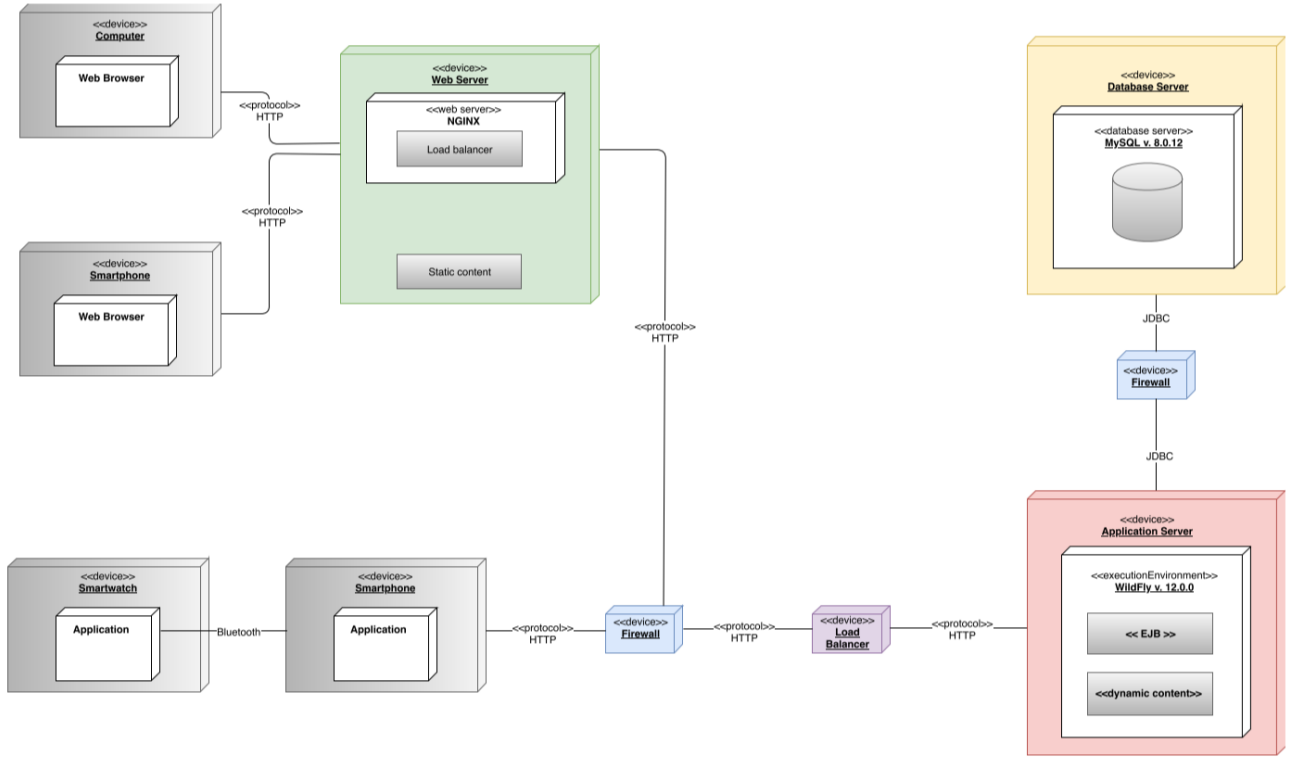
- Third Parties have the possibility to do sample searches according to some filters. TrackMe validates only those searches that result in more than 1000 users (here represented by 1).

Alloy World


IndividualAccessRequest1
accepted: boolean/True
found: boolean/True
IndividualAccessRequest0
accepted: boolean/True
found: boolean/True
thirdParty
thirdParty
ThirdParty
(tp)
AutomatedSOS
ssn
samplings
samplings
samplings
user
partyList
monitoredPeople
accessibleUsers
results
assistances
accessibleUsers
partyList
monitoredPeople
user
results
user
Ssn2
AnonymousSampling1
valid: boolean/False
AnonymousSampling0
valid: boolean/False
AnonymousSampling2
valid: boolean/True
AssistanceRequest
requestSent: boolean/True
User0
sos: boolean/True
User1
(u)
sos: boolean/True
age
healthStatus
ssn
elapsedTime
age
healthStatus
ssn
4
7
6
-8
Ssn1
Ssn0

Architectural Styles

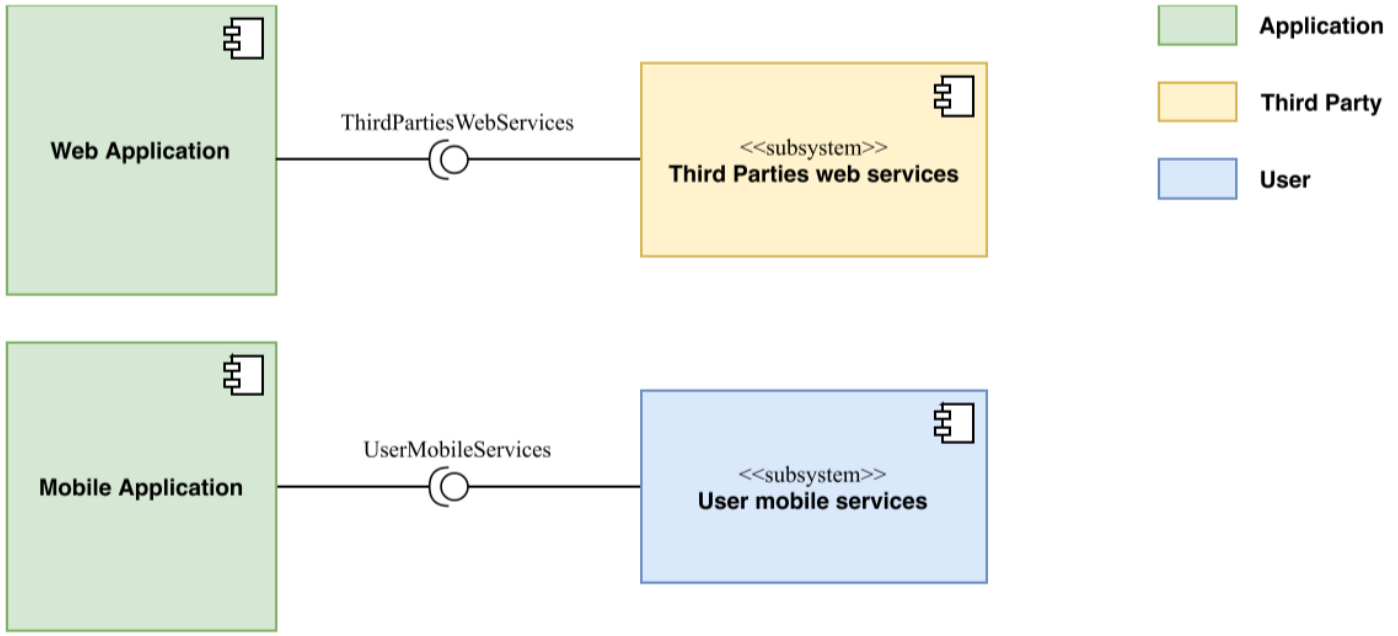
- The Application is based on a 4 tier Architecture
- **Tier 1**: composed by the Client (thin) including the Web Application (run by Web Browser) and the Mobile Application (run by Smartphone)
- **Tier 2**: composed by the Web Server (NGINX)
- **Tier 3**: composed by the Application Server (creates Web Apps and environment in which they can be run)
- **Tier 4**: composed by the Dabase Server on which the DBMS is running



System Components and Interfaces

The following diagrams illustrate the system components and the interfaces through which they interact to fullfil their functionalities. A distinction can be made between Client side and Server side:

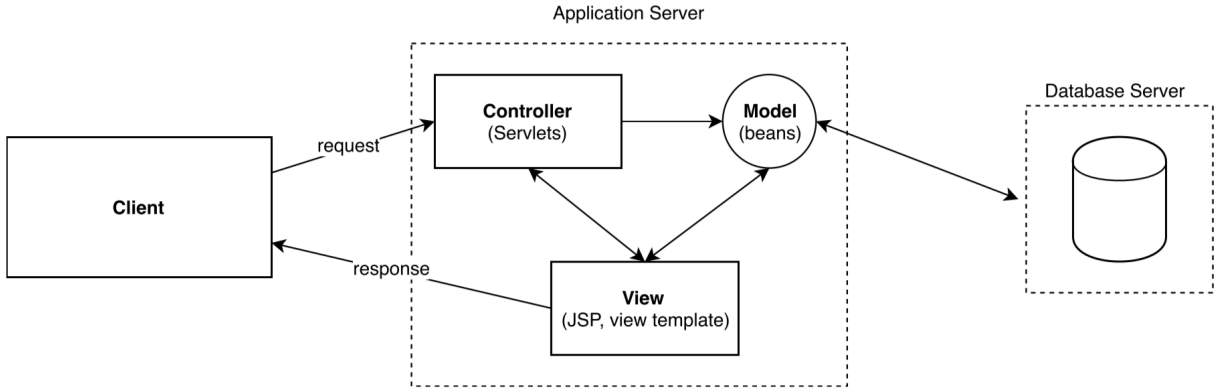
- The Client side is composed by two components, ***Web Application*** and ***Mobile Application***, referring to the following services: ***ThirdPartiesWebServices*** and ***UserMobileServices***.
- The Server side is composed of two main components, ***Third Parties web services*** that will provide functionalities aiming to fullfill third parties needs and ***User mobile services*** that will provide an interface to User in order to manage his own profile, check information about his health status, accept/reject requests, etc.



Design Patterns

In order to better formalize our architecture and make it as flexible as possible and speed up the devolpment process of our system we used different design patterns:

- **Model View Controller**: the main key of this pattern is to create a separation between the User Interface (View), the data (Model) and the response and validity checking of the User's input (Controller)



- **Observer and Observable**: Observable objects changes are obserbed by Observer objects and such changes are notified to the User, refreshing the UI
- **Façade Pattern**: this pattern supports loose coupling. We emphasize the abstraction and hide complex details by exposing a single interface instead of multiple ones

Other Design Decisions

- For what concerns the Mobile App, we decided not to provide a cross-platform web based application.
- Due to the fact that our application must interface with some wearable devices, we prefer providing native applications, both for Android and iOS.
- Doing so the developed applications can guarantee a better performance during their execution and a better user experience: a user who is used to, for example, the iOS graphical interface, will feel more comfortable using an app with the same type of GUI components.
- Furthermore, native applications guarantee their optimal integration with one another and with the entire ecosystem (referring for example to an Apple Watch paired to an iPhone device).

Implementation, Integration and Test Plan (1)

The implementation of our system will be done component by component and module by module, according to a bottom-up approach that will facilitate a high deployment coverage in early phases.

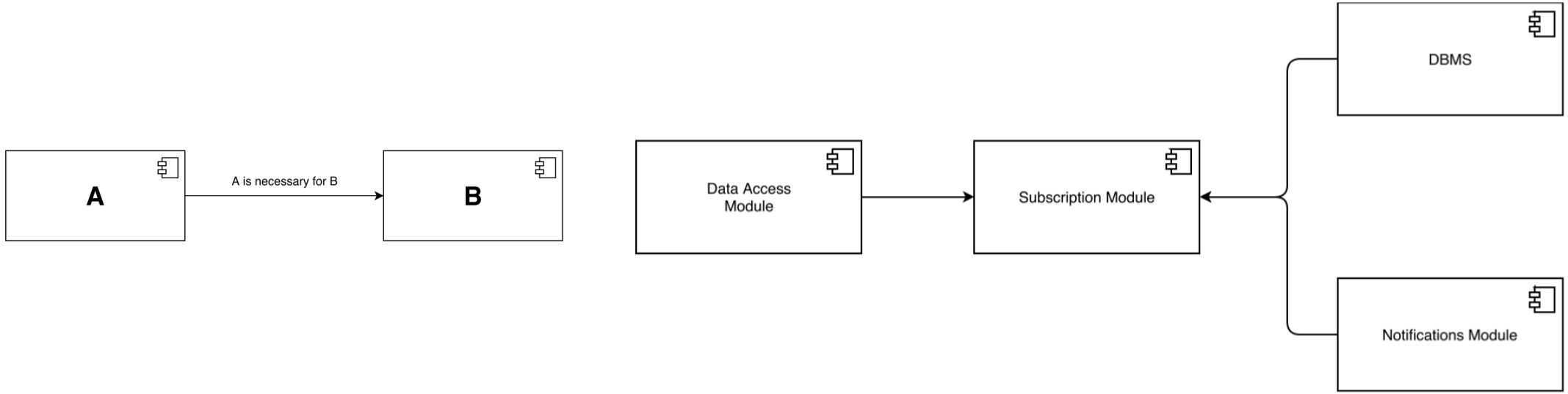
System components can be grouped up as follows:

- **Model**: this component is the first one that must implemented since all the parts of the Application Server will be using its elements and it allows user services to communicate with the DBMS
- **User Mobile Services**: this component is the most complex and important one due to its functionalities of providing all the services necessary for the User
- **Third Party Web Services**: likewise the previous module, its functionalities provide all the services necessary for the Third Parties

Implementation, Integration and Test Plan (2)

The system is composed by the components previously described and will be integrated as follows:

- **Integration of the components with the DBMS**
- **Integration of the components with external services**
- **Integration of the components of the Application Server**
- **Integration of the Client side with the Application Server**



Implementation, Integration and Test Plan (3)

- We opted to use a bottom-up approach for the testing phase
- In the very first place we will start by integrating those components that do not depend on other ones in order to work properly, or those who depend on already developed components (such as external API's).
- After all the single components are tested, we will end up by the subsystems the previous components take part of.
- This allows us to begin to test a component or a subsystem as soon as it's finished (or near completion), allowing us to (partially) parallelize the development phase and the testing phase.