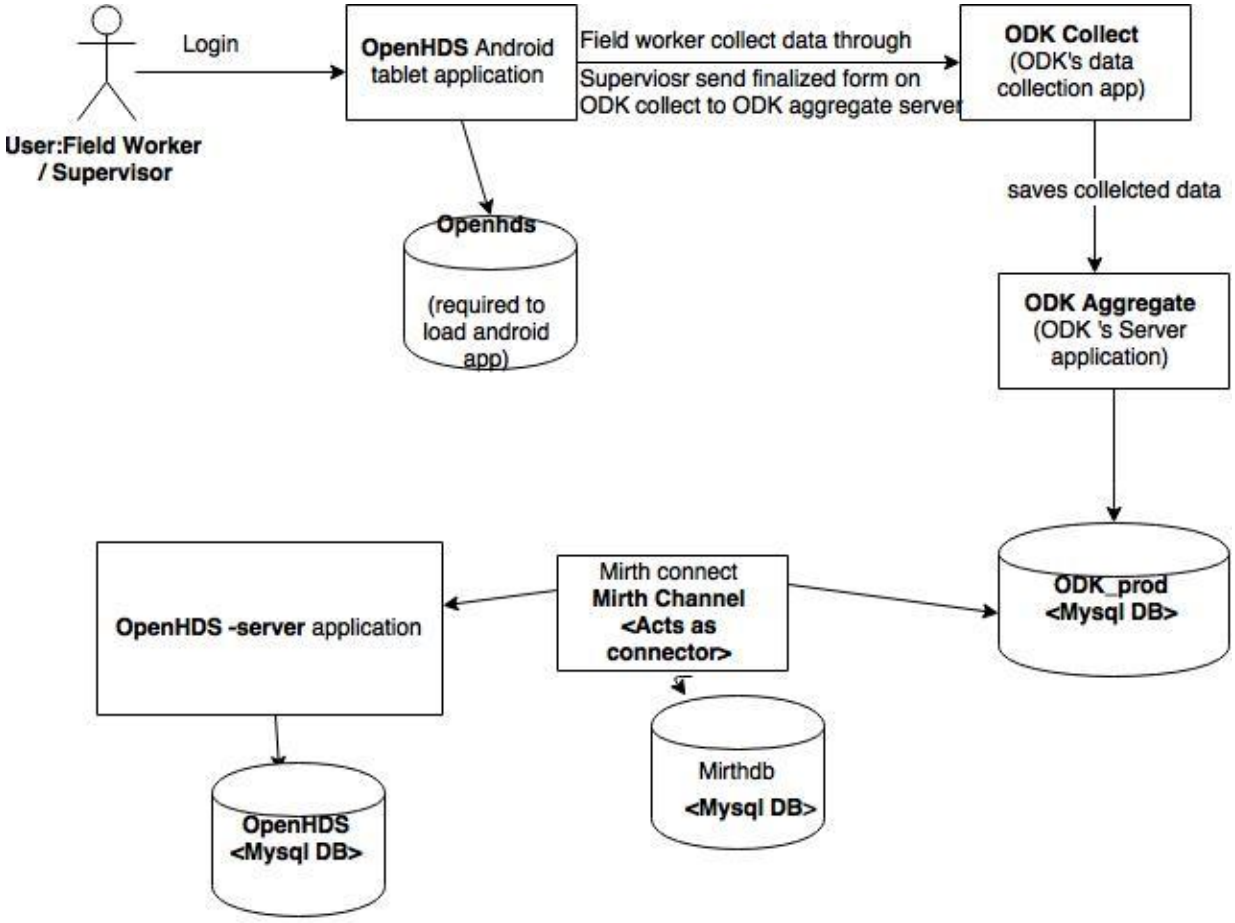
OpenHDS Overview:

The OpenHDS project was implemented as the part of the 'Campaign Information Management System (CIMS) 'of the Bioko Island. The primary goal of the project is to help Bioko Island to combat Malaria by providing an information system to collect and organize the data during the various health campaigns/surveys. The following are the main components of the OpenHDS Information System:

1. OpenHDS server side application :
2. Mobile /tablet OpenHDS app
3. ODK collect and ODK aggregate
4. Mirth Service

The general architecture of OpenHDS system is as shown in the diagram:



WorkFlow:

The registered Field Workers login to the tablet OpenHDS application. They record the data during the health census activities of the Bioko Island. The data is gathered using the data

collection tool called ‘ODK collect’ which is configured with the tablet application. Field workers submit the ODK collect forms on the tablet application. A user called supervisor has an authority to synchronize all the submitted forms in order to save these forms over the ODK aggregate. It is a server side application which is the part ‘Open Data Kit’ that acts as the repository of submitted forms. The OpenHDS system uses the software component called Mirth Service. The main purpose of Mirth Service is to connect two software pieces i.e. ODK and OpenHDS server side application. Mirth channels fetch the data from the ODK aggregate server and send it to the OpenHDS-server.

OpenHDS server is an application build using the JAVA EE , Spring framework, hibernate and web technologies like html and JSF. The goal of OpenHDS server application is to model and store the data send by Mirth service. It has following main modules:

Web: It contains controllers to handle web requests, navigation through web pages and database CRUD operations for the web forms. This module also contains the logic for validation, data conversions, user authentication for the data security purpose and UI.

Domain: This module contains classes to model the data for the entities in the OpenHDS system such as Location, Individual. The data for the entities is validated using the user implemented constraints, java inbuilt annotations and validations packages.

Dao: This layer helps to talk with the database. It uses the Hibernate ORM technology to map data entities with the corresponding tables in the MYSQL database. It also contains the logic for the database CRUD operations and finder method to locate ‘Named queries’.

Report: This module constitutes POJO classes to model the data for the reports such as immigration, household and web controllers to handle the requests for such reports.

The data stored in the database can be converted back to the XML forms. These forms are then pulled back for the tablet applications which enable the field worker to fetch the saved state of forms.

Spring definitions used in OpenHDS:

OpenHDS-server application uses the spring web framework to organize the application in modular fashion and to reduce the dependencies between different components. The different modules have their own spring bean configuration xml and set of defined beans. Spring beans have been used to inject the dependency between different modules such Dao, Domain, controllers. Each module is further divided into sub-modules and these also are wired together using the spring injection styles such auto wiring, annotations, setter-based and constructor injections.

Different beans have been defined to establish the database connection in daoapplicationContext.xml file. We used the apache common BasicDataSource for the database connection pool and spring destroy-method attribute to release the database connection. Spring automatically calls this method which reduces complicated coding required to release the database connection resources. Spring’s property attribute were used to pass the connection

details required to create data source object. We used Hibernate's 'LocalSessionFactoryBean' to create sessionFactory bean. We then injected data source object and annotated classes in the domain module using the property attributes and annotations. This pattern helped to separate the domain and Dao layer. Thus, it reduces the dependency between these two layers.

Example of the one-to-many relations:

In the OpenHDS-server application, we can see many examples of onetoMany relationships such location – residencies, residency-individuals, individual- relations, Location-residencies in the domain module. In order to map One to many relationships among the entities, we used hibernate ORM and annotation @OneToMany. We mark the entities as @Table which identified the entities as relational tables. For example, an individual can have the multiple residencies. Hibernate knows that an instance of individual can have multiple rows in the residency table using the annotation @onetoMany. The other example is an individual can have multiple relations such as mother, father, brother etc. The use of @onetoMany annotation, mappedby attribute in the model classes helps hibernate to understand one to many mapping for different entities.