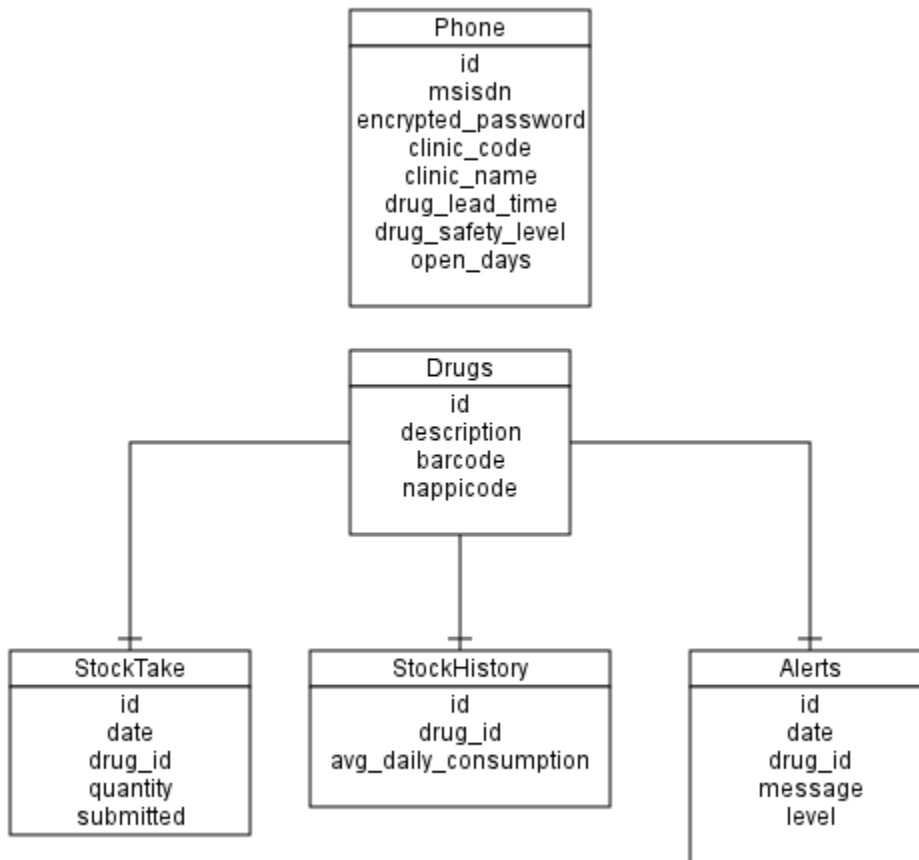


# 0003 - Stock Out App

This page details the system design for the Stock Out App for Android

- Database
- Classes
  - domain
  - dao
  - screens
  - controllers
  - services
- Functionality
  - Setup phone
  - View Alert(s)
  - Stock Take
  - View Stock Take(s)
  - Stock received
  - Enter Pin
  - Receive Alert
  - Generate Alert
  - Background Task
- Average Daily Consumption
- Android Stuff
  - Alert count on the App icon
  - Background Tasks
  - GCM Client documentation

## Database



## Classes

### domain

POJOs (plain old java objects) used to represent entities stored in the database)

- Phone
  - id
  - msisdn: String
  - clinicName: String
  - encryptedPin: String
  - clinic\_code: String
  - clinic\_name: String
  - drug\_lead\_time: Integer (see the Alert calculation)
  - drug\_safety\_level: integer (see the Alert calculation)
  - days\_open: Integer (number of days the clinic is open)
- Drugs
  - id
  - description: String
  - barcode: String
  - nappicode: String

- Alerts
  - id
  - date: Date
  - drug: Drug
  - message: String
  - level: Integer (1=low, 3=high)
- StockTake
  - id
  - date: Date
  - drug: Drug
  - quantity: Integer
  - submitted: Boolean
- StockHistory
  - id
  - drug: Drug
  - avgDailyConsumption: Integer


## dao

data access objects used to extract domain objects out of the database. used by services

- PhoneDao
  - Setup phone
  - Match encrypted password
  - Store new encrypted password
- DrugDao
  - Find a Drug given barcode
  - Find a Drug given an ID
- AlertDao
  - Find all Alerts
  - Find Alert for a specific Drug
  - Create new Alert
  - Delete old Alert
- StockTakeDao
  - Create new Drug StockTake
  - Find StockTake for a specific Drug
  - Find all StockTake done today
  - Mark StockTake as submitted for a specific Drug
- StockHistoryDao
  - Create new Drug StockHistory
  - Find StockHistory for specific Drug
  - Update Drug StockHistory

## screens

classes which are solely concerned with the graphical interface - labels, buttons, etc

- Initial setup (phone, msisdn, clinic\_name, lead\_time, safety\_level)
- Main
- Stock Take
- Pin
- 

## controllers

used in conjunction with the screens to provide access to the functionality - what happens when a button is pressed)

- MainController
- StockTakeController

- PinController

## services

used by controllers to implement the functionality required by the application. these should be discrete and unit testable

- AuthenticationService (login, authenticate via server, update encrypted pin)
- SetupService (creates the phone table with SIM and clinic details)
- StockTakeService (handles creation of a new stock take, sending of stock take to the server, retrieval of today's stock)
- CalculationService (avgDailyConsumption, EstimatedStock + Threshold calculations)
- AlertService (retrieval of alerts)

## Functionality

### Setup phone

- Need to setup the phone with msisdn and system settings.
- The phone will retrieve the clinic information from the server
- The local database will be updated
- System settings include:
  - drug\_lead\_time
  - drug\_safety\_level
  - days\_open
- System settings are synced with Drug Stock Warehouse (DSW)
- Once the setup wizard is completed, initial stock will need to be captured - this includes a checklist of the drugs in the database

### View Alert(s)

- Read Alerts from the database
- Display to the user




### Stock Take

- Open up Zxing application and scan a barcode
- Find Drug by barcode
  - If Drug found, display Drug description to user
  - If no Drug found, display an error message
- Enter in a Stock quantity for Drug
- Find last StockTake for Drug + delete
- Save new StockTake
- Find StockHistory for Drug (Create if it doesn't exist and prompt for initial avgDailyConsumption)
- Calculate new ADC (average daily consumption)
- Find Alert for Drug
- Delete Alert
- Send StockTake to Stock Out App Server using REST service
  - If sending fails, create background procedure to send
  - If sent, update StockTake and set submitted to True

### View Stock Take(s)

- Read today's StockTake from the database
- Display to the user, with the last at the top

### Stock received

- User scans barcode 
-  barcode represents a box of drugs
-  barcode is actually drug package
- Lookup drug in the database
- Get last StockTake for the drug
- Amend the quantity to be old quantity + new quantity




## Enter Pin

- User enters their pin
- Encrypt pin
- Retrieve encryptedPin from the database
  - If encryptedPin found - match encryptedPin with one just entered
    - If they match - authentication is successful
    - If they don't match - authenticate with Stock Out App Service using REST service
      - If authentication is successful, update encryptedPin in the database
      - If authentication fails, user has entered an incorrect pin
      - If server communication fails, user has probably entered an incorrect pin (or has changed their password on the server)
  - If no encryptedPin found - authenticate with Stock Out App Service using REST service
    - If authentication is successful, update encryptedPin in the database
    - If authentication fails, user has entered an incorrect pin
    - If server communication fails, it is not possible for the user to login - you need to have a data connection in order to login the first time

## Receive Alert


- receive GCM push message
- find existing Alert for specified Drug
- Create new Alert
- Delete old Alert

## Generate Alert

- Background job that runs daily
- System properties:
  - Lead\_time indicates how many days it takes to make a delivery - must be entered on phone setup
  - Safety\_level indicates how many days we should keep in stock (this includes the buffer) - must be entered on phone setup
- For each Drug find last StockTake and StockTakeHistory
  -  Calculate:  $(\text{currentDate} - \text{StockTake.date}) * (\text{days\_open} / 7) = \text{days}$
  -  Calculate:  $\text{currentDate} - \text{StockTake.date} = \text{days}$
  - Calculate:  $\text{StockTake.quantity} - (\text{days} * \text{StockTakeHistory.avgDailyConsumption}) = \text{estimated\_stock}$
  - Calculate:  $(\text{Lead\_time} + \text{Safety\_level}) * \text{StockTakeHistory.avgDailyConsumption} = \text{threshold}$
  - If **estimated\_stock** <= **threshold** then generate a Alert
- Send information about the Alert to the server 
- Note: App alert generation will not worry about public holidays or opening days - Alerts will be generated 7 days a week and the clinic will attend to alerts during opening hours.

## Background Task



If sending of Stock Take fails, it must be retried continuously using the following algorithm:

- Once every hour
-  If we use GCM, can we avoid the background task

If another Stock Take is done on the same drug, the background task should be cancelled - the new Stock Take replaces the old.

## Average Daily Consumption

ADC is used to determine to generate an alert that requests the user to perform a stock take

-  (Current StockTake.date - Last StockTake.date) \* (days\_open / 7) = **days**
-  Current StockTake.date - Last StockTake.date = **days**
- Last StockTake.quantity - Current StockTake.quantity = **stock\_used**
- stock\_used / days = new **avgDailyConsumption**

## Android Stuff

### Alert count on the App icon

It appears that the Facebook alert icon "bubble" on the Facebook App icon, is devise independent feature. We can add this as an added feature. See more:

Samsung: <http://stackoverflow.com/questions/20136483/how-do-you-interface-with-badgeprovider-on-samsung-phones-to-add-a-count-to-the/20136484#20136484>

Sony (last answer): <http://stackoverflow.com/questions/17565307/how-to-display-count-of-notifications-in-app-launcher-icon>

## Background Tasks

Should use an [IntentService](#) which is coupled with an [AlarmManager](#) to schedule

## GCM Client documentation

Read more about GCM clients here: <http://developer.android.com/google/gcm/client.html>