

LightSearch Server

[11.09.2018]

Как говорилось в этом документе ([ссылка](#)), я приступил к написанию сервера, к которому присоединялся бы клиент на Android. Я стал думать, как клиент должен присоединяться к серверу, и решил, что общение между клиентом и сервером будет происходить по принципу «команда-ответ»: клиент посылает команду, сервер посылает ответ. Команды и ответы я сделал в виде JSON-файлов. JSON довольно удобный формат, к тому же написано много библиотек, работающих с ним, или же можно самому написать парсер: формат не сложный и понятный, но как говорится, зачем изобретать велосипед? Поэтому я взял готовое решение: библиотека json-simple. При помощи нее можно делать JSON-объекты, вставлять в них строку, массив, структуру, конвертировать в строковую переменную и обратно. Теперь осталось разработать структуру JSON-файлов.

Первоначальный вариант принципа «команда-ответ» был такой(текст из моего черновика, который я сделал для того, чтобы не запутаться и как-то представлять себе, как все это будет работать):

```
Файл  Правка  Поиск  Вид  Документ  Справка
LightSearch Server(в дальнейшем "сервер") работает с android-устройствами(в дальнейшем "клиент").
Когда клиент подключается к серверу, то клиент отправляет следующую информацию серверу:
- IMEI;
- ip-адрес;
- версия ОС;
- модель устройства.

Отправляет клиент это сообщение серверу в формате JSON. Структура данного сообщения такова:
{
  "IMEI": "<IMEI-устройства>",
  "IP": "<ip-адрес>",
  "OS": "<версия ОС>,"
  "model": "<модель устройства>",
  "username": "<Имя пользователя бд>,"
  "password": "<Пароль>"
}

После этого сервер отправляет клиенту сообщение в формате JSON. Структура данного сообщения такова:
{
  "IMEI" : "<IMEI-устройства>",
  "isConnect" "<True или False>",
  "message": "<сообщение>,"
  "skladList": ["<Список складов>"],
  "TKList": "[<Список ТК>]"
}
isConnect - если True, то подключение установлено, и в поле message будет записано сообщение с уведомлением о данном событии. Иначе (False) - подключение не установлено, и в поле message будет записано сообщение об ошибке.
Если пришла ошибка, то skladList и TKList будут иметь пустое значение.

Клиент парсит данный файл и заполняет соответствующие компоненты Spinner.

После этого сервер ждет сообщение от клиента. Клиент посылает одно сообщение в формате JSON.
Структура данного сообщения такова:
{
  "IMEI": "<IMEI-устройства>",
  "query": "<Строка поиска для выполнения запроса>,"
  "sklad": "<Выбранный склад>,"
  "TK": "<Выбранный ТК>"
}
```

IMEI - необходим для того, чтобы действительно удостовериться, что именно данный клиент прислал это сообщение.

query - это то, что ввел клиент в поле поиска. Ключ query никогда не может быть пустым. За данную проверку отвечает клиент.

sklad - содержит в себе название склада, выбранного клиентом при помощи компонента Spinner. Может содержать значение "null" и "all".

TK - содержит в себе название ТК, выбранного клиентом при помощи компонента Spinner. Может содержать значение "null" и "all".

После получения данного сообщения сервер выполняет запрос и отправляет данные запроса клиенту в формате JSON. Структура данного сообщения такова:

```
{
  "IMEI": "<IMEI-устройства>",
  "data": [
    {
      "podrazdelenie": "<Подразделение товара>",
      "ID": "<Идентификатор товара>",
      "name": "<Название товара>",
      "price": "<Текущая цена>",
      "amount": "<Количество товара в подразделении>"
    }
  ]
}
```

Если запрос не вывел ни одного результата, то массив data будет содержать один объект-структуру, в котором каждое поле(podrazdelenie, ID, name, price, amount) будет иметь значение "null".

В случае возникновения ошибки сервер отправляет клиенту следующее сообщение:

```
{
  "IMEI": "<IMEI-устройства>",
  "error": "<Текст ошибки>"
}
```

Администратор - особый клиент, который может подключиться к серверу. При создании сервера он запрашивает пароль администратора, который в дальнейшем будет являться проверкой на администратора. Он присылает серверу следующее сообщение: Отправляет клиент это сообщение серверу в формате JSON. Структура данного сообщения такова:

```
{
  "IMEI": "admin",
  "IP": "admin",
  "OS": "admin",
  "model": "admin",
  "username": "admin, или пользователь с привилегиями администратора",
  "password": "<Пароль>"
}
```

После этого сервер отправляет администратору сообщение:

```
{
  "IMEI": "admin",
  "isConnected": "<True или False>",
  "message": "<Сообщение администратору>"
}
```

При удачном подключении администратор получает доступ к Admin Panel. Ему доступны следующие функции:

1. Рестарт сервера(restart)
2. Задать таймаут сервера(toutServer)
3. Вывести список клиентов(clList)
4. Задать таймаут клиентам(toutClient)
5. Кик клиента(kick)
6. Занести клиента в блэклист(blacklist)
7. Создать нового пользователя с привилегиями администратора(createAdmin)

Пользователь с привилегиями администратора не доступна функция №6.

Файл Правка Поиск Вид Документ Справка		
Далее описывается таблица с сообщениями, в которой в первом столбце указана команда, во втором - сообщение администратора, третье - сообщение сервера.		
Команда	Сообщение администратора	Сообщение сервера
1.restart	{ "name": "Имя", "command": "restart" }	{ "name": "Имя", "isDone": "True/False", "message": "<Сообщение>" }
2.toutServer	{ "name": "Имя", "command": "toutServer", "time": "<таймаут в мс>" }	{ "name": "Имя", "isDone": "True/False", "message": "<Сообщение>" }
3.clList	{ "name": "Имя" "command": "clList" }	{ "name": "Имя", "isDone": "True/False", "list": [{ "IMEI": "<IMEI>", "username": "<Имя пользователя>", }]
4.toutClient	{ "name": "Имя", "command": "toutClient", "time": "<таймаут в часах>" }	{ "name": "Имя", "isDone": "True/False", "message": "<Сообщение>" }

В качестве уникального идентификатора я использую IMEI телефона. При подключении к серверу Android-клиент отправляет следующее сообщение серверу:

```
{
    «IMEI»: «IMEI устройства»
}
```

После того, как сервер получил ответ

Конечный вариант, конечно же, отличается.

Первое, что я исправил — это подключение. Для того, чтобы сервер работал логичнее, я разбил первое сообщение на два: теперь при подключении Android клиент отправляет следующее сообщение серверу:

```
{
    «IMEI»: «IMEI устройства»
}
```

А если клиент — администратор, то:

```
{  
    «IMEI»: «admin»  
}
```

Теперь при подключении например администратора, не надо в JSON записывать лишние поля. Конечно же, для того, чтобы это работало, мне пришлось добавить еще один обработчик, так называемый распределитель. Все новые входящие подключения проходят через него, и он проверяет, кем является клиент: Android клиентом, или администратором.

После этого распределитель после определения типа клиента создает соответствующий обработчик. Если Android клиент, то сервер посылает следующее сообщение:

```
{  
    «IMEI»: «IMEI-устройства»,  
    «isConnect»: «OK»  
}
```

Если администратор, то:

```
{  
    «name»: «Имя администратора»,  
    «isConnect»: «OK»  
}
```

Теперь тип клиента определен и соединение с сервером установлено. Далее сервер ждет сообщение от клиента. Если Android клиент, то:

```
{  
    «IMEI»: «IMEI устройства»,  
    «IP»: «ip-адрес»,  
    «OS»: «версия ОС»,  
    «model»: «модель устройства»,  
    «username»: «Имя пользователя бд»,  
    «password»: «Пароль»  
}
```

Если администратор:

```
{  
    «name»: «Имя администратора»,  
    «password»: «Пароль»  
}
```

В зависимости от введенных параметров, правильности соединения, работы сервера, сервер отправляет сообщение. Если подключение прошло с ошибкой, то сервер отправляет следующее сообщение Android клиенту:

```
{  
    «IMEI»: «IMEI устройства»,  
    «error»: «Текст ошибки»  
}
```

Также сервер обрабатывает и специфические ошибки: ошибка в парсе JSON, ошибка подключения к БД, ошибка в работе самого сервера(ошибки в потоках, в считывании файлов инициализации, ошибка записи в log-файл и т. д.).

Обработчик Android клиента обрабатывает только один вид сообщения, посылая его серверу:

```
{
    «IMEI»: «IMEI устройства»,
    «query»: «Строка» поиска для выполнения запроса»,
    «sklad»: «Выбранный склад»,
    «TK»: «Выбранный ТК»
}
```

Сервер же отвечает так:

```
{
    «IMEI»: «IMEI устройства»,
    «data»: [
        {
            «podrazdelenie»: «Подразделение товара»,
            «ID»: «Идентификатор товара»,
            «name»: «Название товара»,
            «price»: «Текущая цена»,
            «amount»: «Количество товара в подразделении»
        }
    ]
}
```

Такое общение идет до того момента, пока клиент не отключится от сервера.

Администратор работает через приложение LightSearch Admin Panel ([ссылка](#)), и сервер обрабатывает множество команд от администратора. Вот таблица с командами и ответами:

Команда	Администратор	Сервер
1.Перезагрузка сервера	{ «name»: «Имя администратора», «command»: «restart» }	{ «name»: «Имя администратора», «isDone»: «True или False», «message»: «Сообщение» }
	{ «command»: «blist» }	{ «isDone»: «True или False», «list»: [{ «IMEI»: «IMEI устройства или имя администратора» }] }
4. Кик клиента	{ «name»: «Имя администратора», «command»: «blist», «IMEI»: «IMEI устройства» }	{ «name»: «Имя администратора», «isDone»: «True или False», «message»: «Сообщение» }
5. Добавить в черный список	{ «name»: «Имя администратора», «command»: «addBlacklist», «IMEI»: «IMEI устройства» }	{ «name»: «Имя администратора», «isDone»: «True или False», «message»: «Сообщение» }
6. Удалить из черного списка	{ «name»: «Имя администратора», «command»: «delBlacklist», «IMEI»: «IMEI устройства» }	{ «name»: «Имя администратора», «isDone»: «True или False», «message»: «Сообщение» }

Поле isDone необходимо, если на сервере произойдет ошибка, текст который будет записан в поле message.

Сервер при своем запуске считывает следующие файлы:

1) settings – в этом файле содержится два поля: первое поле — время перезагрузки сервера в часах, второе поле — время таймаута клиента со стороны сервера в миллисекундах. Если сервер не находит данный файл, то он устанавливает эти настройки по умолчанию: перезагрузка сервера принимает значение 0, таймаут клиента — 0. То есть сервер будет работать без перезагрузки, и таймаут клиента со стороны сервера также не будет установлен.

2) admin – в этом файле содержится список админов и их хеши паролей. Если данного файла не существует, то сервер предложит создать администратора с именем admin, и предложит ввести для него пароль.

3) blacklist – в этом файле содержится список клиентов и администраторов, попавших в черный список. Если данного файла не существует, то сервер создаст его.

4) db – в этом файле содержатся параметры подключения к базе данных, а именно: имя, адрес, порт. Если данного файла не существует, то сервер предложит ввести имя, порт и адрес базы данных, и создаст данный файл.

После считывания всех файлов сервер начинает свою работу. Сервер введет постоянное логирование и ждет подключения Android-клиентов и администраторов.

Логирование введется как в окне, в котором запущен сервер, так и в файле log_<ДД-ММ-ГГ>. Все логи хранятся в папке logs.

Сообщения на сервере имеют два типа:

1) Info. Событие, которое произошло в ходе определенных действий клиентов и администраторов данного сервера, и так и самого сервера, и закончилось с успехом. Пример сообщения:

[23.07.2018 20:02:28] Info: Server restarted

Это сообщение говорит о том, что сервер в время, указанное в квадратных скобках, был перезагружен.

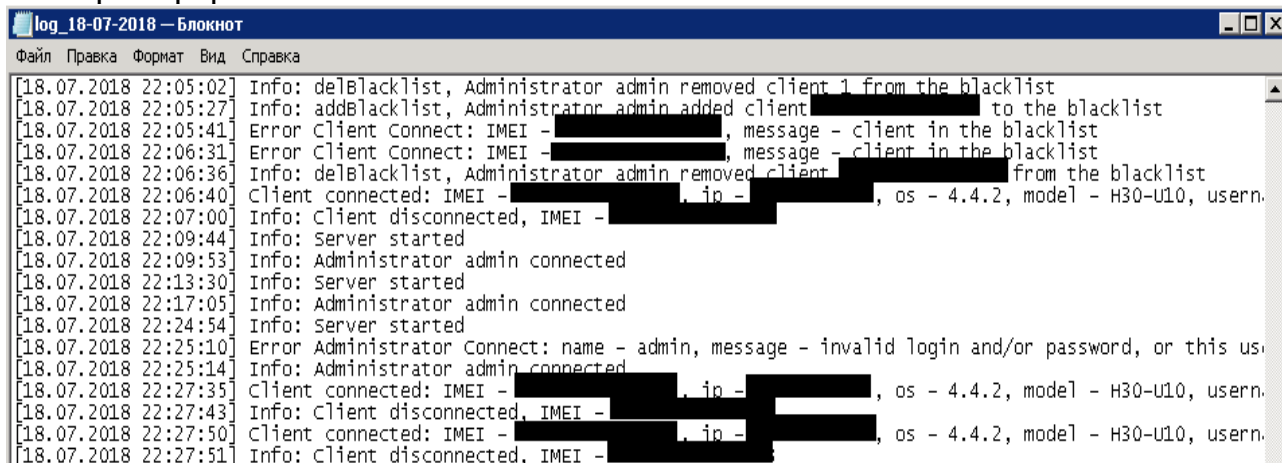
2) Error. Ошибка, произошедшая на сервере. Пример сообщения:

[18.07.2018 14:27:57] Error Admin Connect: name - admin, message - invalid login and/or password, or this user in the blacklist

Ошибки имеют свои названия. Эта ошибка называется Error Admin Connect, далее указываются два параметра — name и message. В name указывается имя администратора, у которого возникла ошибка, а в поле message указывается непосредственно сама ошибка. В данном случае администратор под именем admin ввел неверно пароль, или же он находится в черном списке, или же такого пользователя не существует.

Название ошибок я делал так, чтобы было понятно, в каком блоке программы они возникли. Ошибки, связанные с парсом JSON, потоками и так далее у меня уже не возникают, но они были, и логирование очень помогло мне отладить мой код. Теперь же сообщения типа Error возникают при соединении клиента, и зависят по большей части от неверного имени пользователя и/или пароля.

Вот пример файла лога:



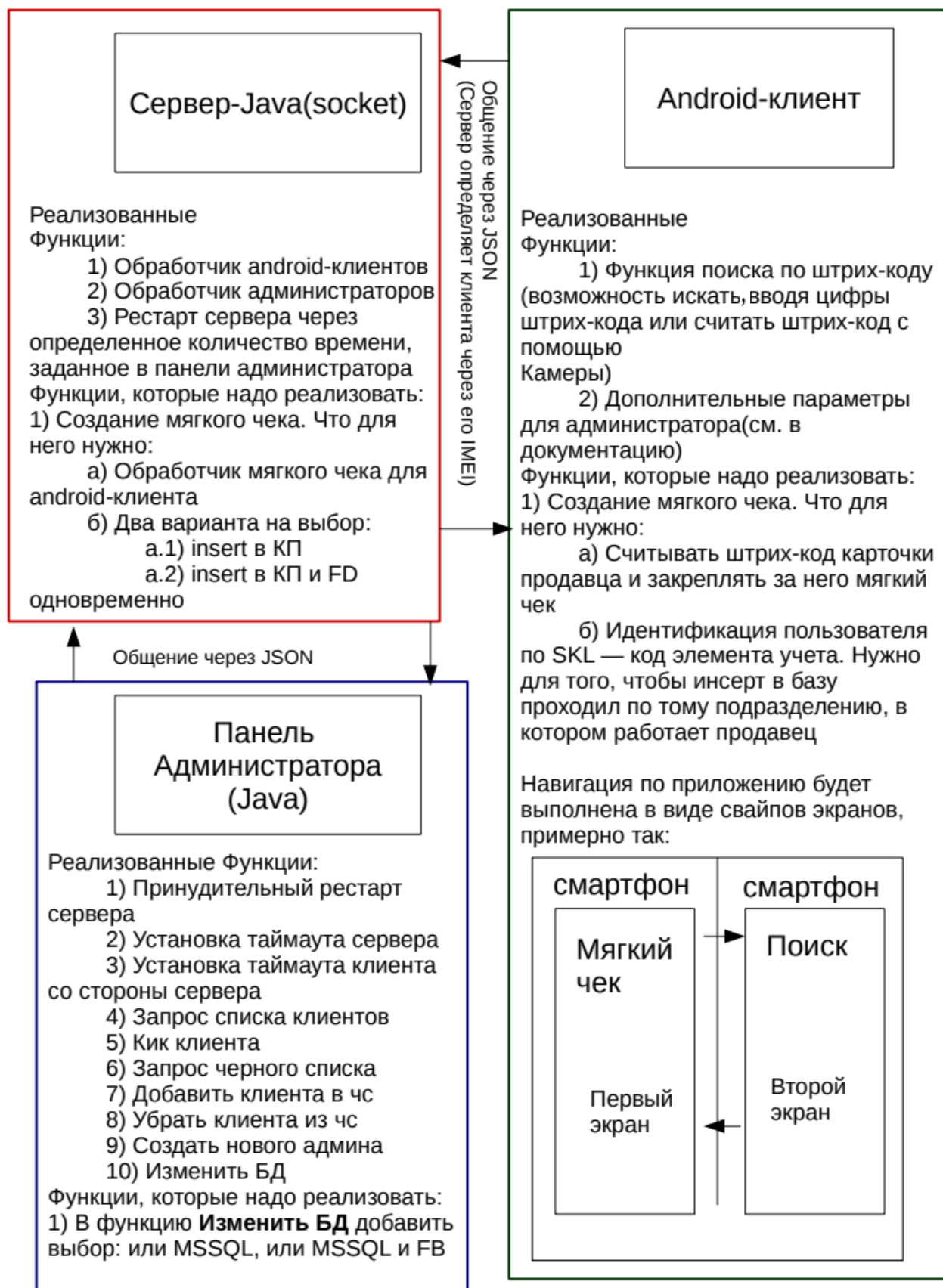
```
log_18-07-2018 - Блокнот
Файл Правка Формат Вид Справка

[18.07.2018 22:05:02] Info: delBlacklist, Administrator admin removed client 1 from the blacklist
[18.07.2018 22:05:27] Info: addBlacklist, Administrator admin added client [REDACTED] to the blacklist
[18.07.2018 22:05:41] Error Client Connect: IMEI - [REDACTED], message - client in the blacklist
[18.07.2018 22:06:31] Error Client Connect: IMEI - [REDACTED], message - client in the blacklist
[18.07.2018 22:06:36] Info: delBlacklist, Administrator admin removed client [REDACTED] from the blacklist
[18.07.2018 22:06:40] Client connected: IMEI - [REDACTED], ip - [REDACTED], os - 4.4.2, model - H30-U10, usern.
[18.07.2018 22:07:00] Info: Client disconnected, IMEI - [REDACTED]
[18.07.2018 22:09:44] Info: Server started
[18.07.2018 22:09:53] Info: Administrator admin connected
[18.07.2018 22:13:30] Info: Server started
[18.07.2018 22:17:05] Info: Administrator admin connected
[18.07.2018 22:24:54] Info: Server started
[18.07.2018 22:25:10] Error Administrator Connect: name - admin, message - invalid login and/or password, or this us
[18.07.2018 22:25:14] Info: Administrator admin connected
[18.07.2018 22:27:35] Client connected: IMEI - [REDACTED], ip - [REDACTED], os - 4.4.2, model - H30-U10, usern.
[18.07.2018 22:27:43] Info: Client disconnected, IMEI - [REDACTED]
[18.07.2018 22:27:50] Client connected: IMEI - [REDACTED], ip - [REDACTED], os - 4.4.2, model - H30-U10, usern.
[18.07.2018 22:27:51] Info: Client disconnected, IMEI - [REDACTED]
```

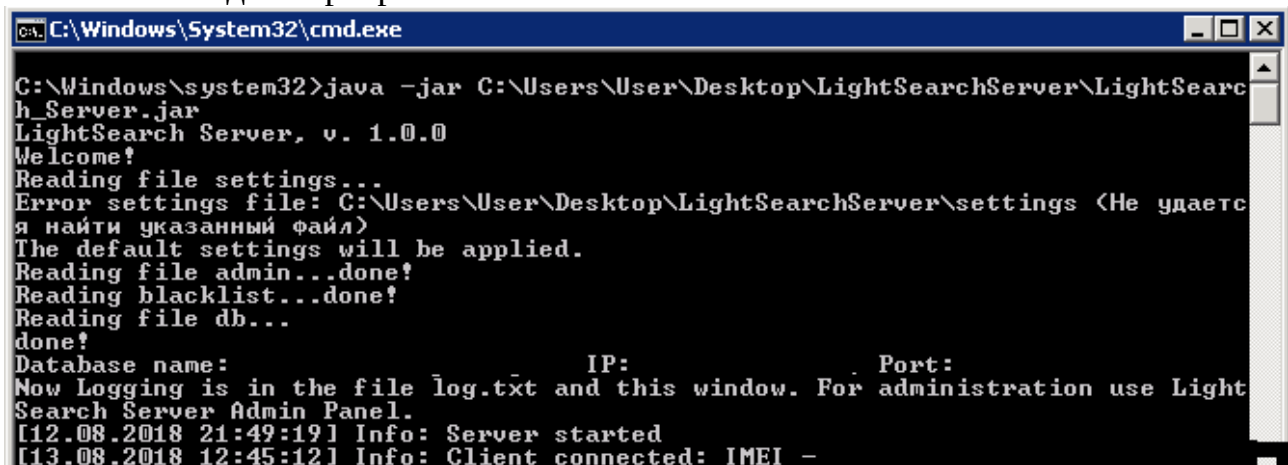
Написав сервер, я начал писать Android клиент ([ссылка](#)), и когда была готова первая версия, я ее стал тестировать. Потом, когда ошибки стали всплывать, я параллельно писал и сервер, и клиент, и панель администратора.

Забыл написать про демона, который перезагружает сервер. Он является небольшой программой на Java, которая сначала завершает процесс, под которым запущен сервер, а затем вызывает команду, которая снова запускает сервер. Для того, чтобы сервер стал полностью кросс-платформенным, надо будет определять в Java, с какой операционной системой работает программа. Но это уже в будущем, так как сервер будет модифицироваться, и станет полностью независимым от бизнес-логики. Также, для обработчика Android клиента необходимо будет добавить еще одну функцию — мягкий чек. Все select'ы необходимо вынести из сервера, и не добавлять в него будущие insert'ы. Все будет работать по принципу «команда-ответ». Пока что идея такова: на стороне предприятия в базе данных будут созданы две таблицы: в одной из них будут храниться от сервера команды, а в другой — ответы для него.

Взаимодействие между бд и сервером будет осуществляться при помощи робота, который будет написан не мной, а предприятием, ведь именно он и будет являться частью их бизнес-логики. Вот схема, которую я нарисовал для того, чтобы у меня было общее представление об проекте:



Вот как выглядит сервер:



```
C:\Windows\System32\cmd.exe

C:\Windows\system32>java -jar C:\Users\User\Desktop\LightSearchServer\LightSearch
h_Server.jar
LightSearch Server, v. 1.0.0
Welcome!
Reading file settings...
Error settings file: C:\Users\User\Desktop\LightSearchServer\settings (Не удаетс
я найти указанный файл)
The default settings will be applied.
Reading file admin...done!
Reading blacklist...done!
Reading file db...
done!
Database name:          IP:          Port:
Now Logging is in the file log.txt and this window. For administration use Light
Search Server Admin Panel.
[12.08.2018 21:49:19] Info: Server started
[13.08.2018 12:45:12] Info: Client connected: IMEI -
```