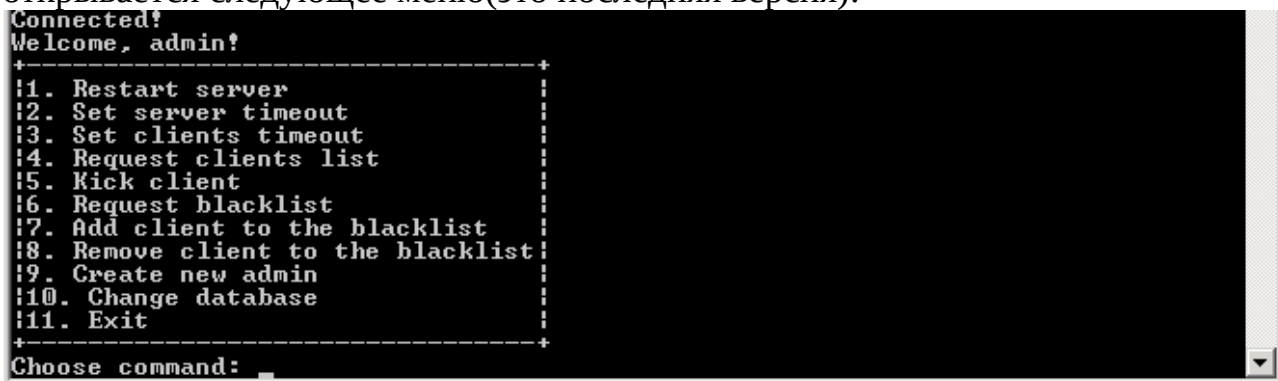


LightSearch Admin Panel

[11.09.2018]

Когда я создавал сервер, я уже думал о том, что надо как-то управлять им. Например, кикать клиентов, добавлять их в черный список, перезагружать сервер, и так далее. Тогда, после того, как я создал первую версию сервера, я приступил к написанию панели администратора. Я решил ее сделать в командной строке, потому что посчитал, что так будет более рационально, ведь администрирование может и проходить на машине, на которой вообще отсутствует графический интерфейс пользователя.

Для подключения к серверу мне пришлось написать обработчик администратора. При подключении панель администратора спрашивает ip-адрес и порт сервера. Затем пытается подключиться к нему, и если подключение прошло успешно, то на сервере создается соответствующий обработчик, а панель спрашивает имя администратора и пароль. Все администраторы и пароли хранятся на сервере в файле admin. Пароли хранятся в виде хешей SHA-256. После правильно введенных параметров подключения администратору открывается следующее меню(это последняя версия):

A screenshot of a terminal window showing the LightSearch Admin Panel interface. The text is as follows:

```
Connected!
Welcome, admin!
+-----+
!1. Restart server
!2. Set server timeout
!3. Set clients timeout
!4. Request clients list
!5. Kick client
!6. Request blacklist
!7. Add client to the blacklist
!8. Remove client to the blacklist
!9. Create new admin
!10. Change database
!11. Exit
+-----+
Choose command: _
```

В первой версии LightSearch Admin Panel я добавил функции рестарта сервера, установка таймаута сервера, время, после которого сервер перезагружается, таймаут для клиента на сервере, запрос списка клиентов, кик клиента, запрос черного списка, добавление в черный список. Потом я добавил функцию создание нового администратора.

После написания клиента на Android я приступил к тестам непосредственно клиента Android, сервера и панели администратора. Первое, что я понял — я забыл добавить одну из важных функций — убирать клиента из черного списка. А то добавил свой телефон, и все, не смог больше подключиться :).

После исправления ошибок и доработок, я добавил еще одну функцию — изменение базы данных. Теперь администратор может изменить имя бд, ip-адрес, порт. После перезагрузки сервер поменяет бд. Параметры бд сохраняются в файле db. Это удобно, когда необходимо менять базу с тестовой на рабочую, и также добавляет универсальности в саму программу.

Взаимодействие между панелью администратора и сервером я описал в документе про сервер (ссылка).

[02.05.2019]

Рефакторинг панели администратора был сделан недавно, несколько недель назад. В отличие от сервера, панель администратора в плане работоспособности сделана полностью.

Как и LightSearch Server, LightSearch Admin Panel написан по принципу SOLID, который кратко характеризуется так: один интерфейс выполняет одну-две функцию, не больше. Это позволяет писать код реализаций интерфейса достаточно маленькими. Подробнее об этом я описал в документе про LightSearch Server.

Принцип SOLID позволяет эффективно использовать еще один принцип — Test-Driven Development. Его суть в том, что мы сначала пишем тест для интересующего интерфейса, а затем на основе теста пишем реализацию интерфейса. Скажу честно, я все еще не могу полностью использовать данный принцип. Иногда мне приходится сначала писать реализацию, а затем только тест. Но практика должна исправить данный момент.

SOLID в совокупности с Test-Driven Development позволяют писать маленькие тесты, и достаточно быстро. Рассмотрим пример. Мне в панели администратора необходимо сделать следующее: считать IP и порт сервера, и проверить эти значения на корректность: IP должен соответствовать IP паттерну в диапазоне 0.0.0.0-255.255.255.255, а порт должен быть в диапазоне 1024-65535.

Для этого необходимо сначала создать два интерфейса: IPValidator и PortValidator, которые будут проверять IP и порт соответственно. Теперь можно писать к ним тесты: отдельно для PortValidator, отдельно для IPValidator. Когда тесты были написаны, можно писать их реализации: IPValidatorDefaultImpl и PortValidatorDefaultImpl. Теперь можно тестировать эти реализации по ранее написанным тестам.

Далее пишем интерфейс ScannerConnection. Он будет содержать в себе два интерфейса: IPValidator и PortValidator, и будет выполнять две функции: считывать IP, введенным пользователем, и считывать порт, введенным пользователем. Пишем для него тест, пишем реализацию, проверяем реализацию ранее написанным тестом.

Таким образом, используя принципы SOLID и Test-Driven Development, мы получили три интерфейса: IPValidator, PortValidator, и ScannerConnection, три реализации этих интерфейсов, и три теста, проверяющие эти интерфейсы. Благодаря такому подходу мы можем писать множество реализаций каждого из этих интерфейсов, например, IPValidator, и это никак не повлияет на работу интерфейсов ScannerConnection и PortValidator: PortValidator вообще никак не связан с IPValidator, а ScannerConnection в текущей его реализации просто просит IPValidator проверить IP, введенный пользователем, на корректность. Тесты же явно показывают, что написанные реализации работают, и что они работают отдельно от этих интерфейсов, так как отвечают только за одну-две функции.

Еще один плюс от применения этих подходов — реализации интерфейсов имеют «маленький» код, как и тесты к ним. Это повышает читаемость кода.

Так же, как и в LightSearch Server, в LightSearch Admin Panel применен паттерн команда с использованием функционального интерфейса Function, доступный в Java, начиная с версии 1.8.

Как было ранее описано в документе про LightSearch Server, в LightSearch Admin Panel изменилась команда подключения к серверу. Теперь она выглядит так:

Если администратор:

```
{  
    «identifier»: «admin»  
}
```

Если сервер прислал сообщение ОК, то это значит, что соединение установлено, и далее администратор вводит имя пользователя и пароль, и посылает на сервер следующее сообщение:

```
{  
    «command»: «connect»,  
    «name»: «имя администратора»,  
    «password»: «пароль администратора»  
}
```

Если такая пара name/password существует, то для администратора на экране показывается меню, и он может продолжать свою работу. Иначе ему предлагается ввести данные аутентификации еще раз. По умолчанию сервер разрешает вводить их три раза. Если все три попытки были неудачными, то сервер отключает этого администратора, и LightSearch Admin Panel завершает свою работу.

При написании панели администратора я столкнулся с одной трудностью. Если LightSearch Server сначала принимает команду, затем отправляет ответ, то LightSearch AdminPanel сначала отправляет команду, потом принимает ответ, конвертирует ответ в удобочитаемый вид для администратора, и затем выводит его на экран.

Для решения этой проблемы пришлось написать два пакета: первый пакет содержит процессоры команд администратора, второй пакет — процессоры команд, которые формируют сообщения на сервер. То есть все происходит примерно так: администратор вводит номер команды, вводит необходимые данные, затем процессор команд администратора вызывает соответствующий процессор команд, формирующий сообщение на сервер, получает от него ответ, и вызывает соответствующий интерфейс, который формирует ответ администратору в удобочитаемом виде.

Зачем было необходимо написать отдельно интерфейсы, которые конвертируют ответ для вывода на экран? Все дело в том, что от сервера может придти ответ, который содержит либо поле message, либо поле data. С полем message все ясно: мы просто выводим содержимое данного поля на экран. С полем data все

сложнее: в зависимости от команды, оно может иметь различное содержание. Обработку этого поля уже нельзя доверить процессору команд администратора или процессору команд, формирующие сообщения серверу: это не их зона ответственности. Поэтому пришлось написать отдельно интерфейсы, которые выполняют эту функцию.

Стоит также сказать, что процессор команд, который отвечает за подключение к серверу, единственный из процессоров, который не использует функциональный интерфейс Function. Это связано с тем, что он отличается от других процессоров: у него нет поля `command`, нет поля `name`, и главное, он логически отличается от других.

Панель администратора полностью протестирована. Ее работоспособность подтверждена как тестами (при использовании пакета TestNG), так и тестом при подключении непосредственно к серверу.

В будущем планируется улучшать код (где-то необходимо сделать инкапсуляцию, заменить наследование на композицию), и добавить новые функции.

Первая функция — работа с переменными командной строки. Чтобы каждый раз не вводить IP и порт сервера, достаточно будет указать IP и порт как переменные командной строки. Ну и как следствие — добавить работу с ключами командной строки.

Вторая функция — автодополнение. Хотя сейчас и реализована возможность вводить не только IMEI, но и номер строки в таблице IMEI, все равно возможность автодополнения была бы удобной. Но пока что ее реализация для меня остается неясной.