МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ОТКРЫТЫЙ УНИВЕРСИТЕТ

имени В.С. Черномырдина

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ И ПРОГРАММИРОВАНИЯ

Развертывание системы доставки учебного расписания на мобильные телефоны с операционной системой Android.

Отчет по учебной практике

Студент:

2 курса, ПОВТ

Шифр 611157

Головань Р.А.

Факультет КиИТ

Руководитель:

_____

_____

_____

Москва 2013 г.

Содержание

Введение

На момент начала практики имеется система доставки учебного расписания на телефоны, через мобильное приложение, разработанная в качестве дипломного проекта студентами 4-го курса. Система представлена в виде исходных текстов для различных компонентов. Для каждого компонента представлена документация, описывающая процесс сборки, настройки и эксплуатации.

Задача заключается в сборке и настройке всех необходимых компонентов, проверке правильности их работы и развертывании всей системы.

Архитектура системы

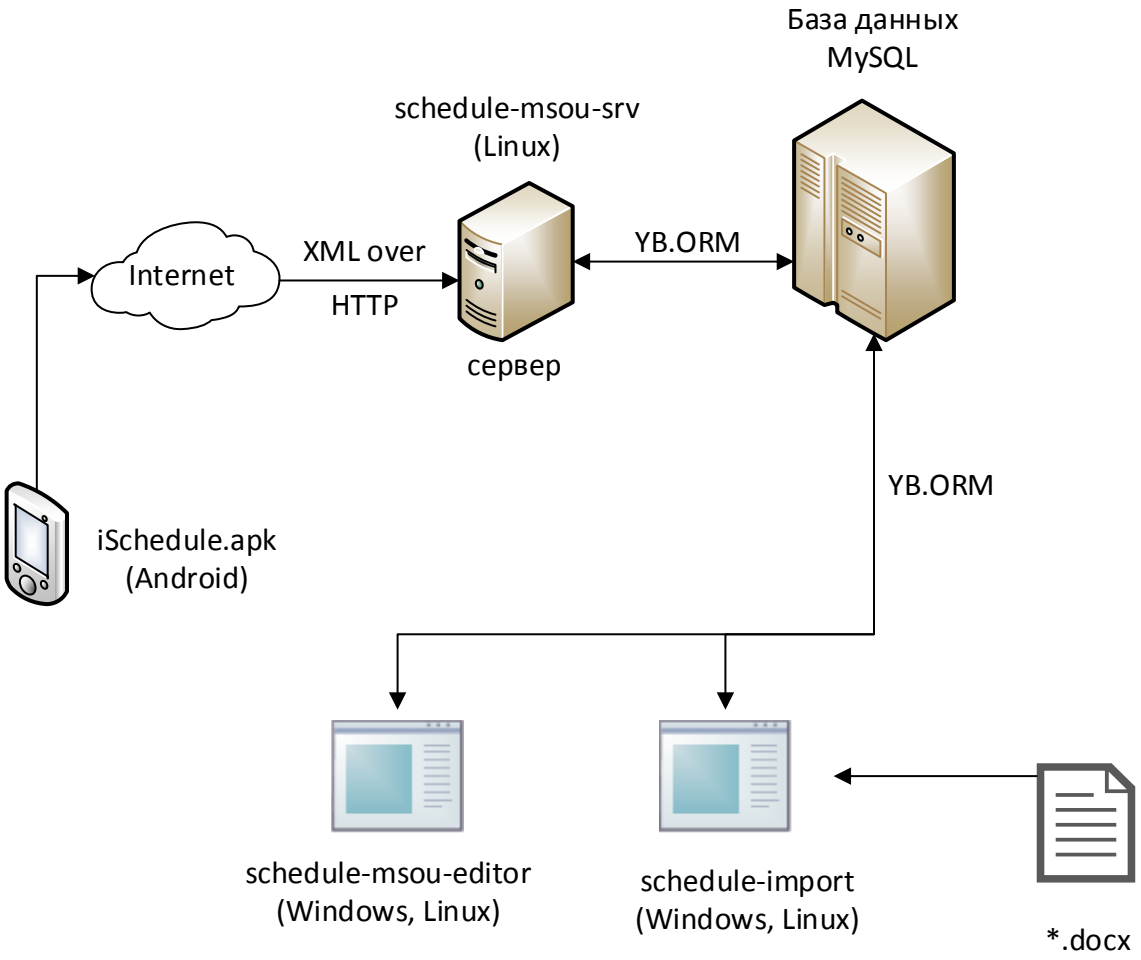

Рис.1 Схема работы системы

Основные узлы системы:

- База данных MySQL представляет из себя хранилище данных, необходимых для правильной работы всей системы.
- Schedule-msou-editor – кросс-платформенный редактор расписания, позволяющий составлять и редактировать данные посредством пользовательского интерфейса. Данные в структурированном виде сохраняются в базу данных.
- Schedule-import – модуль, предназначенный для распознавания специально подготовленного документа Microsoft Office формата .docx, содержащий расписание занятий.
- Schedule-msou-srv – сервер, отвечающий на запросы мобильного приложения на Android и выдающий данные из базы данных через интернет.
- iSchedule.apk – мобильное приложения на Android. Позволяет получить свежую версию расписания предметов через интернет.

Этапы выполнения работы

Сборка пакета yb-orm

Рекомендуется предварительно создать каталог, где будут храниться все данные связанные с проектом и перейти в него:

```
$ mkdir schedule
$ cd schedule
```

Первым шагом к сборке пакета — получение исходных текстов с интернета, при помощи системы контроля версий **git**. Для получения адреса удаленного репозитория необходимо перейти по ссылке:

```
https://code.google.com/p/yb-orm/
```

Там находится вся основная информация по настройке, истории и эксплуатации пакета **yb-orm**. Переходим на вкладку «*Source*» и копируем команду для клонирования репозитория. Выглядит она так:

```
$ git clone https://code.google.com/p/yb-orm/
```

С помощью этой команды происходит копирование удаленного репозитория на локальный компьютер. Если программа **git** на данный момент не установлена на вашем компьютере, её нужно установить, выполнив команду:

```
$ sudo apt-get install git
```

В результате работы программы, в текущем каталоге должна появиться папка **yb-orm**. Вывести список файлов в текущей папке можно при помощи команды **ls**.

Для создания скрипта **configure**, с помощью которого автоматически проверяется доступность ключевых библиотек и заголовочных файлов, необходимо запустить shell–скрипт **autogen.sh**, находящийся в папке **yb-orm**. Переход в папку осуществляется командой **cd yb-orm**. Запустить shell-скрипт можно двумя способами:

1. `$ sh autogen.sh`
2. `$ chmod +x autogen.sh`

   `$ ./autogen.sh`

В результате в папке должен появиться файл **configure**. Следующим шагом станет запуск скрипта **configure** с заданными параметрами.

Запуск скрипта для работы с базой sqlite

Первым делом нужно создать базу данных **sqlite3**, которая в дальнейшем будет использоваться для конфигурирования и работы тестовых приложений.

```
$ sqlite3 /home/user/schedule/yb_test_db.sqlite
```

Выход из интерпретатора команд **sqlite3** производится нажатием клавиш <Ctrl> + <D>.

База создана, осталось запустить **configure** с нужными параметрами. Для просмотра всех возможных установок необходимо набрать:

```
$ ./configure --help
```

Для запуска тестовых приложений нам потребуется только 2 параметра:

- `--with-test-db-url=sqlite+sqlite:///home/user/schedule/yb_test_db.sqlite` — устанавливает диалект, драйвер и путь до базы данных.
- `--prefix=/home/user/schedule/yb-sqlite-inst0` — указывает папку для установки пакета.

```
$./configure --with-test-db-url=sqlite+sqlite:///home/user/schedule/yb_test_db.sqlite –prefix=/home/user/schedule/yb-sqlite-inst0
```

При удачном выполнении скрипта, в папке должен появиться файл **Makefile**.

Следующим шагом станет сборка библиотеки при помощи утилиты автосборки **make**:

```
$ make -j4
```

Параметр **-j4** используется для ускорения процесса компиляции и сборки, разбивая этот процесс на 4 потока.

Для проверки работоспособности пакета, используется несколько тестов, запуск которых выполняется посредством команды:

```
$ make check -j4
```

При первом запуске произойдет компиляция исходных файлов тестового приложения и выявление ошибок работы пакета. Также генерируются **sql** файлы, которые необходимо добавить в используемую базу данных. Для этого необходимо их найти.

```
$ find . -name '*.sql'
```

Для правильной работы системы тестирования необходимо добавить только один файл, лежащий в `./lib/orm/test/mk_tables.sql`:

```
$ sqlite3 /home/user/schedule/yb_test_db.sqlite <./lib/orm/test/mk_tables.sql
```

Снова запускаем тесты. Теперь оба теста должны отработать без ошибок. Осталось только произвести установку пакета:

```
$ make install
```

В результате в папке **schedule** должна появиться папка **yb-sqlite-inst0**.

Запуск скрипта для работы с базой mysql

Первым делом нужно создать базу данных **mysql**, которая в дальнейшем будет использоваться для конфигурирования и работы тестовых приложений.

Для создания базы необходимо подключится к **mysql** под пользователем **root**:

```
$ mysql -u root -p mysql
```

Возможно будет необходимо установить следующие пакеты: **mysql-server**, **mysql-workbench**, **mysql-client**, **unixodbc-dev**, **libmyodbc.**

Если вы забыли пароль для пользователя **root**, то сбросить его можно следующей командой:

```
$ sudo dpkg-reconfigure mysql-server-5.1
```

После входа в интерпретатор вводятся следующий команды:

```
$ create database  test_db default charset utf8;
$ create user 'test'@'localhost' identified by 'test_pwd';
$ grant all on test_db.* to 'test'@'localhost';
```

Первая создает базу данных под названием “**test_db**” и задает кодировку **utf8** по умолчанию. Вторая команда создает пользователя с именем “**test@localhost**” и паролем “**test_pwd**”. Третья команда наделяет пользователя, созданного на предыдущем шаге, всеми привилегиями на базу **test_db**.

После этого необходимо внести изменения в следующий файлы:

```
~/.odbc.ini
/etc/odbcinst.ini
```

Открыть их можно следующими способами:

- Через редактор vim:

  ```
  $ vim ~/.odbc.ini
  ```

- Через редактор gedit:

  ```
  $ gedit ~/.odbc.ini
  ```

В файл **.odbc.ini** необходимо добавить такие строки:

```
[test_dsn]
Description     = Test database
Driver          = MyODBC
Server          = 127.0.0.1
Database        = test_db
Port            = 3306
```

```
Socket          =
Option          = 3
Stmt            =
CharSet         = UTF8
```

В файл **odbcinst.ini** такие:

```
[MyODBC]
Description     = MySQL ODBC driver
Driver          = /usr/lib/i386-linux-gnu/odbc/libmyodbc.so
Driver64        =
Setup           = /usr/lib/odbc/libodbcmyS.so
Setup64         =
UsageCount      = 1
CPTimeout       =
CPReuse         =
```

Если по указанному адресу не находится разделяемая библиотека **libmyodbc.so** (проверить это можно выполнив команду **ls /usr/lib/i386-linux-gnu/odbc/libmyodbc.so**), то её необходимо найти и прописать путь к ней в поле Driver:

```
$ find /usr/lib –name ‘libmyodbc.so’
```

Далее следует запустить скрипт **configure** со следующими параметрами:

```
$ ./configure --with-test-db-
url=mysql+odbc://test:test_pwd@test_dsn prefix=/home/user/schedule/
yb-mysql-inst0
```

При удачном выполнении скрипта, в папке должен появиться файл **Makefile**.

Следующим шагом станет выполнение проверки правильности подключения и работы собранной библиотеки. Для этого необходимо выполнить команду:

```
$ make check –j4
```

При первом запуске она сгенерирует **sql** файлы, которые необходимо добавить в тестовую базу данных **MySQL**. Найти их можно при помощи команды:

```
$ find . -name '*.sql'
```

```
Загрузить в базу:
```

```
$ mysql –u test –ptest_pwd test_db <
./lib/orm/test/mk_tables.sql
```

Снова запускаем тесты. Теперь оба теста должны отработать без ошибок. Осталось только произвести установку пакета:

```
$ make install
```

В результате в папке **schedule** должна появиться папка **yb-mysql-inst0**.

Сборка сервера auth с базой sqlite

Сервер **auth** представляет из себя небольшое приложение, позволяющее протестировать и продемонстрировать работу с базой данных.

Для сборки сервера **auth** необходимо перейти в папку **./examples/auth**:

```
$ cd examples/auth
```

Первым делом необходимо выполнить shell-скрипт **autogen.sh**:

```
$ sh autogen.sh
```

Далее запускается скрипт **configure** с параметрами:

```
$ ./configure --with-test-db-
url=sqlite+sqlite:///home/user/schedule/yb_test_db.sqlite --with-
yborm-root=/home/user/schedule/yb-sqlite-inst0
```

Параметр **--with-yborm-root** указывает путь к папке, где расположен установленный пакет **yb-orm**. В данном случае нужно указывать путь к пакету, сконфигурированному с базой **sqlite**

Далее выполняется **make** и **make check**:

```
$ make –j4 && make check –j4
```

Теперь в используемую базу данных необходимо занести файл **auth_schema.sql**, расположенный в папке **src/domain**:

```
$ sqlite3 /home/user/schedule/yb_test_db.sqlite <
./src/domain/auth_schema.sql
```

Сборка сервера auth с базой MySQL

Для сборки сервера **auth** необходимо перейти в папку **./examples/auth**:

```
$ cd examples/auth
```

Первым делом необходимо выполнить shell-скрипт **autogen.sh**:

```
$ sh autogen.sh
```

Далее запускается скрипт **configure** с параметрами:

```
$ ./configure --with-test-db-
url=mysql+odbc://test:test_pwd@test_dsn --with-yborm-
root=/home/user/schedule/yb-mysql-inst0
```

Параметр **--with-yborm-root** указывает путь к папке, где расположен установленный пакет **yb-orm**. В данном случае нужно указывать путь к пакету, сконфигурированному с базой **MySQL**

Далее выполняется **make** и **make check**:

```
$ make –j4 && make check –j4
```

Теперь в используемую базу данных необходимо занести файл **auth_schema.sql**, расположенный в папке **src/domain**:

```
$ mysql –u test –ptest_pwd test_db <
./src/domain/auth_schema.sql
```

Работа с тестовым сервером auth

Для начала работы тестового сервера, его сначала нужно запустить:

```
$ sh src/auth.sh
```

После этого необходимо перейти в браузере по адресу:

http://localhost:9090

Вы увидите следующее сообщение:

```
<status>NOT</status>
```

Что уже само по себе говорит о работе сервера.

Для проверки работы сервера ему необходимо передать параметры. Создание первого пользователя и остальные запросы расписаны в файле **README.auth** в папке **auth**.

Если в базе еще не существует ни одного пользователя, то его можно создать выполнив запрос:

http://localhost:9090/registration?login=medved&pass=preved&name=Medved&status=0

После его выполнения в базе создастся пользователь с логином ‘**medved**’ и паролем ‘**preved**’, а также наделенный правами администратора (**status=0**).

Следующий шаг – вход под пользователем ‘**medved**’ на сервер **auth**:

http://localhost:9090/login?login=medved&pass=preved

При успешном выполнении, сервер выдаст токен сессии, который необходимо использовать с другими запросами типа:

- http://localhost:9090/logout?token=token_num – производит завершение указанной сессии.
- http://localhost:9090/session_info?token=token_num – выводит информацию о заданной сессии.

В запросах вместо **token_num** необходимо подставить полученный сессионный токен. При успешном выполнении запроса сервер выдаст ‘**OK**’ или предоставит запрашиваемую информацию.

Сборка пакета schedule-msou-srv

Пакет **schedule-msou-srv** очень похож на сервер auth по компоновке и принципу работы с базой данных. Отличие только в более простой архитектуре **auth**.

Для его получения нужно найти на сайте code.google.com проект **schedule-msou-srv** и перейти на вкладку “Sources”. Там вы найдете ссылку для клонирования репозитория к себе на компьютер. Выглядит она так:

```
$ git clone https://code.google.com/p/schedule-msou-srv/
```

Далее переходим в появившийся каталог **schedule-msou-srv**:

```
$ cd schedule-msou-srv
```

И выполняем привычные действия. Первым делом запустим shell-script **autogen.sh**:

```
$ sh autogen.sh
```

Для правильной работы этого компонента, необходимо создать новую **MySQL** базу **schedule_db**, наделить пользователя **test** всеми привилегиями на эту базу и подправить содержимое файла **~/.odbc.ini**, добавив определение **schedule_dsn**.

Запустим появившийся скрипт **configure** со следующими параметрами:

```
$ ./configure --with-test-db-
url=mysql+odbc://test:test_pwd@schedule_dsn --with-yborm-
root=/home/user/schedule/yb-mysql-inst0
```

И выполним команды:

```
$ make –j4
```

```
$ make check –j4
```

Далее нужно внести в базу необходимые таблицы:

```
$ mysql –u test –ptest_pwd schedule_db <
./src/domain/schedule_schema.sql
```

Работа с schedule-msou-srv

Для начала работы сервера **schedule-msou-srv**, его нужно запустить:

```
$ ./src/schedule
```

Обмен данными с сервером происходит при помощь **wget** запросов, набираемых в терминале. Но используемая база данных пуста:

```
$ mysql –u test –ptest_pwd schedule_db
```

```
mysql> SELECT * FROM T_USER;
```

Этот запрос не выведет никаких данных. Перед началом использования базы необходимо создать нового пользователя. Делается это при помощи следующего **sql** запроса:

```
mysql> INSERT INTO T_USER (ID, NAME, PASS, EMAIL, LOGIN, STATUS, FACULT, IS_SUPERUSER, PHONE) VALUES (1, 'Medved', '399a2ece6b34ff6e314d87301af489f0', '', 'medved', 1, 1, 1, '');
```

Где поле **PASS** заполняется значением, полученное в результате хеширования слова “preved”:

```
$ echo –n “preved” | md5sum
```

Следующий запрос выполнил вход пользователя “**medved**” с паролем “**preved**”.

```
$ wget -q -O- --header "Content-type: text/xml" --post-data "<request version='0.1' type='auth'><login>medved</login><pass>preved</pass></request>" 127.0.0.1:19090/main
```

Сервер должен ответить таким сообщением:

```
<response type="auth" version="0.1"><status>OK</status><token>5250230624896698178</token></response>
```

Где **5250230624896698178** – токен сессии.

В следующие запросы вместо **token_num** необходимо подставить выданный на предыдущем шаге сессионный токен:

- `$ wget -q -O- --header "Content-type: text/xml" --post-data "<request version='0.1' type='logout'><token>token_num</token></request>" 127.0.0.1:19090/main` – завершение текущей сессии.
- `$ wget -q -O- --header "Content-type: text/xml" --post-data "<request version='0.1' type='update'><token>token_num</token></request>" 127.0.0.1:19090/main` – обновление сессии.

Остальные запросы можно найти в файле **wget_examples** в папке **schedule-msou-srv**.

Установка и настройка ADT и Android SDK на Eclipse

Переходим на сайт:

http://developer.android.com/sdk/index.html

И скачиваем **`Android SDK for Linux`**. Распаковываем архив в любое удобное место, например, в **`~/androidsdk/`**.

Запускаем среду разработки **Eclipse**, и открываем диалог установки плагина через меню **Help → Install new software**.

В поле **Work with** вписываем адрес загрузки плагина:

`https://dl-ssl.google.com/android/eclipse`

Далее в таблице ниже появится пункт **Developer Tools**, отмечаем его и жмем **Next**.

После успешной установки, можно перезапустить среду **Eclipse**.

После перезагрузки **Eclipse** появиться диалоговое окно, предлагающее установить **SDK**. Выбираем **Install new SDK** и дополнительно отмечаем **Android 2.2**

Соглашаемся с лицензией и жмем кнопку **Finish.**

После успешной установки, можно перезапустить среду **Eclipse**.

Затем идем в меню **Window → Preferences**, далее **Android** и указываем местоположение распакованного SDK. Нажимаем кнопку **Apply.**

Далее переходим в **Window → Android SDK and AVD manager**. В **Available Packages** и устанавливаем все необходимые пакеты, например для **Android 2.2**. и **Android 4.2**

Сборка iSchedule

Запускаем среду разработки **Eclipse**, и открываем диалог установки через меню **Help → Install new software**.

В поле **Work with** вписываем адрес загрузки плагина:

`http://download.eclipse.org/egit/updates-1.3`

Устанавливаем галочку на поле **Eclipse Git Team Provider**, жмем **Next**. Соглашаемся с условиями лицензии и жмем **Next**.

После успешной установки, можно перезапустить среду **Eclipse**.

После перезапуска импортируем репозиторий **GitHub** через меню **File → Import → git → Projects from git** и жмем **Next**.

В поле URL указываем:

`https://github.com/gh0st-dog/iSchedule--MSOU-.git`

Нажимаем **Next**. Ставим галочки на загрузке обеих ветвей разработки и жмем **Next → Next**.

Далее отмечаем **iSchedule** в меню **Import Existing Project** и нажимаем кнопку **Finish**. В древе проектов должна появиться папка **iSchedule**.

Заключение

В ходе прохождения учебной практики была проделана следующая работа:

- Изучена структура предметной области задачи;
- Сконфигурирована и собрана библиотека yb-orm;
- Настроен и установлен сервер schedule-msou-srv;
- Собран редактор расписания schedule-msou-editor;
- Выявлены и задокументированы баги и недочеты в работе редактора.
- Повышены навыки программирования на языках C++ и Java, закреплены навыки написания SQL – запросов и работы в операционной системе Linux;

Поставленная задача учебной практики была выполнена в полном объеме, выбранные средства и способы реализации позволяют реализовать все необходимые функции работы отдельных компонентов и системы распространения расписания в целом.

Дата	Описание работ
1 день 18.06.13	Изучение предметной области. Клонирование настроенной системы Ubuntu Linux.
2 день 19.06.13	Клонирование настроенной системы Ubuntu Linux.
3 день 20.06.13	Клонирование настроенной системы Ubuntu Linux.
4 день 25.06.13	Загрузка исходных файлов и конфигурирование пакета yb-orm. Изучение работы утилиты make.
5 день 26.06.13	Сборка пакета yb-orm. Изучение компонентов Autotools. Настройка тестовой базы данных sqlite3. Запуск тестов.
6 день 27.06.13	Сборка и конфигурирование тестового серверного приложения auth. Проверка его правильности работы.
7 день 2.07.2013	Загрузка и конфигурирование пакета schedule-msou-srv. Конфигурирование yb-orm.
8 день 3.07.2013	Изучение принципов работы MySQL. Создание тестовой базы данных MySQL и импортирование необходимых таблиц.
9 день 4.07.2013	Сборка пакета yb-orm и schedule-msou-srv на базе данных MySQL через интерфейс ODBC.
10 день 5.07.2013	Изучение методов HTTP (POST, GET). Тестирование и работа с базой данных MySQL.
11 день 8.07.2013	Загрузка и сборка пакета schedule-msou-editor. Устранение ошибок. Изучение IDE Qt Creator.
12 день 9.07.2013	Установка плагина Android Development Tools и Android SDK в Eclipse (версии 2.2 и 4.2)
13 день 10.07.2013	Сборка приложения для Android. Изучение объектно-ориентированного языка программирования Java.
14 день 11.07.2013	Подведение итогов практики. Подготовка отчета о проделанной работе.