МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра проектирования безопасности компьютерных систем

Управление мобильными устройствами

Лабораторная работа № 1
по теме «Обработка и тарификация CDR (Call Detail Record)»

Работу выполнил
студент группы №3351
очного отделения
Чопик Степан
Вариант 7

Проверил

_____

Федоров И. Р.

Санкт —Петербург 2020

Цель работы: разработать программный модуль, реализующий простейшее правило тарификации для услуг типа “Телефония” по длительности разговора и “СМС” по общему количеству.

Задачи:

1. Распарсить файл CDR и выбрать нужные строки
2. Протарифицировать указанного в задании абонента

Ход работы:

Условие моего варианта:

Протарифицировать абонента с номером 933156729 с коэффициентом k: 4руб/минута исходящие звонки и входящие звонки до 0:30, 2руб/минута исходящие звонки и входящие звонки после 0:30, смс - 1,5руб/шт.

Правила тарификации услуг “Телефония”:

X = T * k, где

X – итоговая стоимость всех звонков абонента,

T – общая длительность звонков (сумма длительностей всех записей по абоненту),

k – множитель тарифного плана (у каждого варианта свой).

Правила тарификации услуг “СМС”:

Y = N * k, где

Y – итоговая стоимость всех СМС абонента,

N – общее количество СМС (сумма числа всех СМС в записях по абоненту в файле),

k – множитель тарифного плана (у каждого варианта свой).

Таким образом, для тарификации указанного номера необходимо определить:

1) продолжительность входящих и исходящих звонков до 00:30:00,
2) продолжительность входящих и исходящих звонков после 00:30:00,
3) количество отправленных номером смс.

Для получения итоговой стоимости все 3 указанных выше значения необходимо умножить на соответствующий коэффициент k и просуммировать.

Реализуем программный модуль на языке Python 3.7. CDR файл представлен в формате csv, который является обычным текстовым файлом с определенным и одинаковым разделителем в каждой строке.

Значение полей файла CDR:

- timestamp - время звонка
- msisdn_origin - кто совершил звонок
- msisdn_dest - кому звонили
- call_duration - длительность звонка в минутах
- sms_number - количество отправленных смс для абонента msisdn_origin

Файл CDR:

timestamp	msisdn_origin	msisdn_dest	call_duration	sms_number
2020-01-01 0:00:00	915783624	911926375	36.23	15
2020-01-01 0:05:00	911926375	968247916	9.2	5
2020-01-01 0:10:00	936415793	915642913	7.52	24
2020-01-01 0:15:00	914976835	914976835	96.7	97
2020-01-01 0:20:00	962365794	933156729	110.44	15
2020-01-01 0:25:00	966714385	915783624	12.34	5
2020-01-01 0:30:00	968247916	962365794	91.48	57
2020-01-01 0:35:00	933156729	936415793	83.22	73
2020-01-01 0:40:00	915642913	966714385	85.7	18

Исходный код(lab1.py) получившегося программного модуля:

```python
# Вариант 7

import sys

# коэффициент звонков до 0:30
BCOST = 4.0
# коэффициент звонков до 0:30
ACOST = 2.0
# коэффициент СМС
SMSCOST = 1.5

# Время, отноительно которого изменяется коэффциент
TIMECONST = "00:30:00"

# номер из варианта 7
DEFAULT_PHONE = "933156729"

# Считываем данные из файла
mas = []
with open("./data.csv") as cdr:
    for line in cdr:
        mas.append(line[:len(line)-1].split(','))

phone = input("Пожалуйста, введите номер для тарификации:\n")
# если не введен другой номер, используется номер из варианта 7
if not phone.isdigit():
    print("Введенный номер не соответствует формату.\nБудет протарифицирован номер 933156729")
    phone = DEFAULT_PHONE
```

```python
sum = 0
calls_sum = 0
for line in mas:
    # Тарифицируем исходящие звонки и СМС для номера
    if line[1] == phone:

        datetime = line[0]
        time = datetime[datetime.find(":")-2:]

        try:
            duration = float(line[3])
            print("Ошибка в call_duration")
        except:
            sys.exit()


        sms = int(line[-1])

        if time < TIMECONST:
            time = float(datetime[datetime.find(":")+1:datetime.rfind(":")]) + float(datetime[datetime.rfind(":")+1:])/60
            # Сколько осталось до 0:30
            ost = 30.0 - time
            # Если человек начал говорить раньше 0:30, а закончил позже,
            # то находим соответствующие промежутки до(ost) и после(after)
            if duration > ost:
                after = duration - ost
                # Считаем стоимость промежутков по соответствующим коэффициентам
                calls_sum = ost * BCOST + after * ACOST
            else:
                # Иначе считаем по коэффициенту до 0:30
                calls_sum = duration * BCOST
        else:
            # Если звонок начался после 0:30, то тарифицируем по коэффициенту после 0:30
            calls_sum = duration * ACOST
        sum += calls_sum
        sms_sum = sms * SMSCOST
        sum += sms_sum
```

```
    elif line[2] == phone:
        datetime = line[0]
        time = datetime[datetime.find(":")-2:]

        try:
            duration = float(line[3])
        except:
            print("Ошибка в call_duration")
            sys.exit()

        if time < TIMECONST:
            time = float(datetime[datetime.find(":")+1:datetime.rfind(":")]) + float(datetime[datetime.rfind(":")+1:])/60
            # Сколько осталось до 0:30
            ost = 30.0 - time
            # Если человек начал говорить раньше 0:30, а закончил позже,
            # то находим соответствующие промежутки до(ost) и после(after)
            if duration > ost:
                after = duration - ost
                # Считаем стоимость промежутков по соответствующим коэффициентам
                calls_sum = ost * BCOST + after * ACOST
            else:
                # Иначе считаем по коэффициенту до 0:30
                calls_sum = duration * BCOST
        else:
            # Если звонок начался после 0:30, то тарифицируем по коэффициенту после 0:30
            calls_sum = duration * ACOST
        sum += calls_sum

print(round(sum, 2))
```

Выводы:

В ходе работы были получены теоретические знания о биллинге и тарификации абонентов, а так же было практически реализовано простейшее правило тарификации. Был разработан программный модуль для тарификации абонента по заданному в условии варианта правилу. В программном модуле есть возможность изменить номер, для которого проводится тарификация. Таким образом, были получены практические умения разработки компонента возможной биллинговой системы.

Готовый программный модуль был размещен в публичном Github репозитории(07.04.2020):
https://github.com/StepChop/Chopik-N3351-MDM2020