

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**


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 № 3
по теме «Формирование счета на оплату услуг»

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

Проверил

Федоров И. Р.

Цель работы: Сформировать счет на оплату услуг в формате pdf и разработать программный модуль, заполняющий этот счет на основе данных из ранее разработанных модулей.

Ход работы:

Для реализации программного модуля используем язык программирования Python 3.8. В качестве шаблона счета используем html форму, заполнение которой будем производить с помощью шаблонизатора jinja2.

Jinja2 — самый популярный шаблонизатор в языке программирования Python. Jinja2 позволяет использовать тэги для заполнения html документа данными из передаваемых в Jinja2 аргументов. Для этого в исходном html-шаблоне, в местах, которые должны быть заполнены для каждого счета отдельно, расставлены тэги, на место которых шаблонизатор подставляет необходимые данные.

Итоговый вид шаблона представлен в листинге “Шаблон_счета.html”.

```
<table width="100%" cellpadding="2" cellspacing="2" class="invoice_bank_rekv">
  <tr>
    <td colspan="2" rowspan="2" style="min-height:13mm; width: 105mm;">
      <table width="100%" border="0" cellpadding="0" cellspacing="0" style="height: 13mm;">
        <tr>
          <td valign="top">
            <div>{{context.bank_name}}</div>
          </td>
        </tr>
        <tr>
          <td valign="bottom" style="height: 3mm;">
            <div style="font-size:10pt;">Банк получателя </div>
          </td>
        </tr>
      </table>
    </td>
    <td style="min-height:7mm;height:auto; width: 25mm;">
      <div>БИК</div>
    </td>
    <td rowspan="2" style="vertical-align: top; width: 60mm;">
      <div style=" height: 7mm; line-height: 7mm; vertical-align: middle;">{{context.BIK}}</div>
      <div>{{context.bank_account}}</div>
    </td>
  </tr>
  <tr>
    <td style="width: 25mm;">
      <div>СЧ. №</div>
    </td>
  </tr>
</table>
```

Рисунок 1 - Пример содержания шаблона счета.

Для полученных в 1 и 2 лабораторных работах программных модулей, создадим функции, возвращающие итоговые значения.

```
def traffic(settings):
    data = []
    input_file = settings["input_filename"]
    with open(input_file, "r") as file:
        for line in file:
            data.append(line.split(", "))

    # IP-адрес из условия Впра
    IP = settings["IP"]
    k = settings["COST"]
    traffic = 0
    traffic_range = []
    data_mas = {}
    for i in range(len(data)-3):
        line = data[i]
        ts = line[0]
        sa = line[3]
        da = line[4]
        ibyt = line[12]
        if sa == IP or da == IP:
            traffic += int(ibyt)

    traffic /= 1024*1024

    return round(traffic * k, 2)
```

Рисунок 2 - Функция расчета стоимости услуги “Интернет”

```
def mobile(settings):
    BCOST = settings["BCOST"]
    # коэффициент звонков до 0:30
    ACOST = settings["ACOST"]
    # коэффициент СМС
    SMSCOST = settings["SMSCOST"]

    # Время, относительно которого изменяется коэффициент
    TIMECONST = settings["TIMECONST"]

    # номер из варианта 7
    DEFAULT_PHONE = settings["DEFAULT_PHONE"]

    input_filename = settings["input_filename"]
    # считываем данные из файла
    mas = []
    with open(input_filename) as cdr:
        for line in cdr:
            mas.append(line[:len(line)-1].split(','))

    phone = DEFAULT_PHONE
    # 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

    sum = 0
    calls_sum = 0
```

Рисунок 3 - Функция расчета стоимости услуги “Телефония” (Часть 1).

```

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

```

Рисунок 4 - Функция расчета стоимости услуги “Телефония” (Часть 2).

```

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

return round(sum, 2)

```

Рисунок 5 - Функция расчета стоимости услуги “Телефония” (Часть 3).

Далее получим все необходимые для подстановки данные и сформируем подставляемые текстовые значения.

```
# Рассчитываем стоимости услуг
traffic_cost = traffic(traffic_settings)
mobile_cost = mobile(mobile_settings)
total = traffic_cost + mobile_cost
total_rub = int(total//1)

# Выбираем правильные окончания рублей и копеек
rublei = "руб"
if total_rub % 100 in range(10, 20):
    rublei += "лей"
elif total_rub % 10 in [2,3,4]:
    rublei += "ля"
elif total_rub % 10 in [0, 5, 6, 7, 8, 9]:
    rublei += "лей"
elif total_rub % 10 == 1:
    rublei += "ль"

total_santi = int(round(total%1, 2)*100)
santi = "копеек"
if total_rub % 100 in range(10, 20):
    santi += "ек"
elif total_rub % 10 in [2,3,4]:
    santi += "ки"
elif total_rub % 10 in [0, 5, 6, 7, 8, 9]:
    santi += "ек"
elif total_rub % 10 == 1:
    santi += "йка"

# переводим сумму в текст
total_text = num2text(total_rub)
total_text = total_text[:1].upper() + total_text[1:]

# формируем заголовок счета
today = datetime.datetime.now()
day = today.day
month = month[today.month]
year = today.year
bill_name = f"Счет на оплату № {random.randint(1, 1000)} от {day} {month} {year} г."
```

Рисунок 6 - Расчет подставляемых данных.

Далее используя шаблонизатор подставим значения в html-шаблон. Для этого сформируем словарь context с содержимым тегов и передадим его функции render класса Template. Полученный текст сохраним во временный html-файл.

```

# Словарь для шаблонизатора
context = {
    "main_director":bill_settings["руководитель"]["значение"],
    "main_counter":bill_settings["бухгалтер"]["значение"],
    "bill_title":bill_name,
    "INN":bill_settings["ИНН"]["значение"],
    "KPP":bill_settings["КПП"]["значение"],
    "bank_account":bill_settings["счет банка"]["значение"],
    "bank_name":bill_settings["Банк получателя"]["значение"],
    "BIK":bill_settings["БИК"]["значение"],
    "total_text":total_text,
    "rublei":rublei,
    "total_santi":total_santi,
    "santi":santi,
    "total":total,
    "tovar_name_1":bill_settings["товары_1"]["значение"],
    "cost_1":traffic_cost,
    "tovar_name_2":bill_settings["товары_2"]["значение"],
    "cost_2":mobile_cost,
    "client_snp":bill_settings["покупатель"]["значение"],
    "seller":bill_settings["поставщик"]["значение"],
    "seller_account":bill_settings["счет поставщика"]["значение"]
}

template = Template(html)
output = template.render(context=context)

# temp_output - путь до временного файла с расширением html
# output_filename - путь до pdf
with open(temp_output, 'wb') as f:
    f.write(output.encode('utf-8'))

```

Рисунок 7 - Использование шаблонизатора.

Далее, с помощью модуля `pdftk` переводим временный `html` файл в `pdf` используя настройки страницы `options` и удаляем временный файл.

```

pdftk.from_file(temp_output, output_filename, options=options, configuration = config)
os.remove(temp_output)
print("Готово")

```

Рисунок 8 - Сохранение `html` как `pdf`.

```

options = {
    'page-size': 'A5',
    'margin-top': '2cm',
    'margin-left': '3cm',
    'margin-right': '2cm'
}

```

Рисунок 9 - Словарь с параметрами страницы.

Постоянные поля, одинаковые для одно предприятия хранятся в словаре `bill_settings` в файле `settings_2.py`. Так же в нем хранятся словари `traffic_settings` и `mobile_settings` с настройками для тарификации телефонии и интернета. Код программы `lab3.py` и `settings.py` представлен в листинге.

```
bill_settings = {...  
}  
  
traffic_settings = {  
    "IP": "87.245.198.147",  
    "COST": 2,  
    "input_filename": "G:\\Stepa\\Учёба\\УМУ\\Лаб 3\\traffic.csv"  
}  
  
mobile_settings = {  
    "input_filename": "G:\\Stepa\\Учёба\\УМУ\\Лаб 3\\data.csv",  
    "BCOST": 4.0,  
    "ACOST": 2.0,  
    "SMSCOST": 1.5,  
    "TIMECONST": "00:30:00",  
    "DEFAULT_PHONE": "933156729"  
}
```

Рисунок 10 - Содержимое `settings.py`

Вывод

В ходе работы были разработан программный модуль, позволяющий автоматизировать формирование счета на оплату услуг мобильной связи и интернета. При разработке программного модуля, было выявлено, что для упрощения работы с формой выгоднее использовать для ее создания язык разметки html. Тогда для формирования счета можно использовать технологию шаблонизаторов, например Jinja2 в Python 3, по которой в необходимые места html документа проставляются динамические данные. Кроме того, необходима система хранения персональных данных пользователей, так как в случае с несколькими клиентами, использование одного файла с настройками не является возможным. Данные организации, могут либо так же динамически подставляться программным модулем, что позволит сохранять их актуальность без изменения шаблона счета, либо быть частью шаблона, что упрощает разработку модуля для формирования счета. В остальном же, подстановка данных в структуру гипертекста не является проблемой и может быть произведена и без шаблонизатора.

Отдельной задачей является формирование человеко-читаемых формальных элементов, например, склонений при написании чисел текстом.

В результате были получены знания и умения для автоматизации формирования счета на оплаты, например, для биллинговых систем, а также был разработан программный модуль, который с некоторыми корректировками может быть использован на практике.

Листинг программ

lab3.py

```
from openpyxl import Workbook, load_workbook
from ru_number_to_text import *
import random
import datetime
from settings import bill_settings, traffic_settings, mobile_settings
import os
import pdfkit
from jinja2 import Template

path_wkhtmltopdf = r'C:\\Program Files\\wkhtmltopdf\\bin\\wkhtmltopdf.exe'
config = pdfkit.configuration(wkhtmltopdf=path_wkhtmltopdf)
month = ['', 'января', 'февраля', 'марта', 'апреля', 'мая', 'июня',
          'июля', 'августа', 'сентября', 'октября', 'ноября', 'декабря']

def traffic(settings):
    data = []
    input_file = settings["input_filename"]
    with open(input_file, "r") as file:
        for line in file:
            data.append(line.split(","))
    # IP-адрес из условия Бра
    IP = settings["IP"]
    k = settings["COST"]
    traffic = 0
    traffic_range = []
    data_mas = {}
    for i in range(len(data)-3):
        line = data[i]
        ts = line[0]
        sa = line[3]
        da = line[4]
        ibyt = line[12]
        if sa == IP or da == IP:
            traffic += int(ibyt)
    traffic /= 1024*1024
    return round(traffic * k, 2)

def mobile(settings):
    BCOST = settings["BCOST"]
    # коэффициент звонков до 0:30
```

```

ACOST = settings["ACOST"]
# коэффициент СМС
SMSCOST = settings["SMSCOST"]
# Время, относительно которого изменяется коэффициент
TIMECONST = settings["TIMECONST"]
# номер из варианта 7
DEFAULT_PHONE = settings["DEFAULT_PHONE"]
input_filename = settings["input_filename"]
# Считываем данные из файла
mas = []
with open(input_filename) as cdr:
    for line in cdr:
        mas.append(line[:len(line)-1].split(','))
phone = DEFAULT_PHONE
# 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
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
    return round(sum, 2)

# Файл с шаблоном, временный файл, файл с результатом
input_filename = "G:\Stepa\Учёба\УМУ\Лаб 3\invoice.html"
temp_output = "G:\Stepa\Учёба\УМУ\Лаб 3\Счет.html"
output_filename = "G:\Stepa\Учёба\УМУ\Лаб 3\Счет.pdf"
# открываем на чтение файл с шаблоном и считываем содержимое
with open(input_filename, 'r', encoding='utf-8') as f:
    html = f.read()

# Рассчитываем стоимости услуг
traffic_cost = traffic(traffic_settings)
mobile_cost = mobile(mobile_settings)
total = traffic_cost + mobile_cost
total_rub = int(total//1)
# Выбираем правильные окончания рублей и копеек
rublei = "руб"
if total_rub % 100 in range(10, 20):
    rublei += "лей"
elif total_rub % 10 in [2,3,4]:
    rublei += "ля"
elif total_rub % 10 in [0, 5, 6, 7, 8, 9]:
    rublei += "лей"
elif total_rub % 10 == 1:
    rublei += "ль"
total_santi = int(round(total%1, 2)*100)
santi = "копе"
if total_rub % 100 in range(10, 20):
    santi += "ек"
elif total_rub % 10 in [2,3,4]:
    santi += "ки"

```

```

elif total_rub % 10 in [0, 5, 6, 7, 8, 9]:
    santi += "ек"
elif total_rub % 10 == 1:
    santi += "йка"
# переводим сумму в текст
total_text = num2text(total_rub)
total_text = total_text[:1].upper() + total_text[1:]
# формируем заголовок счета
today = datetime.datetime.now()
day = today.day
month = month[today.month]
year = today.year
bill_name = f"Счет на оплату № {random.randint(1, 1000)} от {day}
{month} {year} г."
# настройки для pdf файла
options = {
    'page-size': 'A5',
    'margin-top': '2cm',
    'margin-left': '3cm',
    'margin-right': '2cm'
}
# Словарь для шаблонизатора
context = {
    "main_director":bill_settings["руководитель"]["значение"],
    "main_counter":bill_settings["бухгалтер"]["значение"],
    "bill_title":bill_name,
    "INN":bill_settings["ИНН"]["значение"],
    "KPP":bill_settings["КПП"]["значение"],
    "bank_account":bill_settings["счет банка"]["значение"],
    "bank_name":bill_settings["Банк получателя"]["значение"],
    "BIK":bill_settings["БИК"]["значение"],
    "total_text":total_text,
    "rublei":rublei,
    "total_santi":total_santi,
    "santi":santi,
    "total":total,
    "tovar_name_1":bill_settings["товары_1"]["значение"],
    "cost_1":traffic_cost,
    "tovar_name_2":bill_settings["товары_2"]["значение"],
    "cost_2":mobile_cost,

```

```
        "client_snp":bill_settings["покупатель"]["значение"],
        "seller":bill_settings["поставщик"]["значение"],
        "seller_account":bill_settings["счет поставщика"]["значение"]
    }
    template = Template(html)
    output = template.render(context=context)
    # temp_output - путь до временного файла с расширением html
    # output_filename - путь до pdf
    with open(temp_output, 'wb') as f:
        f.write(output.encode('utf-8'))
    pdfkit.from_file(temp_output, output_filename, options=options,
configuration = config)
    os.remove(temp_output)
    print("Готово")
```

settings.py

```
bill_settings = {

    'Банк получателя':
        {
            'значение': 'БАНК СЕВЕРО-ВОСТОЧНЫЙ БАНК ПАО
СБЕРБАНК Г. САНКТ - ПЕТЕРБУРГ',
            'ячейка': 'bank_name'
        },

    'БИК':
        {
            'значение': '044030653',
            'ячейка': 'BIK'
        },

    'счет банка':
        {
            'значение': '30101810500000000653',
            'ячейка': 'bank_account'
        },

    'счет поставщика':
        {
            'значение': '40702810855000100555',
            'ячейка': 'seller_account'
        },

    'ИНН':
        {
            'значение': '7707049388',
            'ячейка': 'INN'
        },

    'КПП':
        {
            'значение': '784243002',
            'ячейка': 'KPP'
        },

    'поставщик':
        {
            'значение': 'Макрорегиональный филиал "Северо -
Запад" ПАО "РОСТЕЛЕКОМ"',
            'ячейка': 'client_snp'
        },
}
```

```

'покупатель':
    {
        'значение': 'Фёдоров Иван Романович',
        'ячейка': 'client_snpr'
    },

'товары_1':
    {
        'значение': 'Домашний интернет',
        'ячейка': 'tovar_name_1'
    },
'товары_2':
    {
        'значение': 'Мобильная связь',
        'ячейка': 'tovar_name_2'
    },

'руководитель':
    {
        'значение': 'Абрамчук М.В.',
        'ячейка': 'main_director'
    },
'бухгалтер':
    {
        'значение': 'Яковлева М.Н.',
        'ячейка': 'main_counter'
    }
}

```

```

traffic_settings = {
    "IP": "87.245.198.147",
    "COST": 2,
    "input_filename": "G:\Stepa\Учёба\УМУ\Лаб 3\traffic.csv"
}

```

```

mobile_settings = {
    "input_filename": "G:\Stepa\Учёба\УМУ\Лаб 3\data.csv",
    "BCOST": 4.0,
    "ACOST": 2.0,
    "SMSCOST": 1.5,
}

```



```
"TIMECONST": "00:30:00",
"DEFAULT_PHONE": "933156729"
}
```

Шаблон_счета.html

```
<!doctype html>
<html>
<head>
  <title>{{context.bill_title}}</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <style>
    body { width: 210mm; margin-left: auto; margin-right: auto;
border: 1px #efefef solid; font-size: 11pt;}
    table.invoice_bank_rekv { border-collapse: collapse; border:
1px solid; }
        table.invoice_bank_rekv > tbody > tr > td,
table.invoice_bank_rekv > tr > td { border: 1px solid; }
        table.invoice_items { border: 1px solid; border-collapse:
collapse;}
        table.invoice_items td, table.invoice_items th { border: 1px
solid;}
  </style>
</head>
<body>
<table width="100%">
  <tr>
    <td>&nbsp;</td>
    <td style="width: 155mm;">
      <div style="width:155mm; ">Внимание! Оплата данного счета
означает согласие с условиями поставки товара. Уведомление об оплате
обязательно, в противном случае не гарантируется наличие товара на
складе. Товар отпускается по факту прихода денег на р/с Поставщика,
самовывозом, при наличии доверенности и паспорта.</div>
    </td>
  </tr>
  <tr>
  </tr>
</table>
<table          width="100%"          cellpadding="2"          cellspacing="2"
class="invoice_bank_rekv">
```

```
|  |  |  |  |  |  | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| |                                                         | |---------------------------------------------------------| | <div>&lt;div&gt;{{context.bank_name}}&lt;/div&gt;</div> | | <div style="font-size:10pt;">Банк получателя</div>      | | | БИК | {{context.BIK}}  {{context.bank_account}} |
| Сч. № | ИИН {{context.INN}} | КПП {{context.KPP}} |

```

```

                <td rowspan="2" style="min-height:19mm; height:auto;
vertical-align: top; width: 25mm;">
                    <div>Сч. №</div>
                </td>
                <td rowspan="2" style="min-height:19mm; height:auto;
vertical-align: top; width: 60mm;">
                    <div>{{context.seller_account}}</div>
                </td>
            </tr>
            <tr>
                <td colspan="2" style="min-height:13mm; height:auto;">
                    <table border="0" cellpadding="0" cellspacing="0"
style="height: 13mm; width: 105mm;">
                        <tr>
                            <td valign="top">
                                <div>{{context.seller}}</div>
                            </td>
                        </tr>
                        <tr>
                            <td valign="bottom" style="height: 3mm;">
                                <div style="font-size: 10pt;">Получатель</div>
                            </td>
                        </tr>
                    </table>
                </td>
            </tr>
        </table>
        <br/>
        <div style="font-weight: bold; font-size: 16pt; padding-left:5px;">
            {{context.bill_title}}</div>
        <br/>
        <div style="background-color:#000000; width:100%; font-size:1px;
height:2px;">&nbsp;</div>
        <table width="100%">
            <tr>
                <td style="width: 30mm;">
                    <div style="padding-left:2px;">Поставщик: </div>
                </td>
                <td>
                    <div style="font-weight:bold; padding-left:2px;">

```

```

                {{context.seller}}                </div>
            </td>
        </tr>
        <tr>
            <td style="width: 30mm;">
                <div style=" padding-left:2px;">Покупатель:    </div>
            </td>
            <td>
                <div style="font-weight:bold; padding-left:2px;">
                    {{context.client_snp}}                </div>
                </td>
        </tr>
    </table>
    <table        class="invoice_items"        width="100%"        cellpadding="2"
    cellspacing="2">
        <thead>
        <tr>
            <th style="width:13mm;">№</th>
            <th>Товар</th>
            <th style="width:20mm;">Кол-во</th>
            <th style="width:17mm;">Ед.</th>
            <th style="width:27mm;">Цена</th>
            <th style="width:27mm;">Сумма</th>
        </tr>
        </thead>
        <tbody >
            <tr>
                <td style="width:13mm; ">1</td>
                <td>{{context.tovar_name_1}}</td>
                <td style="width:20mm; ">-</td>
                <td style="width:17mm; ">-</td>
                <td style="width:27mm; text-align: center;
">{{context.cost_1}}</td>
                <td style="width:27mm; text-align: center;
">{{context.cost_1}}</td>
            </tr>
            <tr>
                <td style="width:13mm; ">2</td>
                <td>{{context.tovar_name_2}}</td>
                <td style="width:20mm; ">-</td>

```

```
  |
```