Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Работа выполнена в СКБ «Интеллектуальные системы»

СОГЛАСОВАНО

Декан ФКТ

_____ Я.Ю. Григорьев
(подпись)
«____ » _____ 20___ г.

УТВЕРЖДАЮ

Начальник отдела ОПРО

_____ Ю.С. Иванов
(подпись)
«____ » _____ 20___ г.

Заведующий кафедрой МОП ЭВМ
_____ В.А. Тихомиров
(подпись)
«____ » _____ 20___ г.

«Веб-сервис по наложению тематического изображения на изображение человеческого лица»

Комплект проектной документации

Руководитель проекта _____ Е.П. Жарикова
(подпись, дата)

Ответственный исполнитель _____ А.Д. Палыгин
(подпись, дата)

Комсомольск-на-Амуре 2019

Карточка проекта

Название	Веб-сервис по наложению тематического изображения на изображение человеческого лица
Тип проекта	рамках конкурса (инициативный, по заказу, в рамках конкурса, учебная работа, другое)
Исполнители	Палыгин А.Д. – 8ВТб-1 ответственный исполнитель Душкин Е.П. – 9ИБ-1 Макуха А.А – 9ИБ-1 Курбонназаров Д.А. – 7БМб-1
Срок реализации	10.2019-12.2019 Месяц, год

Использованные библиотеки и компоненты

Наименование
PyQt5
Imutils
Numpy
Threading
Time
Sys
cv2
Dlib
Imageio
Math
Imutils
Datetime
Time
Pymysql
Ftplib
Random
String
Qrcode

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ЗАДАНИЕ

на разработку

Выдано студентам: Палыгину А.Д – 8ВТб-1, Душкину Е.П. – 9ИБ-1, Макуха А.А. – 9ИБ-1, Курбонназарову Д.А. – 7БМб-1_____

_____

Название проекта: Веб-сервис по наложению тематического изображения на изображение человеческого лица _____

_____

Назначение: Нанесение на изображение человеческого лица цветов и логотипов популярных спортивных команд г. Хабаровска и г. Комсомольска-на-Амуре в режиме реального времени _____

Область использования: Спортивные мероприятия, городские праздники, различные общественные мероприятия_____

Функциональное описание: Пользователь выбирает маску, программа накладывает на пользователя маску, пользователь нажимает кнопку для фото, программа делает фотографию, показывает пользователю готовое фото вместе с QR-кодом, пользователь скачивает фото по QR-коду_____

Структура программы: _____

Веб сервис стоит из следующих структурных модулей:

1. _____

_____

_____

_____

_____

Требования: Программа должна уметь захватывать более 3 лиц, время отклика должно быть меньше 2 секунд, угол поворота лица необходимый для захвата должен быть до 30^{o} в каждом из направлений_____

План работ:

Наименование работ	Срок
Разработать идею	10.2019
Разработать алгоритм программы	10.2019
Определить список библиотек	10.2019
Написать демо-версию (прототип)	10.2019
Составить блок-схемы	11.2019
Написать итоговую версию	11.2019
Составить документацию	12.2019

Комментарии:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Перечень графического материала:

1. Схема базы данных_____
2. Блок-схема работы_____
3. Скриншоты_____
4. Листинг_____

_____

_____

_____

_____

Руководитель проекта _____ Е.П. Жарикова

(подпись, дата)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

РАБОЧАЯ ДОКУМЕНТАЦИЯ

Программа для ЭВМ

«Веб-сервис по наложению тематического изображения на изображение человеческого лица»

Руководитель проекта _____ Е.П. Жарикова

(подпись, дата)

Ответственный исполнитель _____ А.Д. Палыгин

(подпись, дата)

Комсомольск-на-Амуре 2019

Аннотация

Настоящий документ представляет собой руководство администратора к информационный системе – веб-сервиса по наложению тематического изображения на изображение человеческого лица (далее ИС ВС.)

Руководство определяет порядок установки, настройки и администрирования системы.

Перед установкой и эксплуатацией системы рекомендуется внимательно ознакомиться с настоящим руководством.

Документ подготовлен в соответствии с РД 50-34.698-90 - в части структуры и содержания документов, и в соответствии с ГОСТ 34.201-89 - в части наименования и обозначения документов.

Содержание

1 ОБЩИЕ ПОЛОЖЕНИЯ

1.1 Список обозначений и сокращений

АРМ – Автоматизированное рабочее место

БД – База данных

ИС – Информационная система

ПО – Программное обеспечение

РФ – Российская Федераци

СУБД – Система управления базами данных

ТЗ – Техническое задание

ФЗ – Федеральный закон

Система – Информационная система «Веб-сервиса по наложению тематического изображения на изображение человеческого лица» (ИС ВС)

HTTP - HyperText Transfer Protocol

2 НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Система создана для реализации следующих целей:

– Популяризации спортивных мероприятий

– Популяризация городских/общественных мероприятий

С использованием Системы реализуются следующие функции:

– получение видеопотока

– нахождение лица

– выбор маски с помощью интерфейса, реализованого на PyQt

– изменение изображения путем нахождения лица и наложения изображения маски

– сфотографировать

– сохранение полученного изображения на сервере

– Генерация QR-кода, с помощью которого можно скачать полученное изображение с сервера

Система обеспечивает следующее информационное взаимодействие:

– взаимодействие с ОС телефона путём скаивания через QR-код

2.1 Уровень подготовки персонала

Для штатной эксплуатации ИС ВС необходимо привлечение следующих групп персонала:

Обслуживающий персонал:

− администратор Системы;

− специалист по техническому обслуживанию.

Для штатной эксплуатации ИС ВС нет необходимости в привлечении дополнительного персонала.

3 АРХИТЕКТУРНЫЕ РЕШЕНИЯ

3.1 Архитектурные решения системы

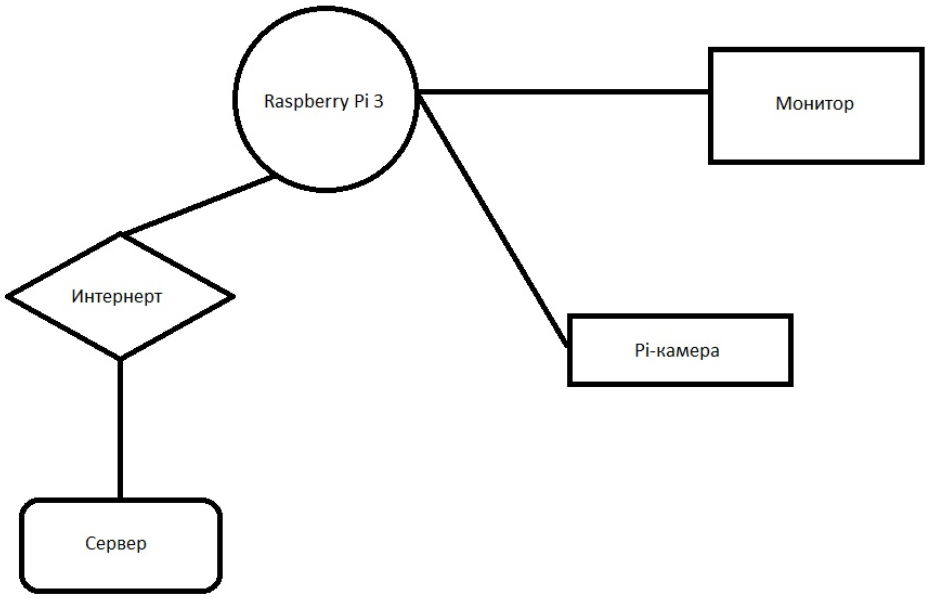


Рисунок 1. Схема ИС ВС

3.1.1 Требования к программному обеспечению системы

ИС ВС состоит из следующих компонентов:

− Веб-сервер;

− Raspberry Pi 3;

− Дисплей;

− Pi-cam.

Для работы администраторов используются АРМ администраторов ИС ВС.

В качестве технологической платформы веб-сервера используется продукт Apache HTTPD 2.4.

Технологическую основу сервера приложений составляют продукты Apache Service Mix 4.5, и сервисная шина Glass Fish Server 3.1.

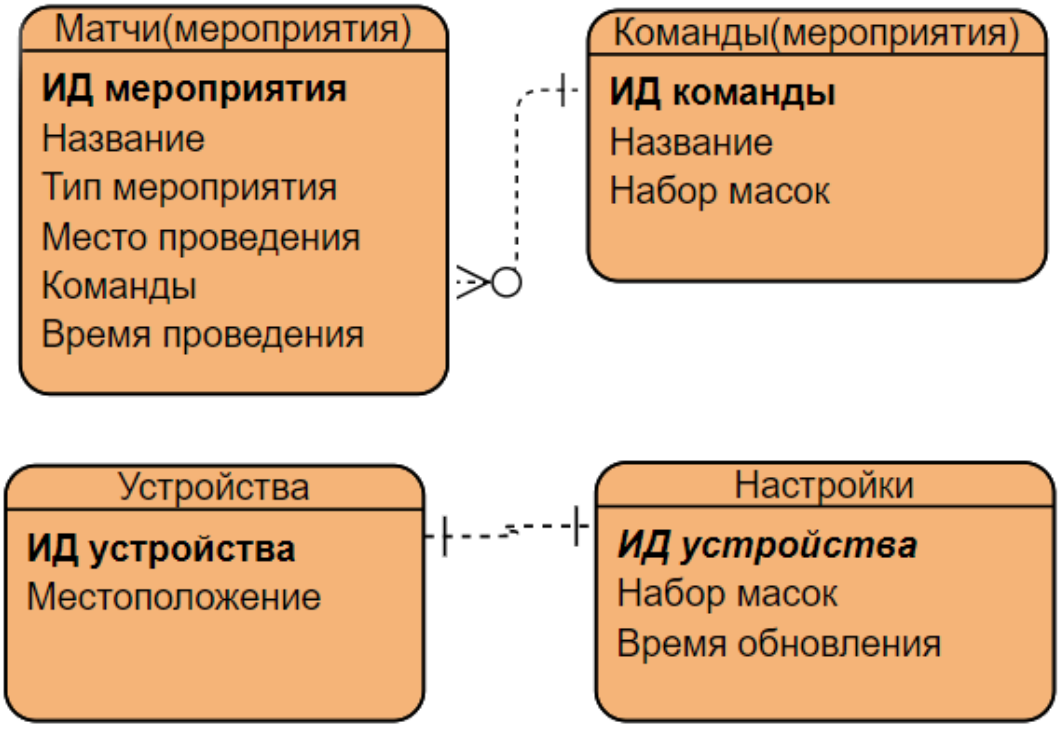
Требования к программному обеспечению УНП представлены в таблице 1 настоящего документа.

Таблица 1. Требования к программному обеспечению

Имя сервера	Требования к операционной системе
Веб-сервер	Microsoft Windows Server 2008 R2 Standard/Enterprise Edition, SP1.
Сервер приложений	Microsoft Windows Server 2008 R2 Standard/Enterprise Edition, SP1.
СУБД	Microsoft Windows Server 2008 R2 Standard/Enterprise Edition, SP1.

3.2 Порядок сопровождения системы программистом/руководство программииста

3.3 Стурктура системы



3.4 API системы/функции

- Функция SAVE_PHOTO - Сохранить фотографию на сервере.
- Функция GET_PHOTO - Получить фотографию с сервера.
- Функция SETTINGS - Передать настройки ИС ВС.

4 УСТАНОВКА И НАСТРОЙКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

4.1 Установка программного обеспечения

Необходимо установить Python 3, скачать програму и установить все необходимые библиотеки, описанные выше в документе.

4.2 Настройка программного обеспечения

Изменить настройки программы в БД.

5 ПОРЯДОК ЭКСПЛУАТАЦИИ СИСТЕМЫ

5.1 Порядок запуска

1. Перейти в директорию программы
2. В терминале ввести команду «python3 main.py»

ПРИЛОЖЕНИЕ А

(обязательное)

```
/*************************************************************/
/*Description:Подключение библиотек                          */
/*Unit has been developed with Python 3                      */
/*Creation Date: 10/11/2019                                  */
/*Author: IT'men                                             */
/*Version 1.0                                                */
/*************************************************************/


from PyQt5 import QtCore
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton,
QColorDialog
from PyQt5 import QtGui
from PyQt5.QtGui import QColor
from imutils import face_utils
import numpy as np
import gui
import threading
import time
import sys
import cv2
import dlib
import imageio
import math
import imutils
import datetime
import time
import pymysql
import ftplib
import random
import string
import qrcode



/*************************************************************/
/*Description:Инициализация программы                        */
/*Unit has been developed with Python 3                      */
/*Creation Date: 10/11/2019                                  */
/*Author: IT'men                                             */
/*Version 1.0                                                */
/*************************************************************/

def main():
    app = QtWidgets.QApplication(sys.argv)
    window = ExampleApp()
    window.show()
```

```
    app.exec_()

if __name__ == '__main__':
    main()



/*************************************************************/
/*Description:Первоночальная настройка программы             */
/*Unit has been developed with Python 3                      */
/*Creation Date: 10/11/2019                                  */
/*Author: IT'men                                             */
/*Version 1.0                                                */
/*************************************************************/

    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.insta_post = False
        self.video_post=False
        self.start_time=time.time()
        t = threading.Thread(target=self.camera)
        t.start()
        self.fourcc = cv2.VideoWriter_fourcc(*'XVID')
        self.counter=1
        self.number=1
        self.add_five = 0

self.label_3.setPixmap(QtGui.QPixmap("resources/logo.png"))
        self.pb_mask1.clicked.connect(self.f_1)
        self.pb_mask2.clicked.connect(self.f_2)
        self.pb_mask3.clicked.connect(self.f_3)
        self.pb_mask4.clicked.connect(self.f_4)
        self.pb_foto.clicked.connect(self.new_post)
        self.mask = "cheeks_1"
        self.groupBox_QR.setVisible(0)
        self.gb_time.setVisible(0)
        self.seting()
        self.f_1()



/*************************************************************/
/*Description:Функция обновления настроек                    */
/*Unit has been developed with Python 3                      */
/*Creation Date: 10/11/2019                                  */
/*Author: IT'men                                             */
/*Version 1.0                                                */
/*************************************************************/

def seting(self):
        con = pymysql.connect('164.132.63.35', 'foto',
```

```
            'zxcasdqwe123', 'foto')
        with con:
            cur = con.cursor()
            cur.execute("SELECT name, coun FROM seting")
            rows = cur.fetchall()
            m=[0,0,0,0]
            c=[0,0,0,0]
            i=0
            for row in rows:
                m[i] = row[0]
                c[i] = row[1]
                i=i+1
        self.mask1=m[0]
        self.num1=c[0]
        self.pb_mask1.setIcon(QtGui.QIcon("resources/" +
str(m[0]) + ".png"))
        self.mask2=m[1]
        self.num2=c[1]
        self.pb_mask2.setIcon(QtGui.QIcon("resources/" +
str(m[1]) + ".png"))
        self.mask3=m[2]
        self.num3=c[2]
        self.pb_mask3.setIcon(QtGui.QIcon("resources/" +
str(m[2]) + ".png"))
        self.mask4=m[3]
        self.num4=c[3]
        self.pb_mask4.setIcon(QtGui.QIcon("resources/" +
str(m[3]) + ".png"))


/************************************************************/
/*Description:Генерация QR                                  */
/*Unit has been developed with Python 3                     */
/*Creation Date: 10/11/2019                                 */
/*Author: IT'men                                            */
/*Version 1.0                                               */
/************************************************************/

    def qr_save(self, key):
        mg =
qrcode.make('http://zvonok666.ru/foto/download?cod=' + key)
        mg.save("QR" + key + ".png")

/************************************************************/
/*Description:Сохраненеи фотографий                         */
/*Unit has been developed with Python 3                     */
/*Creation Date: 10/11/2019                                 */
/*Author: IT'men                                            */
/*Version 1.0                                               */
/************************************************************/

    def server_sql(self, name, key):
```

```
        try:
            con = pymysql.connect('164.132.63.35',
'foto','zxcasdqwe123', 'foto')
            with con:
                cur = con.cursor()
                sql = "INSERT INTO `foto`.`foto` (`id`, `cod`,
`name`, `creat`) VALUES (NULL, %s, %s, CURRENT_TIMESTAMP)"
                cur.execute(sql, (key, name))
        except Exception as e:
            print("error connect mysql")

    def server_ftp(self, filename):
        host = "164.132.63.35"
        ftp_user = "foto"
        ftp_password = "zxcasdqwe123"
        try:
            con = ftplib.FTP(host, ftp_user, ftp_password)
            f = open(filename, "rb")
            con.storbinary("STOR " + filename, f)
            con.close
        except Exception as e:
            print("error connect ftp")


/*************************************************************/
/*Description:Алгоритм                                       */
/*Unit has been developed with Python 3                      */
/*Creation Date: 10/11/2019                                  */
/*Author: IT'men                                             */
/*Version 1.0                                                */
/*************************************************************/

    def image_in_image(self, angle, s_img, l_img, size,
x_offset, y_offset):
        s_img = imutils.resize(s_img, width=size)
        s_h, s_w=s_img.shape[:2]
        y_step=0
        if angle > -180:
            y_step=0.55*math.sin((math.pi/180)*angle)*s_w
        elif angle>-361:
            y_step=-0.55*math.sin((math.pi/180)*angle)*s_w
        y1, y2 = int(y_step+ y_offset), int(y_step+y_offset +
s_img.shape[0])
        x1, x2 = int(x_offset), int(x_offset + s_img.shape[1])
        start_p_x = 0
        start_p_y = 0
        alpha_s = s_img[:, :, 3] / 255.0
        alpha_l = 1.0 - alpha_s
        if x1>0 and y1>0  and x2<639 and y2<479:
            for c in range(0, 3):
                l_img[y1:y2, x1:x2, c] = (alpha_s * s_img[:, :,
c] + alpha_l* l_img[y1:y2, x1:x2, c])
```

```
        return l_img

    def eyes_pose(self, eye_name, shape_add):
        (Start, End) =
face_utils.FACIAL_LANDMARKS_IDXS[eye_name]
        EyePts = shape_add[Start:End]
        return EyePts.mean(axis=0).astype("int")

    def accesuars(self, shape, l_img, det):
        s_img =
cv2.imread("resources/{0}.png".format(self.number), -1)
        shape_add=face_utils.shape_to_np(shape)
        rightEyeCenter = self.eyes_pose("right_eye", shape_add)
        leftEyeCenter = self.eyes_pose("left_eye", shape_add)
        angle = np.degrees(np.arctan2(rightEyeCenter[1] - left-
EyeCenter[1], rightEyeCenter[0] - leftEyeCenter[0])) - 180
        s_img = imutils.rotate_bound(s_img, angle)
        if self.counter == 1:
            sgW = int(math.sqrt((shape.part(16).x -
shape.part(0).x)* (shape.part(16).x - shape.part(0).x) +
(shape.part(16).y - shape.part(0).y)* (shape.part(16).y -
shape.part(0).y)))
            return self.image_in_image(angle, s_img, l_img, sgW,
int( shape.part(27).x - sgW/2),
(shape.part(21).y+shape.part(22).y)/2)
        elif  self.counter == 2:
            sgW = int(math.sqrt((shape.part(15).x -
shape.part(1).x)* (shape.part(15).x - shape.part(1).x) +
(shape.part(15).y - shape.part(1).y)* (shape.part(15).y -
shape.part(1).y))*0.9)
            return self.image_in_image(angle, s_img, l_img, sgW,
int(shape.part(30).x - sgW/2), int(shape.part(29).y +
2*shape.part(28).y)/3)
        elif  self.counter == 3:
            sgW = int(math.sqrt((shape.part(16).x -
shape.part(0).x)* (shape.part(16).x - shape.part(0).x) +
(shape.part(16).y - shape.part(0).y)* (shape.part(16).y -
shape.part(0).y)))
            return self.image_in_image(angle, s_img, l_img, sgW,
int( shape.part(27).x - sgW/2),
(shape.part(21).y+shape.part(22).y)/2)
        else:
            return l_img

    def drawing_makeup (self, shape, p_num, min_1, max_1, min_2,
max_2):
        coords = np.zeros((p_num, 2), np.int32)
        step = max_1-min_1
        for i in range(min_1,max_1):
            coords[i-min_1]=(shape.part(i).x, shape.part(i).y)
        if p_num==12:
            coords[6]=(shape.part(60).x, shape.part(60).y)
```

```
            coords[7]=(shape.part(48).x, shape.part(48).y)
            step=8
        for i in range(max_2,min_2,-1):
            coords[max_2-i+step]=(shape.part(i).x,
shape.part(i).y)
        if p_num==13:
            coords[12]=(shape.part(48).x, shape.part(48).y)
        return coords

/*************************************************************/
/*Description:Накладывание маски и упровление программы      */
/*Unit has been developed with Python 3                       */
/*Creation Date: 10/11/2019                                   */
/*Author: IT'men                                              */
/*Version 1.0                                                 */
/*************************************************************/


    def gen_key(self, size):
        return ''.join(random.choice(string.ascii_letters) for _
in range(size))

    def new_post(self):
        self.insta_post = True
        self.t_c = 50

    def camera(self):
        t1=-1
        t2=0
        detector = dlib.get_frontal_face_detector()
        predic-
tor=dlib.shape_predictor("shape_predictor_68_face_landmarks.dat"
)
        cap = cv2.VideoCapture(0)
        while(True):
            if self.insta_post:
                if (self.t_c < 0):
                    self.gb_time.setVisible(0)
                    time = True
                    if time:
                        name = "123"
                        self.insta_post = False
                        dt = datetime.datetime.now()
                        name = self.gen_key(6) + '.jpg'
                        cv2.imwrite(name,l_img)
                        key = self.gen_key(5)
                        self.server_sql(name, key)
                        self.server_ftp(name)
                        self.qr_save(key)
                        self.qr.setPixmap(QtGui.QPixmap("QR" +
key + ".png"))
                        self.Foto.setPixmap(QtGui.QPixmap(name))
```

```
self.label_COD.setText('<html><head/><body><p><span style=" col-
or:#ff0000;">' + key + '</span></p></body></html>')
                        self.groupBox_QR.setVisible(1)
                        t1=1
                else:
                    self.gb_time.setVisible(1)
                    if(self.t_c == 50 or self.t_c == 40 or
self.t_c == 30 or self.t_c == 20 or self.t_c == 10 or self.t_c
== 0):

self.label_time.setText('<html><head/><body><p
align="center"><span style=" color:#ffffff;">' +
str(int(self.t_c/10)) + '</span></p></body></html>')
                    self.t_c = self.t_c - 1
            if (t2 > 350):
                t2=0
            if (t1 > 0):
                t1 = t1 + 1
                if (t1 > 250):
                    self.groupBox_QR.setVisible(0)
            t2 = t2 + 1
            ret, l_img = cap.read()
            gray = cv2.cvtColor(l_img, cv2.COLOR_BGR2GRAY)
            det_rect=detector(gray)
            for det in det_rect:
                shape = predictor(gray, det)
                l_img=self.accesuars(shape, l_img, det)

            image = QtGui.QImage(l_img, l_img.shape[1],
l_img.shape[0], l_img.shape[1] * l_img.shape[2],
QtGui.QImage.Format_RGB888)
            pixmap = QtGui.QPixmap()
            pixmap.convertFromImage(image.rgbSwapped())
            pixmap = pixmap.scaled(self.label.width(),
self.label.height(), QtCore.Qt.KeepAspectRatio)
            self.label.setPixmap(pixmap)

/*************************************************************/
/*Description:gui.py                                         */
/*Unit has been developed with Python 3                      */
/*Creation Date: 10/11/2019                                  */
/*Author: IT'men                                             */
/*Version 1.0                                                */
/*************************************************************/



from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
```

```
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1221, 965)
        MainWindow.setWindowOpacity(1.0)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(0, 0, 1221, 951))
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePoli-
cy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth()
)
        self.label.setSizePolicy(sizePolicy)
        self.label.setText("")

self.label.setPixmap(QtGui.QPixmap("gui/foto/1571878396.0.jpg"))
        self.label.setScaledContents(True)
        self.label.setAlignment(QtCore.Qt.AlignCenter)
        self.label.setWordWrap(False)
        self.label.setObjectName("label")
        self.groupBox = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox.setGeometry(QtCore.QRect(980, 0, 241,
951))
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePoli-
cy.setHeightForWidth(self.groupBox.sizePolicy().hasHeightForWidt
h())
        self.groupBox.setSizePolicy(sizePolicy)
        self.groupBox.setStyleSheet("background-color: rgba(0,
0, 0, .5);\n"
"")
        self.groupBox.setTitle("")
        self.groupBox.setObjectName("groupBox")
        self.pb_foto = QtWidgets.QPushButton(self.groupBox)
        self.pb_foto.setEnabled(True)
        self.pb_foto.setGeometry(QtCore.QRect(30, 740, 201,
151))
        font = QtGui.QFont()
        font.setPointSize(20)
        font.setBold(True)
        font.setWeight(75)
        self.pb_foto.setFont(font)
        self.pb_foto.setStyleSheet("background-color: rgba(14,
85, 250, .5);")
```

```
        self.pb_foto.setObjectName("pb_foto")
        self.verticalLayoutWidget =
QtWidgets.QWidget(self.groupBox)
        self.verticalLayoutWidget.setGeometry(QtCore.QRect(30,
20, 201, 671))

self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")
        self.verticalLayout =
QtWidgets.QVBoxLayout(self.verticalLayoutWidget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout.setObjectName("verticalLayout")
        self.pb_mask1 =
QtWidgets.QPushButton(self.verticalLayoutWidget)
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePoli-
cy.setHeightForWidth(self.pb_mask1.sizePolicy().hasHeightForWidt
h())
        self.pb_mask1.setSizePolicy(sizePolicy)
        self.pb_mask1.setStyleSheet("background-color:
rgba(96,137,146, .5);")
        self.pb_mask1.setText("")
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("gui/resources/1.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pb_mask1.setIcon(icon)
        self.pb_mask1.setIconSize(QtCore.QSize(100, 100))
        self.pb_mask1.setObjectName("pb_mask1")
        self.verticalLayout.addWidget(self.pb_mask1)
        self.pb_mask2 =
QtWidgets.QPushButton(self.verticalLayoutWidget)
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePoli-
cy.setHeightForWidth(self.pb_mask2.sizePolicy().hasHeightForWidt
h())
        self.pb_mask2.setSizePolicy(sizePolicy)
        self.pb_mask2.setStyleSheet("background-color:
rgba(96,137,146, .5);")
        self.pb_mask2.setText("")
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap("gui/resources/2.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pb_mask2.setIcon(icon1)
        self.pb_mask2.setIconSize(QtCore.QSize(100, 100))
        self.pb_mask2.setObjectName("pb_mask2")
```

```
        self.verticalLayout.addWidget(self.pb_mask2)
        self.pb_mask3 =
QtWidgets.QPushButton(self.verticalLayoutWidget)
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePoli-
cy.setHeightForWidth(self.pb_mask3.sizePolicy().hasHeightForWidt
h())
        self.pb_mask3.setSizePolicy(sizePolicy)
        self.pb_mask3.setStyleSheet("background-color:
rgba(96,137,146, .5);")
        self.pb_mask3.setText("")
        icon2 = QtGui.QIcon()
        icon2.addPixmap(QtGui.QPixmap("gui/resources/3.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pb_mask3.setIcon(icon2)
        self.pb_mask3.setIconSize(QtCore.QSize(100, 100))
        self.pb_mask3.setObjectName("pb_mask3")
        self.verticalLayout.addWidget(self.pb_mask3)
        self.pb_mask4 =
QtWidgets.QPushButton(self.verticalLayoutWidget)
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePoli-
cy.setHeightForWidth(self.pb_mask4.sizePolicy().hasHeightForWidt
h())
        self.pb_mask4.setSizePolicy(sizePolicy)
        self.pb_mask4.setStyleSheet("background-color:
rgba(96,137,146, .5);")
        self.pb_mask4.setText("")
        icon3 = QtGui.QIcon()

icon3.addPixmap(QtGui.QPixmap("mirrow/glasses/glasses_5.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pb_mask4.setIcon(icon3)
        self.pb_mask4.setIconSize(QtCore.QSize(100, 100))
        self.pb_mask4.setObjectName("pb_mask4")
        self.verticalLayout.addWidget(self.pb_mask4)
        self.groupBox_QR =
QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_QR.setEnabled(True)
        self.groupBox_QR.setGeometry(QtCore.QRect(190, 130, 601,
681))
        self.groupBox_QR.setAcceptDrops(False)
        self.groupBox_QR.setStyleSheet("background-color:
rgba(0, 0, 0, .5);\n"
```

```
"")
        self.groupBox_QR.setTitle("")
        self.groupBox_QR.setObjectName("groupBox_QR")
        self.label_2 = QtWidgets.QLabel(self.groupBox_QR)
        self.label_2.setGeometry(QtCore.QRect(50, 10, 501, 51))
        font = QtGui.QFont()
        font.setFamily("DejaVu Sans")
        font.setPointSize(36)
        font.setBold(True)
        font.setWeight(75)
        self.label_2.setFont(font)
        self.label_2.setStyleSheet("background-color: rgba(0, 0,
0, .0);\n"
"")
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setObjectName("label_2")
        self.Foto = QtWidgets.QLabel(self.groupBox_QR)
        self.Foto.setGeometry(QtCore.QRect(110, 70, 381, 261))
        self.Foto.setStyleSheet("background-color: rgba(0, 0, 0,
.9);\n"
"")
        self.Foto.setText("")
        self.Foto.setPixmap(QtGui.QPixmap("gui/mirrow/1.jpg"))
        self.Foto.setScaledContents(True)
        self.Foto.setObjectName("Foto")
        self.label_COD = QtWidgets.QLabel(self.groupBox_QR)
        self.label_COD.setGeometry(QtCore.QRect(50, 620, 501,
31))
        font = QtGui.QFont()
        font.setFamily("DejaVu Sans")
        font.setPointSize(16)
        font.setBold(True)
        font.setWeight(75)
        font.setKerning(False)
        self.label_COD.setFont(font)
        self.label_COD.setAutoFillBackground(False)
        self.label_COD.setStyleSheet("background-color: rgba(0,
0, 0, .0);")
        self.label_COD.setAlignment(QtCore.Qt.AlignCenter)
        self.label_COD.setObjectName("label_COD")
        self.label_4 = QtWidgets.QLabel(self.groupBox_QR)
        self.label_4.setGeometry(QtCore.QRect(50, 530, 501, 81))
        font = QtGui.QFont()
        font.setFamily("DejaVu Sans")
        font.setPointSize(16)
        font.setBold(True)
        font.setWeight(75)
        font.setKerning(False)
        self.label_4.setFont(font)
        self.label_4.setStyleSheet("background-color: rgba(0, 0,
0, .0);\n"
"")
```

```
        self.label_4.setAlignment(QtCore.Qt.AlignCenter)
        self.label_4.setObjectName("label_4")
        self.label_6 = QtWidgets.QLabel(self.groupBox_QR)
        self.label_6.setGeometry(QtCore.QRect(50, 330, 501, 41))
        font = QtGui.QFont()
        font.setFamily("DejaVu Sans")
        font.setPointSize(16)
        font.setBold(True)
        font.setWeight(75)
        self.label_6.setFont(font)
        self.label_6.setStyleSheet("background-color: rgba(0, 0,
0, .0);\n"
"")
        self.label_6.setAlignment(QtCore.Qt.AlignCenter)
        self.label_6.setObjectName("label_6")
        self.qr = QtWidgets.QLabel(self.groupBox_QR)
        self.qr.setGeometry(QtCore.QRect(220, 380, 161, 141))
        font = QtGui.QFont()
        font.setFamily("Ubuntu")
        self.qr.setFont(font)
        self.qr.setStyleSheet("background-color: rgba(0, 0, 0,
.9);")
        self.qr.setText("")
        self.qr.setPixmap(QtGui.QPixmap("Загрузки/qr-code.gif"))
        self.qr.setScaledContents(True)
        self.qr.setObjectName("qr")
        self.gb_time = QtWidgets.QGroupBox(self.centralwidget)
        self.gb_time.setGeometry(QtCore.QRect(400, 400, 141,
121))
        self.gb_time.setStyleSheet("background-color: rgba(0, 0,
0, .5);\n"
"")
        self.gb_time.setTitle("")
        self.gb_time.setObjectName("gb_time")
        self.label_time = QtWidgets.QLabel(self.gb_time)
        self.label_time.setGeometry(QtCore.QRect(10, 0, 131,
111))
        font = QtGui.QFont()
        font.setPointSize(72)
        font.setBold(True)
        font.setWeight(75)
        self.label_time.setFont(font)
        self.label_time.setStyleSheet("background-color: rgba(0,
0, 0, .0);\n"
"")
        self.label_time.setObjectName("label_time")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(0, 0, 111, 101))
        self.label_3.setText("")
        self.label_3.setPixmap(QtGui.QPixmap("2.png"))
        self.label_3.setObjectName("label_3")
        MainWindow.setCentralWidget(self.centralwidget)
```

```
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1221, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow",
"MainWindow"))
        self.pb_foto.setText(_translate("MainWindow", "FOTO"))
        self.label_2.setText(_translate("MainWindow",
"<html><head/><body><p><span style=\" color:#ffffff;\">ВАШЕ
ФОТО</span></p></body></html>"))
        self.label_COD.setText(_translate("MainWindow",
"<html><head/><body><p><span style=\" col-
or:#ff0000;\">XXXXXXXXX</span></p></body></html>"))
        self.label_4.setText(_translate("MainWindow",
"<html><head/><body><p><span style=\" color:#ffffff;\">Вы также
можете скачать фото<br/>перейдя на web-сайт zvo-
nok666.ru<br/>указав уникальный ключ
(ниже)<br/></span></p></body></html>"))
        self.label_6.setText(_translate("MainWindow",
"<html><head/><body><p><span style=\" color:#ffffff;\">Скачайте
сейчас, отсканировав QR код:</span></p></body></html>"))
        self.label_time.setText(_translate("MainWindow",
"<html><head/><body><p align=\"center\"><span style=\" col-
or:#ffffff;\">5</span></p></body></html>"))
```

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

УТВЕРЖДАЮ	СОГЛАСОВАНО
Декан ФКТ	Заведующий кафедрой МОП ЭВМ
_____ Я.Ю. Гриорьев	_____В.А. Тихомиров
(подпись)	*(подпись)*
«____ » _____ 20___ г.	«____ » _____ 20___ г.

АКТ

о приемке в эксплуатацию программы для ЭВМ

«ИС ВС»

г. Комсомольск-на-Амуре « » _____ 20__ г.

Комиссия в составе представителей:

заказчика

- Е.П. Жарикова – руководитель проекта,
- В.А. Тихомиров – Заведующий кафедрой МОП ЭВМ,
- Я.Ю. Григорьев – декана ФКТ

исполнители

- А.Д. Палыгин – группа,
- Е.П. Душкин – 9ИБ-1
- А.А. Макуха – 9ИБ-1
- Д.А. Курбонназаров – 7БМб-1

составили акт о нижеследующем:

«Исполнитель» передает программный комплекс «Веб-сервис по наложению тематического изображения на изображение человеческого лица», в составе:

Программное обеспечение, в том числе:

- Рбочую программу

Эксплуатационная документация:

- Комплект проектной документации

Информационная Система «Веб-сервис по наложению тематического изображения на изображение человеческого лица» прошла опытную эксплуатацию с «____ » _____ по «____ » _____ 20____г. и признана годным к эксплуатации. Были протестированы все режимы функционирования, отказы системы не наблюдались.

Руководитель проекта

_____/ Е.П. Жарикова /

Ответственный исполнитель

_____ /А.Д. Палыгин /

Таблица учета проектной работы в учебных дисциплинах

Дисциплина	Форма учтенной работы (номер ЛР, КП, КР, РГЗ, зачет, итоговая оценка , экзамен)	Преподаватель (дата, ФИО, подпись)	Примечание (ЗУН полученные при выполнении проекта)