**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»**

**Журнал практики**

Тамбова Никиты Витальевича

Студента(-ки) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(ф.и.о.)

##### Институт № 1 «Авиационная техника»

##### Кафедра 107Б «Внешнее проектирование и эффективность авиационных комплексов»

##### Учебная группа \_\_\_\_\_\_\_\_\_\_\_\_

М1О-309С-19

27.05.01

Направление подготовки (специальность) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальные организационно-технические системы

Направление: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Внешнее проектирование и эффективность авиационных комплексов

Специализация: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики

Андрющенко Сергей Владимирович \_\_\_\_\_\_\_\_\_\_\_\_\_

Дата сдачи отчета

25.07.2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись студента)* *(дата)*

Москва, 2022

**1.Место и сроки проведения практики**

*Сроки проведения практики:*

29.06.2022

*-дата начала практики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

26.07.2022

*-дата окончания практики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Наименование предприятия: ФАУ «ГосНИИАС»*

1. **Индивидуальное задание студенту**

Тема: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Разработка симулятора диспетчера аэропорта с целью оценки и прогнозирования эффективности работника.

**3.План выполнения индивидуального задания**

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Наименование этапов выполнения работы** | **Срок выполнения этапов работы** |
| 1. | Утверждение индивидуального задания | 29.06.2022 |
| 2. | Разработка и согласование с руководителем плана выполнения индивидуального задания | 30.06.2022 |
| 3. | Выполнение задания по практике |  |
| 3.1. | Изучение предметной области | 01.07.2022 – 03.07.2022 |
| 3.2. | Выделение системных компонентов | 04.07.2022 – 06.07.2022 |
| 3.3. | Разработка программного комплекса | 07.07.2022 – 14.07.2022 |
| 3.4. | Анализ полученных результатов | 15.07.2022 – 16.07.2022 |
| 3.5. | Подготовка иллюстрационных материалов | 17.07.2022 – 20.07.2022 |
| 4. | Формирование выводов по исследованной проблематике | 21.07.2022 – 22.07.2022 |
| 5. | Представление журнала практики руководителю. Внесения корректировок | 23.07.2022 |
| 6. | Оформление отчета студента о практике и отзыва руководителя | 24.07.2022 |
| 7. | Представление журнала практики на кафедру | 25.07.2022 |

*Руководитель практики*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. В. Андрющенко

25 июля 2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ « \_\_ » \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ г.

*(подпись студента)* *(дата)*

**4.Отзыв руководителя практики**

Студент Тамбов Н.В. выполнил индивидуальное задание на тему: «Разработка симулятора диспетчера аэропорта с целью оценки и прогнозирования эффективности работника» в полном объеме. В период работы над индивидуальным проектом студент Тамбов Н.В. проявил самостоятельность, показал умение работать с литературой и способность решать сложные технические задачи.

К недостаткам разработки можно отнести малое количество доступных аэропортов и диспетчеров, то есть недостаточная гибкость симулятора.

Среди достоинств разработки важно отметить интуитивно понятный дизайн и структуру программного комплекса, доступность обучения. Сама реализация в полной мере отражает поставленную задачу и не имеет дефектов в работе.

В целом, индивидуальное задание Тамбова Н.В. выполнено на высоком уровне, отвечает всем предъявленным требованиям, а сам студент Тамбов Н.В. заслуживает отличной оценки.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Руководитель практики*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. В. Андрющенко

*(подпись)* *(фамилия, инициалы)*

25 июля 2022

«\_\_» \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ г.

Приложение 1 к журналу практики

**Рабочий график проведения практики**

29 июня 22 26 июля 22

в ФАУ «ГосНИИАС» в период с «\_\_\_\_»\_\_\_\_\_\_\_\_\_ 20\_\_ года по «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ года

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ п/п** | **Место проведения** | **Тема** | **Дата** | **Время выполнения (ч.)** |
| 1. | ГосНИИАС | Утверждение индивидуального задания | 29.06.2022 | 2 |
| 2. | ГосНИИАС | Разработка и согласование с руководителем плана выполнения индивидуального задания | 30.06.2022 | 4 |
| 3. | ГосНИИАС | Выполнение задания по практике |  |  |
| 3.1. | ГосНИИАС | Изучение предметной области | 01.07.2022 – 03.07.2022 | 20 |
| 3.2. | ГосНИИАС | Выделение системных компонентов | 04.07.2022 – 06.07.2022 | 15 |
| 3.3. | ГосНИИАС | Разработка программного комплекса | 07.07.2022 – 14.07.2022 | 100 |
| 3.4. | ГосНИИАС | Анализ полученных результатов | 15.07.2022 – 16.07.2022 | 6 |
| 3.5. | ГосНИИАС | Подготовка иллюстрационных материалов | 17.07.2022 – 20.07.2022 | 10 |
| 4. | ГосНИИАС | Формирование выводов по практике | 21.07.2022 – 22.07.2022 | 2 |
| 5. | ГосНИИАС | Представление журнала практики руководителю. Внесения корректировок | 23.07.2022 | 4 |
| 6. | ГосНИИАС | Оформление отчета студента о практике и отзыва руководителя | 24.07.2022 | 4 |
| 7. | ГосНИИАС | Представление журнала практики на кафедру | 25.07.2022 | 1 |

Руководитель практики

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. В. Андрющенко

25 июля 2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ « \_\_ » \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ г.

*(подпись студента)* *(дата)*

Приложение 2 к журналу практики

**Перечень материалов для отчетности**

По завершению практики, каждому из студентов необходимо сдать следующий перечень материалов:

* Отчет о проведенной практики (word);
* Презентация PowerPoint (pptx);
* ПО (исполняемые файлы и исходники);
* Прочее;

Руководитель практики

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. В. Андрющенко

25 июля 2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ « \_\_ » \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ г.

*(подпись студента)* *(дата)*

**МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

(национальный исследовательский университет)

**Отчёт по практике**

Н. В. Тамбов

М1О-309С-19

Студент группы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** /\_\_\_\_\_\_\_\_\_\_\_\_\_/ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Руководитель практики /\_\_\_\_\_\_\_\_\_\_\_\_\_/ С. В. Андрющенко

Москва, 2022 г.

Приложение 3 к журналу практики

**Исходный код программы**

Язык программирования: Python.

from PyQt5.QtWidgets import QApplication

import sys

from StartWindow import StartWindow

app = QApplication(sys.argv)

win = StartWindow(app)

win.show()

sys.exit(app.exec\_())

from PyQt5 import QtGui

from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QGridLayout, QWidget, QHBoxLayout, \

QVBoxLayout, QComboBox

from PyQt5.QtCore import Qt, QObject

from StartWindowWidgets import UserWidget, AirportWidget, NavigationBtn

from MainWindow import MainWindow

from DataBaseMethods import count\_all\_users, count\_all\_airports, getUserInfo, getAirportInfo

class MainWidget(QWidget):

def \_\_init\_\_(self, parent=None):

QWidget.\_\_init\_\_(self, parent)

self.parent = parent

self.main\_window\_create = False # переменная, проверяющая создан ли объект MainWindow

self.btns\_users = [] # карточки пользователей

self.btns\_airports = [] # карточки аэропортов

self.count\_users = count\_all\_users() # кол-во диспетчеров

self.count\_airports = count\_all\_airports() # кол-во аэропортов

self.current\_user = 0 # выбранная карточка пользователя

self.current\_airport = 0 # выбранная карточка аэропорта

# Расстановка элементов

VLayout\_main = QVBoxLayout()

VLayout\_main.setAlignment(Qt.AlignVCenter)

VLayout\_main.setSpacing(50) # расстояние между элементами

VLayout\_main.setContentsMargins(0, 0, 0, 0) # внешние отступы

HLayout\_users = QHBoxLayout()

HLayout\_users.setAlignment(Qt.AlignHCenter)

HLayout\_users.setSpacing(40) # расстояние между элементами

# Кол-во диспетчеров

VLayout\_count\_users = QVBoxLayout()

VLayout\_count\_users.setAlignment(Qt.AlignVCenter)

VLayout\_count\_users.setSpacing(15) # расстояние между элементами

VLayout\_count\_users.setContentsMargins(0, 0, 40, 0) # внешние отступы

self.lbl\_count\_users = QLabel(self) # заголовок

self.lbl\_count\_users.setFont(QtGui.QFont('Helvetica', 32, weight=QtGui.QFont.Bold)) # изменяем шрифт

self.lbl\_count\_users.setFixedWidth(210)

self.lbl\_count\_users.setFixedHeight(78)

self.lbl\_count\_users.setWordWrap(True) # перенос текста

self.lbl\_count\_users.setText("Количество диспетчеров") # меняем текст

self.lbl\_count\_users.setStyleSheet("color: black;") # меняем цвет текста

self.lbl\_count\_users.setAlignment(Qt.AlignCenter)

VLayout\_count\_users.addWidget(self.lbl\_count\_users)

HLayout\_count\_users = QHBoxLayout()

HLayout\_count\_users.setAlignment(Qt.AlignHCenter)

HLayout\_count\_users.setContentsMargins(0, 0, 0, 0) # внешние отступы

self.combobox\_count\_users = QComboBox(self)

self.combobox\_count\_users.setStyleSheet("""

#comboBoxCountUsers {background-color: white; color: black; border: 0px;}

#comboBoxCountUsers QListView {background-color: white; color: black; border: 0px; outline: 0px;}

""")

self.combobox\_count\_users.setObjectName("comboBoxCountUsers")

self.combobox\_count\_users.setFont(QtGui.QFont('Helvetica', 20)) # изменяем шрифт

self.combobox\_count\_users.setFixedWidth(80)

self.combobox\_count\_users.setFixedHeight(40)

for i in range(self.count\_users):

self.combobox\_count\_users.addItem(str(i+1))

self.combobox\_count\_users.setCurrentIndex(self.count\_users-1) # устанавливаем значение по умолчанию

self.combobox\_count\_users.activated[str].connect(self.onComboBoxCountUsers) # при выборе кол-ва диспетчеров

HLayout\_count\_users.addWidget(self.combobox\_count\_users)

VLayout\_count\_users.addLayout(HLayout\_count\_users)

HLayout\_users.addLayout(VLayout\_count\_users)

# Карточки пользователей

for i in range(self.count\_users):

self.btns\_users.append(UserWidget(self,

user\_id=i,

user\_name=getUserInfo(i, "name"),

user\_img=getUserInfo(i, "img")))

self.btns\_users[-1].clicked.connect(self.onBtnUserClick) # при клике на карточку пользователя

HLayout\_users.addWidget(self.btns\_users[-1])

self.updateBtnUser(0)

self.lbl\_user = QLabel(self) # заголовок

self.lbl\_user.setFont(QtGui.QFont('Helvetica', 32, weight=QtGui.QFont.Bold)) # изменяем шрифт

self.lbl\_user.setFixedWidth(237)

self.lbl\_user.setFixedHeight(78)

self.lbl\_user.setWordWrap(True) # перенос текста

self.lbl\_user.setText("Выберите пользователя") # меняем текст

self.lbl\_user.setStyleSheet("color: black;") # меняем цвет текста

HLayout\_users.addWidget(self.lbl\_user)

VLayout\_main.addLayout(HLayout\_users)

self.lbl\_line = QLabel(self) # Заголовок

self.lbl\_line.setFixedWidth(QApplication.desktop().screenGeometry().width())

self.lbl\_line.setFixedHeight(1)

self.lbl\_line.setContentsMargins(0, 0, 0, 0) # внешние отступы

self.lbl\_line.setStyleSheet("background-color: black;") # меняем цвет текста

VLayout\_main.addWidget(self.lbl\_line)

HLayout\_airports = QHBoxLayout()

HLayout\_airports.setAlignment(Qt.AlignHCenter)

HLayout\_airports.setSpacing(80) # расстояние между элементами

VLayout\_title\_btns = QVBoxLayout()

VLayout\_title\_btns.setAlignment(Qt.AlignTop)

VLayout\_title\_btns.setContentsMargins(0, 40, 0, 0) # внешние отступы

VLayout\_title\_btns.setSpacing(25) # расстояние между элементами

self.lbl\_airport = QLabel(self) # заголовок

self.lbl\_airport.setFont(QtGui.QFont('Helvetica', 40, weight=QtGui.QFont.Bold)) # изменяем шрифт

self.lbl\_airport.setFixedWidth(203)

self.lbl\_airport.setFixedHeight(150)

self.lbl\_airport.setAlignment(Qt.AlignTop)

self.lbl\_airport.setWordWrap(True) # перенос текста

self.lbl\_airport.setText("Выберите аэропорт") # меняем текст

self.lbl\_airport.setStyleSheet("color: black;") # меняем цвет текста

VLayout\_title\_btns.addWidget(self.lbl\_airport)

# Кнопки "Далее" и "Выход"

self.btn\_further = NavigationBtn(self, name="further") # кнопка "Далее"

self.btn\_further.clicked.connect(self.btnFurtherClick) # при клике на кнопку

VLayout\_title\_btns.addWidget(self.btn\_further)

self.btn\_exit = NavigationBtn(self, name="exit") # кнопка "Выход"

self.btn\_exit.clicked.connect(parent.close) # при клике на кнопку

VLayout\_title\_btns.addWidget(self.btn\_exit)

HLayout\_airports.addLayout(VLayout\_title\_btns)

# Карточки аэропортов

GridLayout\_airports = QGridLayout()

GridLayout\_airports.setHorizontalSpacing(25) # расстояние между столбцами

GridLayout\_airports.setVerticalSpacing(25) # расстояние между строками

for i in range(self.count\_airports):

self.btns\_airports.append(AirportWidget(self,

airport\_id=i,

airport\_name=getAirportInfo(i, "name"),

count\_runways=getAirportInfo(i, "count\_runways")))

self.btns\_airports[-1].clicked.connect(self.onBtnAirportClick) # при клике на карточку аэропорта

GridLayout\_airports.addWidget(self.btns\_airports[-1], i//2, i%2)

self.updateBtnAirport(0)

HLayout\_airports.addLayout(GridLayout\_airports)

VLayout\_main.addLayout(HLayout\_airports)

self.setLayout(VLayout\_main)

# Отображение определённого кол-ва карточек пользователей

def onComboBoxCountUsers(self, text):

self.count\_users = int(text)

self.current\_user = 0

self.updateBtnUser(0)

# Убираем выделение у остальных пользовательских карточек

for i in range(1, len(self.btns\_users)):

self.btns\_users[i].setStyleSheet("""

QPushButton {background:rgb(135, 136, 160); border-radius: 20px;}

QPushButton:hover {background:rgb(70, 70, 116); border-radius: 20px;}

""")

# Скрываем фото и имена всех пользователей

for btn\_user in self.btns\_users:

btn\_user.user\_img.hide()

btn\_user.lbl\_name\_user.hide()

# Показываем фото и имена только нужных пользователей

for i in range(int(text)):

self.btns\_users[i].user\_img.show()

self.btns\_users[i].lbl\_name\_user.show()

# Изменение стилей у карточек пользователей

def updateBtnUser(self, btn\_id):

# Выделяем карточку пользователя

if self.count\_users >= btn\_id + 1:

self.btns\_users[btn\_id].setStyleSheet("QPushButton {background:rgb(39, 39, 61); border-radius: 20px;}")

if self.current\_user != btn\_id:

# Убираем выделение предыдущей карточки пользователя

if self.count\_users >= btn\_id+1:

self.btns\_users[self.current\_user].setStyleSheet("""

QPushButton {background:rgb(135, 136, 160); border-radius: 20px;}

QPushButton:hover {background:rgb(70, 70, 116); border-radius: 20px;}

""")

self.current\_user = btn\_id # присваиваем новую карточку пользователя

# Выбор карточки пользователя

def onBtnUserClick(self):

btn\_id = int(self.sender().objectName())

self.updateBtnUser(btn\_id)

# Изменение стилей у карточек аэропортов

def updateBtnAirport(self, btn\_id):

# Выделяем карточку аэропорта

self.btns\_airports[btn\_id].setStyleSheet("QPushButton {background:rgb(39, 39, 61); border: 0;}")

if self.current\_airport != btn\_id:

# Убираем выделение предыдущей карточки аэропорта

self.btns\_airports[self.current\_airport].setStyleSheet("""

QPushButton {background:rgb(135, 136, 160); border: 0;}

QPushButton:hover {background:rgb(70, 70, 116); border: 0;}

""")

self.current\_airport = btn\_id # присваиваем новую карточку аэропорта

# Выбор карточки аэропорта

def onBtnAirportClick(self):

btn\_id = int(self.sender().objectName())

self.updateBtnAirport(btn\_id)

# Следующий экран

def btnFurtherClick(self):

self.main\_window = MainWindow(self.count\_users, self.current\_user, self.current\_airport, self)

self.main\_window\_create = True

self.parent.setCentralWidget(self.main\_window) # меняем экран

def \_\_del\_\_(self):

if self.main\_window\_create == True:

del self.main\_window

class StartWindow(QMainWindow):

def \_\_init\_\_(self, app):

super(StartWindow, self).\_\_init\_\_()

self.setGeometry(0, 0, QApplication.desktop().screenGeometry().width(), QApplication.desktop().screenGeometry().height())

self.setWindowTitle('Симулятор диспетчера аэропорта') # название окна

self.setWindowIcon(QtGui.QIcon('imgs/main.ico')) # иконка окна

self.setStyleSheet("background:rgb(172, 177, 202);") # фон окна

self.main\_widget = MainWidget(self)

self.setCentralWidget(self.main\_widget) # устанавливаем главный виджет

# Завершаем отдельный поток, если окно хотят закрыть

def closeEvent(self, event):

del self.main\_widget

from PyQt5 import QtGui

from PyQt5.QtWidgets import QLabel, QPushButton, QVBoxLayout

from PyQt5.QtCore import Qt

class UserWidget(QPushButton):

def \_\_init\_\_(self, parent, user\_id, user\_name, user\_img):

QPushButton.\_\_init\_\_(self, parent)

self.setObjectName(str(user\_id))

self.setFixedWidth(125)

self.setFixedHeight(152)

self.setStyleSheet("""

QPushButton {background:rgb(135, 136, 160); border-radius: 20px;}

QPushButton:hover {background:rgb(70, 70, 116); border-radius: 20px;}

""")

VLayout\_user = QVBoxLayout()

VLayout\_user.setContentsMargins(0, 0, 0, 0) # внешние отступы

VLayout\_user.setAlignment(Qt.AlignVCenter | Qt.AlignHCenter)

self.user\_img = QLabel(self)

self.user\_img.setStyleSheet("background:rgb(217, 217, 217); border-radius: 10px;")

self.user\_img.setFixedWidth(100)

self.user\_img.setFixedHeight(100)

self.pixmap = QtGui.QPixmap(user\_img)

self.user\_img.setPixmap(self.pixmap)

self.user\_img.setAlignment(Qt.AlignVCenter | Qt.AlignHCenter) # выравнивание (центр)

VLayout\_user.addWidget(self.user\_img)

self.lbl\_name\_user = QLabel(self)

self.lbl\_name\_user.setFont(QtGui.QFont('Helvetica', 18, weight=QtGui.QFont.Bold))

self.lbl\_name\_user.setFixedWidth(100)

self.lbl\_name\_user.setFixedHeight(24)

self.lbl\_name\_user.setContentsMargins(0, 5, 0, 0) # внешние отступы

self.lbl\_name\_user.setAlignment(Qt.AlignCenter)

self.lbl\_name\_user.setText(user\_name)

self.lbl\_name\_user.setStyleSheet("background: 0; color: white;")

VLayout\_user.addWidget(self.lbl\_name\_user)

self.setLayout(VLayout\_user)

class AirportWidget(QPushButton):

def \_\_init\_\_(self, parent, airport\_id, airport\_name, count\_runways):

QPushButton.\_\_init\_\_(self, parent)

self.setObjectName(str(airport\_id))

self.setFixedWidth(350)

self.setFixedHeight(163)

self.setStyleSheet("""

QPushButton {background:rgb(135, 136, 160); border: 0;}

QPushButton:hover {background:rgb(70, 70, 116); border: 0;}

""")

VLayout\_airport = QVBoxLayout()

VLayout\_airport.setContentsMargins(0, 0, 0, 0) # внешние отступы

VLayout\_airport.setAlignment(Qt.AlignVCenter | Qt.AlignHCenter)

self.airport\_name = QLabel(self)

self.airport\_name.setStyleSheet("background:rgb(217, 217, 217);")

self.airport\_name.setFixedWidth(300)

self.airport\_name.setFixedHeight(85)

self.airport\_name.setAlignment(Qt.AlignVCenter | Qt.AlignHCenter) # выравнивание (центр)

VLayout\_lbl = QVBoxLayout()

self.lbl\_name = QLabel(self)

self.lbl\_name.setFont(QtGui.QFont('Helvetica', 40))

self.lbl\_name.setFixedWidth(203)

self.lbl\_name.setFixedHeight(42)

self.lbl\_name.setText(airport\_name)

self.lbl\_name.setAlignment(Qt.AlignCenter)

self.lbl\_name.setStyleSheet("background: 0; color: black;")

VLayout\_lbl.addWidget(self.lbl\_name)

VLayout\_lbl.setAlignment(Qt.AlignVCenter | Qt.AlignHCenter) # выравнивание (центр)

VLayout\_lbl.setContentsMargins(0, 0, 0, 0) # внешние отступы

self.airport\_name.setLayout(VLayout\_lbl)

VLayout\_airport.addWidget(self.airport\_name)

self.lbl\_count\_runways = QLabel(self)

self.lbl\_count\_runways.setFont(QtGui.QFont('Helvetica', 24, weight=QtGui.QFont.Bold))

self.lbl\_count\_runways.setFixedWidth(300)

self.lbl\_count\_runways.setFixedHeight(28)

self.lbl\_count\_runways.setContentsMargins(0, 5, 0, 0) # внешние отступы

self.lbl\_count\_runways.setText("Кол-во полос: "+str(count\_runways))

self.lbl\_count\_runways.setStyleSheet("background: 0; color: white;")

VLayout\_airport.addWidget(self.lbl\_count\_runways)

self.setLayout(VLayout\_airport)

# Кнопки "Далее" и "Выход"

class NavigationBtn(QPushButton):

def \_\_init\_\_(self, parent, name):

QPushButton.\_\_init\_\_(self, parent)

self.setObjectName(name)

self.setFixedWidth(182)

self.setFixedHeight(52)

self.setStyleSheet("""

QPushButton {background:rgb(39, 39, 61); border-radius: 20px;}

QPushButton:hover {background:rgb(70, 70, 116); border-radius: 20px;}

QPushButton:pressed {background:rgb(135, 136, 160); border-radius: 20px;}

""")

VLayout\_lbl = QVBoxLayout()

self.lbl\_name = QLabel(self)

self.lbl\_name.setFont(QtGui.QFont('Helvetica', 24))

self.lbl\_name.setFixedWidth(77)

self.lbl\_name.setFixedHeight(26)

if name == "further":

self.lbl\_name.setText("Далее")

elif name == "exit":

self.lbl\_name.setText("Выход")

self.lbl\_name.setAlignment(Qt.AlignCenter)

self.lbl\_name.setStyleSheet("background: 0; color: white;")

VLayout\_lbl.addWidget(self.lbl\_name)

VLayout\_lbl.setAlignment(Qt.AlignVCenter | Qt.AlignHCenter) # выравнивание (центр)

VLayout\_lbl.setContentsMargins(0, 0, 0, 0) # внешние отступы

self.setLayout(VLayout\_lbl)

from PyQt5 import QtGui

from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QScrollArea, QSizePolicy, QMessageBox

from PyQt5.QtCore import Qt, QObject, QThread, pyqtSignal

import time, random

from datetime import datetime, timedelta

from MainWindowWidgets import FlightWidget, NumberBtn, RunwayWidget, NavigationBtn

from DataBaseMethods import getAirportInfo, getFlightInfo, count\_all\_flights, count\_all\_statistics, setUserStatistics, setSessionsInfo

from HelpWindow import HelpWindow

class UpdateInfo(QObject):

updateTimeSignal = pyqtSignal(str)

addFlightSignal = pyqtSignal(dict)

checkCurrentSignal = pyqtSignal()

finishSignal = pyqtSignal()

def \_\_init\_\_(self, count\_users, current\_airport):

super(UpdateInfo, self).\_\_init\_\_()

self.count\_users = count\_users

self.current\_airport = current\_airport

self.btnStartActive = True

def start(self):

self.id = 0

# Синхронизируем время из базы данных с текущим временем

i = 0

while True:

if time.strftime("%H:%M:%S", time.localtime()) == getFlightInfo(i, self.current\_airport, "time\_show"):

self.id = i

break

i += 1

if i == count\_all\_flights(self.current\_airport):

i = 0

# Начинаем добавлять рейсы из базы данных

while self.btnStartActive != False:

self.timeNow()

self.checkCurrentSignal.emit()

if time.strftime("%H:%M:%S", time.localtime()) == getFlightInfo(self.id, self.current\_airport, "time\_show"):

self.addFlights()

# Чем больше диспетчеров в аэропорту, тем меньше нагрузка для пользователя

self.id += self.count\_users

time.sleep(1)

def timeNow(self):

time\_string = time.strftime("%H:%M:%S", time.localtime())

self.updateTimeSignal.emit(time\_string)

def addFlights(self):

self.addFlightSignal.emit(

{"id": self.id,

"type": getFlightInfo(self.id, self.current\_airport, "type"),

"number": getFlightInfo(self.id, self.current\_airport, "number"),

"firm": getFlightInfo(self.id, self.current\_airport, "firm"),

"model": getFlightInfo(self.id, self.current\_airport, "model"),

"time\_show": getFlightInfo(self.id, self.current\_airport, "time\_show"),

"time\_on\_runway": getFlightInfo(self.id, self.current\_airport, "time\_on\_runway")}

)

def stop(self):

self.btnStartActive = False

class MainWindow(QWidget):

def \_\_init\_\_(self, count\_users, current\_user, current\_airport, parent=None):

QWidget.\_\_init\_\_(self, parent)

self.parent = parent

self.count\_users = count\_users

self.current\_user = current\_user

self.current\_airport = current\_airport

self.count\_runways = getAirportInfo(current\_airport, "count\_runways")

self.btnStartActive = True # при True кнопка имеет надпись "НАЧАТЬ", при False - "СТОП"

self.btns\_flights = [] # карточки рейсов

self.btns\_number = [] # кнопки "1", "2", "3", "4"

self.runways = [] # взлётно-посадочные полосы

self.count\_all\_flights = 0 # кол-во всех задействованных рейсов

self.count\_boarding\_flights = 0 # кол-во рейсов на посадку

self.count\_takeoff\_flights = 0 # кол-во рейсов на взлёт

self.count\_flights\_on\_runways = [] # кол-во рейсов на взлётно-посадочных полосах

for i in range(self.count\_runways):

self.count\_flights\_on\_runways.append(0)

self.current\_runway = -1 # номер выбранной взлётно-посадочной полосы

self.current\_flight = -1 # id выбранного рейса

# Расстановка элементов

VLayout\_main = QVBoxLayout()

VLayout\_main.setAlignment(Qt.AlignVCenter)

VLayout\_main.setSpacing(15) # расстояние между элементами

VLayout\_main.setContentsMargins(0, 0, 0, 0) # внешние отступы

# Time

HLayout\_time = QHBoxLayout()

HLayout\_time.setAlignment(Qt.AlignRight)

HLayout\_time.setContentsMargins(10, 10, 10, 0) # внешние отступы

self.lbl\_time = QLabel(self) # заголовок

self.lbl\_time.setFont(QtGui.QFont('Helvetica', 16)) # изменяем шрифт

self.lbl\_time.setFixedWidth(66)

self.lbl\_time.setFixedHeight(18)

self.lbl\_time.setText("00:00:00") # меняем текст

self.lbl\_time.setStyleSheet("color: black;") # меняем цвет текста

HLayout\_time.addWidget(self.lbl\_time)

VLayout\_main.addLayout(HLayout\_time)

self.widget\_flights\_and\_navigation = QWidget()

self.widget\_flights\_and\_navigation.setFixedWidth(QApplication.desktop().screenGeometry().width())

self.widget\_flights\_and\_navigation.setFixedHeight(325)

HLayout\_flights\_and\_navigation = QHBoxLayout()

HLayout\_flights\_and\_navigation.setAlignment(Qt.AlignHCenter)

HLayout\_flights\_and\_navigation.setContentsMargins(0, 0, 0, 0) # внешние отступы

HLayout\_flights\_and\_navigation.setSpacing(40) # расстояние между элементами

# Рейсы на посадку

VLayout\_boarding\_flights = QVBoxLayout() # рейсы на посадку

VLayout\_boarding\_flights.setAlignment(Qt.AlignTop)

VLayout\_boarding\_flights.setContentsMargins(0, 0, 0, 0) # внешние отступы

VLayout\_boarding\_flights.setSpacing(5) # расстояние между элементами

self.lbl\_boarding\_flights = QLabel(self) # заголовок

self.lbl\_boarding\_flights.setFont(QtGui.QFont('Helvetica', 36, weight=QtGui.QFont.Bold)) # изменяем шрифт

self.lbl\_boarding\_flights.setFixedWidth(327)

self.lbl\_boarding\_flights.setFixedHeight(46)

self.lbl\_boarding\_flights.setText("Рейсы на посадку") # меняем текст

self.lbl\_boarding\_flights.setStyleSheet("color: black;") # меняем цвет текста

VLayout\_boarding\_flights.addWidget(self.lbl\_boarding\_flights)

HLayout\_boarding\_flights\_count = QHBoxLayout()

HLayout\_boarding\_flights\_count.setAlignment(Qt.AlignLeft)

HLayout\_boarding\_flights\_count.setContentsMargins(0, 0, 0, 0) # внешние отступы

HLayout\_boarding\_flights\_count.setSpacing(4) # расстояние между элементами

self.lbl\_boarding\_flights\_count = QLabel(self) # заголовок

self.lbl\_boarding\_flights\_count.setFont(QtGui.QFont('Helvetica', 16)) # изменяем шрифт

self.lbl\_boarding\_flights\_count.setFixedWidth(58)

self.lbl\_boarding\_flights\_count.setFixedHeight(22)

self.lbl\_boarding\_flights\_count.setText("кол-во:") # меняем текст

self.lbl\_boarding\_flights\_count.setStyleSheet("color: black;") # меняем цвет текста

HLayout\_boarding\_flights\_count.addWidget(self.lbl\_boarding\_flights\_count)

self.boarding\_flights\_count = QLabel(self) # заголовок

self.boarding\_flights\_count.setFont(QtGui.QFont('Helvetica', 16)) # изменяем шрифт

self.boarding\_flights\_count.setFixedWidth(20)

self.boarding\_flights\_count.setFixedHeight(22)

self.boarding\_flights\_count.setText(str(self.count\_boarding\_flights)) # меняем текст

self.boarding\_flights\_count.setStyleSheet("color: black;") # меняем цвет текста

HLayout\_boarding\_flights\_count.addWidget(self.boarding\_flights\_count)

VLayout\_boarding\_flights.addLayout(HLayout\_boarding\_flights\_count)

# Карточки рейсов на посадку

self.scrollA\_boarding\_flights = QScrollArea()

self.scrollA\_boarding\_flights.setFixedWidth(435)

self.scrollA\_boarding\_flights.setFixedHeight(242)

self.scrollA\_boarding\_flights.setStyleSheet("""

QScrollArea {

background-color: rgb(39, 39, 61);

padding: 10px 5px;

border-top-left-radius: 15px;

border-top-right-radius: 0px;

border-bottom-left-radius: 15px;

border-bottom-right-radius: 0px;

}

QScrollBar:vertical {

background-color: rgb(39, 39, 61);

width: 10px;

}

""")

self.scrollW\_boarding\_flights = QWidget()

self.scrollW\_boarding\_flights.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)

self.scrollW\_boarding\_flights.setFixedWidth(411)

self.scrollW\_boarding\_flights.setStyleSheet("background-color: rgb(39, 39, 61); border-radius: 15px;")

self.VLayout\_scroll\_boarding\_flights = QVBoxLayout()

self.VLayout\_scroll\_boarding\_flights.setAlignment(Qt.AlignTop)

self.VLayout\_scroll\_boarding\_flights.setContentsMargins(5, 5, 5, 5) # внешние отступы

self.VLayout\_scroll\_boarding\_flights.setSpacing(5) # расстояние между элементами

self.scrollW\_boarding\_flights.setLayout(self.VLayout\_scroll\_boarding\_flights)

self.scrollA\_boarding\_flights.setWidget(self.scrollW\_boarding\_flights)

VLayout\_boarding\_flights.addWidget(self.scrollA\_boarding\_flights)

HLayout\_flights\_and\_navigation.addLayout(VLayout\_boarding\_flights)

VLayout\_numbers = QVBoxLayout() # кнопки с номерами

VLayout\_numbers.setAlignment(Qt.AlignBottom)

VLayout\_numbers.setContentsMargins(0, 0, 0, 12) # внешние отступы

VLayout\_numbers.setSpacing(6) # расстояние между элементами

for i in range(self.count\_runways):

self.btns\_number.append(NumberBtn(self, number=i+1))

self.btns\_number[-1].clicked.connect(self.onBtnNumberClick) # при клике на кнопку

VLayout\_numbers.addWidget(self.btns\_number[-1])

HLayout\_flights\_and\_navigation.addLayout(VLayout\_numbers)

# Рейсы на взлёт

VLayout\_takeoff\_flights = QVBoxLayout() # рейсы на посадку

VLayout\_takeoff\_flights.setAlignment(Qt.AlignTop)

VLayout\_takeoff\_flights.setContentsMargins(0, 0, 0, 0) # внешние отступы

VLayout\_takeoff\_flights.setSpacing(5) # расстояние между элементами

self.lbl\_takeoff\_flights = QLabel(self) # заголовок

self.lbl\_takeoff\_flights.setFont(QtGui.QFont('Helvetica', 36, weight=QtGui.QFont.Bold)) # изменяем шрифт

self.lbl\_takeoff\_flights.setFixedWidth(327)

self.lbl\_takeoff\_flights.setFixedHeight(46)

self.lbl\_takeoff\_flights.setText("Рейсы на взлёт") # меняем текст

self.lbl\_takeoff\_flights.setStyleSheet("color: black;") # меняем цвет текста

VLayout\_takeoff\_flights.addWidget(self.lbl\_takeoff\_flights)

HLayout\_takeoff\_flights\_count = QHBoxLayout()

HLayout\_takeoff\_flights\_count.setAlignment(Qt.AlignLeft)

HLayout\_takeoff\_flights\_count.setSpacing(10) # расстояние между элементами

self.lbl\_takeoff\_flights\_count = QLabel(self) # заголовок

self.lbl\_takeoff\_flights\_count.setFont(QtGui.QFont('Helvetica', 16)) # изменяем шрифт

self.lbl\_takeoff\_flights\_count.setFixedWidth(58)

self.lbl\_takeoff\_flights\_count.setFixedHeight(22)

self.lbl\_takeoff\_flights\_count.setText("кол-во:") # меняем текст

self.lbl\_takeoff\_flights\_count.setStyleSheet("color: black;") # меняем цвет текста

HLayout\_takeoff\_flights\_count.addWidget(self.lbl\_takeoff\_flights\_count)

self.takeoff\_flights\_count = QLabel(self) # заголовок

self.takeoff\_flights\_count.setFont(QtGui.QFont('Helvetica', 16)) # изменяем шрифт

self.takeoff\_flights\_count.setFixedWidth(20)

self.takeoff\_flights\_count.setFixedHeight(22)

self.takeoff\_flights\_count.setText(str(self.count\_takeoff\_flights)) # меняем текст

self.takeoff\_flights\_count.setStyleSheet("color: black;") # меняем цвет текста

HLayout\_takeoff\_flights\_count.addWidget(self.takeoff\_flights\_count)

VLayout\_takeoff\_flights.addLayout(HLayout\_takeoff\_flights\_count)

# Карточки рейсов на взлёт

self.scrollA\_takeoff\_flights = QScrollArea()

self.scrollA\_takeoff\_flights.setFixedWidth(435)

self.scrollA\_takeoff\_flights.setFixedHeight(242)

self.scrollA\_takeoff\_flights.setStyleSheet("""

QScrollArea {

background-color: rgb(39, 39, 61);

padding: 10px 5px;

border-top-left-radius: 15px;

border-top-right-radius: 0px;

border-bottom-left-radius: 15px;

border-bottom-right-radius: 0px;

}

QScrollBar:vertical {

background-color: rgb(39, 39, 61);

width: 10px;

}

""")

self.scrollW\_takeoff\_flights = QWidget()

self.scrollW\_takeoff\_flights.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)

self.scrollW\_takeoff\_flights.setFixedWidth(411)

self.scrollW\_takeoff\_flights.setStyleSheet("background-color: rgb(39, 39, 61); border-radius: 15px;")

self.VLayout\_scroll\_takeoff\_flights = QVBoxLayout()

self.VLayout\_scroll\_takeoff\_flights.setAlignment(Qt.AlignTop)

self.VLayout\_scroll\_takeoff\_flights.setContentsMargins(5, 5, 5, 5) # внешние отступы

self.VLayout\_scroll\_takeoff\_flights.setSpacing(5) # расстояние между элементами

self.scrollW\_takeoff\_flights.setLayout(self.VLayout\_scroll\_takeoff\_flights)

self.scrollA\_takeoff\_flights.setWidget(self.scrollW\_takeoff\_flights)

VLayout\_takeoff\_flights.addWidget(self.scrollA\_takeoff\_flights)

HLayout\_flights\_and\_navigation.addLayout(VLayout\_takeoff\_flights)

# Кнопки "НАЧАТЬ", "Справка" и "Выход"

self.widget\_btns\_navigation = QWidget()

self.widget\_btns\_navigation.setFixedWidth(200)

self.widget\_btns\_navigation.setFixedHeight(217)

VLayout\_btns\_navigation = QVBoxLayout() # кнопки с номерами

VLayout\_btns\_navigation.setAlignment(Qt.AlignVCenter)

VLayout\_btns\_navigation.setContentsMargins(30, 0, 0, 0) # внешние отступы

VLayout\_btns\_navigation.setSpacing(8) # расстояние между элементами

self.btn\_start = NavigationBtn(self, name="start", width=170, height=61) # кнопка "НАЧАТЬ"

self.btn\_start.clicked.connect(self.btnStartClick) # при клике на кнопку

VLayout\_btns\_navigation.addWidget(self.btn\_start)

HLayout\_btn\_help = QHBoxLayout()

HLayout\_btn\_help.setAlignment(Qt.AlignRight)

HLayout\_btn\_help.setContentsMargins(0, 30, 0, 0) # внешние отступы

self.btn\_help = NavigationBtn(self, name="help", width=142, height=37) # кнопка "Справка"

self.btn\_help.clicked.connect(self.onBtnHelpClick) # при клике на кнопку

HLayout\_btn\_help.addWidget(self.btn\_help)

VLayout\_btns\_navigation.addLayout(HLayout\_btn\_help)

HLayout\_btn\_exit = QHBoxLayout()

HLayout\_btn\_exit.setAlignment(Qt.AlignRight)

HLayout\_btn\_exit.setContentsMargins(0, 0, 0, 0) # внешние отступы

self.btn\_exit = NavigationBtn(self, name="exit", width=142, height=37) # кнопка "Выход"

self.btn\_exit.clicked.connect(self.parent.parent.close) # при клике на кнопку

HLayout\_btn\_exit.addWidget(self.btn\_exit)

VLayout\_btns\_navigation.addLayout(HLayout\_btn\_exit)

self.widget\_btns\_navigation.setLayout(VLayout\_btns\_navigation)

HLayout\_flights\_and\_navigation.addWidget(self.widget\_btns\_navigation)

self.widget\_flights\_and\_navigation.setLayout(HLayout\_flights\_and\_navigation)

VLayout\_main.addWidget(self.widget\_flights\_and\_navigation)

self.lbl\_line = QLabel(self) # Заголовок

self.lbl\_line.setFixedWidth(QApplication.desktop().screenGeometry().width())

self.lbl\_line.setFixedHeight(1)

self.lbl\_line.setStyleSheet("background-color: black;") # меняем цвет текста

VLayout\_main.addWidget(self.lbl\_line)

# Полосы

HLayout\_runway = QHBoxLayout()

HLayout\_runway.setAlignment(Qt.AlignHCenter)

HLayout\_runway.setSpacing(30) # расстояние между элементами

for i in range(self.count\_runways):

self.runways.append(RunwayWidget(self, id=i+1))

HLayout\_runway.addWidget(self.runways[-1])

VLayout\_main.addLayout(HLayout\_runway)

self.setLayout(VLayout\_main)

# Выбор номера взлётно-посадочной полосы

def onBtnNumberClick(self):

btn\_name = int(self.sender().objectName())

# Выделяем кнопку

self.btns\_number[btn\_name].setStyleSheet("""

QPushButton {background:rgb(70, 70, 116); border-radius: 10px; border: 4px solid rgb(39, 39, 61);}

""")

if self.current\_runway == -1:

self.current\_runway = btn\_name # присваиваем новый номер взлётно-посадочной полосы

if self.current\_runway != btn\_name:

# Убираем выделение предыдущей кнопки

self.btns\_number[self.current\_runway].setStyleSheet("""

QPushButton {background:rgb(39, 39, 61); border-radius: 10px; border: 0px;}

QPushButton:hover {background:rgb(39, 39, 61); border-radius: 10px; border: 2px solid rgb(172, 177, 202);}

""")

self.current\_runway = btn\_name # присваиваем новый номер взлётно-посадочной полосы

# Выбор рейса

def onBtnFlightClick(self):

btn\_name = int(self.sender().objectName())

if self.btns\_flights[btn\_name].can\_click == True:

# Выделяем кнопку

self.btns\_flights[btn\_name].setStyleSheet("""

QPushButton {background-color:rgb(70, 70, 116); border-radius: 10px; border: 3px solid rgb(39, 39, 61);}

""")

self.btns\_flights[btn\_name].lbl\_number.setStyleSheet("background:rgb(70, 70, 116); color:white;")

self.btns\_flights[btn\_name].lbl\_firm.setStyleSheet("background:rgb(70, 70, 116); color:white;")

self.btns\_flights[btn\_name].lbl\_model.setStyleSheet("background:rgb(70, 70, 116); color:white;")

self.btns\_flights[btn\_name].lbl\_time\_show.setStyleSheet("background:rgb(70, 70, 116); color:white;")

if self.current\_flight == -1:

self.current\_flight = btn\_name # присваиваем новую карточку рейса

if self.current\_flight != btn\_name:

# Убираем выделение предыдущей кнопки

self.btns\_flights[self.current\_flight].setStyleSheet("""

QPushButton {background:rgb(217, 217, 217); border-radius: 10px; border: 0px;}

QPushButton:hover {background:rgb(217, 217, 217); border-radius: 10px; border: 3px solid rgb(39, 39, 61);}

""")

self.btns\_flights[self.current\_flight].lbl\_number.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.btns\_flights[self.current\_flight].lbl\_firm.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.btns\_flights[self.current\_flight].lbl\_model.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.btns\_flights[self.current\_flight].lbl\_time\_show.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.current\_flight = btn\_name # присваиваем новую карточку рейса

# Кнопка "Справка"

def onBtnHelpClick(self):

help\_window = HelpWindow(self)

help\_window.show()

# Кнопка "НАЧАТЬ"

def btnStartClick(self, status):

if status != "failed":

status = "succeed"

if self.btnStartActive == True:

self.date\_start = datetime.now()

self.btn\_start.lbl\_name.setText("СТОП")

self.obj\_update\_info = UpdateInfo(self.count\_users, self.current\_airport) # объект, в котором происходит работа со временем и данными из базы данных

self.thread = QThread() # отдельный поток для обновления данных на экране

self.obj\_update\_info.moveToThread(self.thread) # добавляем созданный объект в отдельный поток

self.thread.started.connect(self.obj\_update\_info.start) # при старте потока запускается функция start в объекте

self.obj\_update\_info.updateTimeSignal.connect(self.updateTime) # при сигнале updateTimeSignal вызывается функция updateTime

self.obj\_update\_info.addFlightSignal.connect(self.addFlight) # при сигнале addFlightSignal вызывается функция addFlight

self.obj\_update\_info.checkCurrentSignal.connect(self.updateRunways) # при сигнале checkCurrentSignal вызывается функция updateRunways

self.obj\_update\_info.finishSignal.connect(self.thread.quit) # при сигнале finishSignal завершаем отдельный поток

self.thread.start() # запускаем отдельный поток

self.btnStartActive = False

elif self.btnStartActive == False:

self.date\_end = datetime.now()

self.obj\_update\_info.stop() # останавливаем все процессы внутри объекта

self.thread.quit() # завершаем отдельный поток

self.btn\_start.lbl\_name.setText("НАЧАТЬ")

self.btnStartActive = True

# Собираем и отправляем статистику

setUserStatistics(id=count\_all\_statistics(),

user\_id=self.current\_user,

airport\_id=self.current\_airport,

count\_users\_in\_airport=self.count\_users,

time\_spent=str(self.date\_end-self.date\_start),

data\_start=str(self.date\_start),

data\_end=str(self.date\_end),

status=str(status))

setSessionsInfo(user\_id=self.current\_user, status=str(status))

msg = QMessageBox()

if status == "failed":

msg.setIcon(QMessageBox.Critical)

elif status == "succeed":

msg.setIcon(QMessageBox.Information)

msg.setText("Сессия завершена.")

msg.setInformativeText("user\_id: {0},\nairport\_id: {1},\ncount\_users\_in\_airport: {2},\ntime\_spent: {3},\ndata\_start: {4},\ndata\_end: {5},\nstatus: {6}".format(self.current\_user, self.current\_airport, self.count\_users, str(self.date\_end-self.date\_start), str(self.date\_start), str(self.date\_end), str(status)))

msg.setWindowTitle("Результаты сохранены")

msg.exec\_()

def updateTime(self, time\_string):

self.lbl\_time.setText(time\_string)

def addFlight(self, flight\_info):

self.btns\_flights.append(FlightWidget(self,

object\_name=self.count\_all\_flights,

id=flight\_info["id"],

type=flight\_info["type"],

number=flight\_info["number"],

firm=flight\_info["firm"],

model=flight\_info["model"],

time\_show=flight\_info["time\_show"],

time\_on\_runway=flight\_info["time\_on\_runway"])

)

self.btns\_flights[-1].clicked.connect(self.onBtnFlightClick) # при клике на кнопку

self.count\_all\_flights += 1

if flight\_info["type"] == "boarding\_flight":

self.count\_boarding\_flights += 1

self.VLayout\_scroll\_boarding\_flights.addWidget(self.btns\_flights[-1])

self.scrollW\_boarding\_flights.setFixedHeight(self.count\_boarding\_flights \* (34 + 7))

self.boarding\_flights\_count.setText(str(self.count\_boarding\_flights)) # меняем текст

elif flight\_info["type"] == "takeoff\_flight":

self.count\_takeoff\_flights += 1

self.VLayout\_scroll\_takeoff\_flights.addWidget(self.btns\_flights[-1])

self.scrollW\_takeoff\_flights.setFixedHeight(self.count\_takeoff\_flights \* (34 + 7))

self.takeoff\_flights\_count.setText(str(self.count\_takeoff\_flights)) # меняем текст

def updateRunways(self):

# Главное ограничение для диспетчера - если кол-во рейсов в одном из блоков больше 10, то программа останавливается

if self.count\_boarding\_flights > 10 or self.count\_takeoff\_flights > 10:

self.btnStartActive = False

self.btnStartClick(status="failed")

if self.current\_runway != -1 and self.current\_flight != -1:

if self.btns\_flights[self.current\_flight].type == "boarding\_flight":

self.count\_boarding\_flights -= 1

self.scrollW\_boarding\_flights.setFixedHeight(self.count\_boarding\_flights \* (42 + 7))

self.boarding\_flights\_count.setText(str(self.count\_boarding\_flights)) # меняем текст

elif self.btns\_flights[self.current\_flight].type == "takeoff\_flight":

self.count\_takeoff\_flights -= 1

self.scrollW\_takeoff\_flights.setFixedHeight(self.count\_takeoff\_flights \* (42 + 7))

self.takeoff\_flights\_count.setText(str(self.count\_takeoff\_flights)) # меняем текст

self.count\_flights\_on\_runways[self.current\_runway] += 1

self.btns\_flights[self.current\_flight].current\_runway = self.current\_runway

self.runways[self.current\_runway].VLayout\_runway\_flights.addWidget(self.btns\_flights[self.current\_flight])

self.runways[self.current\_runway].scrollW\_runway.setFixedHeight(self.count\_flights\_on\_runways[self.current\_runway] \* (50 + 13))

self.runways[self.current\_runway].runway\_count.setText(str(self.count\_flights\_on\_runways[self.current\_runway])) # меняем текст

# Убираем выделение у кнопки с номером

self.btns\_number[self.current\_runway].setStyleSheet("""

QPushButton {background:rgb(39, 39, 61); border-radius: 10px; border: 0px;}

QPushButton:hover {background:rgb(39, 39, 61); border-radius: 10px; border: 2px solid rgb(172, 177, 202);}

""")

self.current\_runway = -1

# Устанавливаем новые стили у карточки рейса

self.btns\_flights[self.current\_flight].setStyleSheet("""

QPushButton {background:rgb(217, 217, 217); border-radius: 0px; border: 0px;}

""")

self.btns\_flights[self.current\_flight].setFixedWidth(253)

self.btns\_flights[self.current\_flight].setFixedHeight(50)

self.btns\_flights[self.current\_flight].can\_click = False

self.btns\_flights[self.current\_flight].start\_time\_on\_runway = time.strftime("%H:%M:%S", time.localtime())

self.btns\_flights[self.current\_flight].lbl\_number.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.btns\_flights[self.current\_flight].lbl\_number.setFont(QtGui.QFont('Helvetica', 11, weight=QtGui.QFont.Bold)) # изменяем шрифт

self.btns\_flights[self.current\_flight].lbl\_number.setFixedWidth(38)

self.btns\_flights[self.current\_flight].lbl\_number.setFixedHeight(46)

self.btns\_flights[self.current\_flight].lbl\_number.setWordWrap(True) # перенос текста

self.btns\_flights[self.current\_flight].lbl\_firm.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.btns\_flights[self.current\_flight].lbl\_firm.setFont(QtGui.QFont('Helvetica', 11)) # изменяем шрифт

self.btns\_flights[self.current\_flight].lbl\_firm.setFixedWidth(70)

self.btns\_flights[self.current\_flight].lbl\_firm.setFixedHeight(45)

self.btns\_flights[self.current\_flight].lbl\_firm.setWordWrap(True) # перенос текста

self.btns\_flights[self.current\_flight].lbl\_model.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.btns\_flights[self.current\_flight].lbl\_model.setFont(QtGui.QFont('Helvetica', 11)) # изменяем шрифт

self.btns\_flights[self.current\_flight].lbl\_model.setFixedWidth(70)

self.btns\_flights[self.current\_flight].lbl\_model.setFixedHeight(45)

self.btns\_flights[self.current\_flight].lbl\_model.setWordWrap(True) # перенос текста

self.btns\_flights[self.current\_flight].lbl\_time\_show.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.btns\_flights[self.current\_flight].lbl\_time\_show.setFont(QtGui.QFont('Helvetica', 9)) # изменяем шрифт

self.btns\_flights[self.current\_flight].lbl\_time\_show.setFixedWidth(36)

self.btns\_flights[self.current\_flight].lbl\_time\_show.setFixedHeight(22)

self.btns\_flights[self.current\_flight].lbl\_time\_show.setWordWrap(True) # перенос текста

self.current\_flight = -1

# Скрываем рейсы, которые уже не должны находиться на взлётно-посадочной полосе

for i in range(len(self.btns\_flights)):

if self.btns\_flights[i].can\_click == False and self.btns\_flights[i].start\_time\_on\_runway != "":

if datetime.now() >= datetime(year=datetime.now().year, month=datetime.now().month,

day=datetime.now().day,

hour=int(self.btns\_flights[i].start\_time\_on\_runway[:2]),

minute=int(self.btns\_flights[i].start\_time\_on\_runway[3:5]),

second=int(self.btns\_flights[i].start\_time\_on\_runway[6:])) + timedelta(

seconds=int(self.btns\_flights[i].time\_on\_runway)):

self.btns\_flights[i].hide() # скрываем устаревший рейс

self.btns\_flights[i].start\_time\_on\_runway = ""

self.count\_flights\_on\_runways[self.btns\_flights[i].current\_runway] -= 1

self.runways[self.btns\_flights[i].current\_runway].scrollW\_runway.setFixedHeight(

self.count\_flights\_on\_runways[self.btns\_flights[i].current\_runway] \* (50 + 13))

self.runways[self.btns\_flights[i].current\_runway].runway\_count.setText(

str(self.count\_flights\_on\_runways[self.btns\_flights[i].current\_runway])) # меняем текст

break

def \_\_del\_\_(self):

if self.btnStartActive == False:

self.date\_end = datetime.now()

self.obj\_update\_info.stop() # останавливаем все процессы внутри объекта

self.thread.quit() # завершаем отдельный поток

# Собираем и отправляем статистику

setUserStatistics(id=count\_all\_statistics(),

user\_id=self.current\_user,

airport\_id=self.current\_airport,

count\_users\_in\_airport=self.count\_users,

time\_spent=str(self.date\_end - self.date\_start),

data\_start=str(self.date\_start),

data\_end=str(self.date\_end),

status="error")

setSessionsInfo(user\_id=self.current\_user, status="error")

msg = QMessageBox()

msg.setIcon(QMessageBox.Warning)

msg.setText("Сессия завершена.")

msg.setInformativeText(

"user\_id: {0},\nairport\_id: {1},\ncount\_users\_in\_airport: {2},\ntime\_spent: {3},\ndata\_start: {4},\ndata\_end: {5},\nstatus: {6}".format(

self.current\_user, self.current\_airport, self.count\_users, str(self.date\_end - self.date\_start),

str(self.date\_start), str(self.date\_end), "error"))

msg.setWindowTitle("Результаты сохранены")

msg.exec\_()

from PyQt5 import QtGui

from PyQt5.QtWidgets import QWidget, QVBoxLayout, QHBoxLayout, QLabel, QScrollArea, QPushButton, QSizePolicy

from PyQt5.QtCore import Qt

# Карточки рейсов

class FlightWidget(QPushButton):

def \_\_init\_\_(self, parent, object\_name, id, type, number, firm, model, time\_show, time\_on\_runway):

QPushButton.\_\_init\_\_(self, parent)

self.id = id

self.type = type

self.time\_on\_runway = time\_on\_runway

self.current\_runway = -1

self.can\_click = True

self.start\_time\_on\_runway = ""

self.setObjectName(str(object\_name))

self.setFixedWidth(401)

self.setFixedHeight(34)

self.setStyleSheet("""

QPushButton {background:rgb(217, 217, 217); border-radius: 10px; border: 0px;}

QPushButton:hover {background:rgb(217, 217, 217); border-radius: 10px; border: 3px solid rgb(39, 39, 61);}

""")

HLayout\_flight = QHBoxLayout()

HLayout\_flight.setContentsMargins(0, 0, 0, 0) # внешние отступы

HLayout\_flight.setAlignment(Qt.AlignVCenter | Qt.AlignHCenter)

self.lbl\_number = QLabel(self)

self.lbl\_number.setFont(QtGui.QFont('Helvetica', 16, weight=QtGui.QFont.Bold)) # изменяем шрифт

self.lbl\_number.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.lbl\_number.setFixedWidth(63)

self.lbl\_number.setFixedHeight(18)

self.lbl\_number.setText(str(number))

self.lbl\_number.setAlignment(Qt.AlignCenter)

HLayout\_flight.addWidget(self.lbl\_number)

self.lbl\_firm = QLabel(self)

self.lbl\_firm.setFont(QtGui.QFont('Helvetica', 16)) # изменяем шрифт

self.lbl\_firm.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.lbl\_firm.setFixedWidth(120)

self.lbl\_firm.setFixedHeight(18)

self.lbl\_firm.setText(str(firm))

self.lbl\_firm.setAlignment(Qt.AlignLeft | Qt.AlignVCenter)

HLayout\_flight.addWidget(self.lbl\_firm)

self.lbl\_model = QLabel(self)

self.lbl\_model.setFont(QtGui.QFont('Helvetica', 16)) # изменяем шрифт

self.lbl\_model.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.lbl\_model.setFixedWidth(120)

self.lbl\_model.setFixedHeight(18)

self.lbl\_model.setText(str(model))

self.lbl\_model.setAlignment(Qt.AlignLeft | Qt.AlignVCenter)

HLayout\_flight.addWidget(self.lbl\_model)

self.lbl\_time\_show = QLabel(self)

self.lbl\_time\_show.setFont(QtGui.QFont('Helvetica', 12)) # изменяем шрифт

self.lbl\_time\_show.setStyleSheet("background:rgb(217, 217, 217); color:black;")

self.lbl\_time\_show.setFixedWidth(60)

self.lbl\_time\_show.setFixedHeight(14)

self.lbl\_time\_show.setText(str(time\_show))

self.lbl\_time\_show.setAlignment(Qt.AlignCenter)

HLayout\_flight.addWidget(self.lbl\_time\_show)

self.setLayout(HLayout\_flight)

# Кнопки "1", "2", "3", "4"

class NumberBtn(QPushButton):

def \_\_init\_\_(self, parent, number):

QPushButton.\_\_init\_\_(self, parent)

self.setObjectName(str(number-1))

self.setFixedWidth(50)

self.setFixedHeight(50)

self.setStyleSheet("""

QPushButton {background:rgb(39, 39, 61); border-radius: 10px; border: 0px;}

QPushButton:hover {background:rgb(39, 39, 61); border-radius: 10px; border: 2px solid rgb(172, 177, 202);}

""")

VLayout\_lbl = QVBoxLayout()

self.lbl\_name = QLabel(self)

self.lbl\_name.setText(str(number))

self.lbl\_name.setFont(QtGui.QFont('Helvetica', 32, weight=QtGui.QFont.Bold))

self.lbl\_name.setFixedWidth(50)

self.lbl\_name.setFixedHeight(50)

self.lbl\_name.setAlignment(Qt.AlignCenter)

self.lbl\_name.setStyleSheet("background: 0; color: white;")

VLayout\_lbl.addWidget(self.lbl\_name)

VLayout\_lbl.setAlignment(Qt.AlignVCenter | Qt.AlignHCenter) # выравнивание (центр)

VLayout\_lbl.setContentsMargins(0, 0, 0, 0) # внешние отступы

self.setLayout(VLayout\_lbl)

# Взлётно-посадочные полосы

class RunwayWidget(QWidget):

def \_\_init\_\_(self, parent, id):

QWidget.\_\_init\_\_(self, parent)

self.setObjectName(str(id-1))

self.setFixedWidth(280)

self.setFixedHeight(334)

#self.btns\_runway = [] # карточки рейсов на данную полосу

VLayout\_runway = QVBoxLayout()

VLayout\_runway.setAlignment(Qt.AlignTop)

VLayout\_runway.setSpacing(0) # расстояние между элементами

VLayout\_runway.setContentsMargins(0, 0, 0, 0) # внешние отступы

self.scrollA\_runway = QScrollArea()

self.scrollA\_runway.setFixedWidth(280)

self.scrollA\_runway.setFixedHeight(334)

self.scrollA\_runway.setStyleSheet("""

QScrollArea {

background-color: rgb(135, 136, 160);

padding: 10px 5px;

border-radius: 0px;

}

QScrollBar:vertical {

background-color: rgb(135, 136, 160);

width: 10px;

}

""")

self.scrollW\_runway = QWidget()

self.scrollW\_runway.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)

self.scrollW\_runway.setFixedWidth(253)

self.scrollW\_runway.setStyleSheet("background-color: rgb(135, 136, 160);")

self.VLayout\_runway\_flights = QVBoxLayout()

self.VLayout\_runway\_flights.setAlignment(Qt.AlignTop)

self.VLayout\_runway\_flights.setContentsMargins(5, 0, 0, 0) # внешние отступы

self.VLayout\_runway\_flights.setSpacing(7) # расстояние между элементами

self.scrollW\_runway.setLayout(self.VLayout\_runway\_flights)

self.scrollA\_runway.setWidget(self.scrollW\_runway)

VLayout\_runway.addWidget(self.scrollA\_runway)

# Нижний блок с названием полосы и количеством рейсов

self.title\_runway = QWidget()

self.title\_runway.setFixedHeight(44)

self.title\_runway.setFixedWidth(280)

self.title\_runway.setStyleSheet("background:rgb(70, 70, 116);") # меняем цвет текста

HLayout\_runway\_title = QHBoxLayout()

HLayout\_runway\_title.setAlignment(Qt.AlignHCenter)

HLayout\_runway\_title.setContentsMargins(0, 0, 0, 0) # внешние отступы

HLayout\_runway\_title.setSpacing(5) # расстояние между элементами

self.lbl\_runway\_title = QLabel(self) # заголовок

self.lbl\_runway\_title.setFont(QtGui.QFont('Helvetica', 16)) # изменяем шрифт

self.lbl\_runway\_title.setFixedWidth(150)

self.lbl\_runway\_title.setFixedHeight(18)

self.lbl\_runway\_title.setText("ПОЛОСА №"+str(id)) # меняем текст

self.lbl\_runway\_title.setStyleSheet("color: white;") # меняем цвет текста

HLayout\_runway\_title.addWidget(self.lbl\_runway\_title)

self.lbl\_runway\_count = QLabel(self) # заголовок

self.lbl\_runway\_count.setFont(QtGui.QFont('Helvetica', 16)) # изменяем шрифт

self.lbl\_runway\_count.setFixedWidth(58)

self.lbl\_runway\_count.setFixedHeight(18)

self.lbl\_runway\_count.setText("кол-во:") # меняем текст

self.lbl\_runway\_count.setStyleSheet("color: white;") # меняем цвет текста

self.lbl\_runway\_count.setAlignment(Qt.AlignBottom)

HLayout\_runway\_title.addWidget(self.lbl\_runway\_count)

self.runway\_count = QLabel(self) # заголовок

self.runway\_count.setFont(QtGui.QFont('Helvetica', 16)) # изменяем шрифт

self.runway\_count.setFixedWidth(20)

self.runway\_count.setFixedHeight(18)

self.runway\_count.setText("0") # меняем текст

self.runway\_count.setStyleSheet("color: white;") # меняем цвет текста

self.runway\_count.setAlignment(Qt.AlignBottom)

HLayout\_runway\_title.addWidget(self.runway\_count)

self.title\_runway.setLayout(HLayout\_runway\_title)

VLayout\_runway.addWidget(self.title\_runway)

self.setLayout(VLayout\_runway)

# Кнопки "НАЧАТЬ", "Справка" и "Выход"

class NavigationBtn(QPushButton):

def \_\_init\_\_(self, parent, name, width, height):

QPushButton.\_\_init\_\_(self, parent)

self.setObjectName(name)

self.setFixedWidth(width)

self.setFixedHeight(height)

self.setStyleSheet("""

QPushButton {background:rgb(39, 39, 61); border-radius: 15px;}

QPushButton:hover {background:rgb(70, 70, 116); border-radius: 15px;}

QPushButton:pressed {background:rgb(135, 136, 160); border-radius: 15px;}

""")

VLayout\_lbl = QVBoxLayout()

self.lbl\_name = QLabel(self)

if name == "start":

self.lbl\_name.setText("НАЧАТЬ")

self.lbl\_name.setFont(QtGui.QFont('Helvetica', 24, weight=QtGui.QFont.Bold))

self.lbl\_name.setFixedWidth(101)

self.lbl\_name.setFixedHeight(26)

if name == "help":

self.lbl\_name.setText("Справка")

self.lbl\_name.setFont(QtGui.QFont('Helvetica', 20))

self.lbl\_name.setFixedWidth(82)

self.lbl\_name.setFixedHeight(22)

elif name == "exit":

self.lbl\_name.setText("Выход")

self.lbl\_name.setFont(QtGui.QFont('Helvetica', 20))

self.lbl\_name.setFixedWidth(82)

self.lbl\_name.setFixedHeight(22)

self.lbl\_name.setAlignment(Qt.AlignCenter)

self.lbl\_name.setStyleSheet("background: 0; color: white;")

VLayout\_lbl.addWidget(self.lbl\_name)

VLayout\_lbl.setAlignment(Qt.AlignVCenter | Qt.AlignHCenter) # выравнивание (центр)

VLayout\_lbl.setContentsMargins(0, 0, 0, 0) # внешние отступы

self.setLayout(VLayout\_lbl)

from PyQt5 import QtGui

from PyQt5.QtWidgets import QDialog, QApplication, QVBoxLayout, QLabel, QScrollArea, QWidget, QSizePolicy

from PyQt5.QtCore import Qt

class HelpWindow(QDialog):

def \_\_init\_\_(self, parent=None):

QDialog.\_\_init\_\_(self, parent)

self.setGeometry(QApplication.desktop().screenGeometry().width()//4,

QApplication.desktop().screenGeometry().height()//4,

QApplication.desktop().screenGeometry().width()//2,

QApplication.desktop().screenGeometry().height()//2)

self.setWindowTitle('Справка. Симулятор диспетчера аэропорта') # название окна

self.setWindowIcon(QtGui.QIcon('imgs/main.ico')) # иконка окна

self.setStyleSheet("background-color:rgb(172, 177, 202);") # фон окна

# Расстановка элементов

VLayout\_main = QVBoxLayout()

VLayout\_main.setAlignment(Qt.AlignVCenter)

VLayout\_main.setSpacing(0) # расстояние между элементами

VLayout\_main.setContentsMargins(0, 0, 0, 0) # внешние отступы

self.scrollA = QScrollArea()

self.scrollA.setFixedWidth(QApplication.desktop().screenGeometry().width()//2)

self.scrollA.setFixedHeight(QApplication.desktop().screenGeometry().height()//2)

self.scrollA.setStyleSheet("""

QScrollArea {

background-color: rgb(135, 136, 160);

padding: 10px 5px;

border-radius: 0px;

}

QScrollBar:vertical {

background-color: rgb(135, 136, 160);

width: 10px;

}

""")

self.scrollW = QWidget()

self.scrollW.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)

self.scrollW.setFixedWidth((QApplication.desktop().screenGeometry().width()//2)-20)

self.scrollW.setStyleSheet("background-color: rgb(135, 136, 160);")

VLayout\_lbls = QVBoxLayout()

VLayout\_lbls.setAlignment(Qt.AlignCenter)

VLayout\_lbls.setContentsMargins(0, 20, 0, 40) # внешние отступы

VLayout\_lbls.setSpacing(7) # расстояние между элементами

self.lbl0 = QLabel()

self.lbl0.setFixedWidth(QApplication.desktop().screenGeometry().width() // 2.5)

self.lbl0.setWordWrap(True) # перенос текста

self.lbl0.setFont(QtGui.QFont('Helvetica', 18)) # изменяем шрифт

self.lbl0.setContentsMargins(30, 20, 30, 30) # внешние отступы

self.lbl0.setText("""<html><head/><body>

<p style="line-height:30px;"><span>

<b>Добро пожаловать в программу "Симулятор диспетчера аэропорта".</b>

</span></p>

</body></html>

""")

self.lbl0.setStyleSheet("background-color:rgb(172, 177, 202); border-radius: 10px;")

VLayout\_lbls.addWidget(self.lbl0)

self.lbl\_ogr = QLabel()

self.lbl\_ogr.setFixedWidth(QApplication.desktop().screenGeometry().width() // 2.5)

self.lbl\_ogr.setWordWrap(True) # перенос текста

self.lbl\_ogr.setFont(QtGui.QFont('Helvetica', 18)) # изменяем шрифт

self.lbl\_ogr.setContentsMargins(30, 20, 30, 30) # внешние отступы

self.lbl\_ogr.setText("""<html><head/><body>

<p style="line-height:30px;"><span>

<b>Ограничение для диспетчера:</b> допустимо иметь в блоках "Рейсы на посадку" и "Рейсы на взлёт" не больше 10 карточек рейсов.

</span></p>

</body></html>

""")

self.lbl\_ogr.setStyleSheet("background-color:rgb(172, 177, 202); border-radius: 10px;")

VLayout\_lbls.addWidget(self.lbl\_ogr)

self.lbl1 = QLabel()

self.lbl1.setFixedWidth(QApplication.desktop().screenGeometry().width()//2.5)

self.lbl1.setWordWrap(True) # перенос текста

self.lbl1.setFont(QtGui.QFont('Helvetica', 18)) # изменяем шрифт

self.lbl1.setContentsMargins(30, 20, 30, 30) # внешние отступы

self.lbl1.setText("""<html><head/><body>

<p style="line-height:30px;"><span>

После того, как Вы выберите пользователя и аэропорт на <b>Начальном экране</b>, откроется <b>Главный экран</b> симулятора.<br><br>

После нажатия кнопки <b>"НАЧАТЬ"</b> в блоках <b>"Рейсы на посадку"</b> и <b>"Рейсы на взлёт"</b> будут перечислены все рейсы, ожидающие назначения взлётно-посадочной полосы в реальном времени.

</span></p>

</body></html>

""")

self.lbl1.setStyleSheet("background-color:rgb(172, 177, 202); border-radius: 10px;")

VLayout\_lbls.addWidget(self.lbl1)

self.lbl2 = QLabel()

self.lbl2.setFixedWidth(QApplication.desktop().screenGeometry().width() // 2.5)

self.lbl2.setWordWrap(True) # перенос текста

self.lbl2.setFont(QtGui.QFont('Helvetica', 18)) # изменяем шрифт

self.lbl2.setContentsMargins(30, 20, 30, 30) # внешние отступы

self.lbl2.setText("""<html><head/><body>

<p style="line-height:30px;"><span>

Между двумя блоками находятся кнопки в том же количестве, что и взлётно-посадочные полосы.<br><br>

Чтобы назначить взлётно-посадочную полосу для рейса, нужно нажать на кнопку с номером полосы и карточку рейса в одном из блоков.

</span></p>

</body></html>

""")

self.lbl2.setStyleSheet("background-color:rgb(172, 177, 202); border-radius: 10px;")

VLayout\_lbls.addWidget(self.lbl2)

self.lbl3 = QLabel()

self.lbl3.setFixedWidth(QApplication.desktop().screenGeometry().width() // 2.5)

self.lbl3.setWordWrap(True) # перенос текста

self.lbl3.setFont(QtGui.QFont('Helvetica', 18)) # изменяем шрифт

self.lbl3.setContentsMargins(30, 20, 30, 30) # внешние отступы

self.lbl3.setText("""<html><head/><body>

<p style="line-height:30px;"><span>

В нижней части экрана находятся области, прикреплённые за отдельными взлётно-посадочными полосами аэропорта.<br><br>

Их количество определяется при выборе аэропорта на <b>Начальном экране</b>.<br><br>

После назначения определённой полосы, одна из областей автоматически обновится, добавив выбранный рейс.

</span></p>

</body></html>

""")

self.lbl3.setStyleSheet("background-color:rgb(172, 177, 202); border-radius: 10px;")

VLayout\_lbls.addWidget(self.lbl3)

self.lbl4 = QLabel()

self.lbl4.setFixedWidth(QApplication.desktop().screenGeometry().width() // 2.5)

self.lbl4.setWordWrap(True) # перенос текста

self.lbl4.setFont(QtGui.QFont('Helvetica', 18)) # изменяем шрифт

self.lbl4.setContentsMargins(30, 20, 30, 30) # внешние отступы

self.lbl4.setText("""<html><head/><body>

<p style="line-height:30px;"><span>

Нажмите кнопку <b>"НАЧАТЬ"</b>, чтобы приступить к работе.<br><br>

Кнопка <b>"СТОП"</b> останавливает симулятор.<br><br>

Кнопка <b>"Справка"</b> открывает окно, содержащее всю необходимую информацию по работе программы.<br><br>

Кнопка <b>"Выход"</b> полностью закрывает программу.

</span></p>

</body></html>

""")

self.lbl4.setStyleSheet("background-color:rgb(172, 177, 202); border-radius: 10px;")

VLayout\_lbls.addWidget(self.lbl4)

self.lbl5 = QLabel()

self.lbl5.setFixedWidth(QApplication.desktop().screenGeometry().width() // 2.5)

self.lbl5.setWordWrap(True) # перенос текста

self.lbl5.setFont(QtGui.QFont('Helvetica', 18)) # изменяем шрифт

self.lbl5.setContentsMargins(30, 20, 30, 30) # внешние отступы

self.lbl5.setText("""<html><head/><body>

<p style="line-height:30px;"><span>

<b>Продуктивной работы и хороших полётов!</b>

</span></p>

</body></html>

""")

self.lbl5.setStyleSheet("background-color:rgb(172, 177, 202); border-radius: 10px;")

VLayout\_lbls.addWidget(self.lbl5)

self.scrollW.setLayout(VLayout\_lbls)

self.scrollA.setWidget(self.scrollW)

VLayout\_main.addWidget(self.scrollA)

self.setLayout(VLayout\_main)

import sqlite3

def count\_all\_users():

cursor\_info.execute("SELECT id FROM users\_info ORDER BY id DESC")

return int(cursor\_info.fetchone()[0]) + 1

def count\_all\_airports():

cursor\_info.execute("SELECT id FROM airports\_info ORDER BY id DESC")

return int(cursor\_info.fetchone()[0]) + 1

def count\_all\_flights(airport\_id):

cursor\_flights.execute("SELECT id FROM airport{} ORDER BY id DESC".format(airport\_id))

return int(cursor\_flights.fetchone()[0]) + 1

def count\_all\_statistics():

cursor\_info.execute("SELECT id FROM users\_statistics ORDER BY id DESC")

return int(cursor\_info.fetchone()[0]) + 1

def getUserInfo(id, parameter):

cursor\_info.execute("SELECT {0} FROM users\_info WHERE id={1}".format(parameter, id))

return cursor\_info.fetchone()[0]

def getAirportInfo(id, parameter):

cursor\_info.execute("SELECT {0} FROM airports\_info WHERE id={1}".format(parameter, id))

return cursor\_info.fetchone()[0]

def getFlightInfo(id, airport\_id, parameter):

cursor\_flights.execute("SELECT {0} FROM airport{1} WHERE id={2}".format(parameter, airport\_id, id))

return cursor\_flights.fetchone()[0]

def setUserStatistics(id, user\_id, airport\_id, count\_users\_in\_airport, time\_spent, data\_start, data\_end, status):

cursor\_info.execute("INSERT INTO users\_statistics VALUES (?,?,?,?,?,?,?,?)",

tuple([id, user\_id, airport\_id, count\_users\_in\_airport, time\_spent, data\_start, data\_end, status]))

conn\_info.commit() # Сохраняем изменения

def setSessionsInfo(user\_id, status):

cursor\_info.execute("UPDATE users\_info SET count\_all\_sessions=count\_all\_sessions+1 WHERE id={0}".format(user\_id))

conn\_info.commit() # Сохраняем изменения

cursor\_info.execute("UPDATE users\_info SET count\_sessions\_{0}=count\_sessions\_{0}+1 WHERE id={1}".format(status, user\_id))

conn\_info.commit() # Сохраняем изменения

conn\_info = sqlite3.connect("./db/info.db", check\_same\_thread=False)

cursor\_info = conn\_info.cursor()

conn\_flights = sqlite3.connect("./db/flights.db", check\_same\_thread=False)

cursor\_flights = conn\_flights.cursor()