Руководство разработчика

программного обеспечения для просмотра статистики использования социальной сети ВКонтакте

Требования, предъявляемые к рабочей станции пользователя ПО

1. ОС – Windows 10
2. Процессор – 2.4 ГГц или больше
3. Оперативная память – 8 Гб
4. Жесткий диск – 1024 Мб свободного пространства
5. Графика – Встроенная графика
6. Дистрибутив Anaconda версии 2021.11 или интерпретатор языка Pyhon версии не ниже 3.9.7

Список использованных библиотек

Через «==» указана использованная версия библиотеки

1. matplotlib==3.5.2
2. numpy==1.22.3
3. pandas==1.4.2
4. PyQt5==5.15.4
5. pyqt5-plugins==5.15.4.2.2
6. PyQt5-Qt5==5.15.2
7. pyqt5-tools==5.15.4.3.2
8. SQLAlchemy==1.4.36

Структура файлов программы

.

├── App.exe

├── CHANGELOG

├── README.md

├── data

│ ├── old\_project.db

│ ├── project.db

│ └── raw\_data

│ └── ….csv

├── requirements.txt

└── scripts

├── App.source

├── DB\_fill.py

├── constants

│ ├── platforms.py

│ └── users.py

├── graphs

│ ├── analysis.ipynb

│ ├── graph\_avg\_median.py

│ ├── graph\_avg\_time\_by\_platform.py

│ ├── graph\_bars\_accumulated.py

│ ├── graph\_bars\_accumulated\_by\_platforms.py

│ ├── graph\_boxplot\_daily.py

│ ├── graph\_daily\_activity.py

│ ├── graph\_daily\_activity\_multiple.py

│ ├── graph\_histogram\_sessions.py

│ ├── graph\_histogram\_sessions\_platforms.py

│ ├── graph\_line\_online\_counter.py

│ └── graph\_scatter.py

├── main.py

├── models

│ ├── \_\_init\_\_.py

├── ui

│ ├── ChooseUsers.py

│ ├── ChooseUsers.ui

│ ├── Dialog1.ui

│ ├── Dialog2.ui

│ ├── Graphs.py

│ ├── Interface.ui

│ ├── TextReport1.py

│ ├── TextReport2.py

│ ├── TextReport3.py

│ ├── dialog1.py

│ ├── dialog2.py

│ ├── logo.ico

│ └── logo.png

└── utils

├── calculate\_all\_sessions.py

├── create\_platform.py

├── create\_sessions.py

├── create\_user.py

├── find\_sessions\_by\_period.py

├── find\_user\_by\_id.py

├── find\_user\_by\_name.py

├── find\_user\_sessions\_by\_period.py

├── get\_platforms.py

├── get\_users.py

├── initialize\_db.py

├── online\_checker.py

├── parse\_all\_sessions\_from\_raw.py

├── parse\_user\_sessions\_from\_json.py

├── save\_platforms.py

└── save\_users.py

Описание архитектуры программы

Весь программный код находится в папке scripts. Скрипт DB\_fill.py может применяться для заполнения базы данных, но только после консультации с разработчиками (на данный момент работает медленно и требует редактирования кода основной программы для корректной работы). В файле main.py находится код основной логики программы, а также интерфейс код реализации главного окна.

Папка scripts/constant содержит список отслеживаемых пользователей и платформ. Эти данные используются только скриптом DB\_fill.py и предназначены исключительно для первоначального заполнения базы данных информацией.

Папка scripts/graphs содержит набор из 9 файлов, реализующих все графики, содержащиеся в программном продукте. Код здесь упорядочен по одной системе: модуль принимает набор входных значений (в том числе настроенное подключение к базе данных), делает запрос к БД, обрабатывает информацию и возвращает matplotlib.Figure в качестве результата работы.

Папка scripts/models является модулем, описывающим объекты пользователя, сессии и платформы для корректной работы SQLAlchemy ORM.

Папка scripts/ui содержит интерфейсы всех окон программы (за исключением главного окна), а также программное описание процесса создания текстовых отчётов. Такое решение принято по причине глубокой интеграции отчётов в графический интерфейс.

Папка scripts/utils содержит методы для работы с базой данных, используемые при построении графиков. Инструменты устроены однотипно: на вход они получают подключение к базе данных и некоторые входные параметры, а на выходе возвращают список найденных объектов, обработанных SQLAlchemy ORM. Также в папке находятся инструменты, применяющиеся при первоначальном создании базы данных скриптом DB\_fill.py.

Листинг программного кода проекта

**main.py**

# -\*- coding: utf-8 -\*-

import sys

from datetime import datetime

sys.path.append('../')

from sqlalchemy import create\_engine

from sqlalchemy.orm import sessionmaker

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtSql import QSqlDatabase, QSqlTableModel # Работа с бд

from ui.dialog1 import Ui\_Dialog as Ui\_Dialog1

from ui.dialog2 import Ui\_Dialog as Ui\_Dialog2

from ui.TextReport1 import TextReport1

from ui.TextReport2 import TextReport2

from ui.TextReport3 import TextReport3

from ui.Graphs import GraphWidget

from graphs.graph\_scatter import graph\_scatter

from graphs.graph\_histogram\_sessions import graph\_histogram\_sessions

from graphs.graph\_histogram\_sessions\_platforms import graph\_histogram\_sessions\_platforms

from graphs.graph\_bars\_accumulated import graph\_bars\_accumulated

from graphs.graph\_bars\_accumulated\_by\_platforms import graph\_bars\_accumulated\_by\_platforms

from graphs.graph\_boxplot\_daily import graph\_boxplot\_daily

from graphs.graph\_daily\_activity\_multiple import graph\_daily\_activity\_multiple

from graphs.graph\_line\_online\_counter import graph\_line\_online\_counter

from graphs.graph\_avg\_time\_by\_platform import graph\_avg\_time\_by\_platform

from graphs.graph\_avg\_median import graph\_avg\_median

from utils.save\_platforms import save\_platforms

from utils.save\_users import save\_users

class Ui\_MainWindow(object):

'''

Класс, описывающий главное окно программы и логику его работы

Автор

-----

Илья Абрамов, Иван Чеканов, Яна Евдокимова, Егор Волков

'''

def lock\_ui(func):

'''

Декоратор, запрещающий изменение базы данных во время выбора настроек

создания графика или текстового отчёта.

Автор

-----

Иван Чеканов

'''

def wrapper(self):

self.pushButton.setDisabled(True)

self.pushButton\_4.setDisabled(True)

self.tab\_4.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.tab\_5.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.tab\_6.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

return\_value = func(self)

self.tab\_4.setEditTriggers(QtWidgets.QAbstractItemView.AllEditTriggers)

self.tab\_5.setEditTriggers(QtWidgets.QAbstractItemView.AllEditTriggers)

self.tab\_6.setEditTriggers(QtWidgets.QAbstractItemView.AllEditTriggers)

self.pushButton.setEnabled(True)

self.pushButton\_4.setEnabled(True)

return return\_value

return wrapper

def Report1(self):

'''

Метод создания текстового отчёта "Список сессий за выбранный период"

Автор

-----

Илья Абрамов

'''

self.textReport1 = TextReport1(self.db\_session, self.connection, self.ui1.selected\_users, self.ui1.dateEdit.dateTime().toPyDateTime(), self.ui1.dateEdit\_2.dateTime().toPyDateTime())

self.textReport1.resizeColumnsToContents()

self.scrollArea.setWidget(self.textReport1)

self.statusbar.showMessage("Отчёт успешно создан.")

def Report2(self):

'''

Метод создания текстового отчёта "Список активных пользователей за выбранный период"

Автор

-----

Илья Абрамов

'''

self.textReport2 = TextReport2(self.db\_session, self.ui1.dateEdit.dateTime().toPyDateTime(), self.ui1.dateEdit\_2.dateTime().toPyDateTime())

self.textReport2.resizeColumnsToContents()

self.scrollArea.setWidget(self.textReport2)

self.statusbar.showMessage("Отчёт успешно создан.")

def Report3(self):

'''

Метод создания текстового отчёта "Сводная таблица"

Автор

-----

Иван Чеканов

'''

self.textReport3 = TextReport3(self.db\_session, self.ui1.selected\_users, self.ui1.dateEdit.dateTime().toPyDateTime(), self.ui1.dateEdit\_2.dateTime().toPyDateTime())

self.scrollArea.setWidget(self.textReport3)

self.statusbar.showMessage("Отчёт успешно создан.")

def Graph(self, mode: int):

'''

Метод интеграции графического отчёта в интерфейс программы

Параметры

---------

mode : int

Номер графика для создания

Автор

-----

Илья Абрамов

'''

users = self.ui2.selected\_users

tstart = self.ui2.dateEdit.dateTime().toPyDateTime()

tend = self.ui2.dateEdit\_2.dateTime().toPyDateTime()

if mode == 1: w = GraphWidget(graph\_bars\_accumulated(self.db\_session, users, tstart, tend))

elif mode == 2: w = GraphWidget(graph\_boxplot\_daily(self.db\_session, users, tstart, tend))

elif mode == 3: w = GraphWidget(graph\_daily\_activity\_multiple(self.db\_session, users, tstart, tend))

# elif mode == 4: w = GraphWidget(graph\_histogram\_sessions(self.db\_session, tstart, tend))

elif mode == 4: w = GraphWidget(graph\_histogram\_sessions\_platforms(self.db\_session, tstart, tend))

elif mode == 5: w = GraphWidget(graph\_line\_online\_counter(self.db\_session, tstart, tend))

elif mode == 6: w = GraphWidget(graph\_scatter(self.db\_session, tstart, tend))

elif mode == 7: w = GraphWidget(graph\_avg\_time\_by\_platform(self.db\_session, tstart, tend))

elif mode == 8: w = GraphWidget(graph\_avg\_median(self.db\_session, tstart, tend))

elif mode == 9: w = GraphWidget(graph\_bars\_accumulated\_by\_platforms(self.db\_session, users, tstart, tend))

self.scrollAreaWidgetContents\_3.close()

self.scrollAreaWidgetContents\_3.deleteLater()

self.scrollAreaWidgetContents\_3 = QtWidgets.QWidget()

self.scrollArea\_2.setWidget(self.scrollAreaWidgetContents\_3)

self.scrollAreaWidgetContents\_3.setLayout(w.layout)

self.statusbar.showMessage("График успешно построен.")

@lock\_ui

def dialog1ButtonClicked(self):

'''

Метод обработки нажатия кнопки "Создать текстовый отчёт"

Автор

-----

Илья Абрамов, Иван Чеканов

'''

self.dialog1.show()

self.dialog1.exec()

if not self.dialog1.result():

return

if not self.ui1.dateEdit.dateTime() < self.ui2.dateEdit\_2.dateTime():

QtWidgets.QMessageBox().critical(self.centralwidget, "Ошибка", "Начальная дата больше конечной. Повторите попытку ввода.")

return

multiple\_buttons = [self.ui1.radioButton.isChecked(), self.ui1.radioButton\_2.isChecked()]

if any(multiple\_buttons) and not self.ui1.selected\_users:

QtWidgets.QMessageBox().critical(self.centralwidget, "Ошибка", "Не было выбрано ни одного пользователя. Повторите попытку ввода.")

return

if self.ui1.radioButton\_2.isChecked():

self.Report1() # Список сессий для выбранных пользователей

elif self.ui1.radioButton\_3.isChecked():

self.Report2() # Список активных пользоватлей

elif self.ui1.radioButton.isChecked():

self.Report3() # Количество сессий с платформ по пользователям (сводная таблица)

else:

QtWidgets.QMessageBox().critical(self.centralwidget, "Ошибка", "Не был выбран ни один графический отчёт. Повторите попытку ввода.")

return

self.tabWidget.setCurrentIndex(1)

@lock\_ui

def dialog2ButtonClicked(self):

'''

Метод обработки нажатия кнопки "Создать графический отчёт"

Автор

-----

Илья Абрамов, Иван Чеканов

'''

self.dialog2.show()

self.dialog2.exec()

if not self.dialog2.result():

return

if not self.ui2.dateEdit.dateTime() < self.ui2.dateEdit\_2.dateTime():

QtWidgets.QMessageBox().critical(self.centralwidget, "Ошибка", "Начальная дата больше конечной. Повторите попытку ввода.")

return

multiple\_buttons = [

self.ui2.radioButton\_2.isChecked(),

self.ui2.radioButton\_5.isChecked(),

self.ui2.radioButton\_6.isChecked(),

self.ui2.radioButton\_9.isChecked()

]

if any(multiple\_buttons) and not self.ui2.selected\_users:

QtWidgets.QMessageBox().critical(self.centralwidget, "Ошибка", "Не было выбрано ни одного пользователя. Повторите попытку ввода.")

return

if self.ui2.radioButton\_2.isChecked():

self.Graph(1) # Столбчатый график активности

elif self.ui2.radioButton\_5.isChecked():

self.Graph(2) # Бокс-Уискер

elif self.ui2.radioButton\_6.isChecked():

self.Graph(3) # Активность в течении одного для для пользователя

elif self.ui2.radioButton\_8.isChecked():

self.Graph(4) # Гистограмма

elif self.ui2.radioButton\_7.isChecked():

self.Graph(5) # График кол-ва пользователей в сети

elif self.ui2.radioButton.isChecked():

self.Graph(6) # Диаграмма рассеивания

elif self.ui2.radioButton\_3.isChecked():

self.Graph(7) # Сравнение среднего времени по платформам

elif self.ui2.radioButton\_4.isChecked():

self.Graph(8) # Сравнение медианной и средней длительности

elif self.ui2.radioButton\_9.isChecked():

self.Graph(9) # Столбчатый график активности с платформами

else:

QtWidgets.QMessageBox().critical(self.centralwidget, "Ошибка", "Не был выбран ни один графический отчёт. Повторите попытку ввода.")

return

self.tabWidget.setCurrentIndex(2)

def saveReport(self):

'''

Метод обработки нажатия кнопки "сохранить" для текстового отчёта

Автор

-----

Иван Чеканов

'''

if not self.scrollArea.widget():

QtWidgets.QMessageBox().critical(self.centralwidget, "Ошибка", "Чтобы сохранить что-нибудь ненужное, надо сначала сгенерировать что-нибудь ненужное. Пожалуйста, создайте отчёт, чтобы сохранить его.")

try:

if self.scrollArea.widget().saveReport(self.db\_session):

self.statusbar.showMessage("Отчёт успешно сохранён.")

else:

self.statusbar.showMessage("Не был выбран файл.")

except AttributeError:

self.statusbar.showMessage("Проблема при сохранении отчёта.")

def addRecordBtnClicked(self):

'''

Метод добавления строки в базу данных

Автор

-----

Илья Абрамов

'''

if self.tabWidget\_2.currentIndex() == 0:

r = self.model1.record()

r.setValue("name", "Username\_" + str(self.model1.rowCount() + 1))

r.setValue("vk\_id", self.model1.rowCount() + 1)

r.setValue("university\_group", None)

self.model1.insertRecord(-1, r)

self.model1.select()

elif self.tabWidget\_2.currentIndex() == 1:

r = self.model2.record()

n = self.model2.rowCount()+1

r.setValue("slug", f"SLUG\_{str(n)}")

r.setValue("description", f"DESCRIPTION\_{n}")

self.model2.insertRecord(-1, r)

self.model2.select()

elif self.tabWidget\_2.currentIndex() == 2:

r = self.model3.record()

r.setValue("user\_id", 0)

time = QtCore.QDateTime.currentDateTime()

r.setValue("session\_start", time)

r.setValue("session\_end", time)

r.setValue("platform\_id", 0)

self.model3.insertRecord(-1, r)

self.model3.select()

self.statusbar.showMessage("Строка успешно добавлена.")

def deleteSelectedRows(self):

'''

Метод удаления строки из базы данных

Автор

-----

Иван Чеканов

'''

opened\_tab\_index = self.tabWidget\_2.currentIndex()

current\_tab = [self.tab\_4, self.tab\_5, self.tab\_6][opened\_tab\_index]

current\_model = [self.model1, self.model2, self.model3][opened\_tab\_index]

indices = current\_tab.selectionModel().selectedRows()

for index in sorted(indices):

current\_model.removeRow(index.row())

n = len(indices)

if n == 0:

QtWidgets.QMessageBox().critical(self.centralwidget, "Ошибка", "Не выбрано ни одной строки.")

elif n == 1:

self.statusbar.showMessage("Успешно удалена 1 строка.")

else:

word = "строки" if n in (2,3,4) else "строк"

self.statusbar.showMessage(f"Успешно удалено {n} {word}.")

def dataBases(self, \*args):

'''

Метод получения данных из базы данных для заполнения графического интерфейса программы

Автор

-----

Илья Абрамов

'''

self.connection = QSqlDatabase.addDatabase("QSQLITE")

self.connection.setDatabaseName(self.filepath)

self.connection.open()

self.model1 = QSqlTableModel()

self.model1.setTable("user")

self.model1.setEditStrategy(QSqlTableModel.OnFieldChange)

self.model1.setHeaderData(0, QtCore.Qt.Horizontal, "id")

self.model1.setHeaderData(1, QtCore.Qt.Horizontal, "Name")

self.model1.setHeaderData(2, QtCore.Qt.Horizontal, "VK id")

self.model1.setHeaderData(3, QtCore.Qt.Horizontal, "Group")

self.model1.select()

if not self.model1.rowCount():

self.connection.close()

return False

self.tab\_4 = QtWidgets.QTableView()

self.tab\_4.setModel(self.model1)

self.tab\_4.resizeColumnsToContents()

self.tab\_4.setColumnWidth(2, 100)

self.tab\_4.setColumnWidth(3, 100)

self.model2 = QSqlTableModel()

self.model2.setTable("platform")

self.model2.setEditStrategy(QSqlTableModel.OnFieldChange)

self.model2.setHeaderData(0, QtCore.Qt.Horizontal, "id")

self.model2.setHeaderData(1, QtCore.Qt.Horizontal, "Slug")

self.model2.setHeaderData(2, QtCore.Qt.Horizontal, "Description")

self.model2.select()

self.tab\_5 = QtWidgets.QTableView()

self.tab\_5.setModel(self.model2)

self.tab\_5.resizeColumnsToContents()

self.model3 = QSqlTableModel()

self.model3.setTable("activity")

self.model3.setEditStrategy(QSqlTableModel.OnFieldChange)

self.model3.setHeaderData(0, QtCore.Qt.Horizontal, "id")

self.model3.setHeaderData(1, QtCore.Qt.Horizontal, "User id")

self.model3.setHeaderData(2, QtCore.Qt.Horizontal, "Session start")

self.model3.setHeaderData(3, QtCore.Qt.Horizontal, "Session end")

self.model3.setHeaderData(4, QtCore.Qt.Horizontal, "Platform id")

self.model3.select()

self.tab\_6 = QtWidgets.QTableView()

self.tab\_6.setModel(self.model3)

self.tab\_6.sortByColumn(0, QtCore.Qt.DescendingOrder) # Qt.DescendingOrder Qt.AscendingOrder

self.tab\_6.resizeColumnsToContents()

return True

def openDB(self, MainWindow: QtWidgets.QMainWindow):

'''

Метод инициализации баз данных QT и SQLAlchemy

Параметры

---------

MainWindow : QtWidgets.QMainWindow

Объект главного окна программы

Автор

-----

Иван Чеканов

'''

self.filepath = QtWidgets.QFileDialog().getOpenFileName(filter="База данных (\*.db \*.sqlite \*.sqlite3)")[0]

# self.filepath = "C:\\Users\\is-20\\Documents\\GitHub\\vk\_online\_analysis\\data\\project.db"

if not self.filepath:

return

if not self.dataBases(MainWindow):

QtWidgets.QMessageBox().critical(

self.centralwidget, "Ошибка", "Структура выбранной базы данных не соответствует требуемой для работы программы. Выберите другую базу данных.")

return

engine = create\_engine(f"sqlite:///{self.filepath}")

self.db\_session = sessionmaker(engine)()

# setting up the UI

self.statusbar.showMessage("Подключение к базе данных выполнено успешно.")

self.action.triggered.disconnect()

self.action.triggered.connect(lambda: QtWidgets.QMessageBox().warning(

self.centralwidget, "Предупреждение", "База данных уже выбрана. Чтобы сменить базу данных перезапустите приложение."))

self.ui2.setupUi(self.dialog2, self, self.connection)

self.ui1.setupUi(self.dialog1, self, self.connection)

self.tab\_4.setObjectName("tab\_4")

self.tabWidget\_2.addTab(self.tab\_4, "")

self.tab\_5.setObjectName("tab\_5")

self.tabWidget\_2.addTab(self.tab\_5, "")

self.tab\_6.setObjectName("tab\_6")

self.tabWidget\_2.addTab(self.tab\_6, "")

\_translate = QtCore.QCoreApplication.translate

self.tabWidget\_2.setTabText(self.tabWidget\_2.indexOf(self.tab\_4), \_translate("MainWindow", "Пользователи"))

self.tabWidget\_2.setTabText(self.tabWidget\_2.indexOf(self.tab\_5), \_translate("MainWindow", "Платформы"))

self.tabWidget\_2.setTabText(self.tabWidget\_2.indexOf(self.tab\_6), \_translate("MainWindow", "Активность"))

self.action\_7.setEnabled(True)

self.pushButton.setEnabled(True)

self.pushButton\_2.setEnabled(True)

self.pushButton\_3.setEnabled(True)

self.pushButton\_4.setEnabled(True)

self.action\_3.setEnabled(True)

self.action\_5.setEnabled(True)

self.action\_7.setEnabled(True)

self.action\_8.setEnabled(True)

self.action\_11.setEnabled(True)

self.action\_12.setEnabled(True)

def btnClickSaveUsers(self, \*args):

'''

Метод обработки нажатия кнопки "Сохранить список пользователей"

Автор

-----

Илья Абрамов, Иван Чеканов

'''

if self.connection:

save\_users(self.db\_session)

self.statusbar.showMessage("Список пользователей успешно сохранён.")

else:

QtWidgets.QMessageBox().critical(self.centralwidget, "Ошибка", "Невозможно выполнить сохранение. Сначала откройте базу данных.")

def btnClickSavePlatforms(self, \*args):

'''

Метод обработки нажатия кнопки "Сохранить список платформ"

Автор

-----

Илья Абрамов, Иван Чеканов

'''

if self.connection:

save\_platforms(self.db\_session)

self.statusbar.showMessage("Список платформ успешно сохранён.")

else:

QtWidgets.QMessageBox().critical(self.centralwidget, "Ошибка", "Невозможно выполнить сохранение. Сначала откройте базу данных.")

def setupUi(self, MainWindow: QtWidgets.QMainWindow):

'''

Метод инициализации интерфейса диалога выбора пользователей

Автор

-----

Илья Абрамов, Яна Евдокимова, Иван Чеканов

'''

self.connection = None

self.GraphLayout = None

self.dialog1 = QtWidgets.QDialog()

self.ui1 = Ui\_Dialog1()

self.dialog2 = QtWidgets.QDialog()

self.ui2 = Ui\_Dialog2()

MainWindow.setObjectName("MainWindow")

MainWindow.resize(775, 652)

MainWindow.showMaximized()

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.horizontalLayout = QtWidgets.QHBoxLayout(self.centralwidget)

self.horizontalLayout.setObjectName("horizontalLayout")

self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)

self.tabWidget.setObjectName("tabWidget")

self.tab = QtWidgets.QWidget()

self.tab.setObjectName("tab")

self.verticalLayout\_2 = QtWidgets.QVBoxLayout(self.tab)

self.verticalLayout\_2.setObjectName("verticalLayout\_2")

self.tabWidget\_2 = QtWidgets.QTabWidget(self.tab)

self.tabWidget\_2.setObjectName("tabWidget\_2")

self.verticalLayout\_2.addWidget(self.tabWidget\_2)

self.horizontalLayout\_2 = QtWidgets.QHBoxLayout()

self.pushButton = QtWidgets.QPushButton(self.tab) # Добавить новую строку

self.pushButton.setObjectName("pushButton")

self.pushButton.clicked.connect(self.addRecordBtnClicked) # Установка триггера кнопки

self.pushButton.setDisabled(True)

self.horizontalLayout\_2.addWidget(self.pushButton)

self.pushButton\_4 = QtWidgets.QPushButton(self.tab) # Удалить строки

self.pushButton\_4.setObjectName("pushButton\_4")

self.pushButton\_4.clicked.connect(self.deleteSelectedRows) # Установка триггера кнопки

self.pushButton\_4.setDisabled(True)

self.horizontalLayout\_2.addWidget(self.pushButton\_4)

self.verticalLayout\_2.addLayout(self.horizontalLayout\_2)

self.tabWidget.addTab(self.tab, "")

self.tab\_2 = QtWidgets.QWidget() # Текстовые отчёты

self.tab\_2.setObjectName("tab\_2")

self.verticalLayout\_5 = QtWidgets.QVBoxLayout(self.tab\_2)

self.verticalLayout\_5.setObjectName("verticalLayout\_5")

self.verticalLayout\_7 = QtWidgets.QVBoxLayout()

self.verticalLayout\_7.setObjectName("verticalLayout\_7")

self.scrollArea = QtWidgets.QScrollArea(self.tab\_2)

self.scrollArea.setWidgetResizable(True)

self.scrollArea.setObjectName("scrollArea")

self.verticalLayout\_7.addWidget(self.scrollArea)

self.verticalLayout\_5.addLayout(self.verticalLayout\_7)

self.pushButton\_2 = QtWidgets.QPushButton(self.tab\_2) # Сгенерировать текстовый отчёт

self.pushButton\_2.setObjectName("pushButton\_2")

self.pushButton\_2.clicked.connect(self.dialog1ButtonClicked)

self.pushButton\_2.setDisabled(True)

self.verticalLayout\_5.addWidget(self.pushButton\_2)

self.tabWidget.addTab(self.tab\_2, "")

self.tab\_3 = QtWidgets.QWidget() # Графические отчёты

self.tab\_3.setObjectName("tab\_3")

self.verticalLayout\_4 = QtWidgets.QVBoxLayout(self.tab\_3)

self.verticalLayout\_4.setObjectName("verticalLayout\_4")

self.verticalLayout\_8 = QtWidgets.QVBoxLayout()

self.verticalLayout\_8.setObjectName("verticalLayout\_8")

self.scrollArea\_2 = QtWidgets.QScrollArea(self.tab\_3)

self.scrollArea\_2.setWidgetResizable(True)

self.scrollArea\_2.setObjectName("scrollArea\_2")

self.scrollAreaWidgetContents\_3 = QtWidgets.QWidget()

self.scrollAreaWidgetContents\_3.setGeometry(QtCore.QRect(0, 0, 630, 442))

self.scrollAreaWidgetContents\_3.setObjectName("scrollAreaWidgetContents\_3")

self.scrollArea\_2.setWidget(self.scrollAreaWidgetContents\_3)

self.verticalLayout\_8.addWidget(self.scrollArea\_2)

self.verticalLayout\_4.addLayout(self.verticalLayout\_8)

self.pushButton\_3 = QtWidgets.QPushButton(self.tab\_3) # Построить график

self.pushButton\_3.setObjectName("pushButton\_3")

self.pushButton\_3.clicked.connect(self.dialog2ButtonClicked)

self.pushButton\_3.setDisabled(True)

self.verticalLayout\_4.addWidget(self.pushButton\_3)

self.tabWidget.addTab(self.tab\_3, "")

self.horizontalLayout.addWidget(self.tabWidget)

MainWindow.setCentralWidget(self.centralwidget)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 775, 26))

self.menubar.setObjectName("menubar")

self.menu = QtWidgets.QMenu(self.menubar)

self.menu.setObjectName("menu")

self.menu\_2 = QtWidgets.QMenu(self.menubar)

self.menu\_2.setObjectName("menu\_2")

self.menu\_3 = QtWidgets.QMenu(self.menubar)

self.menu\_3.setObjectName("menu\_3")

MainWindow.setMenuBar(self.menubar)

self.action = QtWidgets.QAction(MainWindow)

self.action.setObjectName("action")

self.action.triggered.connect(self.openDB)

self.action\_3 = QtWidgets.QAction(MainWindow)

self.action\_3.setObjectName("action\_3")

self.action\_3.setDisabled(True)

self.action\_3.triggered.connect(self.dialog1ButtonClicked)

self.action\_5 = QtWidgets.QAction(MainWindow)

self.action\_5.setObjectName("action\_5") # Сохранить отчёт

self.action\_5.triggered.connect(self.saveReport)

self.action\_5.setDisabled(True)

self.action\_7 = QtWidgets.QAction(MainWindow)

self.action\_7.setObjectName("action\_7")

self.action\_7.setDisabled(True)

self.action\_7.triggered.connect(lambda: self.statusbar.showMessage("База данных успешно сохранена."))

self.action\_8 = QtWidgets.QAction(MainWindow)

self.action\_8.setObjectName("action\_8")

self.action\_8.setDisabled(True)

self.action\_8.triggered.connect(self.dialog2ButtonClicked)

self.action\_11 = QtWidgets.QAction(MainWindow)

self.action\_11.setObjectName("action\_11")

self.action\_11.triggered.connect(self.btnClickSaveUsers)

self.action\_11.setDisabled(True)

self.action\_12 = QtWidgets.QAction(MainWindow)

self.action\_12.setObjectName("action\_12")

self.action\_12.triggered.connect(self.btnClickSavePlatforms)

self.action\_12.setDisabled(True)

self.menu.addAction(self.action)

self.menu.addSeparator()

self.menu.addAction(self.action\_7)

self.menu.addAction(self.action\_11)

self.menu.addAction(self.action\_12)

self.menu\_2.addAction(self.action\_3)

self.menu\_2.addSeparator()

self.menu\_2.addAction(self.action\_5)

self.menu\_3.addAction(self.action\_8)

self.menu\_3.addSeparator()

self.menubar.addAction(self.menu.menuAction())

self.menubar.addAction(self.menu\_2.menuAction())

self.menubar.addAction(self.menu\_3.menuAction())

self.retranslateUi(MainWindow)

self.tabWidget.setCurrentIndex(0)

self.tabWidget\_2.setCurrentIndex(0)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

'''

Метод добавления подписей к элементам интерфейса

Автор

-----

Яна Евдокимова

'''

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowIcon(QtGui.QIcon('ui/logo.png'))

MainWindow.setWindowTitle(\_translate("MainWindow", "Анализ активности ВК"))

self.pushButton.setText(\_translate("MainWindow", "Добавить новую строку"))

self.pushButton\_4.setText(\_translate("MainWindow", "Удалить выбранные строки"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), \_translate("MainWindow", "Просмотр/редактирование БД"))

self.pushButton\_2.setText(\_translate("MainWindow", "Сгенерировать"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_2), \_translate("MainWindow", "Текстовые отчёты"))

self.pushButton\_3.setText(\_translate("MainWindow", "Построить"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_3), \_translate("MainWindow", "Графические отчёты"))

self.menu.setTitle(\_translate("MainWindow", "Файл"))

self.menu\_2.setTitle(\_translate("MainWindow", "Отчёт"))

self.menu\_3.setTitle(\_translate("MainWindow", "График"))

self.action.setText(\_translate("MainWindow", "Открыть базу данных"))

self.action\_3.setText(\_translate("MainWindow", "Сгенерировать"))

self.action\_5.setText(\_translate("MainWindow", "Сохранить отчёт"))

self.action\_7.setText(\_translate("MainWindow", "Сохранить базу данных"))

self.action\_8.setText(\_translate("MainWindow", "Построить"))

self.action\_11.setText(\_translate("MainWindow", "Сохранить список пользователей в файл"))

self.action\_12.setText(\_translate("MainWindow", "Сохранить список платформ в файл"))

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

MainWindow = QtWidgets.QMainWindow()

ui = Ui\_MainWindow()

ui.setupUi(MainWindow)

MainWindow.show()

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