**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ**

**Кафедра веб-технологий и компьютерного моделирования**

**СОЗДАНИЕ АГРЕГАТОРА НОВОСТЕЙ НА PYTHON**

Курсовая работа

Мелешко Романа Дмитриевича

студента 3 курса,

специальность «научно-педагогическая

деятельность»

Научный руководитель:

Доцент, кандидат физико-математических наук,

Кремень Е.В.

Минск, 2020

Оглавление

[Введение 3](#_Toc41861450)

[Глава 2. Обзор используемых методов и технологий 5](#_Toc41861451)

[1.1 Описание предметной области 5](#_Toc41861452)

[1.2 Модель предметной области 5](#_Toc41861453)

[1.3 Python 5](#_Toc41861454)

[1.4 PyQt5 7](#_Toc41861455)

[1.5 RSS 8](#_Toc41861456)

[Глава 2. Реализация агрегатора 10](#_Toc41861457)

[3.1 Парсинг данных 10](#_Toc41861458)

[3.2 GUI 12](#_Toc41861459)

[3.3 связь виджетов и функций 13](#_Toc41861460)

[3.4 Результат выполнения 16](#_Toc41861461)

[Заключение 18](#_Toc41861462)

[Список использованных источников 19](#_Toc41861463)

[Приложение1 20](#_Toc41861464)

[Приложение 2 24](#_Toc41861465)

Введение

В современном мире самым используемым источником для получения информации является интернет. Люди из практически любой точки Земли могут получить неограниченный доступ к контенту, собранному со всего мира и представленному в удобном формате. Почти все печатные издания, многие из которых существуют десятки лет, имеют свои цифровые версии.

С развитием информационной среды и увеличением публикуемых новостей количество всевозможных интернет порталов растет. Человек, порой, не в состоянии самостоятельно отслеживать все интересующие его темы из различных источников, по причине их большого количества. Для упрощения поиска, структуризации и ускорении потребления интересующей информации были созданы новостные агрегаторы.

Агрегатор (от латинского aggregation(накопление)) —тот, кто собирает и группирует объекты в категории более высокого уровня.

Новостной агрегатор — программа, сканирующая в автоматическом режиме новостные сайты и с помощью семантических ухищрений создающая роботизированную ленту новостей.   
Самый известный новостной агрегатор рунета — Яндекс. Новости. Новостные агрегаторы, как правило, не пишут заголовки самостоятельно, но используют заголовки самых влиятельных СМИ на своих лентах.   
Правительства периодически пытаются приравнять новостные агрегаторы к СМИ, хотя те, по сути, никакими СМИ не являются, а всего лишь собирают удобный дайджест из ежедневного новостного потока.

RSS-агрегатор — клиентская программа или веб-приложение для автоматического сбора сообщений из источников, экспортирующих в формат RSS, например, заголовки новостей, блогов, подкастов и видеохостингов.RSS-агрегатор удобен для личного использования, можно настроить его под свои нужды и получать необходимую информацию с интернет-порталов.

RSS — семейство XML-форматов, предназначенных для описания лент новостей, анонсов статей, изменений в блогах и т. п. Информация из различных источников, представленная в формате RSS, может быть собрана, обработана и представлена пользователю в удобном для него виде специальными программами-агрегаторами.

Основная цель курсовой работы —создать RSS-агрегатор на языке Python.На данный момент новостные агрегаторы очень востребованы. Ярким примером служат такие проекты, как: Flipboard, Яндекс. Новости, Feedly, Squid, Google Новости, News Tab и многие другие. Этими сервисами ежедневно пользуются миллионы людей по всему миру. Это доказывает то, что тема курсовой работы актуальна.

Отсюда вытекают задачи курсовой работы:

* Выбрать интернет-порталы для получения информации;
* Изучить и научиться использовать доступные библиотеки Python для достижения поставленной цели;
* Создать простое в использовании десктопное приложение.

Глава 2. Обзор используемых методов и технологий

1.1 Описание предметной области

Предметной областью является изучение библиотек Python, RSS-формата, QtDesigner для создания GUI. На примере создания агрегатора все это можно изучить и применять на практике

1.2 Модель предметной области

Десктопное приложение должно позволять пользователю выбирать интересующие его сайты, также выбирать категорию(и) получаемой с них информации, иметь понятный и простой интерфейс.

1.3 Python

В качестве языка программирования был выбран Python.

Отличительные характеристики языка:

* очень низкий порог вхождения, уже после одного дня изучения можно начать писать простые программы;
* минималистичный язык, с небольшим количеством конструкций;
* краткий код;
* очень хорошая поддержка математических вычислений (библиотеки NumPy, SciPy);
* большое количество развитых web-фреймворков (Django, TurboGear, CherryPy, Flask);
* Отличается высокой скоростью выполнения и разработки, поддерживает огромное количество как встроенных, так и подключаемых библиотек для всевозможных задач.

Универсальная природа языка обеспечивает возможность его применения в самых разных областях. Фактически с определенной долей уверенности можно утверждать, что Python так или иначе используется практически каждой достаточно крупной организацией, занимающейся разработкой программного обеспечения, - как для решения краткосрочных тактических задач, так и для разработки долгосрочных стратегических проектов.

Для выполнения более узких задач в состав Python также входит большая коллекция библиотечных инструментов, которые поддерживают практически все, что только может потребоваться, - от поиска с использованием регулярных выражений до работы в сети. Библиотечные инструменты языка Python — это то место, где выполняется большая часть операций [1].

Средой разработки был выбран PyCharm—интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django. PyCharm разработана компанией JetBrainsна основе IntelliJ IDEA. Возможности:

 Статический анализ кода, подсветка синтаксиса и ошибок.

 Навигация по проекту и исходному коду: отображение файловой структуры проекта, быстрый переход между файлами, классами, методами и использованиями методов.

 Рефакторинг: переименование, извлечение метода, введение переменной, введение константы, подъём и спуск метода и т. д.

 Инструменты для веб-разработки с использованием фреймворка Django

 Встроенный отладчик для Python

 Встроенные инструменты для юнит-тестирования

 Разработка с использованием Google App Engine [2]

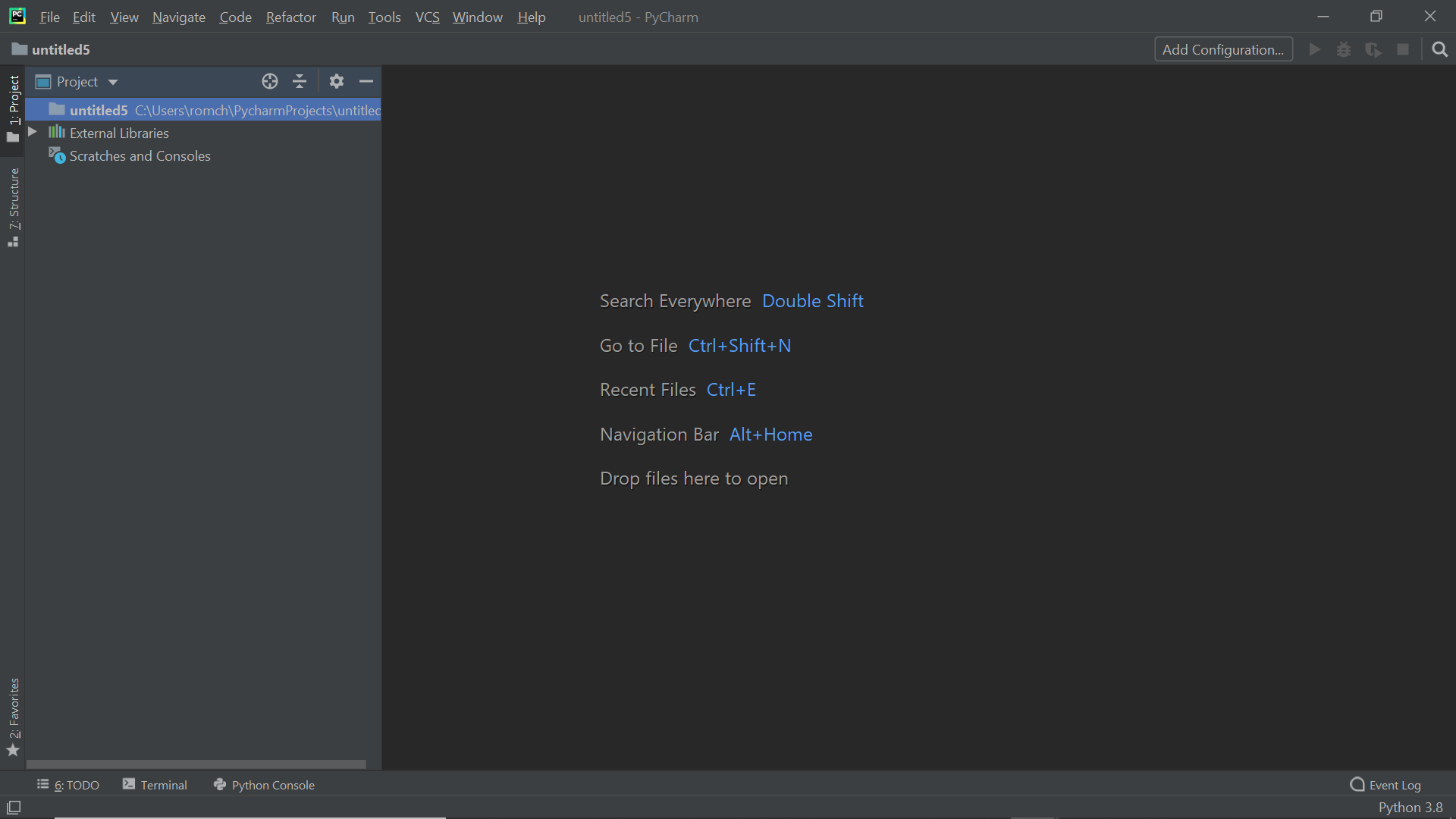


Рисунок 1.PyCharm

1.4 PyQt5

PyQt — набор расширений графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python. P**yQt5** является одним из наиболее часто используемых модулей для создания **GUI приложений** в Python, и это связанно с его простотой

PyQt разработан британской компанией Riverbank Computing. PyQt работает на всех платформах, поддерживаемых Qt: Linux и другие UNIX-подобные ОС, Mac OS X и Windows.

PyQt также включает в себя QtDesigner (QtCreator) — дизайнер графического интерфейса пользователя. Программа pyuic генерирует Python код из файлов, созданных в QtDesigner. Это делает PyQt очень полезным инструментом для быстрого прототипирования. Кроме того, можно добавлять новые графические элементы управления, написанные на Python, в QtDesigner[3].

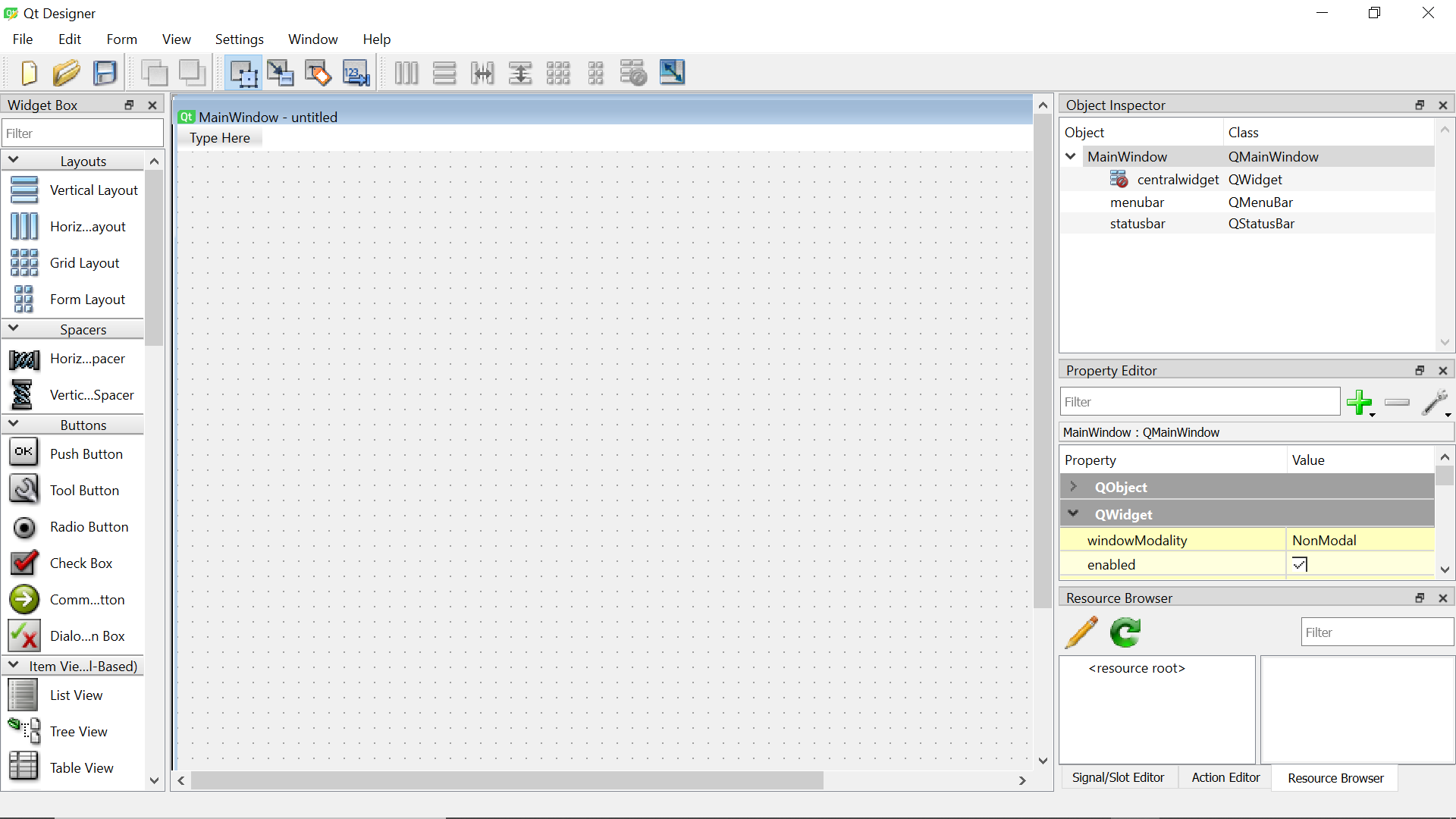


Рисунок 2.Рабочая область QtDesigner

1.5 RSS

RSS – это специальный файл в формате (rss или xml), который используется для описания новостей сайтов или их анонсов со ссылкой на полную версию текста RSS ленты сегодня одна из лучших возможностей держать руку на пульсе событий[4]. Не нужно пересматривать десятки ресурсов, чтобы найти актуальную информацию, достаточно подписаться на RSS каналы и получить сортированные по категориям новости.

Данный формат отлично подходит для инфопорталов, где нет высокой вовлеченности аудитории в материал. То есть новости не комментируют, не лайкают, не делятся в соцсетях. RSS поток – это ознакомление с информацией. Что с ней делать дальше решает сам пользователь: идти на блог и как-то реагировать на статью или просто принимать данные к сведению. В 90% случаев формат RSS успешно используется порталами, блогами (в качестве анонсирования новых материалов), библиотечными ресурсами, досками объявлений.

Глава 2. Реализация агрегатора

3.1 Парсинг данных

Сперва создадим проект в PyCharm. Подключим необходимые библиотеки.

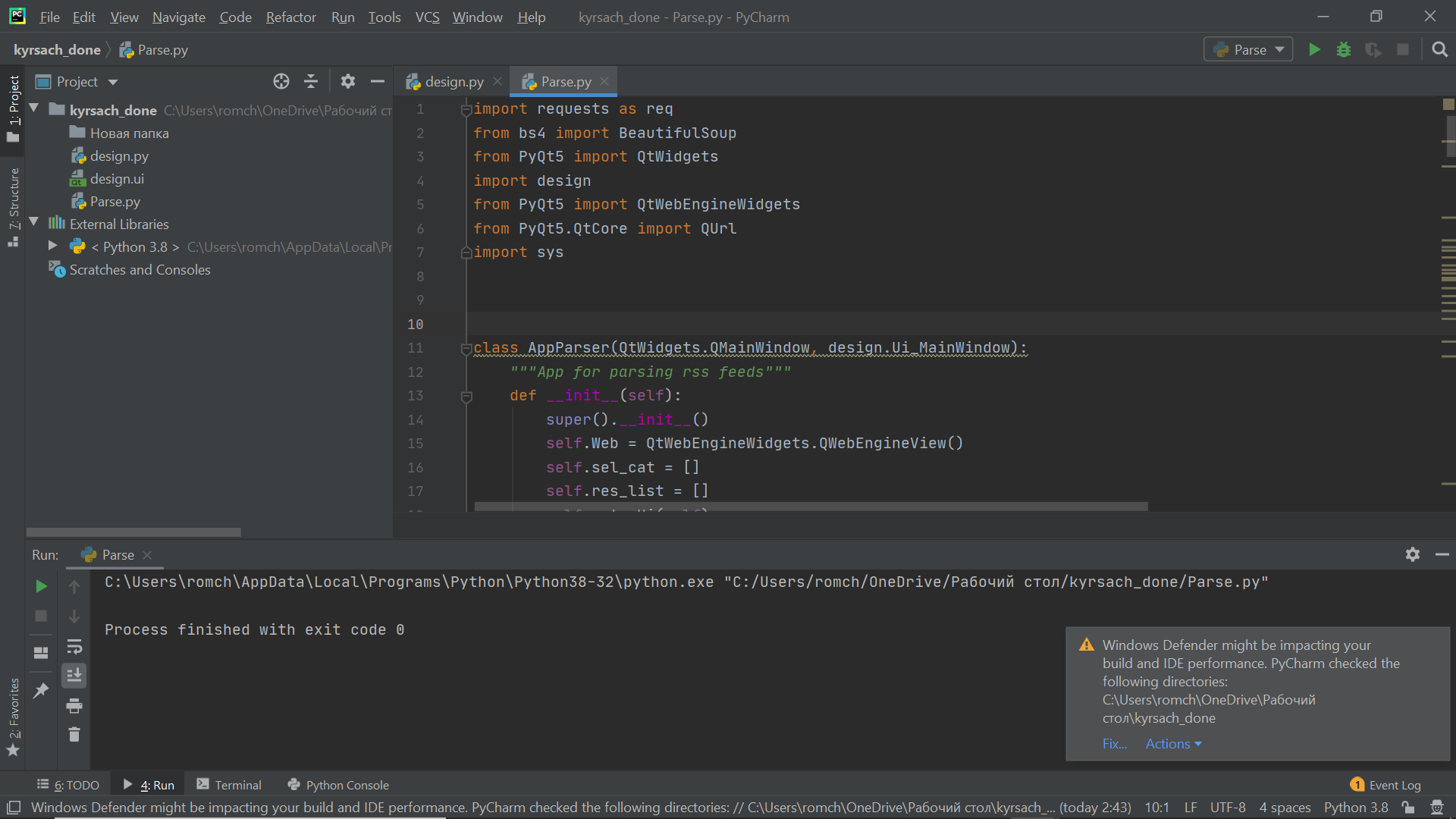


Рисунок 3

Библиотека requests служит для выполнения http запросов. Beautiful Soup - это парсер для синтаксического разбора файлов HTML/XML, написанный на языке программирования Python, который может преобразовать даже неправильную разметку в дерево синтаксического разбора. Он поддерживает простые и естественные способы навигации, поиска и модификации дерева синтаксического разбора [5]. Также импортируем библиотеки для работы со встроенным браузеров, отображением виджетов. Виджеты в Qt5 — это «строительные блоки» для создания пользовательского интерфейса. Визуальные объекты на форме, такие как кнопки, метки, поля, меню, раскрывающиеся списки и т.д. — всё это относится к виджетам. Модуль design напишем позже. В нем содержится информация о графической оболочке будущего приложения.

Далее создадим класс AppParser.В нем описываются все функции.

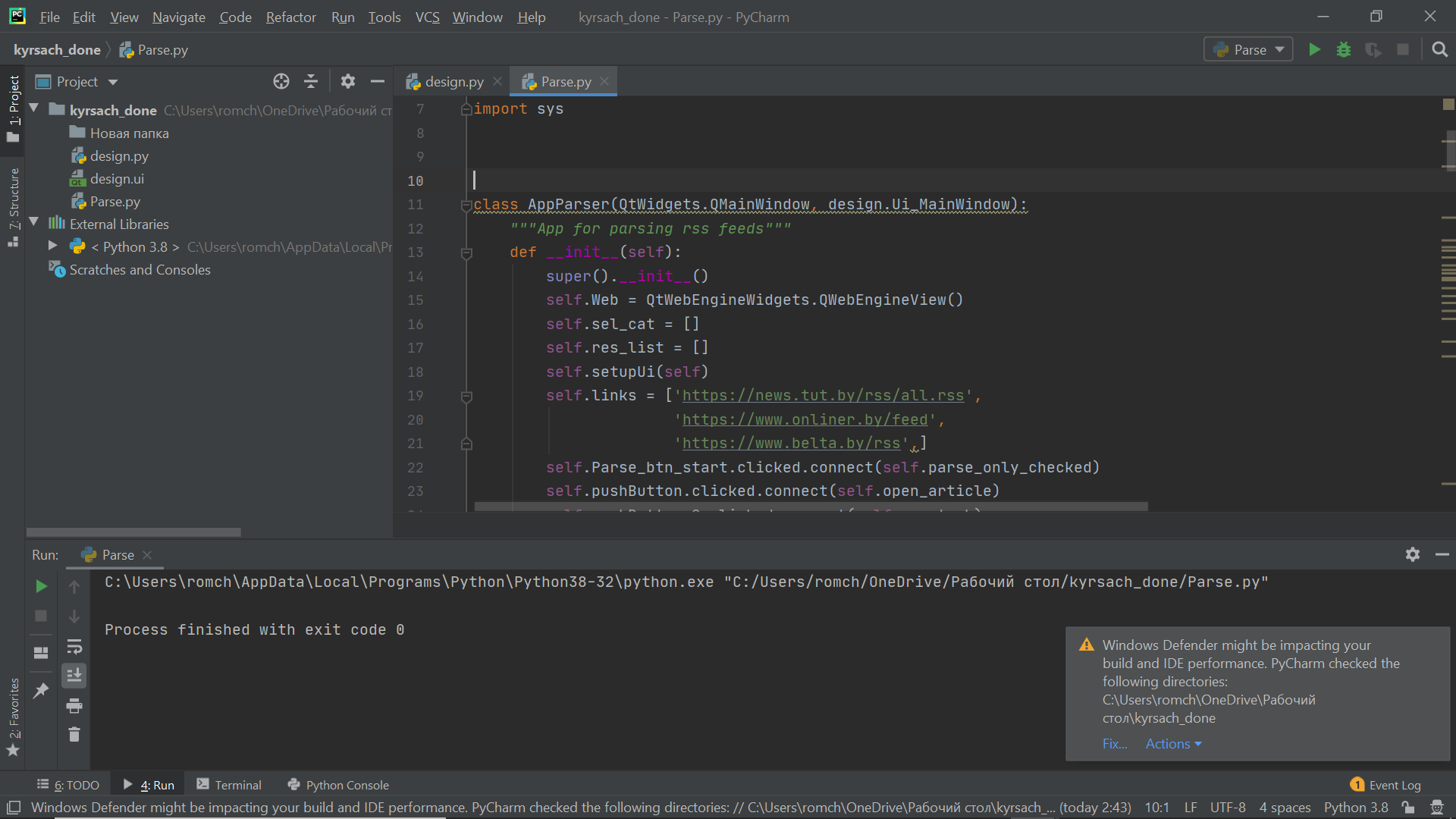


Рисунок 4

В качестве источников информации возьмем 3 наиболее популярные новостные порталы РБ. Tut.by, Onliner, Belta. Все они имеют RSS файл. Создадим список из ссылок на эти ресурсы. Списки для категорий и данных, которые будем парсить.

Теперь опишем функцию parser.

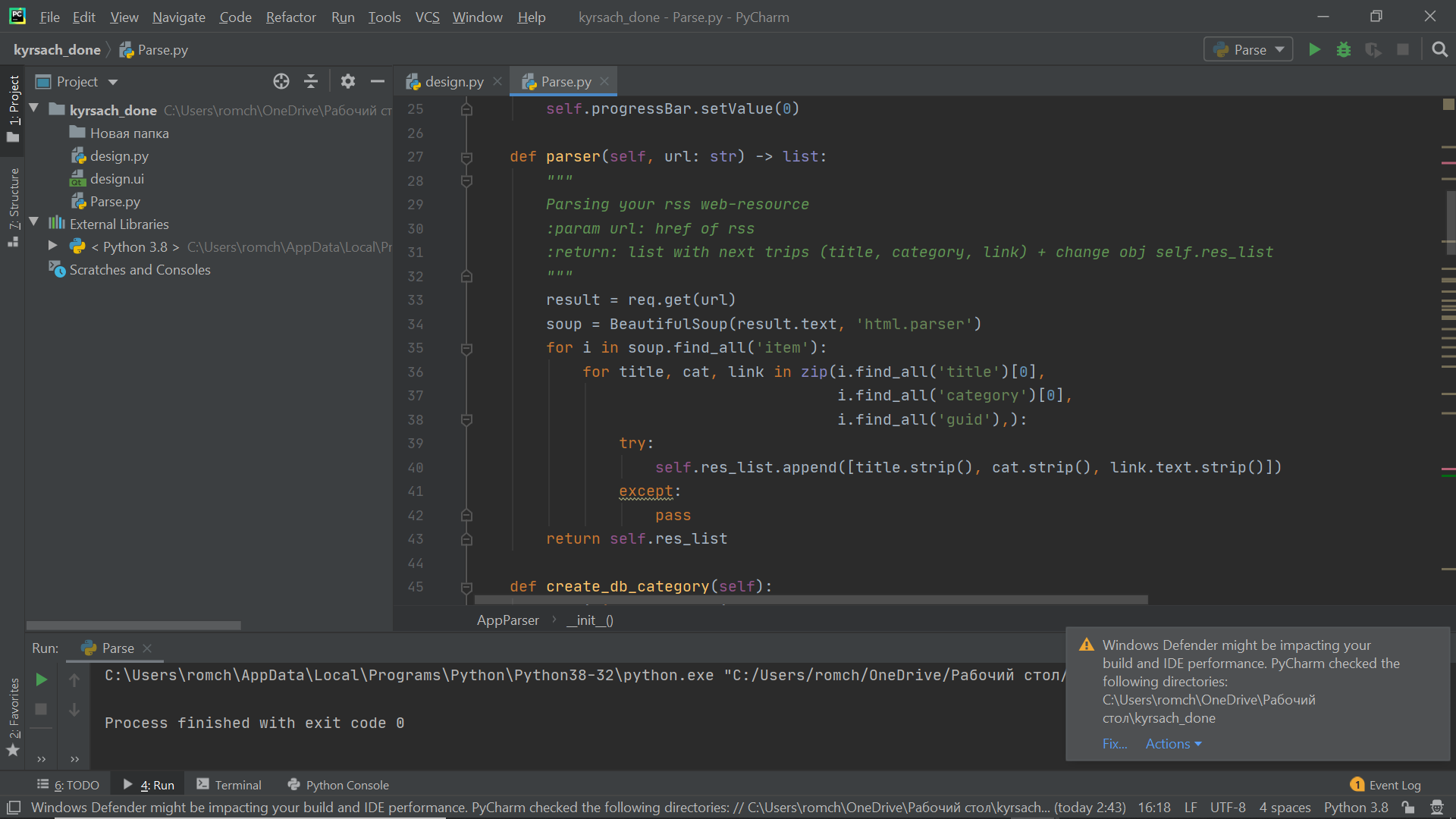


Рисунок 5

Эта функции с помощью библиотеки BS и Request посылает запрос на сайт. Изучив структуру RSS файла, получаем необходимые нам данные. А именно: из title получаем заголовок новости, category позволит нам сгруппировать новости по категориям, а из guid вытаскиваем ссылку на источник. Всю информацию помещаем в список res\_list.

3.2 GUI

Теперь откроем QtDesigner и создадим графический интерфейс.

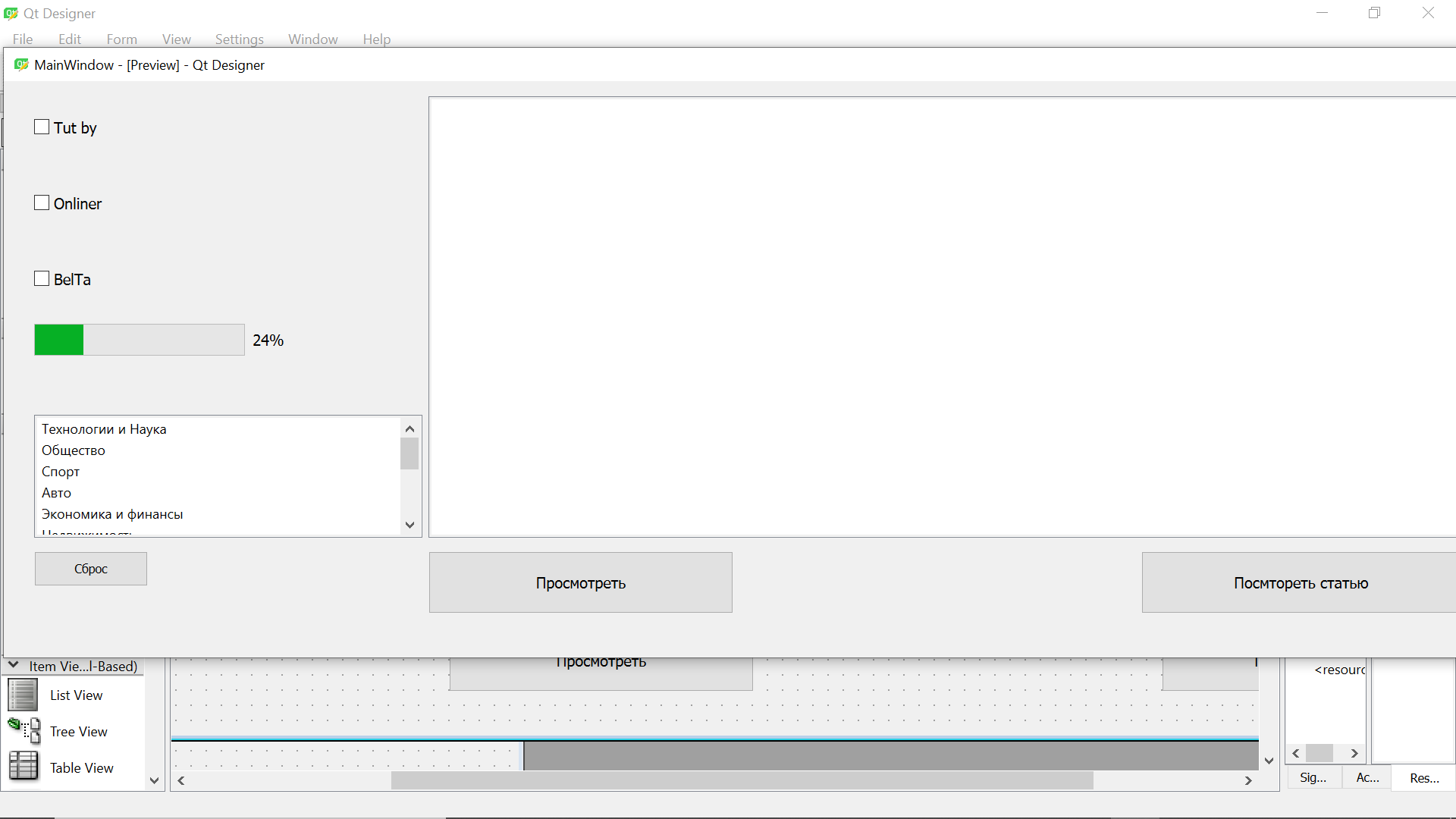


Рисунок 6

Добавим 3 check\_box для возможности выбора сайтов, LabelWidget с полосой прокрутки для категорий новостей, которые хотим получить, окно вывода всех заголовков новостей, удовлетворяющих выбранным параметрам, кнопку просмотреть для старта парсинга, посмотреть статью для открытия во встроенном браузере интересующей статьи, сброс для удобного выбора категорий заново и ProgressBar для слежения за ходом выполнения парсинга.

Файл созданный в QtDesigner сохраняется с расширением .ui. Для того чтобы конвертировать этот файл в воспринимаемый python-ом .py необходимо открыть терминал из папки где находится этот файл и выполнить следующую команду:

python -m PyQt5.uic.pyuic -x [FILENAME].ui -o [FILENAME].py [6]

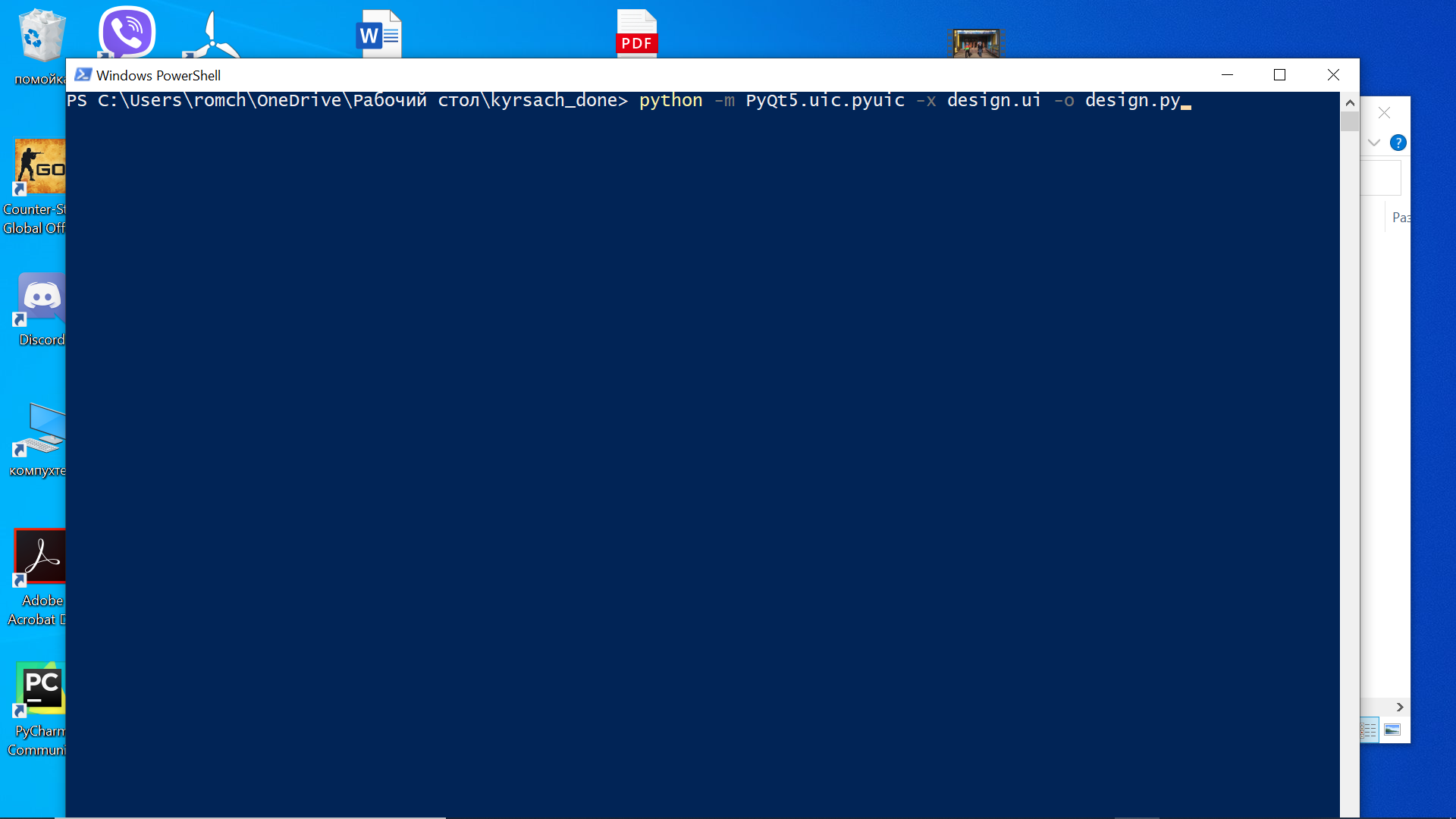


Рисунок 7

В результате мы получим .py файл, который мы может использовать в своем коде.

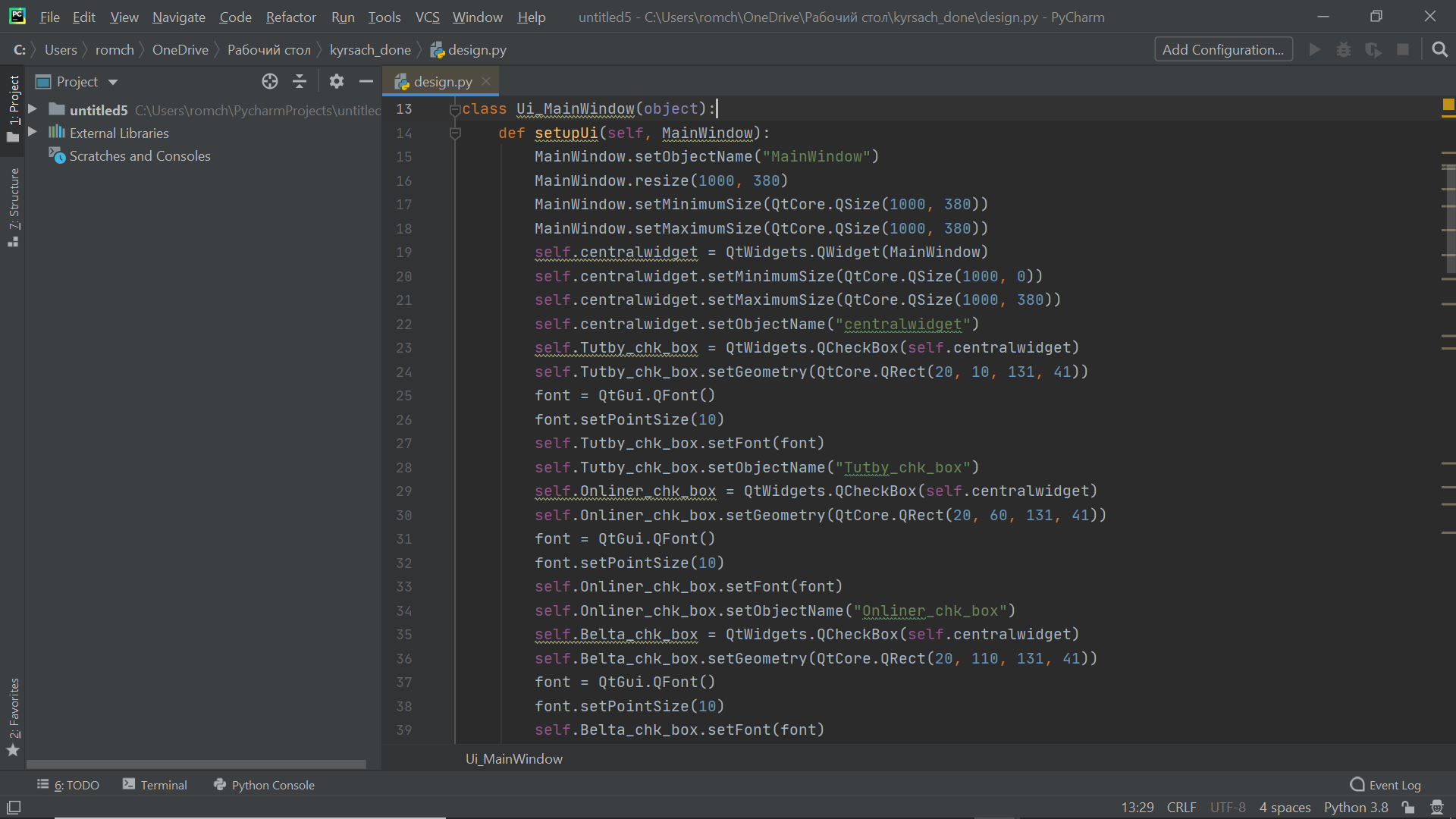


Рисунок 8

Получили все виджеты, которые добавляли и редактировали в QtDesigner.

3.3 связь виджетов и функций

Теперь разобьем полученные категории в общие группы, по ключевым словам. Функцией create\_db\_category. Всего выделим 12 различных групп новостей.

Вернемся к check\_box.

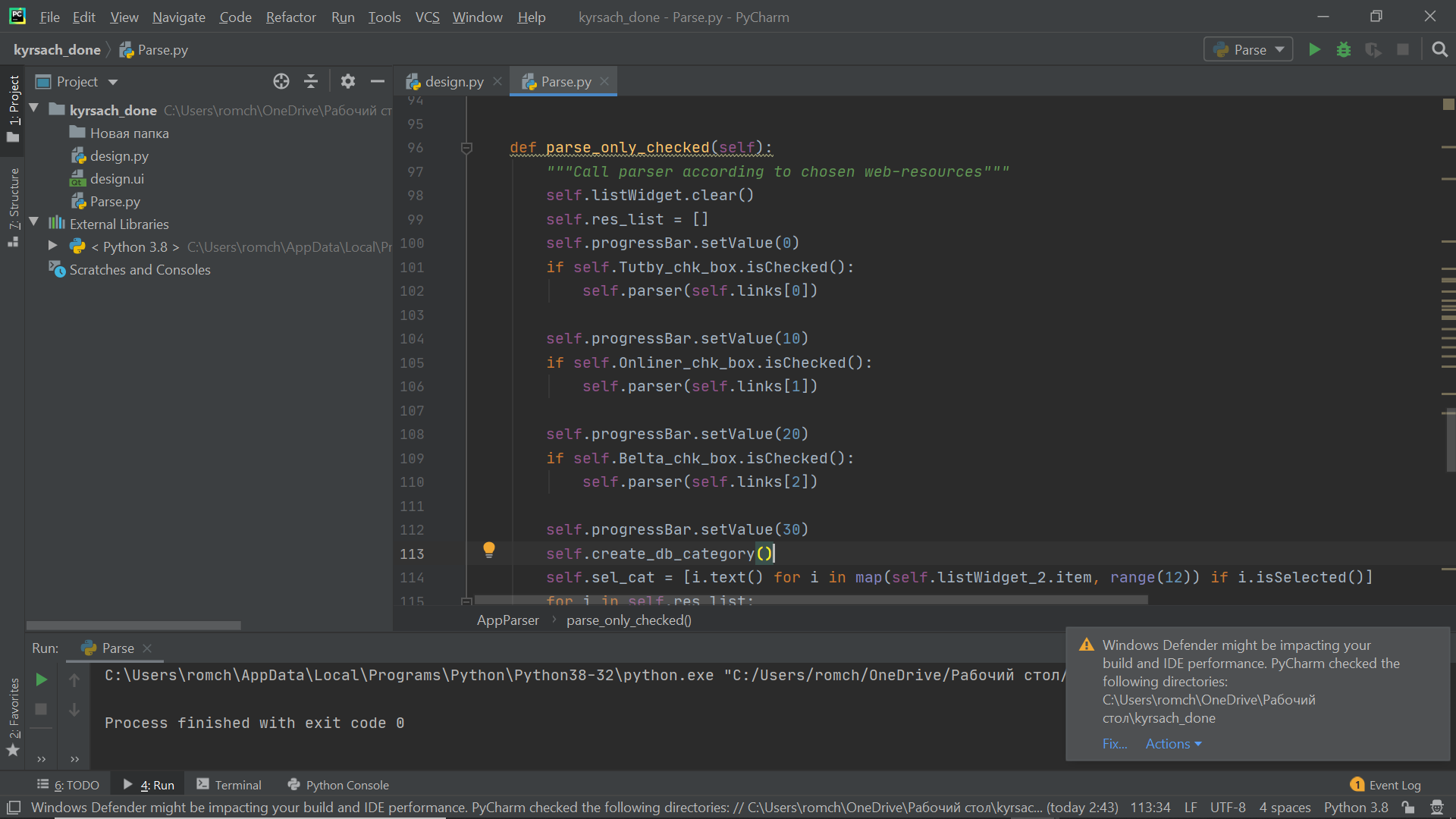


Рисунок 9

Данная функция (рисунок 7) проверяет, какие из чек-боксов имеют статус True. Запросы на сайт будут отправлены если пользователь выбрал данный сайт. Ссылки на сайты хранятся в links[].

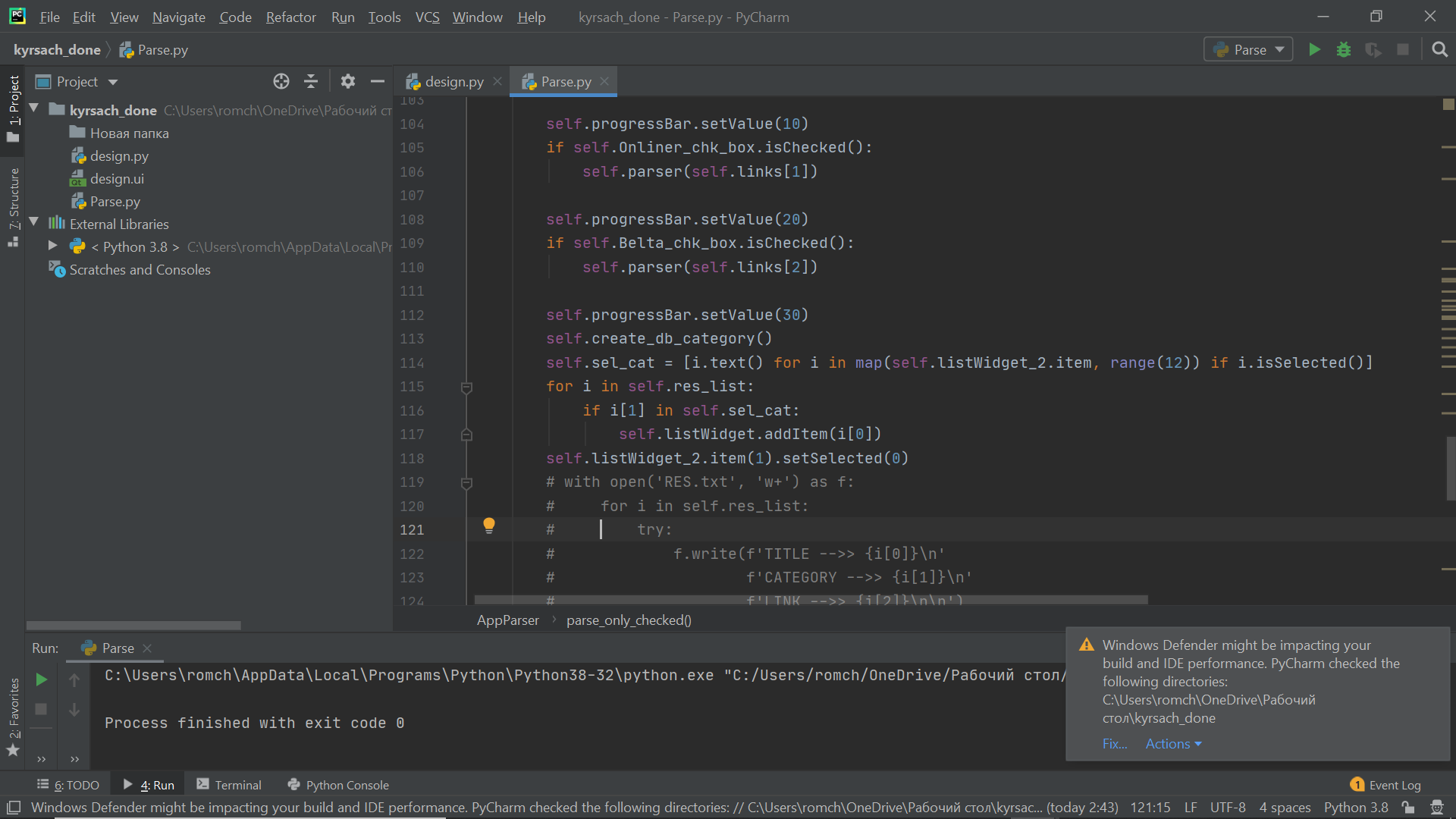


Рисунок 10

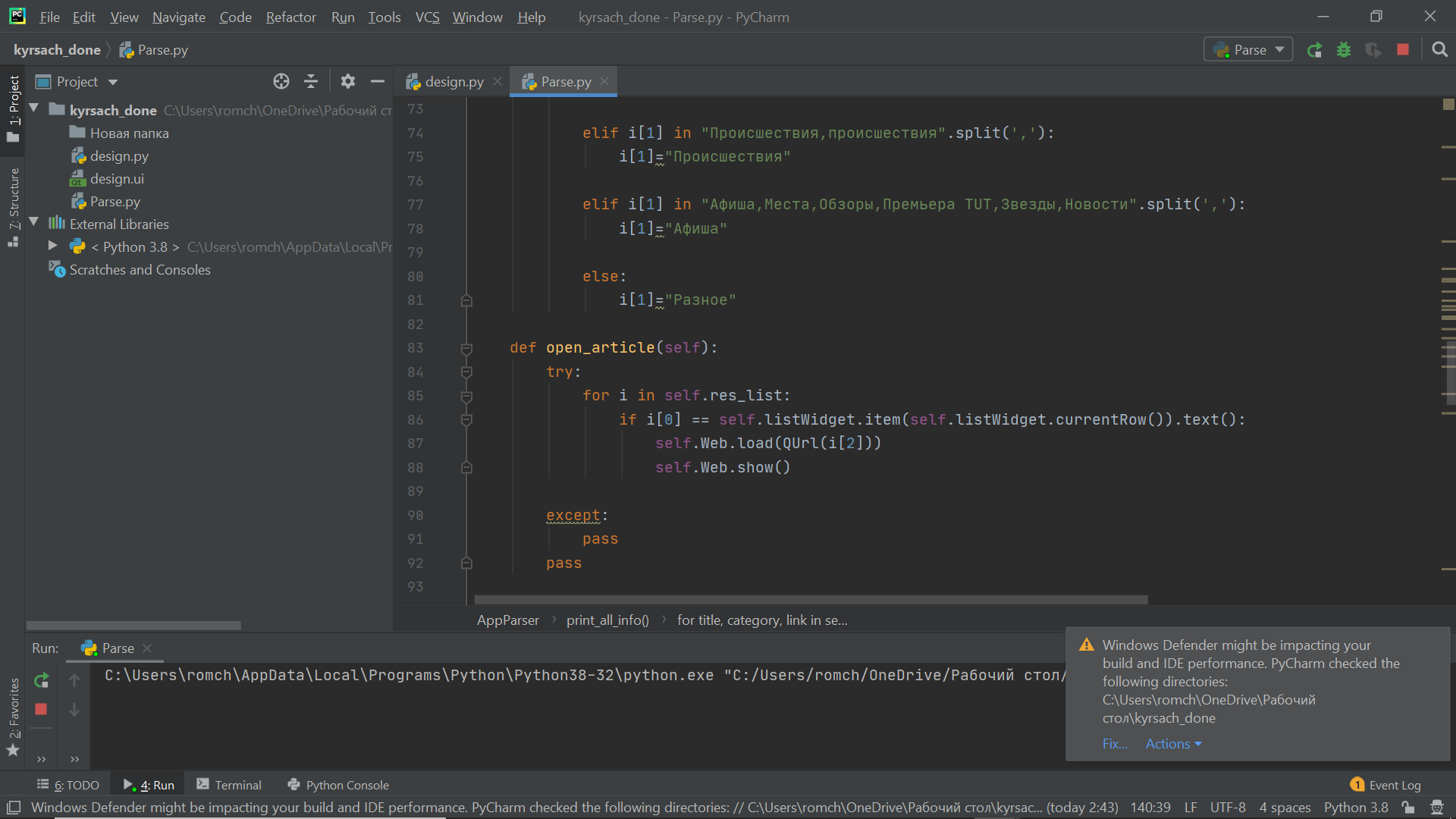
Поскольку нам не просто нужно получить новости с сайта, а с учетом выбранных категорий, то создадим цикл, который проверяет, есть ли выбранная категория в res\_list. В итоге мы получаем новости с выбранного сайта(ов) определенных категорий с выводов в виджет ListWidget\_2.

После того, как выбор сделан, все заголовки отображаются ListWidget\_2



Рисунок 11

Теперь при просмотре заголовков можно выбрать любой и нажать просмотреть статью.



Функция open\_article сравнивает текущую выбранную строку с i[0] в res\_list ( там хранятся заголовки). Если они равны, то соответствующая ссылка i[2] открывается во встроенном браузере[7]. Браузер открывается в новом окне, после его закрытия можно вернуться обратно к списку заголовком, или задать новые условия отображения и получить другие заголовки.

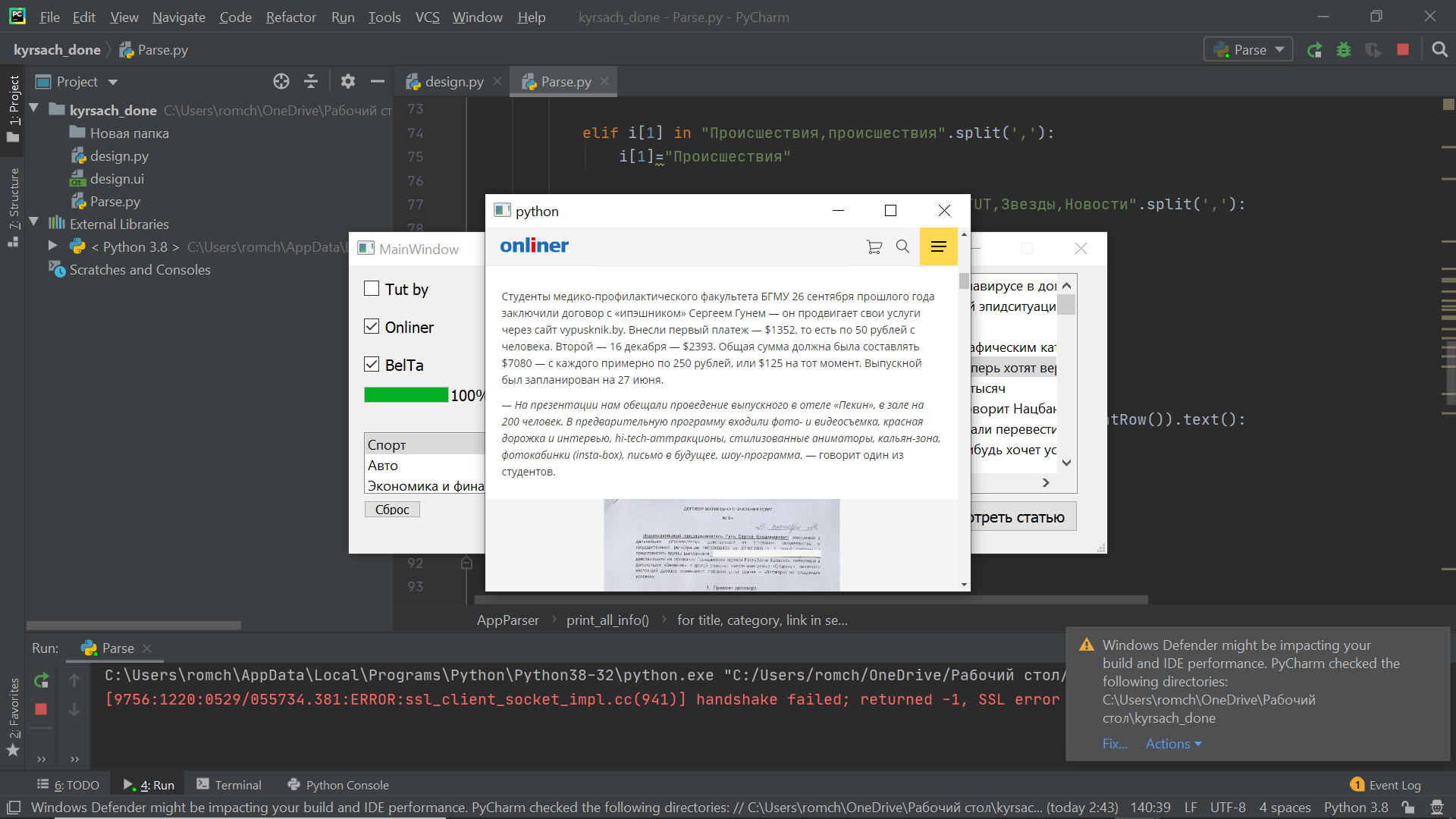
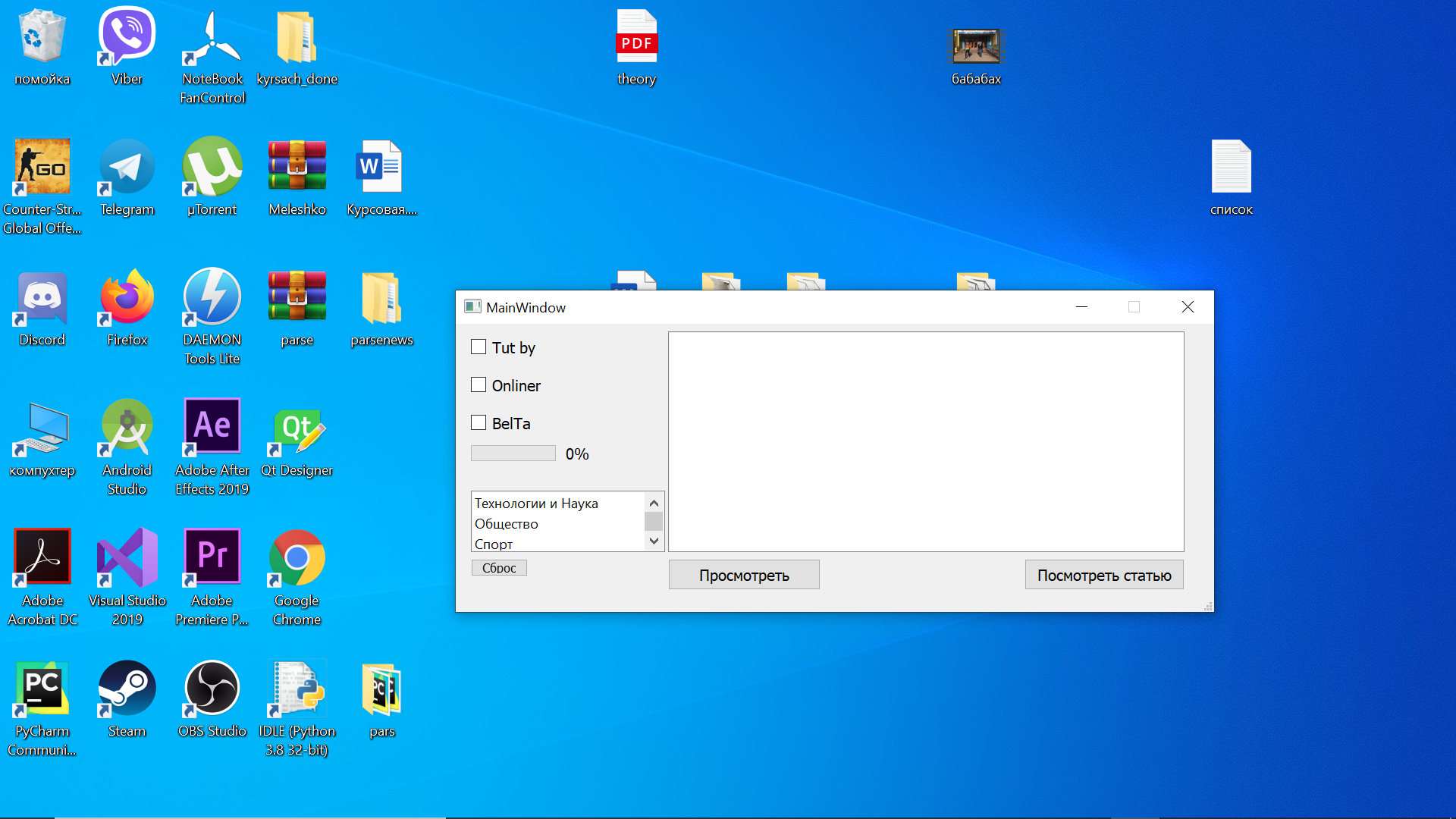
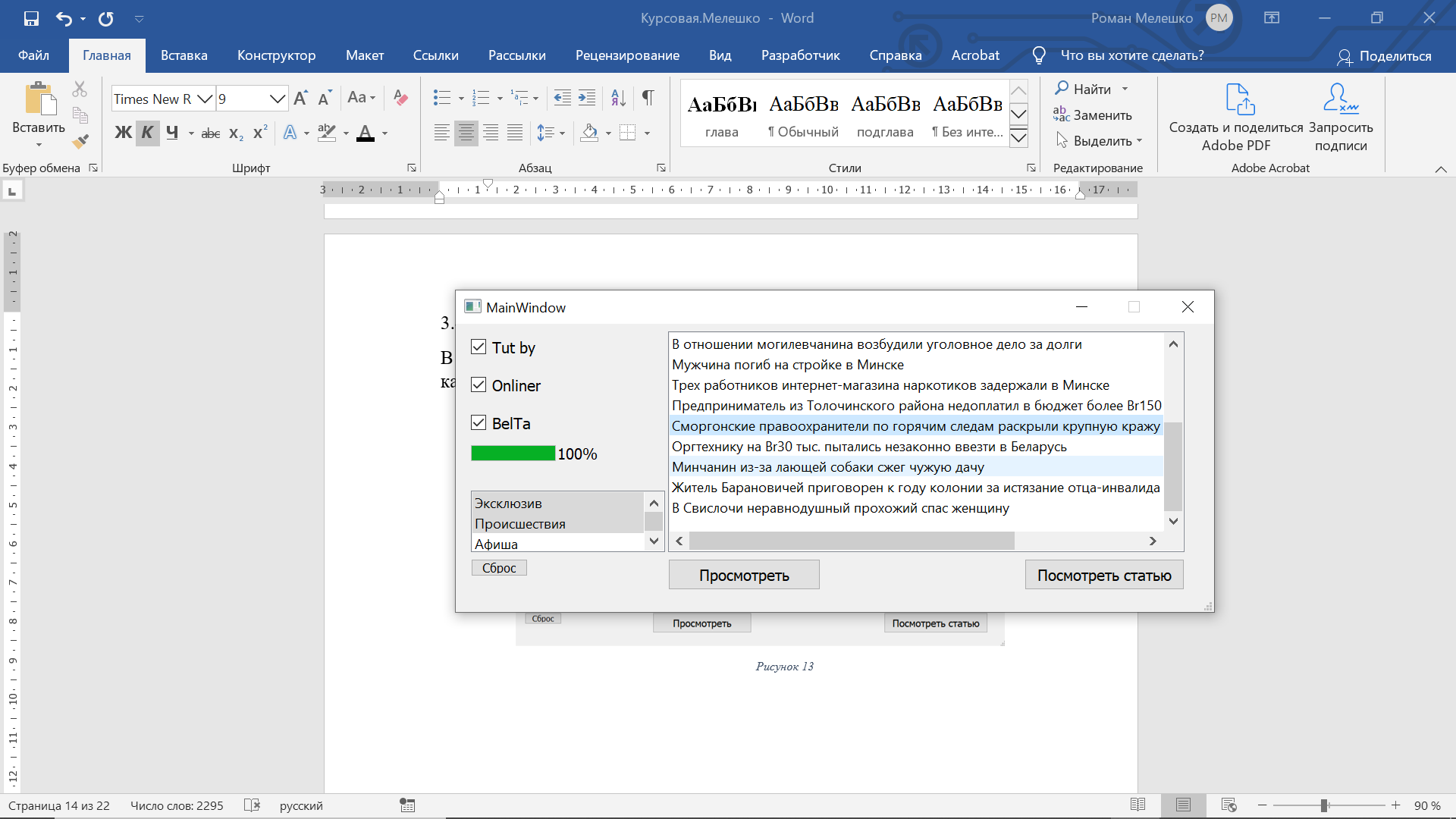


Рисунок 12

3.4 Результат выполнения

В итоге получили приложение, которое с определенных сайтов по заданным категориям выдает заголовки новостей, которые можно открыть.



Рисунок 14

Рисунок

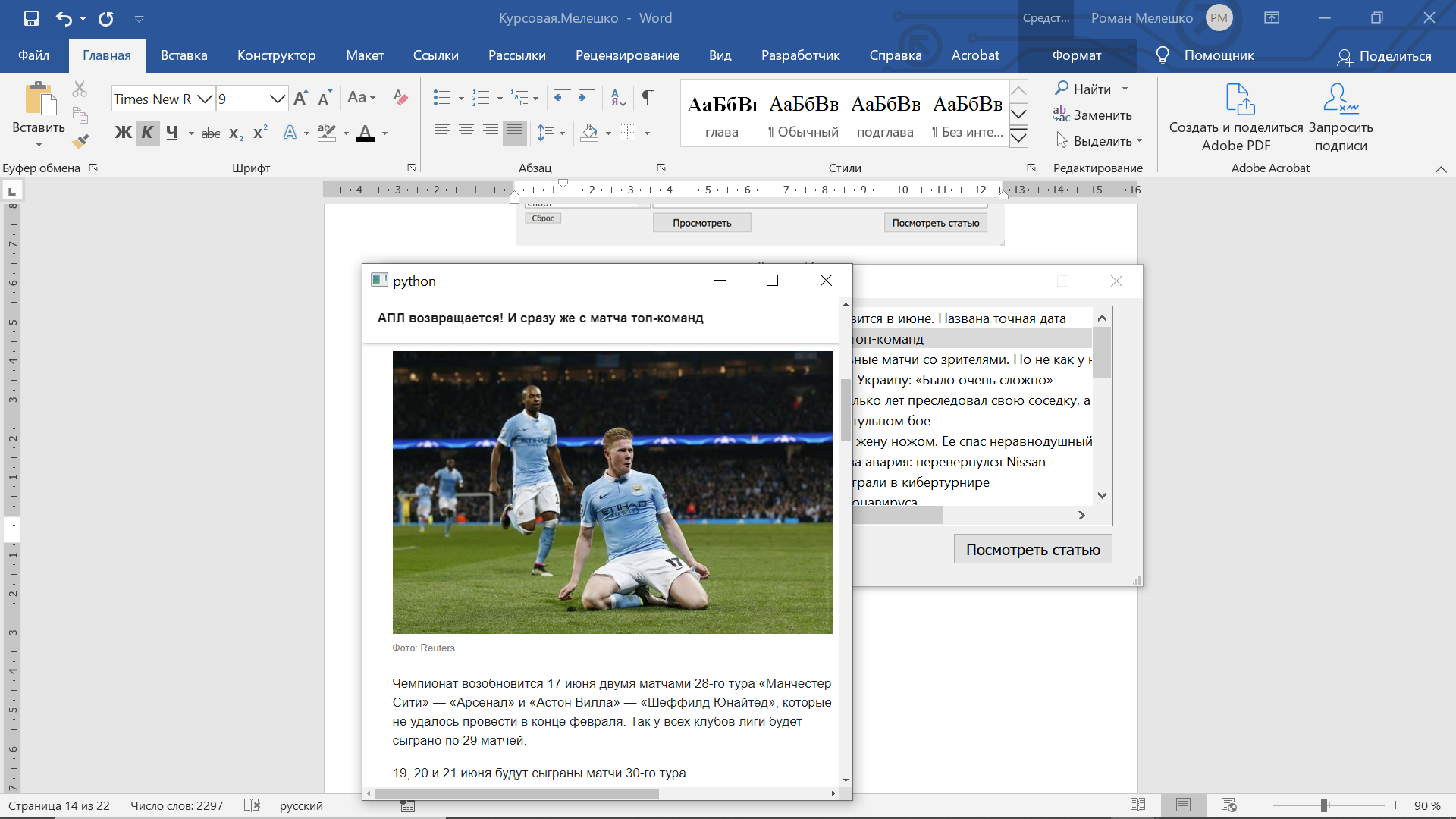


Рисунок 14

Заключение

В ходе выполнения курсовой работы был создан десктопный RRS-агрегатор, позволяющий получать информацию в компактном виде с популярных белорусских интернет порталов с возможностью выбора сайтов, желаемых категорий, просмотра заголовков новостей и, при желании, просмотра полного содержании новости с использованием встроенного браузера. Были описаны используемые инструменты, показан процесс непосредственного написания агрегатора.

Все поставленные цели и задачи курсовой работы выполнены.

https://drive.google.com/drive/folders/1TKy6DB2oqlNVG3NdNr4b3kWb2Iv5VBKA?usp=sharing

Список использованных источников

1. Python 3.8.3 documentation [Электронный ресурс] Режим доступа: https://docs.python.org/3/- Дата доступа: 10.05.2020.
2. PyCharm documentation[ Электронный ресурс] Режим доступа: https://intellij-support.jetbrains.com/hc/en-us/?pycharm&\_ga=2.153623100.545254317.1590746838-498862250.1589377038 Дата доступа:10.05.2020
3. Qt tech talk&totorials [Электронный ресурс] Режим доступа: https://resources.qt.io/resources-by-content-type-videos-demos-tutorials Дата доступа:12.05.2020.
4. Wikipedia Электронный ресурс] Режим Доступа: https://ru.wikipedia.org/wiki/RSS-агрегатор Дата доступа 12.05.2020
5. Shaw Z. A.Learn Python The Hard Way,3 издание/ Shaw Z. A-2013 год-320 страниц
6. Прохоренок Н.А. Python 3 и PyQt 5. Разработка приложений. Прохоренок Н.А.-2016 год-833 страницы
7. Райан Митчел. Веб-парсинг на Python,2 издание/ Райан Митчел-2018 год-308 страниц

Приложение1

import requests as req  
from bs4 import BeautifulSoup  
from PyQt5 import QtWidgets  
import design  
from PyQt5 import QtWebEngineWidgets  
from PyQt5.QtCore import QUrl  
import sys  
  
  
  
class AppParser(QtWidgets.QMainWindow, design.Ui\_MainWindow):  
 *"""App for parsing rss feeds"""* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.Web = QtWebEngineWidgets.QWebEngineView()  
 self.sel\_cat = []  
 self.res\_list = []  
 self.setupUi(self)  
 self.links = ['https://news.tut.by/rss/all.rss',  
 'https://www.onliner.by/feed',  
 'https://www.belta.by/rss',]  
 self.Parse\_btn\_start.clicked.connect(self.parse\_only\_checked)  
 self.pushButton.clicked.connect(self.open\_article)  
 self.pushButton\_2.clicked.connect(self.reset\_ch)  
 self.progressBar.setValue(0)  
  
 def parser(self, url: str) -> list:  
 *"""  
 Parsing your rss web-resource  
 :param url: href of rss  
 :return: list with next trips (title, category, link) + change obj self.res\_list  
 """* result = req.get(url)  
 soup = BeautifulSoup(result.text, 'html.parser')  
 for i in soup.find\_all('item'):  
 for title, cat, link in zip(i.find\_all('title')[0],  
 i.find\_all('category')[0],  
 i.find\_all('guid'),):  
 try:  
 self.res\_list.append([title.strip(), cat.strip(), link.text.strip()])  
 except:  
 pass  
 return self.res\_list  
  
 def create\_db\_category(self):  
 for i in self.res\_list:  
 if i[1] in "Технологии,В Беларуси,Наука,Интернет и связь,Гаджеты,Игры,Оружие".split(','):  
 i[1] = "Технологии и Наука"  
  
 elif i[1] in "В мире,Попкорн,Люди,Мнения,Комментарии,Интервью,Колумнисты,Кругозор,Калейдоскоп,Общество,Регионы,Люди,Культура".split(','):  
 i[1] = "Общество"  
  
 elif i[1] in "Единоборства,Спорт,Чемпионат Беларуси по футболу,Биатлон,Хоккей,Футбол,Теннис,Баскетбол,Гандбол,Околоспорт".split(','):  
 i[1]="Спорт"  
  
 elif i[1] in "Видео,Авто,Дорога,Тест-драйвы,Автобизнес,Автоновости".split(','):  
 i[1]="Авто"  
  
 elif i[1] in "Деньги и власть,Президент,Политика,Экономика,Личный счет,Публичный счет,Банки".split(','):  
 i[1]="Экономика и финансы"  
  
 elif i[1] in "Недвижимость,Экспертиза,От застройщика,Строительство,Аренда,Деньги,Интерьер,дизайн,ремонт".split(','):  
 i[1]="Недвижимость"  
  
 elif i[1] in "Лекарства,ЗОЖ,Правильное питание,Врачи,Болезни,Тренировки,Красота,Медицинские новости".split(','):  
 i[1]="Здоровье"  
  
 elif i[1] in "Тело,Вкус жизни,Отношения,Стиль,Карьера,Звезды,Вдохновение,Еда,Анонсы".split(','):  
 i[1]="Леди"  
   
 elif i[1] in "эксклюзив,Эксклюзив".split(','):  
 i[1]="Эксклюзив"  
  
 elif i[1] in "Происшествия,происшествия".split(','):  
 i[1]="Происшествия"  
  
 elif i[1] in "Афиша,Места,Обзоры,Премьера TUT,Звезды,Новости".split(','):  
 i[1]="Афиша"  
  
 else:   
 i[1]="Разное"  
  
 def open\_article(self):  
 try:  
 for i in self.res\_list:  
 if i[0] == self.listWidget.item(self.listWidget.currentRow()).text():  
 self.Web.load(QUrl(i[2]))  
 self.Web.show()  
  
 except:  
 pass  
 pass  
  
  
  
 def parse\_only\_checked(self):  
 *"""Call parser according to chosen web-resources"""* self.listWidget.clear()  
 self.res\_list = []  
 self.progressBar.setValue(0)  
 if self.Tutby\_chk\_box.isChecked():  
 self.parser(self.links[0])  
  
 self.progressBar.setValue(10)  
 if self.Onliner\_chk\_box.isChecked():  
 self.parser(self.links[1])  
  
 self.progressBar.setValue(20)  
 if self.Belta\_chk\_box.isChecked():  
 self.parser(self.links[2])  
  
 self.progressBar.setValue(30)  
 self.create\_db\_category()  
 self.sel\_cat = [i.text() for i in map(self.listWidget\_2.item, range(12)) if i.isSelected()]  
 for i in self.res\_list:  
 if i[1] in self.sel\_cat:  
 self.listWidget.addItem(i[0])  
 self.listWidget\_2.item(1).setSelected(0)  
 # with open('RES.txt', 'w+') as f:  
 # for i in self.res\_list:  
 # try:  
 # f.write(f'TITLE -->> {i[0]}\n'  
 # f'CATEGORY -->> {i[1]}\n'  
 # f'LINK -->> {i[2]}\n\n')  
 # except:  
 # print(f'{i[0]}\n{i[1]}\n{i[2]}')  
 self.progressBar.setValue(100)  
   
 def print\_all\_info(self) -> None:  
 *"""  
 Show all parsed info  
 Do Print  
 :return: All parsed info >>> TITLE:: CATEGORY:: LINK::  
 """* if not self.res\_list:  
 print('Choose web-resource, use method check\_box\_value\_return to chose')  
 for title, category, link in self.res\_list:  
 print(f'{"<<< NEW BLOCK >>>":^100}\n'  
 f'{title:^100}\n'  
 f'{category:^100}\n'  
 f'{link}\n')  
  
 def reset\_ch(self):  
 [i.setSelected(0) for i in map(self.listWidget\_2.item, range(12))]  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 Main = QtWidgets.QApplication(sys.argv)  
 window = AppParser()  
 window.show()  
 Main.exec\_()

Приложение 2

# -\*- coding: utf-8 -\*-  
  
# Form implementation generated from reading ui file 'design.ui'  
#  
# Created by: PyQt5 UI code generator 5.14.2  
#  
# WARNING! All changes made in this file will be lost!  
  
  
from PyQt5 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 MainWindow.setObjectName("MainWindow")  
 MainWindow.resize(1000, 380)  
 MainWindow.setMinimumSize(QtCore.QSize(1000, 380))  
 MainWindow.setMaximumSize(QtCore.QSize(1000, 380))  
 self.centralwidget = QtWidgets.QWidget(MainWindow)  
 self.centralwidget.setMinimumSize(QtCore.QSize(1000, 0))  
 self.centralwidget.setMaximumSize(QtCore.QSize(1000, 380))  
 self.centralwidget.setObjectName("centralwidget")  
 self.Tutby\_chk\_box = QtWidgets.QCheckBox(self.centralwidget)  
 self.Tutby\_chk\_box.setGeometry(QtCore.QRect(20, 10, 131, 41))  
 font = QtGui.QFont()  
 font.setPointSize(10)  
 self.Tutby\_chk\_box.setFont(font)  
 self.Tutby\_chk\_box.setObjectName("Tutby\_chk\_box")  
 self.Onliner\_chk\_box = QtWidgets.QCheckBox(self.centralwidget)  
 self.Onliner\_chk\_box.setGeometry(QtCore.QRect(20, 60, 131, 41))  
 font = QtGui.QFont()  
 font.setPointSize(10)  
 self.Onliner\_chk\_box.setFont(font)  
 self.Onliner\_chk\_box.setObjectName("Onliner\_chk\_box")  
 self.Belta\_chk\_box = QtWidgets.QCheckBox(self.centralwidget)  
 self.Belta\_chk\_box.setGeometry(QtCore.QRect(20, 110, 131, 41))  
 font = QtGui.QFont()  
 font.setPointSize(10)  
 self.Belta\_chk\_box.setFont(font)  
 self.Belta\_chk\_box.setObjectName("Belta\_chk\_box")  
 self.Parse\_btn\_start = QtWidgets.QPushButton(self.centralwidget)  
 self.Parse\_btn\_start.setGeometry(QtCore.QRect(280, 310, 201, 41))  
 font = QtGui.QFont()  
 font.setPointSize(10)  
 self.Parse\_btn\_start.setFont(font)  
 self.Parse\_btn\_start.setObjectName("Parse\_btn\_start")  
 self.progressBar = QtWidgets.QProgressBar(self.centralwidget)  
 self.progressBar.setEnabled(True)  
 self.progressBar.setGeometry(QtCore.QRect(20, 160, 171, 21))  
 font = QtGui.QFont()  
 font.setPointSize(10)  
 self.progressBar.setFont(font)  
 self.progressBar.setProperty("value", 24)  
 self.progressBar.setObjectName("progressBar")  
 self.listWidget = QtWidgets.QListWidget(self.centralwidget)  
 self.listWidget.setGeometry(QtCore.QRect(280, 10, 681, 291))  
 self.listWidget.setObjectName("listWidget")  
 self.listWidget\_2 = QtWidgets.QListWidget(self.centralwidget)  
 self.listWidget\_2.setGeometry(QtCore.QRect(20, 220, 256, 81))  
 self.listWidget\_2.setSelectionMode(QtWidgets.QAbstractItemView.MultiSelection)  
 self.listWidget\_2.setObjectName("listWidget\_2")  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 item = QtWidgets.QListWidgetItem()  
 self.listWidget\_2.addItem(item)  
 self.pushButton = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton.setGeometry(QtCore.QRect(750, 310, 211, 41))  
 font = QtGui.QFont()  
 font.setPointSize(10)  
 self.pushButton.setFont(font)  
 self.pushButton.setObjectName("pushButton")  
 self.pushButton\_2 = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton\_2.setGeometry(QtCore.QRect(20, 310, 75, 23))  
 self.pushButton\_2.setObjectName("pushButton\_2")  
 MainWindow.setCentralWidget(self.centralwidget)  
 self.statusbar = QtWidgets.QStatusBar(MainWindow)  
 self.statusbar.setObjectName("statusbar")  
 MainWindow.setStatusBar(self.statusbar)  
  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))  
 self.Tutby\_chk\_box.setText(\_translate("MainWindow", "Tut by"))  
 self.Onliner\_chk\_box.setText(\_translate("MainWindow", "Onliner"))  
 self.Belta\_chk\_box.setText(\_translate("MainWindow", "BelTa"))  
 self.Parse\_btn\_start.setText(\_translate("MainWindow", "Просмотреть"))  
 \_\_sortingEnabled = self.listWidget\_2.isSortingEnabled()  
 self.listWidget\_2.setSortingEnabled(False)  
 item = self.listWidget\_2.item(0)  
 item.setText(\_translate("MainWindow", "Технологии и Наука"))  
 item = self.listWidget\_2.item(1)  
 item.setText(\_translate("MainWindow", "Общество"))  
 item = self.listWidget\_2.item(2)  
 item.setText(\_translate("MainWindow", "Спорт"))  
 item = self.listWidget\_2.item(3)  
 item.setText(\_translate("MainWindow", "Авто"))  
 item = self.listWidget\_2.item(4)  
 item.setText(\_translate("MainWindow", "Экономика и финансы"))  
 item = self.listWidget\_2.item(5)  
 item.setText(\_translate("MainWindow", "Недвижимость"))  
 item = self.listWidget\_2.item(6)  
 item.setText(\_translate("MainWindow", "Здоровье"))  
 item = self.listWidget\_2.item(7)  
 item.setText(\_translate("MainWindow", "Леди"))  
 item = self.listWidget\_2.item(8)  
 item.setText(\_translate("MainWindow", "Эксклюзив"))  
 item = self.listWidget\_2.item(9)  
 item.setText(\_translate("MainWindow", "Происшествия"))  
 item = self.listWidget\_2.item(10)  
 item.setText(\_translate("MainWindow", "Афиша"))  
 item = self.listWidget\_2.item(11)  
 item.setText(\_translate("MainWindow", "Разное"))  
 self.listWidget\_2.setSortingEnabled(\_\_sortingEnabled)  
 self.pushButton.setText(\_translate("MainWindow", "Посмотреть статью"))  
 self.pushButton\_2.setText(\_translate("MainWindow", "Cброс"))  
  
  
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\_())