Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра математического анализа и моделирования
Направление подготовки – 01.03.02 Прикладная математика и информатика

ДОПУСТИТЬ К ЗАЩИТЕ
И.о. зав. кафедрой
_____Н.Н. Максимова
«_____»_____2021 г.

БАКАЛАВРСКАЯРАБОТА

на тему: Разработка программного обеспечения для парсинга сайтов с обработкой данных в табличном процессоре

Исполнитель
студент группы 752об _____ Д.А. Турышев
(подпись,дата)

Руководитель
доцент, канд. тех. наук _____Т.В. Труфанова
(подпись, дата)

Нормоконтроль
доцент, канд. физ.-мат. наук _____ Е.М. Веселова
(подпись, дата)

Благовещенск2021

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра математического анализа и моделирования

УТВЕРЖДАЮ
И.о. зав. кафедрой
_____Н.Н. Максимова
«_____»_____2021

г.

З А Д А Н И Е

К бакалаврской работе студента Турышева Дмитрия Андреевича

1.Темабакалаврскойработы:Разработка программного обеспечения для парсинга сайтов с обработкой данных в табличном процессоре (утвержденаприказомот23.04.2021№812-уч)

2.Сроксдачистудентомзаконченнойработы:23.06.2021 г.

3. Исходные данные к бакалаврской работе: формулировка темы ВКР, литературные источники, среда разработки PyCharm для языка программирования Python.

4. Содержание бакалаврской работы (перечень подлежащих разработке вопросов): разработка программного продукта, который будет способствовать созданию единого информационного пространства рынка недвижимости на сайте Авито.

5. Перечень материалов приложения: листинг компьютерной программы;SQL скрипт для создания БД .

6. Консультанты по бакалаврской работе: Нормоконтроль – Веселова Е.М., доцент, канд. физ.-мат. наук.

7.Датавыдачизадания:07.05.2021г.

Руководитель бакалаврской работы: Труфанова Татьяна Вениаминовна, доцент, канд. тех. наук

Задание принял к исполнению(07.05.2021):_____Турышев Д.А.

РЕФЕРАТ

Бакалаврская работа содержит 65 страниц, 8 рисунков, 19 источников, 7 таблиц.

PYTHON, HTML, ОБРАБОТКА, ПАРСИНГ, ПРОГРАММА, МОДУЛИ, СВОДНЫЕ ТАБЛИЦЫ, НЕДВИЖИМОСТЬ, БАЗА ДАННЫХ.

Данная работа посвящена разработке программного продукта, который будет способствовать созданию единого информационного пространства рынка недвижимости.

В работе проделан обзор программных средств для разработки ПО.

Создана база данных с использованием системы управления базами данных (СУБД) MySQL Work Bench от компании Oracle.

Разработан алгоритм для получения данных о недвижимости с применением модулей Requestsи Beautiful Soop из языка программирования Python. Представлены полученные данные и сводные таблицы о недвижимости в Microsoft Excel.

Реализовано программное обеспечение для сбора и анализа данных о рынке недвижимости в Амурской области. Написаны алгоритмы для реализации основных функций в программе PyCharm.Установлено соединения MicrosoftExcel с базой данных.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В течение нескольких лет профессионалы рынка недвижимости работали и работают над созданием системы информационного обеспечения деятельности риэлторов - информационных стандартов, алгоритмов, программных и технических средств, обеспечивающих информационное взаимодействие риэлторских фирм и ассоциаций при решении ими всего разнообразия профессиональных задач. В этом круге задач выделяются три самостоятельных, но тесно связанных направления:

- информационное обеспечение операций с недвижимостью (прием, хранение и переработка в фирме заявок на продажу и покупку объектов, межфирменный, межрегиональный, международный обмен)

- информационное обеспечение технологии риэлтерской деятельности (прием, хранение и использование юридической и иной общей информации, необходимой риелторам)

- информационное и методическое обеспечение исследований рынка недвижимости (мониторинг, анализ состояния и прогнозирование тенденций развития рынка)

Исследование (анализ) рынка недвижимости представляет собой самостоятельный вид деятельности, имеющий целью обеспечение объективной информацией лиц, принимающих решения о проведении тех или иных операций на рынке.

Исследование (анализ) рынка недвижимости может проводиться с различными частными целями и представлять собой элемент, этап других видов деятельности: маркетингового исследования с целью продвижения конкретного товара или услуги; оценочной деятельности с целью определения стоимости конкретного объекта; инвестиционной деятельности с целью оценки эффективности инвестиционных решений; анализа и прогнозирования тенденций развития рынка с самыми общими целями – для выработки риэлтерским сообществом, руководителями риэлтерских фирм и

ассоциаций, потенциальными инвесторами, органами управления стратегических решений по развитию бизнеса, совершенствованию механизмов функционирования рынка.

Общим элементом анализа рынка в интересах всех видов деятельности является мониторинг рынка, т.е. сбор и обработка информации о рынке.

В последнее время все большее понимание находит необходимость преодоления информационного разрыва между оценщиками, брокерами, девелоперами, финансовыми аналитиками и другими риелторами - профессионалами рынка недвижимости, а также их объединениями.

Данная тема является актуальной, так как разработка программного обеспечения позволит анализировать рынок недвижимости в удобном для пользователя формате.

Работа посвящена созданию единого информационного пространства рынка недвижимости и технологией, и организацией сбора и обработки данных.

Объектом исследования является программное обеспечение для парсинга сайтов недвижимости.

Предметом исследования являются организационно экономические отношения, возникающие при разработке программного обеспечения для парсинга данных о недвижимости в табличном процессоре Microsoft Excel.

Целью бакалаврской работы является разработка программного обеспечения, который будет способствовать созданию единого информационного пространства рынка недвижимости.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- построить базу данных для хранения информации о недвижимости
- реализовать алгоритм для получения информации с сайта Авито
- ввести обработку информации для стандартизации полученных данных о недвижимости
- настроить подключение к базе данных в Microsoft Excel

- апробация данных с помощью сводных таблиц в табличном процессоре Excel

С точки зрения практической значимости, разработанное программное обеспечение можно использовать для анализа рынка недвижимости инвесторами, риелторами и аналитика рынка.

Результаты проделанной работы апробированы на XXX научных конференциях «День науки АмГУ», а также в XXII региональной научно-практической конференции «Молодежь XXI века: шаг в будущее».По результатам работы опубликована научная статья [19].

В первой главе бакалаврской работы описаны программные продукты, которые использовались в разработке ПО.

Во второй главе постановка и анализ рассматриваемой задачи.

В третьей главе дипломной работы рассмотрен подход к построению базы данных, расписано логическое проектирование связей между таблицами в базе данных. Взаимодействие базы данных с программным продуктом. Практическая часть выполненных исследований представлена в четвертой главе бакалаврской работы. Разработка ПО и представление полученных данных о недвижимости в Microsoft Excel.

1 ОБЗОР ПРОГРАММНЫХ СРЕДСТВ ДЛЯ РАЗРАБОТКИ ПО

1.1 Язык программирования Python

Python (в русскоязычное наименование питон или пайтон) — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным — всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации, сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как Си или C++.

В нем поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты. По состоянию на 2019 год в нём содержалось более 175 тысяч пакетов. Основные пакеты для выполнения запросов и обработки полученного результат Requestsи BeautifulSoup.

Библиотека Requestsявляется стандартным инструментом для составления HTTP-запросов в Python. Простой и аккуратный API значительно облегчает трудоемкий процесс создания запросов. Таким образом, можно сосредоточиться на взаимодействии со службами и использовании данных в приложении.

Основные задачи Requests:

1. Создание запросов при помощи самых популярных HTTP методов;

2. Редактирование заголовков запросов и данных при помощи строки запроса и содержимого сообщения;

3. Анализ данных запросов и откликов;

4. Создание авторизированных запросов;

5. Настройка запросов для предотвращения сбоев и замедления работы приложения.

BeautifulSoup является библиотекой Python для парсинга HTML и XML документов. Часто используется для скрапинга веб-страниц. BeautifulSoup позволяет трансформировать сложный HTML-документ в сложное древо различных объектов Python. Это могут быть теги, навигация или комментарии.

1.2. Среда программирования PyCharm

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юниттестов и поддерживает веб-разработку на Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

PyCharm кроссплатформенна и совместима с Windows, Linux и MacOS. Она поддерживает, как вторую, так и третью версию Питона и, кроме всего прочего, имеет красивый и функциональный UI.

Инструменты и функции, предоставляемые этой интегрированной средой, помогают Python-программистам быстро и эффективно писать код, синхронизироваться с системой контроля версий, использовать фреймворки и плагины, позволяя при этом настроить интерфейс так, как удобно самому пользователю и даже дают возможность дополнительного расширения IDE.

В PyCharm, конечно же, есть встроенный анализатор кода, который действительно помогает при его написании. Благодаря анализатору можно весьма комфортно перемещаться по проекту, использовать быстрый поиск, а также исправлять ошибки, которые PyCharm любезно подсвечивает и подробно описывает.

В PyCharm реализованы системы автодополнения и контроля качества кода в соответствии со стандартом PEP8. Всё для того, чтобы сделать текст ваших программ красивее, чище и структурированнее.

В интегрированной среде от JetBrains поддерживаются основные современные Python-фреймворки для веб-разработки, в ней вы сможете работать с Jupyter-notebook, подключать Anaconda, а также прочие библиотеки для научных вычислений и Data Science. Помимо, собственно, питона, PyCharm отлично ладит и с другими языками программирования – JS, TypeScript-a, SQL или шаблонизаторами.

В PyCharm открывается разумное обновление кода с безопасным удалением и переименованием, методом извлечения, вводной переменной, встроенной переменной или методом, и другими видами рефакторинга. Ориентированные на фреймворк и язык программирования рефакторинги помогут выполнить любое изменение в рамках целого проекта.

PyCharm предоставляет первоклассную поддержку для JavaScript, CoffeeScript, TypeScript, HTML и CSS, а также их современных преемников. Отладчик JavaScript также включен в PyCharm и является интегрированным с конфигурацией запуска сервера Django.

PyCharm Professional Edition имеет несколько вариантов лицензий, которые отличаются функциональностью, стоимостью и условиями использования, а также является бесплатным для образовательных учреждений и проектов с открытым исходным кодом.

Существует также бесплатная версия Community Edition, обладающая усеченным набором возможностей. Распространяется под лицензией Apache 2.

1.3. MySQL Workbench

MySQL Workbench от Oracle — это графический инструмент для работы с базами данных MySQL. MySQL — это система управления реляционными базами данных с открытым исходным кодом, и это самая

популярная система баз данных, используемая с PHP. Он доступен в Windows, Linux и Mac OS X.

MySQL Workbench — мощный визуальный инструмент для администраторов баз данных, архитекторов баз данных и разработчиков MySQL. Этот инструмент предлагает моделирование данных, разработку SQL и инструменты администрирования для настройки и администрирования сервера.

MySQL Workbench — это кроссплатформенный инструмент проектирования реляционных баз данных с открытым исходным кодом, который добавляет функциональность и упрощает разработку MySQL и SQL. Он объединяет проектирование, разработку, создание, администрирование и обслуживание SQL, а также предлагает графический интерфейс для структурированной работы с вашими базами данных.

MySQL Workbench предоставляет возможности для управления моделями баз данных, такими как:

1. Создание графической модели;
2. Обратный инжиниринг живых баз данных в модели (моделирование данных);
3. Прямая инженерная модель в скрипт / живую базу данных и больше;

Существует несколько систем управления реляционными базами данных, таких как Microsoft SQL Server, Microsoft Access, Oracle и DB2. MySQL Workbench предлагает некоторые преимущества, которые следует учитывать при выборе инструмента.

MySQL поддерживает несколько механизмов хранения, каждый со своими спецификациями, в отличие от других инструментов. Он также предлагает высокую производительность благодаря своему дизайну и простоте.

Он также известен своей рентабельностью. Версия сообщества бесплатна для пользователей, а корпоративная версия имеет низкую лицензионную плату.

Моделирование — отличный способ визуализировать требования и создавать хорошо работающие базы данных, которые могут не отставать от постоянно меняющихся требований к данным. MySQL Workbench позволяет создавать модели и управлять ими, преобразовывать динамическую базу данных в модель, а также создавать и редактировать таблицы и вставлять данные.

Инструмент имеет возможность преобразовывать диаграммы ER в операторы SQL и отправлять их на сервер SQL. Также может создавать модели из целевой базы данных или даже из импортированных файлов SQL.

MySQL Workbench предлагает различные инструменты, которые позволяют просматривать и улучшать производительность. Панель управления производительностью дает быстрый обзор различных показателей производительности. Отчеты о производительности позволяют анализировать производительность баз данных.

MySQL Workbench упрощает процесс миграции. Он позволяет выполнять миграцию с Microsoft SQL Server, SQLite, Microsoft Access и многих других. СУБД также позволяет настраивать, планировать, редактировать и выполнять проекты миграции.

MySQL Workbench значительно упрощает управление пользователями:

1. Просмотр информации об учетных записях всех пользователей на сервере MySQL;
2. Добавить и удалить пользователей;
3. Предоставлять и отзывать привилегии;
4. Изменить глобальные разрешения и разрешения базы данных;
5. Сменить пароли;
6. Проведите аудит, чтобы узнать, кто что и когда делал;

MySQL Workbench позволяет создавать, управлять и организовывать подключения к базе данных. Управление объектами Обозреватель объектов MySQL Workbench позволяет визуально выбирать таблицы и столбцы. Имеется возможность перемещаться между схемами, выбирать таблицы и поля, создавать новые или отбрасывать их.

1.4. Табличный процессорMicrosoft Excel

Программа Microsoft Excel предназначена для работы с таблицами данных, преимущественно числовых. При формировании таблицы выполняют ввод, редактирование и форматирование текстовых и числовых данных, а также формул. Наличие средств автоматизации облегчает эти операции. Созданная таблица может быть выведена на печать.

Электронными таблицами называются программы, предназначенные для проведения расчетов на компьютере в табличной форме.

Документом Excel является файл с произвольным именем и расширением .XLS. В каждом файле XLS может размещаться от 1 до 255 электронных таблиц, каждая из которых называется рабочим листом (Sheet). В любую ячейку электронной таблицы можно ввести: число, формулу(Возможность использования формул и функций является одним из важнейших свойств программы обработки электронных таблиц. Вычисления задаются формулами. Текст формулы, которая вводится в ячейку таблицы, должен начинаться со знака равенства =, чтобы программа Excel могла отличить формулу от текста. После знака равенства в ячейку записывается математическое выражение, содержащее аргументы, арифметические операции и функции. В качества аргументов в формуле обычно используются числа и адреса ячеек. Для обозначения арифметических операций могут использоваться следующие символы: + (сложение); - (вычитание); * (умножение); / (деление);^ (возведение в степень), текстовую (алфавитно-цифровую) информацию. Адрес ячейки определяется ее местоположением в таблице. Ячейка задается своими координатами, в которых на первом месте стоит буква, обозначающая колонку, а на втором -

число, обозначающее ряд. Ячейка, в которую в данный момент вводятся данные, называется активной. В каждый момент времени активной может быть только одна ячейка, она всегда выводится на индикацию.

Современные табличные процессоры обеспечивают:

1. ввод, хранение и корректировку большого количества данных;
2. автоматическое проведение вычислений при изменении исходных данных;
3. наглядность и естественную форму документов, представляемых пользователю на экране;
4. графическую интерпретацию данных в виде диаграмм;

Основные функции табличных процессоров:

1. Защита данных.Организована через меню Файл в диалоговом окне Сохранение и реализовано с помощью паролей на открытие файлов, разрешение записи и чтение.

2. В основном табличный процессор предназначен для работы с числами, поэтому предусматриваются расширенные возможности по их форматированию.Все табличные вычисления реализуются с использованием формул и функций. Встроенный менеджер формул помогает пользователю найти ошибку или неверную ссылку в большой таблице. При переходе в текстовый режим в ячейках отображаются сами формулы, а не результат вычисления.

3. Ввод формул. Excel интерпретирует вводимые данные либо как текст (выравнивание по левому краю), либо как числовое значение (выравнивание по правому краю), либо в виде формулы со ссылками на ячейки данных. Для редактирования формул необходимо использовать клавишуF2 или двойной щелчок мышью.

4. Информационные связи. Предназначены для установления ссылок на значения в ячейках, которые каждый раз будут автоматически обновляться. Связывание производится через адрес ссылки или из меню Правка Специальная вставка.

5. Групповые имена. Для упрощения задач, связанных с целой группой ячеек. Для присвоения имени выделенной группе ячеек в меню Вставка открывается подменю Имя и директива Присвоить. Имя группы должно начинаться с буквы и содержать не более 255 символов. Не допускается использование пробелов, также имя не должно совпадать с адресами ячеек. Заголовки строк и столбцов также могут использоваться в качестве имён этих областей. Реализуется с помощью директивы Создать из меню Вставка Имя. Адресация может осуществляться ссылкой на другой лист в той же книге, ссылкой на лист в другой книге, при этом Excel автоматически обновляет ссылки только для открытых книг. Когда книга закрыта, ссылка содержит полный путь к данным. Эффективное использование ссылок обеспечивает:

1. Слияние данных нескольких книг;
2. Создание различных представлений одних и тех же данных;
3. Последовательную разработку больших и сложных моделей обработки данных.

6. Построение и оформление диаграмм. Excel обладает специальным конструктором, позволяющим улучшить качество документа и разрабатывать трёхмерные диаграммы. Конструктор называется ChartWizard. Он обеспечивает построение, выбор типов, организацию надписей на осях, внедрение дополнительных объектов и актуализацию диаграммы.

7. Работа с функциями. Табличный процессор имеет более 300 заранее запрограммированных формул, которые называются функциями. Все функции разделены по категориям, а встроенный конструктор функций облегчает работу с ними. Функция оперирует некоторыми данными, которые называются её аргументами. Аргумент функции может занимать одну ячейку или размещаться в целой группе. Конструктор функций позволяет самостоятельно добавлять, редактировать и комбинировать различными функциями.

8. Создание простых баз данных.Для организации поиска записей в базе данных применяются фильтры. Фильтры могут быть использованы

только для одной базы данных на листе. Фильтры находятся в меню Данные.Реализуются фильтры в двух вариантах: автофильтр и расширенный фильтр.

В Excelимеется возможность импорт данных через постоянное подключение к базе данных и импорт из csv файла, благодаря этому можно производить обработку данных в нем: использовать сводные таблицы, графики, диаграммы и так далее. Что позволит не тратить время разработку этих инструментов вручную. Экономит время и ресурсы.

1.5. Браузер Google Chrome.

На сегодняшний день Google Chrome самым популярный и стабильный веб-обозреватель. Он составляет здоровую конкуренцию таким мощным программам для работы в интернете, как Opera, Internet Explorer, FireFox. Хороший, многофункциональный и безопасный браузер.

Впервые браузер был выпущен 2 сентября 2008 года, но бета-версия не функционировала корректно, поэтому путем многократных изменений, главный релиз Google Chrome состоялся 11 декабря 2008 года. Суть продукта была такая же, как и других веб-обозревателей – серфинг по интернет-страницам. Однако в основу разработок компания решила заложить высокий уровень безопасности и высокоскоростную работу, что в результате ей удалось.

На сегодняшний день последним релизом Гугл Хрома является 79 версия. За годы усовершенствования браузер прошел очень много обновлений и доказал, что он лучший среди своих конкурентов. Об этом говорит и статистика количества скачиваний. В 2014 году браузером Chrome пользовалась треть человечества на земле, а в 2018 году он побил все рекорды, так как была зафиксирована цифра – 61% скачиваний. Поэтому Гугл Хром можно смело называть самым популярным и прогрессивным веб-обозревателем.

Интерфейс Google Chrome очень простой. Разработчики изначально делали акцент на подобный дизайн, однако некоторые пользователи

называют его «пустым» и вроде как чего-то не хватает. Но ввиду огромного количества доступных расширений, можно создавать дизайн и наполнять браузер разными программами.

Специалисты на основе проведенных тестов признали браузер Гугл Хром самым безопасным.

Кроме безопасности Google Chrome покорил большое количество пользователей своей скоростью и быстрой работой. В первую очередь это связано с использованием высокопроизводительного движка Java Script последнего поколения и протокол чтения DNS. Во встроенном функционале браузера можно посмотреть нагрузку процессов на оперативную память устройства и тем самым через Диспетчер задач отключать тормозящие сайты или расширения. Что касается стабильности, то и здесь разработчики Гугл хорошо постарались. Они ввели многопроцессную архитектуру и тем самым разделили открытие вкладок на отдельные процессы, не мешающие друг другу. Изолировав работу сайтов, вы можете не переживать, что сбой или зависание на одном ресурсе скажется на другом.

Хром имеет много полезных инструментов для анализа страниц, запросов, скриптов и тд. Один из таких инструменты разработчика. Инструменты разработчика Chrome позволяют быстро анализировать контент и ресурсы веб-страницы.

2АНАЛИЗ ПОСТАВЛЕННОЙ ЗАДАЧИ

2.1 Постановка и описании задачи разработки.

На сегодняшний день в интернете много информации, которую сложно оценивать и анализировать. Авито – это сайт для размещения объявлений о товарах, недвижимости, резюме и тому подобное. Товары могут быть как новыми, так и бывшими. Одной из основных проблем, встающих перед предпринимателем, риэлтором, инвесторами, аналитиками при анализе рынка недвижимости на сайте Авито, отсутствие единой базы, с помощью которой бы удобно определять емкость рынка в целом или его отдельных сегментов, строить графики, искать зависимости, а также чтобы это база была актуальной. От сюда и вытекает необходимость в программе для сбора информации в единый кластер.

Для полного анализа были определены следующие характеристики для одного объекта недвижимости:

1. Название
2. Цена
3. Адрес
4. Область
5. Район
6. Населенный пункт
7. Номер дома
8. Номер квартиры
9. Общая площадь
10. Цена за квадратный метр
11. Вид объявления – форма имущественного договора (продажа, аренда на 1 день, аренда на 30 дней)
12. Состояние жилища – первичное или вторичное жилье
13. Количество просмотров объявления

14. Собственник
15. Ссылка на собственника
16. Вид объекта
17. Номер этажа
18. Количество этажей в доме
19. Количество комнат
20. Жилая площадь
21. Площадь кухни
22. Город постройки
23. Отделка
24. Тип участия
25. Официальный застройщик
26. Название новостройки
27. Корпус/строение
28. Площадь участка
29. Расстояние до города
30. Материал стен
31. Возможность онлайн просмотра
32. Дата публикации объявления
33. Дата снятия с публикации объявления

Для этих целей выбран язык программирования Python, так как он содержит мощные модули для работы chttpзапросами, обработки html иподключение к базе данных.

Поддержания актуальности базы данных следует делать повторную выгрузку с интервалом в 2 дня.

Для исследования собранной информации представить данные в удобном пользователю виде.Табличный процессор Excelотлично подходит, поскольку он имеет мощные инструменты для обработки большого объема данных, такие как: импорт данных, создание таблиц, создание сводных

таблица с настройками фильтров по выбранным полям, построение графиков и диаграмм.

2.2 Описание существующего решения.

На сегодняшний деньв интернете много сервисов предлагающие услуги парсинга Авито. Парсинг досок объявлений позволяет получить информацию обо всех объявлениях в определенных категориях. В том числе и номера телефонов тех, кто их размещает. В большинстве случаев это работает так:

1. На «Авито» указываете в строке поиска нужное вам ключевое слово, выбираете категорию и параметры фильтра, обновляете страницу, чтобы доска показывала объявления по выбранным параметрам;
2. Копируется ссылка из адресной строки сайта, и она вставляется в парсер;
3. Выбираются настройки — какие данные собирать, какие можно игнорировать, за какой период интересны объявления и так далее;
4. Парсер собирает информацию по заданным на Avito и внутри него фильтрам и отдает ее в виде таблицы или в другом виде в зависимости от настроек;

В течение нескольких минут или часов в зависимости от размера выборки получаются все необходимые данные — имена и контакты людей, которые их подали, цены на определенные товары, описания, а в некоторых случаях даже фотографии. Эти данные можно использовать для разных целей:

1. Массовых рассылок. Можно рассылать по собранным номерам телефона смс-сообщения со ссылкой на продвигаемый ресурс или отправлять голосовые сообщения с помощью ботов. А еще писать в мессенджеры — WhatsApp, Viber, Telegram и другие. Да, это спам, но в случае правильной выборки он может принести свои плоды.
2. Холодного обзвона базы. Таким способом часто пользуются продавцы различных финансовых продуктов. Можно обзванивать владельцев объявлений самостоятельно или передать базу call-центру.

3. Заполнения своей площадки. Например, если вы решили создать свою группу в социальной сети, можно не копировать все объявления вручную, а использовать парсер. То же самое актуально для наполнения некоторых страниц сайта, например, если вы перепродаете товары или привлекаете трафик рекламодателям и пишете нормальное описание продуктов.

4. Анализа конкурентов. Можно с помощью парсера собирать текст объявлений или количество просмотров, добавления в избранное. Так вы поймете, на какие формулировки и на какую подачу лучше реагируют пользователи, и сможете создать свое эффективное описание продукта.

В принципе возможности парсинга практически безграничны. Можно собрать цены, чтобы выставить оптимальную для аудитории стоимость товара, или собрать номера тех, кто ищет работу, и предложить им обучение популярной IT-профессии.

2.3 Проблема выбранной темы.

Первая проблема состояла в организации сбора и дальнейшего использования информации, представляемой регионами. Результатом решения этой задачи стали ежегодные сводные отчеты о ситуации на рынке недвижимости России, которые, начиная с 1994 года, выпускает РГР. В этих отчетах обрабатываются, анализируются и обобщаются данные о рынке жилья, а затем и других видов недвижимости 30-50 городов, представляющих большинство регионов нашей страны. Отчеты содержат обобщенную информацию по рынку недвижимости отдельных городов, сопоставление данных по группам городов и обобщение по России в целом. Также выявляются специфические закономерности становления и развития рынка недвижимости России, в увязке с общими процессами, происходящими в экономике страны в целом и с учетом региональных особенностей локальных рынков. Разрабатываются прогнозы на предстоящий год, что стало особенно актуально после разразившегося в августе 1998 года экономического кризиса.

Вторая проблема - обеспечить единство методологии, форм и способов сбора/обработки информации. Для решения этой задачи за прошедшие годы были разработаны три методических пособия, которые были изданы и направлены всем членам РГР и заинтересованным организациям. В Москве и регионах проводилось обучение методам анализа рынка недвижимости на курсах повышения квалификации риэлторов и оценщиков, организуемых РГР. В феврале и мае 2000 года проведены первые семинары по подготовке аналитиков на тему “Технология анализа рынка недвижимости”.

По мере расширения круга организаций, обеспечивающих сбор данных и их обработку, и углубления анализа, возникла настоятельная потребность в стандартизации содержания и форм представления данных, а также автоматизации процессов их обработки. С целью удовлетворения этой потребности были разработаны стандартизированные формы регистрации данных о рынке и специализированные программные продукты для статистической обработки данных.

Основные недостатки при использовании услуг парсинга существующих сервисов:

1. монетизация – сервисы оказывают услуги парсинг по платным тарифам, к примеру, за 1000 объявления платится 100 рублей. Объявлений в категории Недвижимость из региона Амурская область около 8 тысяч штук, за которые по выше указанному тарифу придется отдать 800 рублей. Для использования такого сервиса на постоянной основе,для поддержания актуальности данных, обойдется пользователю в немалую сумму

2. данных, с которыми невозможно работать – ресурсы, оказывающие услуги выгрузки данных с авито, выдают информацию, не поддающуюся глубокому анализу, что важно для решения профессиональных задач аналитиков и финансистов. Допустим требуется проанализировать срез данных в городе Благовещенск на улице Горького, чтобы это сделать, необходимо разбить его по полям регион, район, город,

улица, и тогда производится выборка по этим полям отдельно. У ресурсов нет таких услуг.

3. неполнота – огромная часть из необходимых полей в поставленной задачи для необходимого анализа отсутствуют. К примеру: материал стен, количество просмотров, застройщик, общая площадь и так далее.

4. большая нагрузка на пользователя – использование сторонних сервисов по парсинг неудобно для анализа на долгосрочной основе. Пользователь будет каждый раз вручную выгружать данные с сервиса, платить за их услуги, хранение данных в неструктурированной форме, в конечном итоге становится сложно проводить аналитику рынка. Огромная трата времени и денежных ресурсов.

Все эти недостатки усложняют эффективно оценивать ситуацию на рынки недвижимости.

ЗБАЗА ДАННЫХ

3.1 Подход к построению базы данных.

Одно из основных свойств БД – независимость данных от программы, использующих эти данные. Работа с базой данных требует решения различных задач, основные из них следующие:

1. создание базы
2. заполнение начальными данными
3. сохранение спарсенных данных
4. выборка данных из базы по запросам пользователя

Для создания базы данныхиспользуется системы управления базами данных (СУБД)MySQLWorkBench от компании Oracle.

База данных состоит из трех основных таблиц:

1. Объявление – таблица используется для хранения собранной информации об одной единицы недвижимости с интернета. В которой хранятся все необходимые поля. Программа обращается к ней чтобы записать полученную информации с интернета.
2. Населённых пункты – эта таблица содержит в себе информацию обо всех населённых пунктах в Амурской области с количеством жителей в них. Заполняется после создания базы данных и хранит в себе наименование поселений и районов в едином формате. Пример: район – Зейский, поселение – Зея, население – 21965 человек.
3. История – таблица служит для хранения информации о даты публикации и снятия объявления публикации.

Все таблицы приведены до третьей нормальной формы. Иными словами, неключевые столбцы не пытаются играть роль ключа в таблице, т.е. они действительно должны быть неключевыми столбцами, такие столбцы не дают возможности получить данные из других столбцов, они дают

возможность посмотреть на информацию, которая в них содержится, так как в этом их назначение.

Схема базы данных – еёструктура, описанная на формальном языке, поддерживаемом СУБД. В реляционных базах данных схема определяет таблицы, поля в каждой таблице, и ограничения целостности (первичный, потенциальные и внешние ключи и другие ограничения). Схема базы данных представлена на рис. 1 ниже.

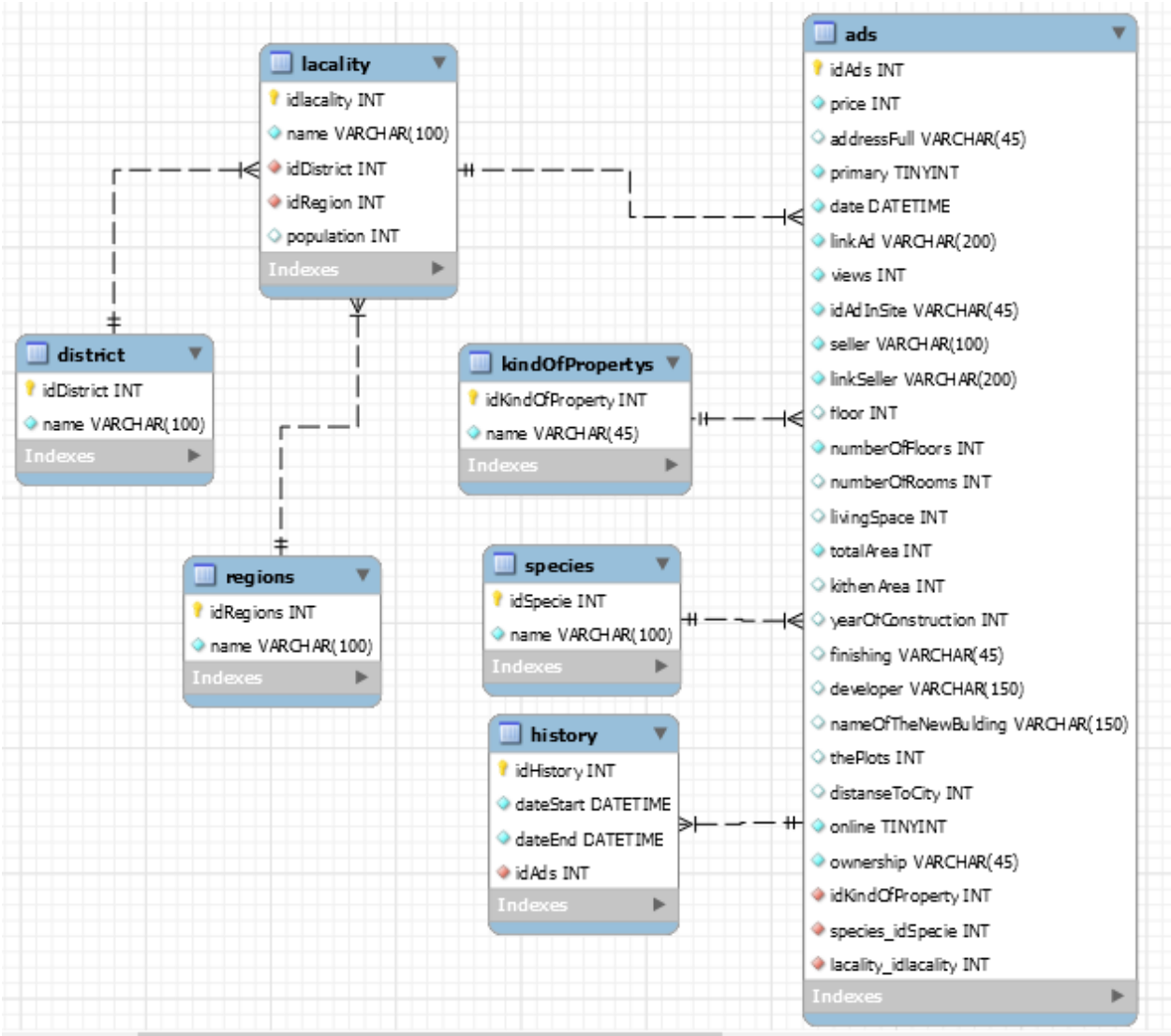

Рисунок 1 – Графическое представление структуры базы данных.

3.2Логическое проектирование базы данных.

Одно из основных свойств БД – независимость данных от программы, использующих эти данные.Логическим проектированием баз данных формально можно считать процесс создания модели мира самой базы

данных, без той системы, которая будет позволять ей работать, и прочих физических деталей. Точность и полнота играют в этом процессе ключевую роль. Одним из главных преимуществ этого этапа является то, что всегда можно взять черновой проект, отложить его в сторону и начать все заново или просто внести желаемые поправки. Гораздо легче менять те или иные детали на этапе проектирования, чем иметь дело с проблемами уже реализованной производственной базы данных, которая плохо спроектирована.

Первая таблица «Регионы» хранит регионы, поля описываются в таблице 1.Состоит в связи «один ко многим» с таблицей «Населенные пункты».

Таблица 1 – Спецификация атрибутов таблицы «Регионы»

Наименование поля	Тип поля	Описание
код региона	целое	автоинкрементен, главный ключ
наименование региона	строка	подпись региона

Вторая таблица «Районы» хранит данные о всех районах в области, поля представлены в таблице 2. Заполняется после создания БД. Состоит в связи «один ко многим» с таблицей «Населенные пункты».

Таблица 2 – Спецификация атрибутов таблицы «Регионы»

Наименование поля	Тип поля	Описание
код района	целое	автоинкремент, главный ключ
название района	строка	наименование районов

Третья таблица «Населенные пункты» хранит все данные о всех населённых пунктах во всех районах в регионе. Заполняется после создания базы данных. Данная таблица включает в себя поля код региона и код района

для создания связи «один ко многим». Так же для вычисления коэффициента количество продаж к числу населению, присутствует поле «население» Описание полей приводится в таблице 3.

Таблица 3 – Спецификация атрибутов таблицы «Населённые пункты»

Наименование поля	Тип поля	Описание
код поселения	целое	автоинкремент, главный ключ
наименование поселения	строка	подпись поселения
код региона	целое	ключ из таблицы «Регионы»
код района	целое	ключ из таблицы «Районы»
население	целое	количество населения

Таблица четыре «Виды объекта» хранит все данные о всех типа построек, такие как квартира, дом, дача, коттедж и т.д.. Данная таблица включается в таблицу «Объявления» по связи «один ко многим». Описание полей приводится в таблице 4.

Таблица 4 – Спецификация атрибутов таблицы «Виды объекта»

Наименование поля	Тип поля	Описание
код вида объекта	целое	автоинкремент, главный ключ
наименование	строка	тип объекта

Пятая таблица «Материал стен» хранит все данные о всех материалах из которых строятся жилье. Данная таблица включается в таблицу «Объявления» по связи «один ко многим». Таблица дополняется при встрече не знакомого материала. Описание полей приводится в таблице 5.

Таблица 5 – Спецификация атрибутов таблицы «Материал стен»

Наименование поля	Тип поля	Описание
код материала	целое	автоинкремент, главный ключ
название материала	строка	наименование материала

Шестая таблица «Тип договора» хранит все данные о трех видах договора: продажа, аренда за сутки, аренда за месяц Данная таблица включается в таблицу «Объявления» по связи «один ко многим». Таблица заполняется при создании бд . Описание полей приводится в таблице 6.

Таблица 6 – Спецификация атрибутов таблицы «Тип договора»

Наименование поля	Тип поля	Описание
код договора	целое	автоинкремент, главный ключ
наименование договора	строка	название договора

Седьмая таблица «Объявления» хранит все данные о полученных объявлений с помощью парсинг. Таблица участвует со многими таблицами по связи «один ко многим». Заполняется таблица ежедневно. С «объявлений» данные выгружаются в Excel, Описание полей приводится в таблице 7.

Таблица 7 – Спецификация атрибутов таблицы «Объявления»

Наименование поля	Тип поля	Описание
код объявления	целое	автоинкремент, главный ключ
цена договора	целое	цена сделки по продаже или аренды
полный адрес	строка	адрес полученный со

		странички на сайте авито, который хранится в необработанном виде
дата	дата	дата, которая устанавливается при записи объявления
ссылка	строка	ссылка на объявление в интернете
просмотры	число	количество просторов объявления на авито пользователями
продавец	строка	имя продавца
ссылка на продавца	строка	ссылка на страницу профиля продавца на авито
этаж	число	номер этажа
этажность здания	число	число этажей в доме
количество комнат	число	общее число комнат у недвижимости
жилая площадь	число	жилая площадь объекта торга в квадратах
общая площадь	число	суммарная площадь объекта торга в квадратах
площадь кухни	число	кухонная площадь объекта торга в квадратах
дистанция до города	число	дистанция до столицы

		района в км
застройщик	строка	наименование застройщика, если таковой есть
онлайн	логический	имеется ли у продавца, возможность онлайн показа недвижимости
код материала	число	ключ из таблицы «Материал стен»
код вида объекта	число	ключ из таблицы «Виды объекта»
код населённого пункта	число	ключ из таблицы «Населенные пункты»
код типа договора	число	ключ из таблицы «Тип договора»

SQLскрипт для создания базы данных приведен в приложении Б.

3.3Работа ПО с базой данных.

Работа с базой данных осуществляется с помощью модуля PyMySQL в языке программирования Python. Данный модуль представляет собой «драйвер» для контакта с базой данных. Подключение к базе данных осуществляется по логину и пароль созданного пользователя, которые хранятся в файле конфигурации. Файл конфигурации хранит все данные, которые вызываются в программе. Эти данные влияют на поведение программы. Конфигурации имеет читаемый вид, что позволяет пользователь вносить свои поправки, таким образом пользователю нет обходимости редактировать код программы.

Таблицы «Регионы», «Районы» и «Поселения» заполняются после создания бд с помощью программы приведенной в приложении В. Данные для их наполнения берутся с сайта федеральной службы государственной

статистики – Росстата(https://rosstat.gov.ru/mission). При необходимости данные обновляются, при загрузке файла со статистикой в папку с программой. Так же эта группа таблиц выступает в качестве эталона наименований населений, районов и регионов, поэтому при получении адреса недвижимости с сайта, преобразуется к виду эталона.

Следующие таблицы «Тип договора», «Материал стен» и «Вид объекта» относятся к таблице «Объявления» по типу связи один-ко-многим.

Таблица «Объявления» выступает хранилищем всей информации о недвижимости полученной из страницы на сайте Авито. Перед сохранением в таблицу, данные проходят обработку и трансформации для приведения ее в единый формат, для удобства последующего анализа. Для уменьшения погрешности, при анализе, была создана таблица «Заблокированные объявления», а которую записываются идентификаторы объявления из таблицы «Объявления» в связи с дублированием объявления на недвижимость недобросовестными риелторами. Для этого объявление проходит проверку по параметрам этажность, площадь и т.д.

Таблица «История» используется для автоматической записи даты публикации объявления и даты снятия с публикации. Записи в эту таблицу вносятся с помощью триггеров, которые вызываются при выполнении вставки новой недвижимости или обновлении информации о недвижимости в таблице «Объявления».

Такой подход к построению связей таблиц позволит значительно упростит задачу по хранению данных о недвижимости, полученных и обработанных программой.

4СОЗДАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНЬЯ

4.1 Основные этапы программирования.

Определение проблемы. Перед тем как приступать к кодированию, необходимо четко сформулировать проблему, которую будущая программа должна решать. Так как, не имея хорошего определения проблемы, можно потратить много усилий и времени на решение не той проблемы, которую требуется решить.

На данном этапе проводится простая формулировка сути проблемы без каких-либо намеков на ее возможные решения, при этом формулировать ее следует на языке, понятном пользователю, т.е. она должна быть описана с пользовательской точки зрения.

Требования вырабатывают для того, чтобы свести к минимуму изменения системы после начала непосредственной разработки. Такие требования должны быть обязательно официальными, т.е. документально оформлены. Так как это гарантирует то, что функциональность системы определяется заказчиком, а не программистом. Даже в случае с внутрикорпоративными разработками такие требования должны быть зафиксированы, например, в виде технического задания, подписанного всеми задействованными лицами, тем самым избежать лишних разговоров и споров, например, о том, что реализованный функционал делает не все или не так.

На этапе создания плана разработки уже должны в формальном виде составить план разработки программного обеспечения с учётом существующей проблемы и выработанных требований. То есть составляется план того, как будем действовать дальше.

Детальное проектирование. На этом этапе проводится проектирование программы на низком уровне, иными словами, здесь проектируются классы и методы, рассматриваются, оцениваются и сравниваются различные варианты и причины выбора окончательных подходов и способов реализации.

При разработке небольших программ программисты обычно сами проектируют программу на таком уровне, это выглядит как написание псевдокода или рисование схем, поэтому часто этот этап рассматривается как часть непосредственного кодирования и в таких случаях итоговый документ (если того требует формальность) состоит преимущественно из различных набросков и заметок программистов.

Но при реализации крупных проектов данному процессу отводится отдельный этап и проектирование в этом случае проводится с очень высокой степенью детальности.

Кодирование. Это как раз тот этап, который все знают и, наверное, думают, что это единственный этап в процессе разработке программного обеспечения – это непосредственное написание кода и его отладка. Но это далеко не первый и не единственный этап разработки ПО.

Если все вышеперечисленные этапы выполнены, то данный этап подразумевает чисто механическую работу, т.е. кодинг. Программисту в этом случае не нужно что-то выдумывать и самостоятельно разрабатывать, ему нужно просто написать код, который реализует заданный, очень детально описанный в проекте, алгоритм.

После того как код написан, программисту необходимо отладить этот код, чтобы в нем не было никаких ошибок.

Тестирование всей системы. На данном этапе проводится тестирование всей системы, уже с учётом интеграции всех компонентов. На этом этапе можно выявить проблемы взаимодействия компонентов и устранить их. Также на этом этапе основным предметом тестирования является безопасность, производительность, утечка ресурсов и другие моменты, которые невозможно протестировать на более низких уровнях тестирования.

Сопровождение. После запуска программы в промышленную эксплуатацию осуществляется сопровождение этой программы, т.е. внесение изменений на основе выявленных недочетов в процессе эксплуатации

системы, а также проводится оптимизация функционала или добавление нового.

4.2 Получение данных о недвижимости с сайта Авито.

Программа по сбору и обработки данных о недвижимости осуществляется с помощью языка программирования Python[5]. Программа поделена на 4 основных блока. Стоит отметить, что разделение условно.

1. Первая часть логики программы – это получение html страницы с помощью библиотеки(модуля) Requests, которая позволяет выполнять разного рода http запросы на сервер. Проверку на состояния подключения. Имитация действий пользователя (добавление рандомного интервала ожидания между запросами). Получение страницы категории «Все квартиры» по ссылкеhttps://www.avito.ru/amurskaya_oblast_blagoveschensk/kvartiry/prodam-ASgBAgICAUSSA8YQ?cd=1. Для получения полной информации потребуются еще категории: «Квартиры в аренду», «Квартиры посуточно», «Дома, дачи, коттеджи». Они получатся с помощью http запроса по ссылкам методом «GET»[17]. В каждом разделе есть пагинация (номера станиц), при переходе на страницу 2 передается параметр «p=2» через метод «GET», где цифра 2 отвечает за номер страницы. Пагинацией называется разделение большого массива данных, имеющихся на сайте, на отдельные страницы для удобства использования (рис. 2). При получении количества страниц в категории, циклично получаются и сами страницы, передавая параметр[18] «p» с нужным номером. Таким образом происходит получения всех страниц в категории.

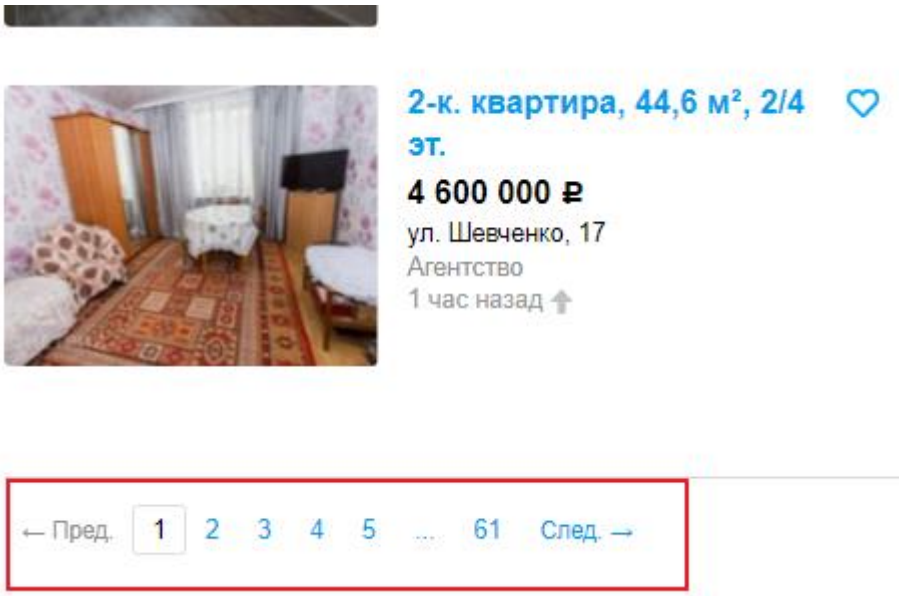

Рисунок 2 – Пагинация на сайте Авито

2. Вторая обработка полученной страницы, которая хранится в формате html[16]. Для работы с этим форматом у Python присутствует библиотека Beautiful Soup. Для анализа[6]html(страницы) помогает функция «Инструменты разработчика» в браузеры. В начале получается количество страниц в категории, которые находятся в теге «div»с классом «pagination-root-2oCjZ», в нем последний тег «span» содержит количество страниц (рис. 3). Далее получение информации самих объявлений. Делается аналогичным образом, как и анализ пагинации. Из тега объявления получается название объявления и ссылка на подробную страницу о нем (рис. 4). При выполнении httpзапроса на полученную ссылку получим страницу с подробным описанием[7] объявления.

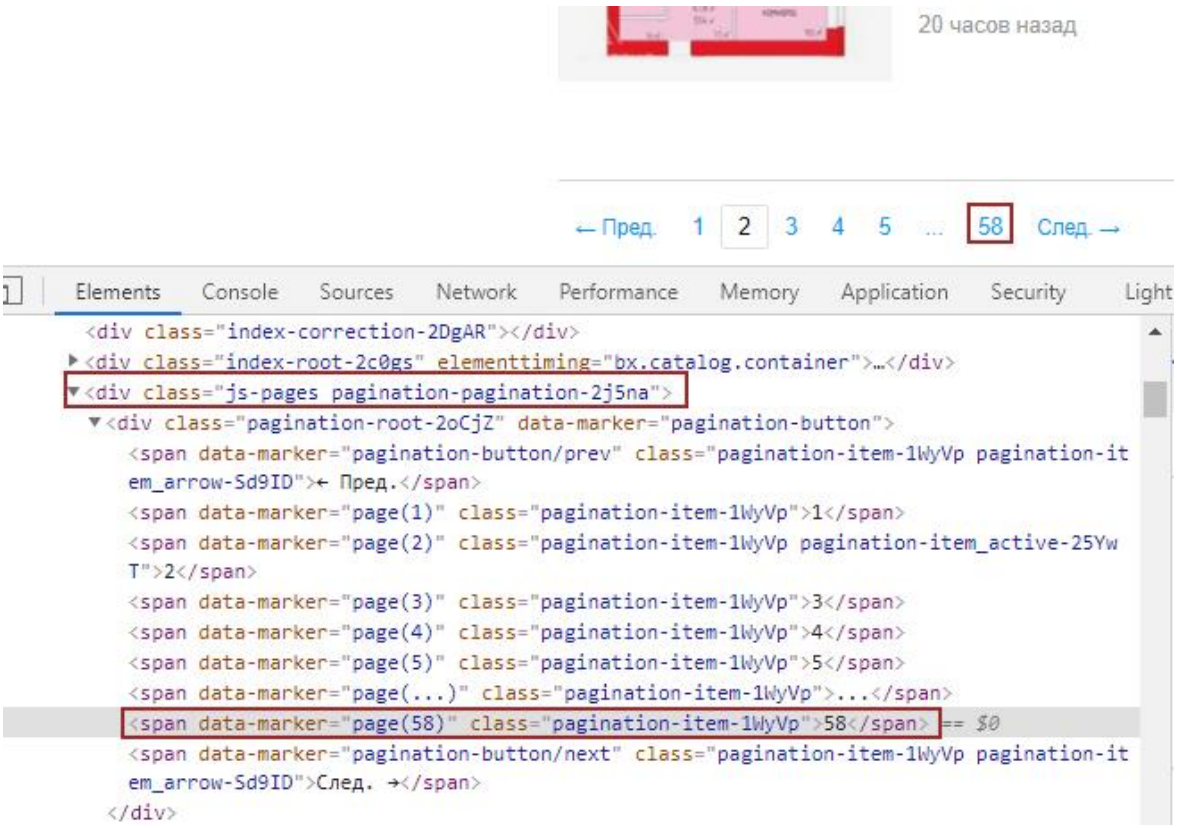

Рисунок 3 – Анализ пагинации для получения количества страниц

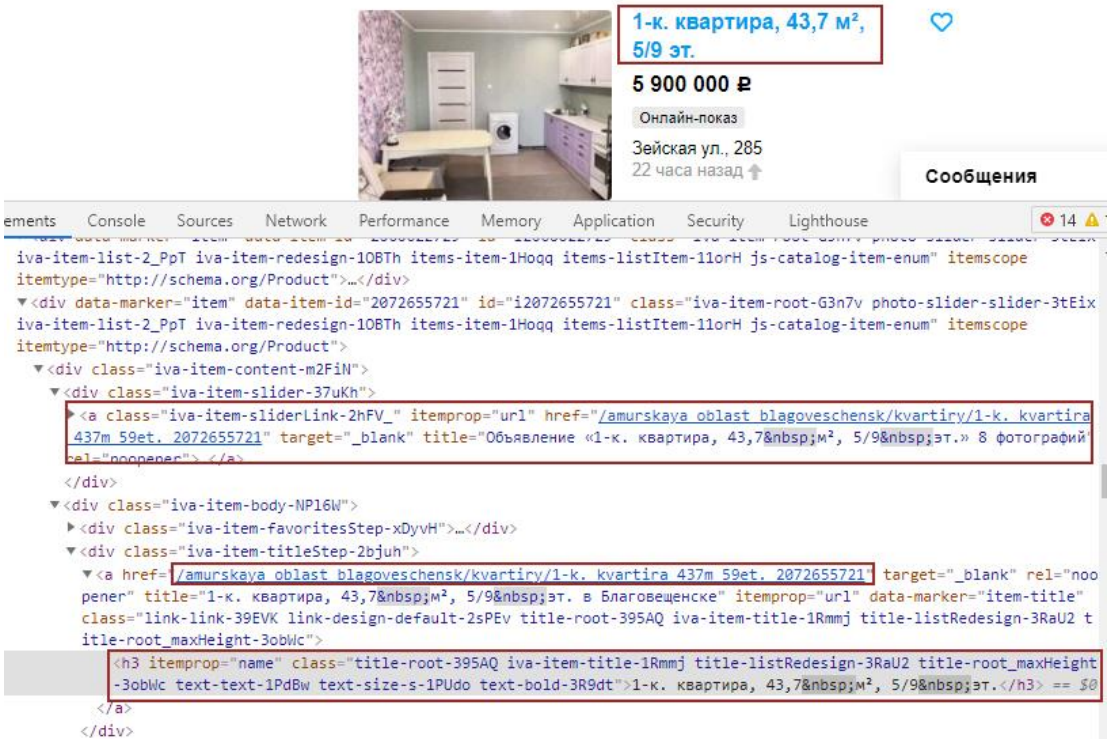

Рисунок 4 – Теги для получения названия и ссылки объявления

3. Третья часть программы работает с полученным данными и записывает их в базу данных[4]. Приводит данные к одному виду, так как на Авито нет единого формата отображения информации, такой как адрес. Приходится использовать кучу проверок и обработок адреса[10]. Пример: Получается адрес недвижимости из объявления «Амурская область, Благовещенск, Зейская ул., 285». Такой формат не подходит для построения сносок и сводных таблиц. Приводится к формату, Район, насланный пункт, улица, дом, кв., этаж. Благовещенский, Благовещенск, Зейская ул., 285.

Также для уменьшения погрешности [11] есть проверка на дубликат объявления, так как риелторы делаю огромное количество дубликатов одних и тех же объявлений. И так образом проходит обработка каждого объявления.

4. Последний блок – запись данных в базе данных.

Листинг программы приведет в приложении А.

4.3 Представление полученных данных о недвижимости в MicrosoftExcel.

Для представления аккумулированных данных используется табличный процессорMicrosoft Excel [12], так как он имеет мощные инструменты для создания сводных таблиц и построения графиков любой сложности.

Данные, которые хранятся в базе данных, получаются напрямую из базы данных [8] для Excel. Подключение к Excelосуществляется через вкладку «Данные», там выбирается пункт «Получение внешних данных», следующим пунктом идет «Из других источников», далее «С сервера SQL Server». Откроется «Мастер подключения к данным» в нем вводится пароль доступа к БД и заносится скрипт SQLдля выборки данных о недвижимости с базы данных(рис. 5).

После импорта данных о недвижимости с БД, они представлены одной таблицей(далее главная таблица). На основе этой таблицы строятся графики,

диаграммы, сводные таблицы и проводится глубокий анализ рынка недвижимости.

Сводные таблицы или сводки на основе главной таблицы, имеют возможность настройкиразных фильтров. Фильтры предоставляют возможность по управлению данными и настройки критерии отбора.

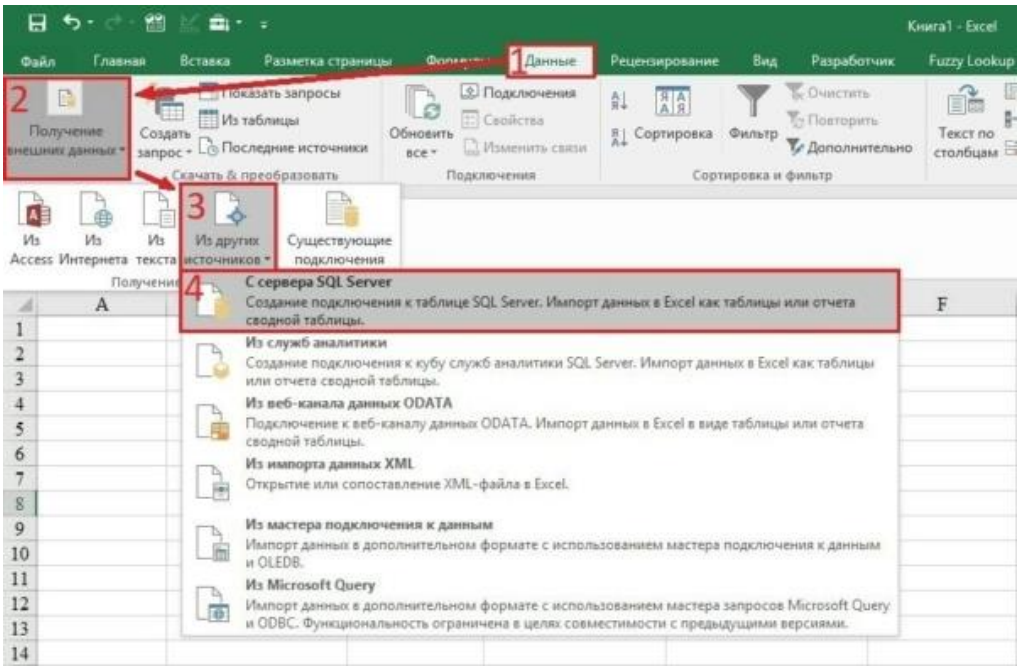

Рисунок 5– Подключение к БД в Excel.

Первая сводная таблица – это таблица населённые пункты (рис. 6). Она показывает количество объявлений, цена квадрата и сумму всей недвижимости. Таблица масштабируется от населенного пункта до района. Так же у таблицы есть два фильтра: по виду (тип договора) и первичка/вторичка. При изменении фильтров, меняются и данные в сводной таблице.

Вид	продажа		
Первичка/Вторичка	(Все)		
Названия строк	**Количество**	**Цена**	**Цена кв.**
⊞Архаринский	**52**	**58 541 403**	**17 139,0**
⊞Белогорский	**418**	**874 768 546**	**34 400,6**
⊟Благовещенский	**2 435**	**11 048 286 569**	**72 115,1**
г. Благовещенск	1 912	8 690 749 274	76 683,8
железнодорожная станция Призейская	1	800 000	16 000,0
посёлок Заречный	3	59 600 000	80 027,5
село Верхнеблаговещенское	15	57 640 000	58 488,7
село Владимировка	37	134 188 997	29 540,6
село Волково	26	82 489 000	33 963,1
село Грибское	10	27 050 000	38 348,2

Рисунок 6 – Сводная таблица по районам и населённым пунктам

Следующая сводная таблица «Количество комнат» показывает полное количество комнат у продаваемого объекта. Так же как и у предыдущей таблице по районам и населенным пунктам, имеется два фильтра.

11	Количество комнат	
12	Район	(Все)
13	Вид объекта	(Все)
14		
15	**Названия строк**	**Количество**
16	1	1 121
17	2	1 099
18	3	618
19	4	156
20	5	10
21	6	2
22	7	1
23	нет	2 142
24	своб, планировка	4
25	студия	278
26	**Общий итог**	**5 431**

Рисунок 7 – Сводная таблица «Количество комнат»

ЗАКЛЮЧЕНИЕ

Данная работы показывает, парсинг выступает хорошим элементом анализа рынка и вдругих видах деятельности.

В данной бакалаврской работе основное внимание уделялось созданию программного обеспечения для получения данных о рынке недвижимости.

Для достижения поставленной цели были решены следующие задачи:

- построены базы данных для хранения информации о недвижимости
- реализован алгоритм для получения информации с сайта Авито
- введена обработка информации для стандартизации полученных данных о недвижимости
- настроено подключение к базе данных в MicrosoftExcel
- проведена апробация данныхс помощью сводных таблиц в табличном процессоре Excel

Результаты проделанной работы апробированы на XXXнаучных конференциях «День науки АмГУ», а также в XXII региональной научно-практической конференции «Молодежь XXI века: шаг в будущее».По результатам работы опубликована научная статья [19].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Маккинли, Уэс Python и анализ данных / Уэс Маккинли; перевод А. Слинкина. – 2-е изд. – Саратов: Профобразование, 2019. – 482 с. – ISBN 978-5-4488-0046-7. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/88752.html

2 Программные системы статистического анализа. Обнаружение закономерностей в данных с использованием системы R и языка Python: учебное пособие / В. М. Волкова, М. А. Семёнова, Е. С. Четвертакова, С. С. Вожов. – Новосибирск: Новосибирский государственный технический университет, 2017. – 74 с. –ISBN 978-5-7782-3183-2. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/91682.html

3 Дроботун, Н. В. Алгоритмизация и программирование. Язык Python: учебное пособие / Н. В. Дроботун, Е. О. Рудков, Н. А. Баев. – Санкт-Петербург: Санкт-Петербургский государственный университет промышленных технологий и дизайна, 2020. – 119 с. –ISBN 978-5-7937-1829-5. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/102400.html

4 Каташинских, В. С. Методы сбора социальной информации: учебное пособие / В. С. Каташинских; под редакцией Ю. Р. Вишневского. – Екатеринбург: Издательство Уральского университета, 2017. – 124 с. –ISBN 978-5-7996-2069-1. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/106418.html

5 Креопалов, В. В. Технические средства и методы защиты информации: учебное пособие / В. В. Креопалов. – Москва: Евразийский открытый институт, 2011. – 278 с. –ISBN 978-5-374-00507-3. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/10871.html

6 Борисова, И. В. Цифровые методы обработки информации: учебное пособие / И. В. Борисова. – Новосибирск: Новосибирский государственный технический университет, 2014. – 139 с. –ISBN 978-5-7782-2448-3. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/45061.html

7 Туркин, В. С. Методы обработки маркетинговой информации: учебное пособие / В. С. Туркин. – Москва: Евразийский открытый институт, Московский государственный университет экономики, статистики и информатики, 2005. – 245 с. – ISBN 2227-8397. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/10722.html

8 Ревунков, Г. И. Базы и банки данных: методические указания по курсу «Банки данных» / Г. И. Ревунков. – Москва: Московский государственный технический университет имени Н.Э. Баумана, 2011. –69 с. – ISBN 2227-8397. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/30921.html

9 Ревунков, Г. И. Проектирование баз данных: учебное пособие по курсу «Банки данных» / Г. И. Ревунков. – Москва: Московский государственный технический университет имени Н.Э. Баумана, 2009. – 20 с. – ISBN 2227-8397. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/31513.html

10Кей, Хорстманн Scala для нетерпеливых / Хорстманн Кей; перевод А. Н. Киселев. – Саратов: Профобразование, 2019. – 414c. – ISBN 978-5-4488-0434-2. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/87976.html

11 Власовец, А. М. Основы информационных технологий решения экономических задач в табличном процессоре Excel: учебное пособие / А. М. Власовец, Е. А. Осипова, О. М. Сметкина. – Санкт-Петербург: Российский государственный гидрометеорологический университет, 2005. – 145 с. –

ISBN 5-7310-1762-X. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL:https://www.iprbookshop.ru/12510.html

12 Иванец, Г. Е. Табличный процессор MS Excel: учебное пособие / Г. Е. Иванец, Г. Е. Ивина. – Кемерово: Кемеровский технологический институт пищевой промышленности, 2007. – 107 с. – ISBN 978-5-89289-403-7. – Текст: электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. – URL:https://www.iprbookshop.ru/14391.html

13 Вычисления, графики и анализ данных в Excel 2010: самоучитель / М. П. Айзек, В. В. Серогодский, М. В. Финков, Р. Г. Прокди. – Санкт-Петербург: Наука и Техника, 2013. – 352 с. – ISBN 978-5-94387-921-0. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/35392.html

14 Пуговкин, А. В. Сети передачи данных: учебное пособие / А. В. Пуговкин. – Томск: Томский государственный университет систем управления и радиоэлектроники, 2015. – 138 с. – ISBN 2227-8397. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/72179.html

15 Борисова, И. В. Цифровые методы обработки информации: учебное пособие / И. В. Борисова. – Новосибирск: Новосибирский государственный технический университет, 2014. – 139 с. –ISBN 978-5-7782-2448-3. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/45061.html

16 Зудилова, Т. В. Web-программирование HTML / Т. В. Зудилова, М. Л. Буркова. – Санкт-Петербург: Университет ИТМО, 2012. –70 с. – ISBN 2227-8397. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/65748.html

17 Кириченко, А. В. Динамические сайты на HTML, CSS, Javascript и Bootstrap. Практика, практика и только практика / А. В. Кириченко, Е. В. Дубовик. – Санкт-Петербург: Наука и Техника, 2018. – 272 с. – ISBN 978-5-

94387-763-6. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/77578.html

18 Информационные технологии. HTML и XHTML: учебное пособие / А. И. Костюк, С. М. Гушанский, М. Ю. Поленов, Б. В. Катаев. – Таганрог: Издательство Южного федерального университета, 2015. – 131 с. –ISBN 978-5-9275-1329-1. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/78670.html

19Турышев, Д. А. Парсинг и обработка данных с помощью языка программирования Pythonи табличного процессора / Д. А. Турышев// Материалы XXII региональной научно-практической конференции «Молодежь XXI века: шаг в будущее». – 2021. – С. 788-789.

ПРИЛОЖЕНИЕ А

Код основной программы, которая осуществляет сбор и мониторинг объявлений в категории недвижимость.

```
import re
import requests
import csv
from bs4 import BeautifulSoup
import time
from datetime import datetime
from fake_useragent import UserAgent
import pymysql
from random import randint
from time import sleep

connection = pymysql.connect(
host="localhost",
user="vladimir",
password="_Lbvf_82461973",
db="avito_db",
database="avito_db",
port=3306
)
ALL_NEED_URL_FROM_CATEGORY={
'аренда домов':
'https://www.avito.ru/amurskaya_oblast/doma_dachi_kottedzhi/sdam-
ASgBAgICAUSUA9IQ',
'аренда квартир': 'https://www.avito.ru/amurskaya_oblast/kvartiry/sdam-
ASgBAgICAUSSA8gQ?cd=1',

'продажа квартир': 'https://www.avito.ru/amurskaya_oblast/kvartiry/prodam-
ASgBAgICAUSSA8YQ?cd=1',
'продажа
домов':'https://www.avito.ru/amurskaya_oblast/doma_dachi_kottedzhi/prodam-
ASgBAgICAUSUA9AQ?cd=1'
}
HEADERS = {
'User-Agent': 'Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36',
'upgrade-insecure-requests': '1',
'cookie': 'mos_id=CllGxlx+PS20pAxcIuDnAgA=; session-
cookie=158b36ec3ea4f5484054ad1fd21407333c874ef0fa4f0c8e34387efd5464a1e9500e22
77b0367d71a273e5b46fa0869a; NSC_WBS-QUBG-jo-nptsv-WT-
443=ffffffff0951e23245525d5f4f58455e445a4a423660; rheftjdd=rheftjddVal;
_ym_uid=1552395093355938562; _ym_d=1552395093; _ym_isad=2'
}

HOST='https://www.avito.ru'
FILE='price.csv'
Month=
{'января':1,'февраля':2,'марта':3,'апреля':4,'мая':5,'июня':6,'июля':7,'авгус
та':8,'сентября':9,'октября':10,'ноября':11,'декабря':12}
TYPES_OF_HOME=['квартира','студия', 'дом','коттедж','дача','студия','своб.
планировка']
flag_break_from_url=False
DISTRICT={'Архара':'Архаринский',
'Белогорск':'Белогорский',
'Благовещенск':'Благовещенский',
'Новобурейский':'Бурейский',
```

```
'Завитинск':'Завитинский',
'Зея':'Зейский',
'Ивановка':'Ивановский',
'Константиновка':'Константиновский',
'Магдагачи':'Магдагачинский',
'Новокиевский Увал':'Мазановский',
'Поярково':'Михайловский',
'Екатеринославка':'Октябрьский',
'Ромны':'Ромненский',
'Экимчан':'Селемджинский',
'Серышево':'Серышевский',
'Сковородино':'Сковородинский',
'Тамбовка':'Тамбовский',
'Тында':'Тындинский',
'Шимановск':'Шимановский',
'Свободный':'Свободненский',
'Райчихинск':'Райчихинск', #тут кончились районы
'Циолковский':'ЗАТО Циолковский',
'Прогресс':'г.о. Прогресс',
'Бурея': 'Бурейский'
}
# district_read_from_db={}
# lacality_read_from_db={}
# property read from db={}
# species_read_from_db={}
# material_read_from_db={}
G_O_= {'г.о. Прогресс':'Прогресс',
'г.о. Райчихинск':'Райчихинск',
'г.о. Город Зея':'Зейский',
'г.о. Благовещенск':'Благовещенский',
'г.о. Циолковский':'Циолковский'
}
CITY_ENG={
'belogorsk': 'Белогорск',
'blagoveschensk': 'Благовещенск',
'zavitinsk': 'Завитинск',
'zeya': 'Зея',
'raychihinsk': 'Райчихинск',
'svobodnyy': 'Свободный',
'skovorodino': 'Сковородино',
'tynda': 'Тында',
'ciolkovskij': 'Циолковский',
'shimanovsk': 'Шимановск',
'arhara':'Архара',
'progress':'Прогресс',
'novobureyskiy':'Новобурейский',
'tambovka':'Тамбовка',
'ekaterinoslavka':'Екатеринославка'
}
VILLAGE_BELONGS={
'Благовещенск':['Белогорье','Мухинка','Белогорье','Призейская','Плодопитомник
','Садовое','Чигири'],
'Белогорск':['Низинное'],
'Зея':['Широкий','Угольное','Зельвино'],
'Прогресс':['Кивдинский'],
'Завитинск':['Новоалексеевка','Червоная Армия',' Тур'],
'Архара':['Бон','Журавли'],
'Новобурейский':['Николаевка']
}
```

```
defcsv_dict_reader(path,delimiter=',',code='utf-8'): #?????? ?? csv ????a
with open(path, encoding=code) as file_obj:

reader = list(csv.DictReader(file_obj, delimiter=delimiter))
return reader

def writer_str_csv(data,path,type='w'): #?????? ? csv ????
with open(path,type, encoding='utf-8',newline='') as file:
        columns=data[0].keys()
        writer=csv.DictWriter(file,fieldnames=columns)
        writer.writeheader()
        writer.writerows(data)
return

def get_html(url, header=None, params=None): # Функциядляполучениястраницыпо
url
if header is None:
        header = HEADERS
while True:
try:
            r=requests.get(url, headers=header, params=params)
if r.status_code!=200:
print(r.status_code)
                sleep(randint(5,10))
continue
sleep(randint(5,10))
return r
except requests.exceptions.ConnectionError:
print('Connection Error')
        sleep(randint(5,10))

def get_pages_count(html):
    soup = BeautifulSoup(html, 'html.parser')
    pagination=soup.find_all('span', class_='pagination-item-1WyVp')
if pagination:
return int(pagination[-2].get_text())
else:
return 1

def get_last_link_ad(html):
pass

def read_from_txt(file):
    file=open(file,'r',encoding='utf-8')
    lines=file.readlines()
    lines=[line.rstrip('\n') for line in lines]
    file.close()
return lines

def check_type_of_home(str):
    str=str.lower()
for type_of_home in TYPES_OF_HOME:
if type_of_home in str:
return type_of_home
else:
return 'нет'
def writer_txt(feature,file_name,type='w'): #?????? ? ?????? ? ????????? ????
```

```
withopen(file_name,type, encoding='utf-8',newline='') as file:
        file.write(str(feature)+'\n')
```

```
return

def get_district_from_village(village):

village=village.replace('пос.','').replace('с.','').replace('село','').strip(
)
for city in VILLAGE_BELONGS.keys():
for inner_village in VILLAGE_BELONGS[city]:
if inner_village in village:
return {'Район':DISTRICT[city]}

def unification_address_with_population(address):
global population
    split_address={}
flag_break=False
split_address_from_line = address.split(',')
    first_district = 'нет'
second_district = 'нет'
split_address['Область'] = 'Амурскаяобл.'
for vilage in population:

        lacality=vilage['Населёныйпункт'].replace('г.
','').replace('г.','').replace('пгт ','').replace('село
','').replace('поселок ','').replace('посёлок
','').replace('железнодорожнаястанция ','')
        lacality=lacality.replace('железнодорожныйблок-пост
','').replace('железнодорожныйразъезд ','').replace('железнодорожныйблок-пост
','').replace('населённыйпунктБлок-Пост ','').rstrip().rstrip()
        district = vilage['Регион'].replace('Городскойокругпгт ',
'').replace('Городскойокругг.','').rstrip().rstrip()

for address_ad in split_address_from_line:
# if re.search(r''+lacality+'\b',address_ad):
            #     split_address['Пунктнаселения(Город,
селоит.д.)']=vilage['Населёныйпункт']

if lacality+' ' in address_ad+' ' and\
not lacality+'ий' in address_ad and\
not 'ул.' in address_ad and\
not 'ш.' in address_ad and\
not 'переулок' in address_ad and\
not 'пер.' in address_ad and\
not 'улица' in address_ad:
# if re.search(r''+lacality.lower()+'\b',address_ad.lower()):

                # split_address['Область']=vilage['Область']

                # split_address['Район']=vilage['Регион']
first_district = vilage['Регион']
                split_address['Пунктнаселения(Город,
селоит.д.)']=vilage['Населёныйпункт']
if district in address_ad:
                second_district=vilage['Регион']
# flag_break=True
                # break
        # if flag break:
        #     break
if second_district=='нет':
```

```
            split_address['Район']=first_district
elif second_district!=first_district:
```

```
split_address['Район']=second_district
else:
            split_address['Район']=first_district
# print(split_address)
return split_address

def get_split_address_from_url(url,district):
    address={'Пунктнаселения(Город, селоит.д.)':'нет',
'Район':district}
for city in CITY_ENG.keys():
if city in url:
            address['city']=CITY_ENG[city]
if district =='нет':
                address['Район']=DISTRICT[CITY_ENG[city]]
return address
return address

def get_split_address(address,url,show_d=False):
#функциявозвращаетадрессприведенныйкнормальномувиду
list_address={
'Улица':'нет',
'Номердома':'нет'
}
    address=address.split(',')
try:
for current_address in address:
            current_address = current_address.strip().strip()
if 'область' in current_address:
continue
            elif ('р-н' in current_address) or ('район' in current_address)
or ('р-н' in current_address):
continue
            elif current_address in DISTRICT.keys():
continue
            elif current_address in G_O_:
continue
            elif ('ул.' in current_address) or ('ш.' in current_address) or
('пр-т' in current_address) or (
'шоссе' in current_address) or ('улица' in current_address) or (
'ул ' in current_address) or (' ул' in current_address) or (
'пер.' in current_address) or (' пер' in current_address) or (
'б-р' in current_address) or (' лет ' in current_address):
                list_address['Улица'] = current_address
continue
            elif ('пос.' in current_address) or ('с.' in current_address) or
('село' in current_address):
continue
            else:
                list_address['Номердома'] = current_address
continue
    except:
print(address)
return list_address

def filling_empty_features(features):
    need_features=read_from_txt('features.txt')
```

```
for need_feature in need_features:
if need_feature in features.keys():
continue
        else:
            features[need_feature]='нет'
return features
def get_content(html,pages_count,page_count,category=''): #
возвращаетконтентизстраницы
soup = BeautifulSoup(html, 'html.parser')
    items = soup.find_all('div', class_='item_table-wrapper')
try:
        count_ad = int(soup.find('span', class_='page-title-count-
1oJOc').text.replace(' ', ''))
except:
print('Ошибкаприполучениеколичествообъявленийнастранице')
return []
if category=='':
        last_viewed=[]
else:
        last_viewed=read_from_txt(str(category)+'.txt')
    data=[]
for index, item in enumerate(items):
        features={}
if pages_count==page_count and ((pages_count - 1) * 50 + index) >= count_ad:
# if ((pages_count - 1) * page_count + index) >= count_ad:
print('Проверканаколичествостраницсработала')
break
print('Проходпообъявлениям: ', index + 1, ' из ', count_ad, '(', len(items),
')')
try:
            link = HOST + item.find('a', class_='snippet-link').get('href')
except:
            link = 'нет'
features['Ссылканаобъявление'] = link
if link in last_viewed:
global flag_break_from_url
            flag_break_from_url = True
            break
        try:
            title = item.find('div', class_='snippet-title-row').find('a',
class_='snippet-link').text.strip()
except:
            title = 'нет'
features['Название'] = title
        features['Видобъекта'] = check_type_of_home(title).title()
try:
online_view=0
online_view = item.find('div', class_='snippet-title-row').find('span',
class_='snippet-tag').text.strip()
            online_view = 1
except:
            online_view = 0
features['Возможностьонлайнпросмотра'] = online_view
try:
            date_all = item.find('div', class_='snippet-date-
info').get('data-tooltip').strip().split(' ')
if date_all == ['']:
                date_all = item.find('div', class_='snippet-date-
info').text.strip().split(' ')
```

```
date =str(datetime.now().year) + '.' + str(Month[date_all[1]]) + '.' +
date_all[0]
except:
            date = 'нет'
features['Датаразмещения'] = date
try:
            price_and_rent = item.find('div', class_='snippet-price-
row').text.strip().split('₽')
if (price_and_rent[1]) == '':
                type_ad = 'продажа'
else:
                type_ad = price_and_rent[1].strip().split('\n')[0].strip()
            price = price_and_rent[0].strip()
except:
            price = 0
type_ad = 'нет'
features['Цена'] = int(price.replace(" ",'').replace(" ",'').replace("
",'').replace(" ",'').replace(" ",'').replace(" ",'').replace(" ",''))
        features['Вид'] = type_ad
        data.append(features)
return data

def get_inner_content(url):
    html=get_html(url)
    soup = BeautifulSoup(html.text,'html.parser')
try:
        views=soup.find('div', class_='title-info-metadata-
views').text.rstrip().split('(+')
        views=int(views[0])+int(views[1].replace(')', ''))
except:
        views=0
try:
        id_ad=soup.find('div',class_='item-view-search-info-
redesign').find('span').text.replace('№','').strip()
        id_ad=int(id_ad)
except:
        id_ad=0
try:
        link_seller = HOST + soup.find('div', class_='seller-info-name js-
seller-info-name').find('a').get('href')
except:
        link_seller = 'нет'
secondary = 0
try:
        spans_of_secondary=soup.find('div', class_='item-
navigation').find('div',class_='breadcrumbs').find_all('span')
for span_of_secondary in spans_of_secondary:
if 'Новостройки' in span_of_secondary.text:
                secondary=1
continue
    except:
print('Ошибкаприполученииисвойствапервичка/вторичка')
try:
        name_seller=soup.find('div',class_='seller-info-name js-seller-info-
name').text.strip()
except:
        name_seller='нет'
try:
```

```
address_inner=soup.find('span',class_='item-address__string').text.strip()
        split_address=unification_address_with_population(address_inner)
        split_address.update(get_split_address(address_inner,url))
except Exception as e:
        split_address = {'Область': 'нет',
'Район': 'нет',
'Пунктнаселения(Город, селоит.д.)': 'нет',
'Улица': 'нет',
'Номердома': 'нет'}
print(e.__class__)
        writer_txt(':  '+str(e.__class__)+'\n', 'log.txt', 'a')
# writer_txt('Ошибкапричтенииадреса'+datetime.now(),'log.txt','a')
address_inner='нет'
propertys=soup.find_all('li',class_='item-params-list-item')
    features={}
for feature in propertys:
try:
            params=feature.text.strip().split(': ')
            name_feature=params[0].strip()
            value_feature=params[1].strip()
if name_feature=='Площадьдома':
                name_feature='Общаяплощадь'
if name_feature=='Типдома':
                name_feature='Материалстен'
if name_feature=='Этаж':
                value_feature=value_feature.split(' из ')
                features['Этажейвдоме']=value_feature[1].strip()
                features[name_feature]=value_feature[0].strip()
continue
features[name_feature]=value_feature.rstrip('\xa0м²').rstrip('\xa0км').rstrip
('сот.').rstrip(' м²').rstrip('\xa0сот.').rstrip('\xa0сот.').replace('.',',')

except:
continue
features['Просмотров']=views
    features['id_ad']=id_ad
    features['Имяпродовца']=name_seller
    features['Ссылканапрдовца']=link_seller
    features['АдресПолный']=address_inner
    features['Первичка/Вторичка']=secondary
    features.update(split_address)
return features
def check_from_verified_ad(curret_ad):
    mycursor = connection.cursor()
    sql = 'SELECT linkAd,status,idAds,price FROM avito_db.ads where linkAd=
"' + curret_ad[
'Ссылканаобъявление'] + '";'
try:
        mycursor.execute(sql)
        query = mycursor.fetchall()
except Exception as e:
print(e.__class__)
print('Ошибка: проверкунаналичиевтаблицеад')
        writer_txt('Ошибка: проверкунаналичиевтаблицеад','log.txt','a')
        writer_txt('Ошибка: проверкунаналичиевтаблицеад - ' +
str(datetime.now()) + ' - ' + curret_ad[
'Ссылканаобъявление'], 'log.txt', 'a')
        writer_txt(e.__class__, 'log.txt', 'a')
```

```
return False

status_of_original=1

if len(list(query)) != 0:
        ans = list(query[0])
        status_of_original = ans[1]
id_of_original = ans[2]
if ans:
if ans[1] == 0:
                sql = "UPDATE `ads` set `status`='1',`price`='" +
str(curret_ad['Цена']) + "' WHERE (`idAds` = '" + str(
                    ans[2]) + "');"
try:
                    mycursor.execute(sql)
                    connection.commit()
except Exception as e:
print(e.__class__)
print('Ошибка: обновление (наналичиевтаблицеад)')
                    writer_txt('Ошибка: обновление (наналичиевтаблицеад) -
'+str(datetime.now())+' - '+curret_ad['Ссылканаобъявление'],'log.txt','a')
                    writer_txt(e.__class__, 'log.txt', 'a')
return False
            return False

sql = 'SELECT linkAd,status,idAds,price FROM avito_db.adsbanned where linkAd=
"' + curret_ad[
'Ссылканаобъявление'] + '";'
try:
        mycursor.execute(sql)
        query = mycursor.fetchall()
if len(list(query)) != 0:
            ans_b = list(query[0])
if ans_b:
if status_of_original == 0:
                    sql = "UPDATE `ads` set `status`='1',`price`='" +
str(curret_ad['Цена']) + "' WHERE (`idAds` = '" + str(ans[2]) + "');"
mycursor.execute(sql)
return False
            return False
    except Exception as e:
print(e.__class__)
        writer_txt(e.__class__, 'log.txt', 'a')
        writer_txt('Ошибка: обновление (наналичиевтаблицеадбан) - ' +
str(datetime.now()) + ' - ' + curret_ad[
'Ссылканаобъявление'], 'log.txt', 'a')
return False
    return True

def check_from_writer(curret_ad):
    mycursor=connection.cursor()
try:
        sql = 'SELECT
price,`numberOfFloors`,`floor`,`totalArea`,`addressFull`,idAds FROM
avito_db.ads;'
mycursor.execute(sql)
except:
        writer_txt('Ошибка')
```

```
clear_data_of_ads = mycursor.fetchall()
# try:
forsaved_adinclear_data_of_ads:
saved_ad=list(saved_ad)
ifcurret_ad['Цена'] == saved_ad[0]  andcurret_ad['Этажейвдоме']==saved_ad[1]
andcurret_ad['Этаж']==saved_ad[2] andcurret_ad['Общаяплощадь'] != saved_ad[3]
andcurret_ad['АдресПолный']==saved_ad[4]:
writer_db(curret_ad,['idOfOriginalAd'],[str(saved_ad[5])],'adsbanned')
# if
returnFalse
# exceptExceptionase:
    #     print('Ошибкаприпроверкинабан')
    #     print(e.__class__)
    #     writer_txt(e.__class__, 'log.txt', 'a')
    #     returnFalse
returnTrue

defwriter_db(line,fieldnames=[],values=[],table="ads"):
#записываетполученныеданныевбд
try:
# globaldistrict_read_from_db
        # globallacality_read_from_db
mycursor = connection.cursor()

sql = "INSERTINTO "+table+"
(`name`,`price`,`addressFull`,`primary`,`date`,`linkAd`,`views`,`idAdInSite`,
`seller`,`linkSeller`,`floor`,`numberOfFloors`,`numberOfRooms`,`livingSpace`,
`totalArea`,`kithenArea`,`yearOfConstruction`,`finishing`,`developer`,`nameOf
TheNewBulding`,`thePlots`,`distanseToCity`,`online`,`ownership`,`idKindOfProp
erty`,`species_idSpecie`,`lacality_idlacality`,`idMaterial`"
forfieldnameinfieldnames:
sql+=", `"+fieldname+"`"

sql+=") VALUES ("

sql+="'"+(line['Название'])+"'"
sql+=","+"'"+str(line['Цена'])+"'"
sql+=","+"'"+(line['АдресПолный'])+"'"
sql+=","+"'"+str(line['Первичка/Вторичка'])+"'"
sql+=","+"'"+(line['Датаразмещения'])+"'"
sql+=","+"'"+(line['Ссылканаобъявление'])+"'"
sql+=","+"'"+str(line['Просмотров'])+"'"
sql+=","+"'"+str(line['id_ad'])+"'"
sql+=","+"'"+(line['Имяпродовца'])+"'"
sql+=","+"'"+(line['Ссылканапрдовца'])+"'"
sql+=","+"'"+(line['Этаж'])+"'"
sql+=","+"'"+(line['Этажейвдоме'])+"'"
sql+=","+"'"+(line['Количествокомнат'])+"'"
sql+=","+"'"+(line['Жилаяплощадь'])+"'"
sql+=","+"'"+(line['Общаяплощадь'])+"'"
sql+=","+"'"+(line['Площадькухни'])+"'"
sql+=","+"'"+(line['Годпостройки'])+"'"
sql+=","+"'"+(line['Отделка'])+"'"
sql+=","+"'"+(line['Официальныйзастройщик'])+"'"
sql+=","+"'"+(line['Названиеновостройки'])+"'"
sql+=","+"'"+(line['Площадьучастка'])+"'"
sql+=","+"'"+(line['Расстояниедогорода'])+"'"
sql+=","+"'"+str(line['Возможностьонлайнпросмотра'])+"'"
sql+=","+"'"+(line['Правособственности'])+"'"
```

```
sql+=","+"'"+str(property_read_from_db[line['Видобъекта'].lower()])+"'"
sql+=","+"'"+str(species_read_from_db[line['Вид']])+"'"
# try:
        #     line['Пунктнаселения(Город, селоит.д.)']

# except:
        #     print(line)
        # if 'г.' in line['Пунктнаселения(Город, селоит.д.)'] and not 'г. '
in line['Пунктнаселения(Город, селоит.д.)']:
        #     line['Пунктнаселения(Город,
селоит.д.)']=line['Пунктнаселения(Город, селоит.д.)'].replace('г.','г. ')
get_id_of_lacality='SELECT idlacality FROM avito_db.lacality inner join
district on (lacality.idDistrict=district.idDistrict) inner join regions on
(lacality.idRegion=regions.idRegion) where
lacality.name="'+line['Пунктнаселения(Город, селоит.д.)']+'" and
district.name = "'+line['Район']+'";'
# print(get_id_of_lacality)
mycursor.execute(get_id_of_lacality)
        list_of_lacality=list(mycursor.fetchall())
# print('list: ',list_of_lacality)
        # for x in list_of_lacality:
try:
            id_of_lacality=list(list_of_lacality[0])[0]
# print('id: ', id_of_lacality)

except:
            id_of_lacality=list_of_lacality
print('Ошибкаприполучении id')
        sql+=","+"'"+str(id_of_lacality)+"'"
# print(id_of_lacality)
sql+=","+"'"+str(material_read_from_db[line["Материалстен"]])+"'"
for value in values:
            sql+=","+"'"+str(value)+"'"
sql+=");"
mycursor.execute(sql)
print('Строказаписанна')

        connection.commit()
except:
        writer_txt(sql,'write_bag.txt','a')
        writer_txt(line,'write_bag.txt','a')
        writer_txt(datetime.now(),'write_bag.txt','a')
print("Ошибкапризаписивбд: ", sql)

def parse(category):
# data=[]
global flag_break_from_url
    flag_break_from_url = False
url=ALL_NEED_URL_FROM_CATEGORY[category]
    html=get_html(url)
    data=[]
if html.status_code==200:
try:
            pages_count=get_pages_count(html.text)
# pages_count=1

except:
print('Ошибкаприполучениистраниц')
```

```
pages_count=1
```

```
# pages_count=5
for page in range(1,pages_count+1):
print(f'Парсингстраницы{page}из{pages_count}')
            html=get_html(url, params={'p':page})
            data+=get_content(html.text,pages_count,page,category)
if  flag_break_from_url:
break

        if data==[]:
print("Зашлаприпустом data листе")
return

        for index,line in enumerate(data):

            writer_txt(line['Ссылканаобъявление'], str(category) + '.txt',
'a')

# mycursor = connection.cursor()
            # sql = 'SELECT linkAd,status,idAds,price FROM avito_db.ads where
linkAd= "' + line[
            #     'Ссылканаобъявление'] + '";'
            # mycursor.execute(sql)
            # query = mycursor.fetchall()
            # if len(list(query)) != 0:
            #     if list(query)[1]==1:
            #         continue
if check from verified ad(line)==False:
print('Проверканауникальность')
continue

line.update(get_inner_content(line['Ссылканаобъявление']))
            line = filling_empty_features(line)
if line['Район']=='нет' :
print('Непрошлапоадресу')
print(line)
                writer_txt(line['Ссылканаобъявление'],'emtyaddress.txt','a')
continue
# writer_txt(line, str(category) + '.txt')
print('ПРоверкаданныхдлязаписи: ',index,' из ',len(data))
if check_from_writer(line):
print('Прошла')
# writer_str_csv(line,'new_data.csv','a')
                # try:
writer_db(line)
continue
# except Exception as e:
                #     print(e.__class__)
                #     writer_txt(e.__class__, 'log.txt', 'a')
print('Строканепрошланазапись')


else:
print('Неудалосьполучить html')

def check_status_ads():
# data=[]
global flag_break_from_url
    data = []
```

```
forcategory in ALL_NEED_URL_FROM_CATEGORY.keys():

flag_break_from_url = False
url=ALL_NEED_URL_FROM_CATEGORY[category]
        html=get_html(url)
if html.status_code==200:
try:
                pages_count = get_pages_count(html.text)
except:
print('Ошибкаприполученииистраниц')
# pages count = 1
            # pages count=1
for page in range(1,pages_count+1):
print(f'Проверкастатусов: страница{page}из{pages_count}')
                html = get_html(url, params={'p': page})
                data+=get_content(html.text,pages_count,page)
if  flag_break_from_url:
break
        else:
print('Неудалосьполучить html')
        sleep(30)
    mycursor = connection.cursor()
    sql='SELECT linkAd,price,status FROM avito_db.ads;'
mycursor.execute(sql)
    ads_query=mycursor.fetchall()
    ads=[ list(i) for i in ads_query]
data_clear=[]
    count_enters=0
count_write=0
for ad in ads:
        flag_status_ad=False
# print(ad[0])
for line in data:
# print(line['Ссылканаобъявление'])
if line['Ссылканаобъявление'] in ad[0]:
print('ПРоверкасработалананаличиевнайденомсписке')
                flag_status_ad=True
count_enters+=1
break
        if not flag_status_ad:
            sql="update ads set status=0 where linkAd='"+ad[0]+"'"
# print('Записьнаобновление')
try:
                mycursor.execute(sql)
print('Произошлообновление')
                count_write+=1
except:
print('Ошибкаприобновлениистатусавобъявления')

writer_txt('Ошибкаприобновлениистатусавобъявления'+str(datetime.now()),'log.t
xt','a')
print('Всегообъявленийв data листе: ', len(data))
print('Всегообъявлениевбд(ads): ', len(ads))
print('Количествосработанныхпроверок, наличиевнайденомсписке:
',count_enters)
print('Количествоназапись: ', count_write)
    data_clear=data
for index, line in enumerate(data_clear):
if check_from_verified_ad(line) == False:
```

```
print('Проверканауникальность(статус)')
continue
```

```
line.update(get_inner_content(line['Ссылканаобъявление']))
line = filling_empty_features(line)
ifline['Район'] == 'нет':
print('Непрошлапоадресу')
print(line)
writer_txt(line['Ссылканаобъявление'], 'emtyaddress.txt', 'a')
continue
print('Проверкаданныхдлязаписи (статус): ', index, ' из ', len(data_clear))
ifcheck_from_writer(line):
writer_db(line)

continue
print('Строканепрошланазапись')
return

if __name__ == "__main__":
global district_read_from_db
global property_read_from_db
global species_read_from_db
global material_read_from_db
global lacality read from db
global population
    population=csv_dict_reader('population_data.csv')


    district_read_from_db = {}
    lacality_read_from_db = {}
    property_read_from_db = {}
    species_read_from_db = {}
    material_read_from_db = {}
try:
with connection.cursor() as cursor:
            query = 'SELECT * FROM district'
cursor.execute(query)

for row in cursor:
                district_read_from_db[row[1]] = row[0]


with connection.cursor() as cursor:
            query = 'SELECT * FROM lacality'
cursor.execute(query)
for row in cursor:
                lacality_read_from_db[row[1]] = row[0]

with connection.cursor() as cursor:
            query = 'SELECT * FROM kindofpropertys'
cursor.execute(query)
for row in cursor:
                property_read_from_db[row[1]] = row[0]


with connection.cursor() as cursor:
            query = 'SELECT * FROM buildingmaterials'
cursor.execute(query)
for row in cursor:
                material_read_from_db[row[1]] = row[0]

with connection.cursor() as cursor:
```

```
            query = 'SELECT * FROM species'
cursor.execute(query)
for row in cursor:
                species_read_from_db[row[1]] = row[0]
except:
print('Неудачапричтениипервыхсвойств')
    flag_check_all_close_ads=True
    while True:

if False and (flag_check_all_close_ads and
(datetime(1,1,1,1,1,1).time()>datetime.now().time() or
(datetime(1,1,1,13,1,1).time()< datetime.now().time() and
datetime(1,1,1,15,1,1).time()> datetime.now().time()))):

print('Проверкастатусов ')
            check_status_ads()
            flag_check_all_close_ads=False
        if not flag_check_all_close_ads and
((datetime(1,1,1,2,1,1).time()<datetime.now().time() and
datetime(1,1,1,13,1,1).time()>datetime.now().time()) or
datetime(1,1,1,15,1,1).time() < datetime.now().time()):
            flag_check_all_close_ads=True
        for category in ALL_NEED_URL_FROM_CATEGORY.keys():
print(category.upper())
            parse(category)
            sleep(randint(2,5))
        sleep(60)
```

ПРИЛОЖЕНИЕ Б

SQLскрипт, который создаст базу данных со всеми взаимосвязями, процедурами, макросами и т.д.

```
-- MySQL Script generated by MySQL Workbench
-- Fri Jun  4 03:44:31 2021
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema avito_db
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema avito_db
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `avito_db` DEFAULT CHARACTER SET utf8 ;
USE `avito_db` ;

-- -----------------------------------------------------
-- Table `avito_db`.`district`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `avito_db`.`district` (
```

```
  `idDistrict` INT NOT NULL AUTO_INCREMENT,
```

ПродолжениеПриложенияБ

```
  `name` VARCHAR(100) NOTNULL,
PRIMARY KEY (`idDistrict`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `avito_db`.`regions`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `avito_db`.`regions` (
  `idRegions` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`idRegions`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `avito_db`.`lacality`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `avito_db`.`lacality` (
  `idlacality` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL,
  `idDistrict` INT NOT NULL,
  `idRegion` INT NOT NULL,
  `population` INT NULL,
  PRIMARY KEY (`idlacality`),
  INDEX `fk_lacality_district_idx` (`idDistrict` ASC) VISIBLE,
  INDEX `fk_lacality_regions1_idx` (`idRegion` ASC) VISIBLE,
  CONSTRAINT `fk_lacality_district`
    FOREIGN KEY (`idDistrict`)
    REFERENCES `avito_db`.`district` (`idDistrict`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
```

```
  CONSTRAINT `fk_lacality_regions1`
```



```
    FOREIGN KEY (`idRegion`)
    REFERENCES `avito_db`.`regions` (`idRegions`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `avito_db`.`kindOfPropertys`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `avito_db`.`kindOfPropertys` (
  `idKindOfProperty` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idKindOfProperty`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `avito_db`.`species`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `avito_db`.`species` (
  `idSpecie` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`idSpecie`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `avito_db`.`ads`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `avito_db`.`ads` (
  `idAds` INT NOT NULL AUTO_INCREMENT,
```

```
`price` INTNOTNULL,
```

ПродолжениеПриложенияБ

```
`addressFull` VARCHAR(45) NULL,
  `primary` TINYINT NOT NULL,
  `date` DATETIME NOT NULL,
  `linkAd` VARCHAR(200) NOT NULL,
  `views` INT NOT NULL,
  `idAdInSite` VARCHAR(45) NOT NULL,
  `seller` VARCHAR(100) NOT NULL,
  `linkSeller` VARCHAR(200) NOT NULL,
  `floor` INT NULL,
  `numberOfFloors` INT NOT NULL,
  `numberOfRooms` INT NULL,
  `livingSpace` INT NULL,
  `totalArea` INT NOT NULL,
`kithenArea` INT NULL,
  `yearOfConstruction` INT NULL,
  `finishing` VARCHAR(45) NULL,
  `developer` VARCHAR(150) NULL,
  `nameOfTheNewBulding` VARCHAR(150) NULL,
  `thePlots` INT NULL,
  `distanseToCity` INT NULL,
  `online` TINYINT NOT NULL,
  `ownership` VARCHAR(45) NOT NULL,
  `idKindOfProperty` INT NOT NULL,
  `species_idSpecie` INT NOT NULL,
  `lacality_idlacality` INT NOT NULL,
  PRIMARY KEY (`idAds`),
  UNIQUE INDEX `linkAd_UNIQUE` (`linkAd` ASC) VISIBLE,
  UNIQUE INDEX `idAdInSite_UNIQUE` (`idAdInSite` ASC) VISIBLE,
  INDEX `fk_ads_kindOfPropertys1_idx` (`idKindOfProperty` ASC) VISIBLE,
  INDEX `fk_ads_species1_idx` (`species_idSpecie` ASC) VISIBLE,
  INDEX `fk_ads_lacality1_idx` (`lacality_idlacality` ASC) VISIBLE,
  CONSTRAINT `fk_ads_kindOfPropertys1`
```

```
    FOREIGN KEY (`idKindOfProperty`)
```



```
    REFERENCES `avito_db`.`kindOfPropertys` (`idKindOfProperty`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_ads_species1`
    FOREIGN KEY (`species_idSpecie`)
    REFERENCES `avito_db`.`species` (`idSpecie`)
ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_ads_lacality1`
    FOREIGN KEY (`lacality_idlacality`)
    REFERENCES `avito_db`.`lacality` (`idlacality`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `avito_db`.`history`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `avito_db`.`history` (
  `idHistory` INT NOT NULL,
  `dateStart` DATETIME NOT NULL,
`dateEnd` DATETIME NOT NULL,
  `idAds` INT NOT NULL,
  PRIMARY KEY (`idHistory`),
  INDEX `fk_history_ads1_idx` (`idAds` ASC) VISIBLE,
  CONSTRAINT `fk_history_ads1`
    FOREIGN KEY (`idAds`)
    REFERENCES `avito_db`.`ads` (`idAds`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
SETSQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```