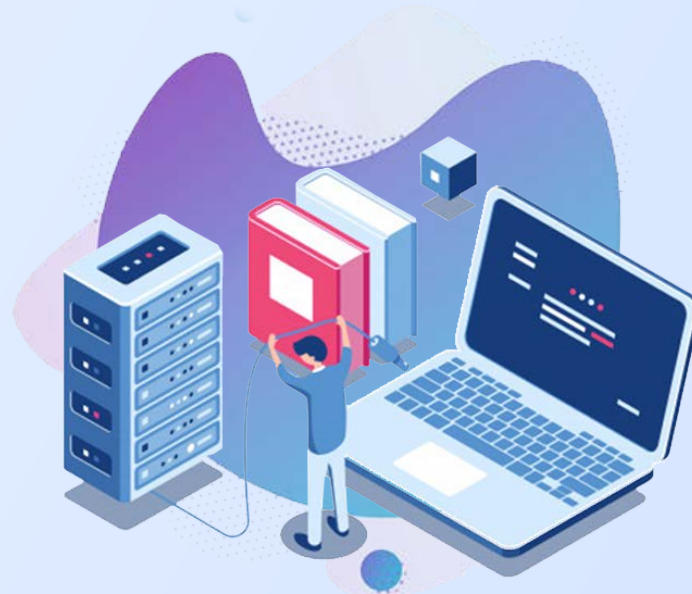




НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЛАБОРАТОРИЯ
ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Сергей Левашкин ЗНЧ-2023



ФЕРМА ДАННЫХ



'Ферма данных' - это программно-инструментальная система, предназначенная для сбора и анализа данных из структурированных и неструктурированных открытых источников сети Интернет для получения из нее информации в виде заданных параметров.

Уникальность проекта состоит в том, что у него быстро разворачиваемый сбор данных, построенный на Open-Source компонентах. Система также содержит модуль оповещений в мессенджеры о работе всех используемых в ней скриптов. Это модульная система, построенная по принципу конструктора «лего»: любой компонент может быть заменен другим, если результаты работы системы не будут удовлетворительными.

Области применения данного проекта различны: легкая перестройка и масштабирование системы под разные задачи, сбор и анализ данных в любой предметной области. Система была протестирована на данных по Ковид-19, которые собирались, начиная с марта 2020 г.

Все данные были визуализированы и проанализированы, как методами искусственного интеллекта, так и с помощью оригинальной прогнозной математической модели эпидемии.

Проведенный патентный поиск показал, что аналога фермы данных не существует

<https://lab.ci.ru/docs#patent-search>

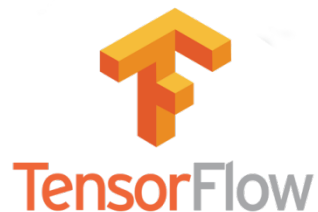


Публикации по теме

- Математическая биология и биоинформатика. *Исследование адаптивно-компарментной модели распространения КОВИД-19 в некоторых регионах РФ методами оптимизации*
https://www.matbio.org/article.php?journ_id=38&id=477
- CONTROLO 2022. *Human-Machine Interaction for Monitoring COVID-19 Internet Data in Russia and the World*
https://link.springer.com/chapter/10.1007/978-3-031-10047-5_30
- Информационное общество. *Информационная система для визуализации данных КОВИД-19 и изучения контрольных параметров модели распространения пандемии в России и мире*
[№3 30.06.2023](#)
- Труды ИСП РАН. *Ферма данных: Информационная система сбора, хранения и обработки неструктурированных данных из разнородных источников*
https://www.researchgate.net/publication/369189085_Data_farm_Information_system_for_collecting_storing_and_processing_unstructured_data_from_heterogeneous_sources

Патенты и свидетельства

- Патент № 2022617423. Система получения данных со «СтопКоронаВирус» через репозиторий github
- Патент № 2022617725. Модульная система сбора данных
- Патент № 2022617780. Бэкап система и система оповещений проекта «Вирусы»
- Патент № 2022618496. Система получения сообщений по ключевым словам
- Патент № 2022618430. Система сбора данных по параметрам с Росстат
- Патент № 2022618365. Система получения данных по вакцинации из открытых источников



КОМПОНЕНТЫ АРХИТЕКТУРЫ

В основу легли следующие программные компоненты:

Ubuntu server 20.4.01 LTS

MongoDB 4.4.1

Conda 4.9.0

Jupyter-notebook 6.0.3

Python 3.8.3

и многое другое







Преимущества MongoDB:

- Легкая масштабируемость
- Гибкая схема добавления документа
- Хранение данных в формате JSON
- Возможность глубоких запросов

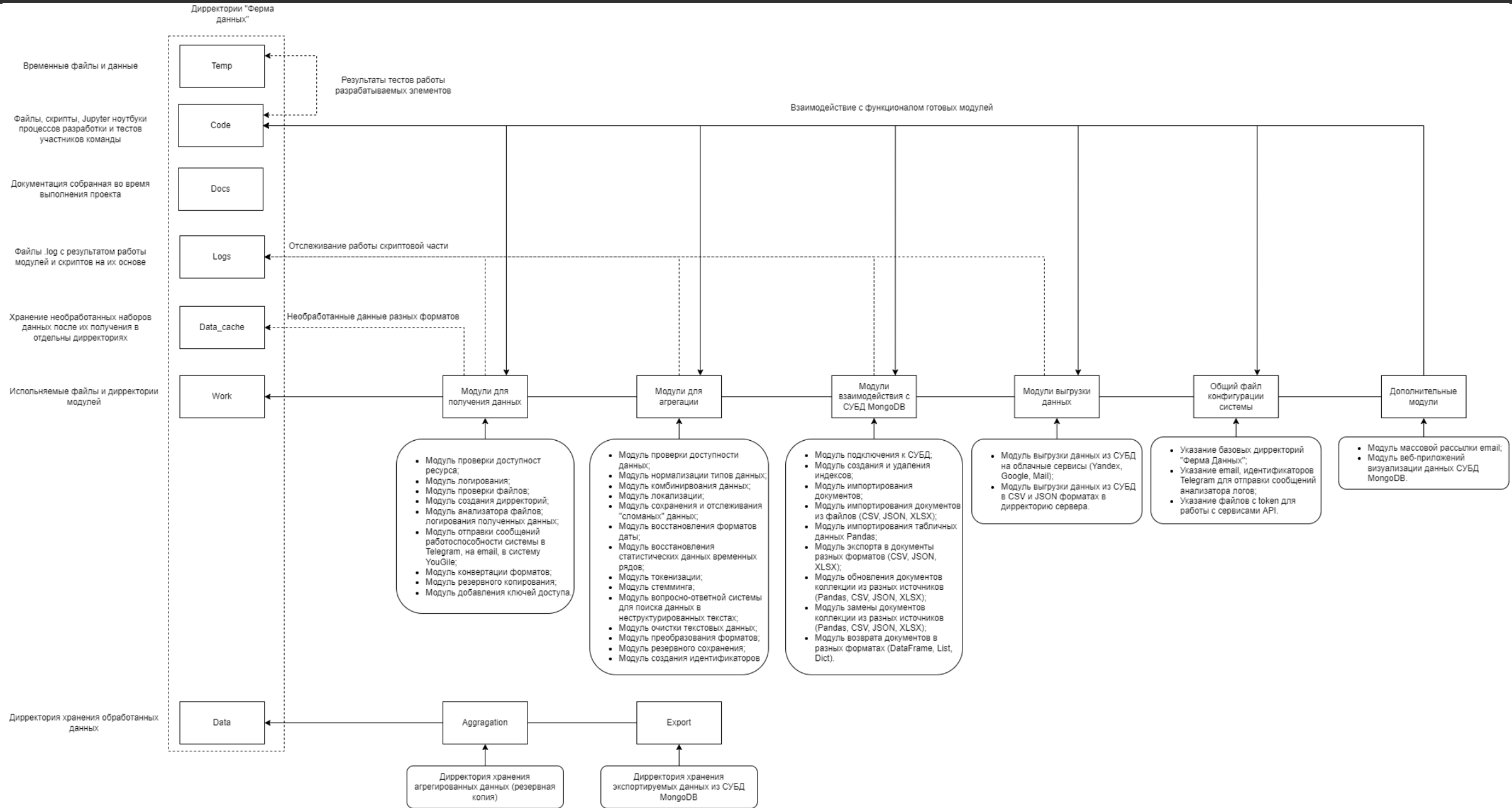


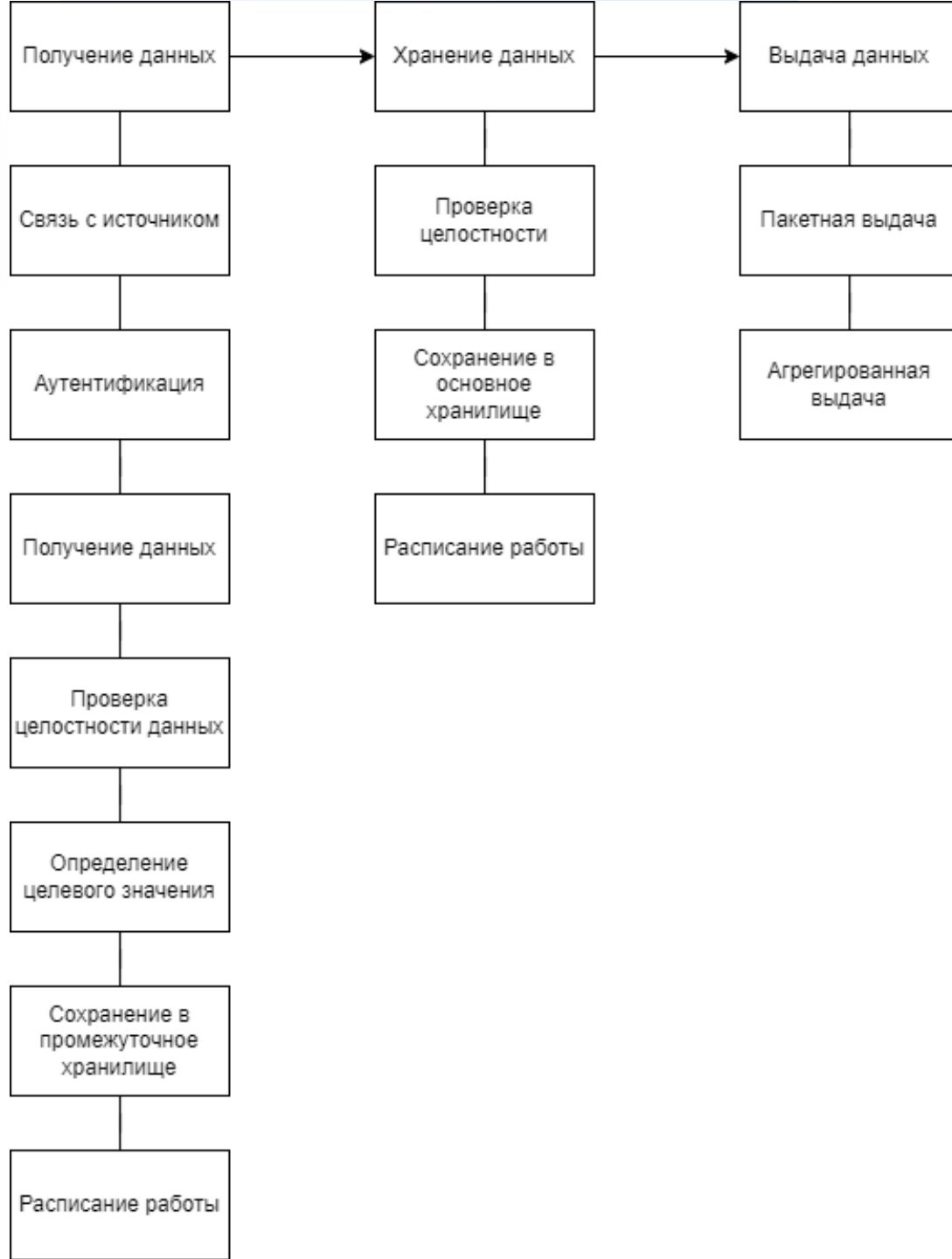
Структура директорий системы направлена на соблюдение определенных правил, чтобы каждый участник проекта делал изменения в конкретном месте, а целостность файлов в процессе работы не была потеряна.

Name	Size (KB)	Last modified	Owner	Group	Access
..					
Work		2020-10-29 ...	administrator	administrator	drwxrwxr-x
Temp		2020-10-29 ...	administrator	administrator	drwxrwxr-x
Docs		2020-10-29 ...	administrator	administrator	drwxrwxr-x
Data_cache		2020-11-16 ...	administrator	administrator	drwxrwxr-x
Data		2020-10-29 ...	administrator	administrator	drwxrwxr-x
Code		2020-10-31 ...	administrator	administrator	drwxrwxr-x

-  Work – утвержденные скрипты и коды моделей, прошедшие ряд тестирования
-  Temp – временные файлы
-  Docs – документация, собранная в ходе реализации проекта
-  Data_cache – временное хранилище файлов, скаченных с API и Datasets в формате CSV
-  Data – данные базы MongoDB
-  Code – скрипты участников проекта

Общая архитектура сбора и хранения информации



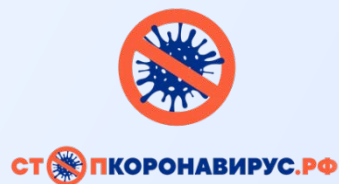


- Подключение к ресурсу - обеспечивает функционал подключения к ресурсу данных по различным протоколам, как в Интернете, так и подключение к другим базам данных
- Аутентификация - обеспечивает механизмы безопасности при подключении, такие как хранение и предоставление паролей и ключей
- Получение данных - механизм опроса для синхронных и асинхронных источников данных, пакетная обработка
- Проверка данных на целостность - уверенность в качестве полученных данных

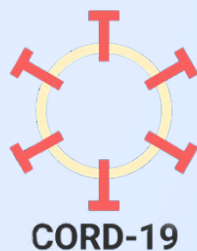


СБОР ДАННЫХ

Используемые ресурсы



РОСПОТРЕБНАДЗОР



При сборе данных было проанализировано более 200 различных ресурсов.

Выбор основывался на определенных параметрах, благодаря которым результат будет более точным.

Основными критериями выбора являлись данные о случаях заражения, выздоровления и летальных исходах, о мерах предпринимаемых руководством страны, реакция населения, а также данные по России.

СБОР ДАННЫХ

Структура скриптов



```
rewriting.py
mkdir.py
copyrate_hdx.py
copyrate.py
connect.py
config.py
dogs.py
check_size.py
check_dirrectory.py
check.py
```

Мы используем единый подход к сбору данных. Он заключается в стандартизации используемых функций, которые могут быть применимы к различным источникам. Такой подход позволяет быстро добавлять скрипты для ранее не рассматриваемых ресурсов, а также не вызывает затруднений в работе команды, так как в процессе используются одинаковые наименования и конструкции кода.

```
API_N00001050 = '/home/user/Data_cash/API_N00001050/'
API_N00001010 = '/home/user/Data_cash/API_N00001010/'
API_N00101000 = '/home/user/Data_cash/API_N00101000/'
DS_N00001400 = '/home/user/Data_cash/DS_N00001400/'
DS_N00001410 = '/home/user/Data_cash/DS_N00001410/'
DS_N00001420 = '/home/user/Data_cash/DS_N00001420/'
DS_N00001430 = '/home/user/Data_cash/DS_N00001430/'
GHR_N00001160 = '/home/user/Data_cash/GHR_N00001160/'
GHR_N00001170 = '/home/user/Data_cash/GHR_N00001170/'
GHR_N00001180 = '/home/user/Data_cash/GHR_N00001180/'

director_conf = '/home/user/Data_cash'

log_conf = '/home/user/Data_cash/log_script.txt'
```

Применяются общие файлы настройки для определения директорий хранения, что позволяет облегчить код скриптов, а также при необходимости ускорить процесс переноса системы на другие носители.

СБОР ДАННЫХ

Примеры скриптов



```
1 # API No N00001010
2 # Novel Corona Virus 2019 Dataset
3 # https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset
4 # Last update a month ago
5
6 from kaggle.api.kaggle_api_extended import KaggleApi
7 import sys
8 sys.path.append('/home/user/Code/')
9 from check import checks
10 from connect import connect
11 from makedir import makedir
12 from clogs import set_log
13 from copyrate import cops
14 from check_size import checking
15 from config import *
16
17 log = log_conf
18
19 director = API_N00001010
20 directory = director + 'Data/'
21 directory_backup = director + 'Backup/'
22
23 api = KaggleApi()
24
25 # Authentication
26 def authentication():
27     api.authenticate()
28
29 # Save to datacash
30 def save(source, name, direct):
31     print("Download ", name, " from Kaggle")
32     api.dataset_download_files(dataset=source, path=direct, force=False, quiet=True, unzip=True)
33     print("Download complete")
34
35 query_dataset = 'https://www.kaggle.com/'
36 query_dataset_url = 'sudalairajkumar/novel-corona-virus-2019-dataset'
37 name_source = 'API_N00001010'
38 set_log(log)
39
40 if connect(query_dataset+query_dataset_url, name_source, log) == True:
41     makedir(directory)
42     makedir(directory_backup)
43     authentication()
44     before_size = checking.size_change_before(directory_backup)
45     save(source=query_dataset_url, name=name_source, direct=directory)
46     after_size = checking.size_change_after(directory)
47     if checking.final_comparison(before_size, after_size, name_source) == True:
48         if cops(directory, directory_backup, name_source) == True:
49             checks(director, directory_backup)
50
```

Скрипт N00001010 (Kaggle)

```
23829299
Download API_N00001010 from Kaggle
Download complete
23829299
%s Downloading data without updating API_N00001010
/home/user/Data_cash/API_N00001010/log_csv.txt
covid_19_data.csv
covid_19_data.csv - file check passed
time_series_covid_19_confirmed.csv
time_series_covid_19_confirmed.csv - file check passed
time_series_covid_19_confirmed_US.csv
time_series_covid_19_confirmed_US.csv - file check passed
time_series_covid_19_deaths.csv
time_series_covid_19_deaths.csv - file check passed
time_series_covid_19_deaths_US.csv
time_series_covid_19_deaths_US.csv - file check passed
time_series_covid_19_recovered.csv
time_series_covid_19_recovered.csv - file check passed
```

Результ



ССЫЛКА НА ИСТОЧНИК



```
284 2021-02-06 => 00:25:02 => WARNING => API_N00001010 => source is working
285 2021-02-06 => 00:25:04 => WARNING => API_N00001010 => downloading data without updating
286 2021-02-06 => 00:25:04 => WARNING => API_N00001010 => backup completed successfully
287
288 2021-02-06 => 00:30:03 => WARNING => API_N00001050 => source is working
289 2021-02-06 => 00:30:05 => WARNING => API_N00001050 => downloading data without updating
290 2021-02-06 => 00:30:05 => WARNING => API_N00001050 => backup completed successfully
291
292 2021-02-06 => 00:35:01 => WARNING => API_N00101000 => source is working
293 2021-02-06 => 00:35:02 => WARNING => API_N00101000_daily => source is working
294 2021-02-06 => 00:35:02 => WARNING => API_N00101000_current => source is working
295 2021-02-06 => 00:36:02 => WARNING => API_N00101000 => successful download of updated data
296 2021-02-06 => 00:36:02 => WARNING => API_N00101000 => backup completed successfully
297
298 2021-02-06 => 00:40:04 => WARNING => DS_N00001400 => source is working
299 2021-02-06 => 00:41:32 => WARNING => DS_N00001400 => successful download of updated data
300 2021-02-06 => 00:41:33 => WARNING => DS_N00001400 => backup completed successfully
301
302 2021-02-06 => 00:45:04 => WARNING => DS_N00001410 => source is working
303 2021-02-06 => 00:45:09 => WARNING => DS_N00001410 => successful download of updated data
304 2021-02-06 => 00:45:09 => WARNING => DS_N00001410 => backup completed successfully
305
306 2021-02-06 => 00:50:04 => WARNING => DS_N00001420 => source is working
307 2021-02-06 => 00:51:48 => WARNING => DS_N00001420 => successful download of updated data
308 2021-02-06 => 00:52:00 => WARNING => DS_N00001420 => backup completed successfully
309
310 2021-02-06 => 01:05:02 => WARNING => GHR_N00001160 => source is working
311 2021-02-06 => 01:09:22 => WARNING => GHR_N00001160 => successful download of updated data
312 2021-02-06 => 01:09:23 => WARNING => GHR_N00001160 => backup completed successfully
313
314 2021-02-06 => 01:15:03 => WARNING => GHR_N00001170 => source is working
315 2021-02-06 => 01:15:06 => WARNING => GHR_N00001170 => downloading data without updating
316 2021-02-06 => 01:15:06 => WARNING => GHR_N00001170 => backup completed successfully
317
318 2021-02-06 => 01:20:02 => WARNING => GHR_N00001180 => source is working
319 2021-02-06 => 01:20:06 => WARNING => GHR_N00001180 => downloading data without updating
320 2021-02-06 => 01:20:06 => WARNING => GHR_N00001180 => backup completed successfully
321
322 2021-02-06 => 17:04:22 => WARNING => API_N00001010 => source is working
323 2021-02-06 => 17:04:24 => WARNING => API_N00001010 => downloading data without updating
324 2021-02-06 => 17:04:24 => WARNING => API_N00001010 => backup completed successfully
325
326 2021-02-06 => 17:32:25 => WARNING => DS_N00001400 => source is working
327 2021-02-06 => 17:33:51 => WARNING => DS_N00001400 => successful download of updated data
328 2021-02-06 => 17:33:52 => WARNING => DS_N00001400 => backup completed successfully
```

Для отслеживания ошибок в системе сбора данных задействовано логирование. В лог файл, в режиме реального времени, поступает вся ключевая информация о работе того или иного скрипта.

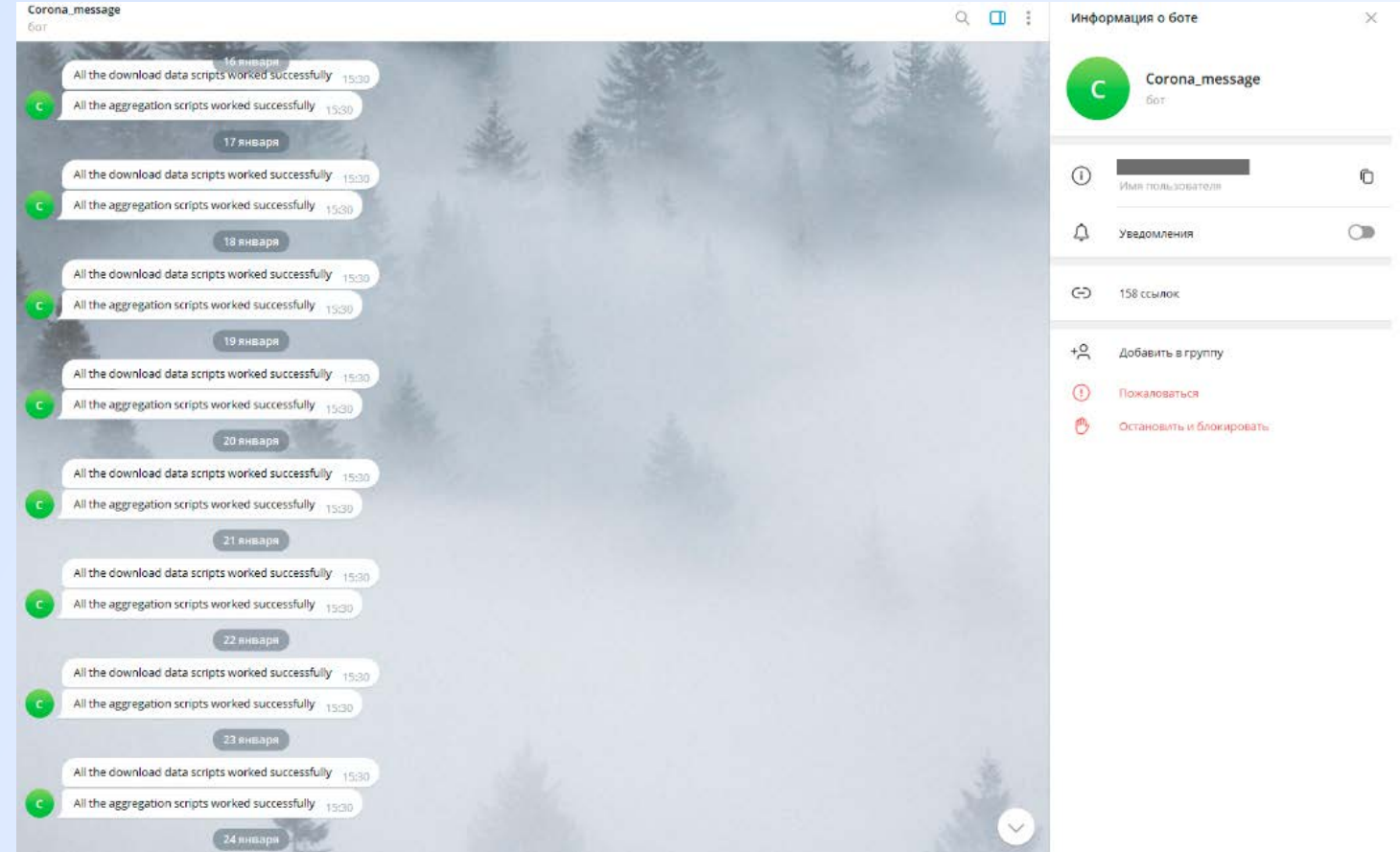
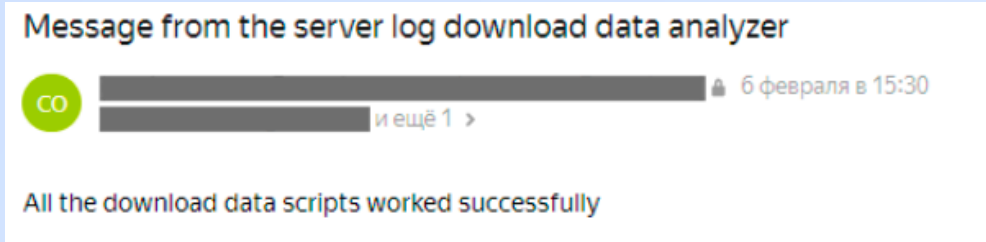
```
# Aggregation
0 3 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_AMU.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
10 3 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_ALT.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
20 3 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_KEM.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
30 3 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_CHE.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
40 3 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_TVA.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
50 3 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_KK.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
0 4 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_AD.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
0 4 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_BA.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
10 4 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_VLA.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
20 4 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_VDR.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
30 4 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_ARK.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
40 4 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_SEV.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
50 4 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_SPE.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
0 5 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_TAM.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
10 5 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_ULY.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
20 5 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_LEN.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
30 5 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_MOS.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
40 5 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_MOW.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
0 6 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_KGN.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
10 6 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_KL.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
20 6 */1 * * /home/user/anaconda3/bin/python3 /home/user/RosAtom-V/Work/aggregation_data/RU_KLU.py >> /home/user/RosAtom-V/Logs/cron_aggregation.log 2>&1
```

Системный сервис Cron

Фрагмент данных в файле логирования

СБОР ДАННЫХ

Анализатор логов



В системе присутствует анализатор файлов логирования для оповещения об ошибках при загрузке данных. Сообщения отправляются как на почту, так и в telegram канал.



```
# Setting up mailings (address, application password)
email_covid = '[REDACTED]'
password = "[REDACTED]"

# the one who receives the messages (list of emails)
emails_dev = ['[REDACTED]']

# Telegram setup (token, send user)
token = '[REDACTED]'
chat_id_dev_group = ['[REDACTED]']
```

Система предусматривает возможности расширения и добавления новых участников проекта. Причем, можно добавлять неограниченное число лиц для получения информации о работоспособности системы получения данных.

ХРАНЕНИЕ ДАННЫХ

Агрегация данных



```
import os
from collections import defaultdict
import sys
sys.path.append('/home/user/Code/ivanov/aggregation_world/')
from config import *

raw_data_path = RAW_DATA + 'API_N00101000/Backup/'
raw_columns_path = RAW_COLUMNS + 'API_N00101000/'
save_data1 = AGGREGATED_DATA
save_data2 = AGGREGATED_DATA2

sources = ['all_states_daily.csv',
           'us_daily.csv']

columns_sources = ['col_dict_1.json',
                  'col_dict_2.json']

def read_from_source(sn):
    df = pd.read_csv(raw_data_path + sources[sn])
    return df

def read_from_columns(cl):
    with open(raw_columns_path + columns_sources[cl]) as f:
        col = json.load(f)
        return col

states_df = read_from_source(0)
us_df = read_from_source(1)
states_columns = read_from_columns(0)
us_columns = read_from_columns(1)

states_df['date'] = pd.to_datetime(states_df['date'].astype(str), format='%Y%m%d')
us_df['date'] = pd.to_datetime(us_df['date'].astype(str), format='%Y%m%d')

def change_name(col, rev_columns):
    if rev_columns[col] == '':
        return col
    else:
        return rev_columns[col]

def raw_to_std_columns(raw_columns, s_columns):
    rev_columns = {value:key for key, value in s_columns.items()}
    rev_columns = defaultdict(str, rev_columns)
    std_columns = [change_name(column, rev_columns) for column in raw_columns]
    return std_columns

std_columns = raw_to_std_columns(states_df.columns, states_columns)
states_df.columns = std_columns
```

- Агрегация данных из предварительного хранилища. Сбор одинаковых данных из разных источников
- Проверка целостности данных – проверка и обработка недостающих, дублирующийся данных, проверка ошибки формата данных
- Сохранение в MongoDB – размещение в основном хранилище
- Расписание работы – запуск скриптов для получения данных

ХРАНЕНИЕ ДАННЫХ

MongoDB

MongoDB - документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы. MongoDB предоставляет драйверы и инструменты для взаимодействия с хранилищем данных MongoDB с использованием различных языков программирования, включая Python, JavaScript, Java, Go и C#.

Pymongo - официальная библиотека Python для взаимодействия с MongoDB, работает на Python 3.9

```
> use CovidRussia1
switched to db CovidRussia1
> db.AggregationRU.findOne()
{
  "_id" : ObjectId("601402fd69e10d521e200712"),
  "date" : "2021-01-23",
  "confirmed" : 12638,
  "deaths" : 113,
  "recovered" : 10710,
  "region_name" : "Республика Адыгея",
  "region_code" : "RU-AD",
  "isolation_start" : "16.07.2020 21:58:11",
  "level" : 3,
  "self_isolation" : NaN
}
> db.AggregationRU.count()
25847
> db.AggregationRU.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "unique" : true,
    "key" : {
      "date" : 1,
      "region_code" : 1
    },
    "name" : "date_1_region_code_1"
  }
]
```

Пример документа хранящегося в коллекции

ХРАНЕНИЕ ДАННЫХ

Примеры скриптов

Работа с базой MongoDB предусматривает не только использование встроенной консоли mongoshell, но и скрипты, по образу и подобию системы получения данных Интернет ресурсов.

```
from config import *
from mkdir import mkdir
from tools_mongo import *
from tools_aggregation import *

from clogs import set_log
import logging
log = log_conf_agg
set_log(log)

# Import db name from config file
type_agg = name_covid_city_agg

msg_welcome = f'START {type_agg}'
print(msg_welcome)
logging.warning(msg_welcome)

# Setup add to config file
database = dashboard_database_ver_3
address = dashboard_address

list_csv_path = find_final_csv(agggregated_final + covid_city_path)
start_quick_save_2(type_agg, list_csv_path, database, address, key_for_update='_id')
```

Процесс добавления данных в коллекцию базы с использованием Python

```
from pymongo.errors import BulkWriteError
from pymongo.errors import ConnectionFailure
from pymongo.errors import ConfigurationError
from pymongo.errors import DuplicateKeyError
from pymongo.errors import OperationFailure
from pymongo.errors import PyMongoError
from pymongo.errors import AutoReconnect
import pandas as pd
from pymongo import MongoClient
import csv
import json
from mkdir import mkdir
import time
from bson.json_util import dumps
import logging
from clogs import set_log

from config import log_conf_mongo

set_log(log_conf_mongo)
log = log_conf_mongo

class tools(object):

    def __init__(self, database=None, collection=None, address=None):

        self.dBName = database
        self.collectionName = collection
        self.addressName = address

        self.client = MongoClient(self.addressName)
        self.DB = self.client[self.dBName]
        self.collection = self.DB[self.collectionName]

    def connect(self):

        """ The function connecting to databases and collection. """

        waiting = 2000
        if self.DB != None and self.collection != None:

            print('Try connect on database...')

            try:
                client = MongoClient(self.addressName, serverSelectionTimeoutMS=waiting)
                client.server_info()
            except ConnectionFailure as e:
```

АНАЛИЗ ДАННЫХ

Результат преобразований



7 октября в регионе подтверждено 644 случая новой коронавирусной инфекции: - 248 в г. самара - 118 в г. тольятти - 26 в г. новокуйбышевск - 23 в г. отрядный - 21 в г. кинель - 16 в нефтегорском районе - 16 в г. сызрань - 12 в кинель-черкасском районе - 12 в сергиевском районе - 11 в волжском районе - 10 в г. чапаевск - 9 в красноармейском районе - 9 в ставропольском районе - 8 в приволжском районе - 7 в алексеевском районе - 7 в кинельском районе - 7 в клявлинском районе - 7 в кошкинском районе - 6 в похвистневском районе - 6 в большеглушицком районе - 6 в богатовском районе - 6 в г. похвистнево - 5 в исаклинском районе - 5 в пестравском районе - 5 в борском районе - 5 в г. жигулевск - 5 в г. октябрьск - 5 в хворостянской районе - 4 в большечерниговском районе - 4 в челно-вершинском районе - 3 в красноварском районе - 3 в шигонском районе - 3 в елховском районе - 2 в камышлинском районе - 2 в безенчукском районе - 1 в шенталинском районе - 1 в сызранском районе

👉 612 человек обследовано с диагнозом внебольничная пневмония и орви; 👉 32 человека выявлено при профилактическом обследовании лиц, не имеющих клинических проявлений. 📄 нарастающим итогом в регионе зафиксировано 109622 случая коронавируса. скончались 4048 человек, в том числе женщины 60, 60, 60, 81, 87, 87, 84, 84 лет, мужчины 73, 78 лет, страдавшие заболеванием сердечно-сосудистой системы; женщины 58, 62, 84 лет, мужчина 81 года, страдавшие заболеваниями сердечно-сосудистой и эндокринной систем; женщины 72, 78, 78, 79, 79, 79, 81, 83 лет; мужчины 55, 59 лет; женщина 67 лет, страдавшая заболеванием сердечно-сосудистой системы, онкологическим заболеванием; женщины 72, 85 лет, страдавшие заболеваниями сердечно-сосудистой системы и органов дыхания; мужчина 58 лет, страдавший инфекционным заболеванием; мужчина 67 лет, страдавший заболеванием пищеварительной системы; показатель смертности на 100 тыс. населения в области - 126,78, по стране - 146,27. коэффициент летальности в области - 3,69, в стране - 2,78. 📝 проведено лабораторных исследований на кви за сутки - 14432, нарастающим итогом 3741448.

Пример полученной записи из группы оперативного штаба Вконтакте по Самарской области

date	city	infected_city	pneumonia_ARVI	death_rate_per_100_thousand_population_region	death_rate_per_100_thousand_population_country	without_clinical_manifestation
01-01-2021	самара	127	258	19.35	39.42.	45
01-04-2021	самара	80	169	34.67	67.97.	30
01-05-2021	самара	74	145	47.35	75.69.	15
01-06-2021	самара	18	64	61.35	83.47.	8
01-07-2020	самара	41	32	1.75	6.53.	19

Пример полученного набора данных с использованием методов обработки естественного языка



ВИЗУАЛИЗАЦИЯ ДАННЫХ

Дашборд – модуль проекта «Ферма данных», позволяющий визуализировать и сравнивать собранные и обработанные данные о ходе заболевания в странах и регионах РФ.

Уникальность проекта состоит в размещении результатов исследования моделей распространения коронавирусной эпидемии с возможностью изменения ряда управляющих параметров в регионах РФ.

Все размещенные данные доступны для скачивания в формате CSV.

С результатом практической реализации проекта можно ознакомиться по ссылке <https://lab-ai.ru/dashboard>

А с подробным описанием фермы данных в препринте по [ссылке](#) (статья выйдет в одном из ближайших выпусков [Труды ИСП РАН](#)).

Последние новости

Мы заботимся о качестве нашего проекта. Если у вас есть предложение, вопрос или вы заметили ошибку, пожалуйста, напишите нам. Для этого мы добавили форму обратной связи.

С 2021-08-04 Университет Джона Хопкинса больше не публикует информацию о случаях заболевания Covid-19. Данную метрику мы получаем из множества других источников. Обновление данных по данной метрике может быть не своевременным.

Обновлен раздел для скачивания данных. Добавлены данные по городам Российской Федерации.

О проекте

Проект, предназначенный для визуализации статистической информации по случаям Covid-19 в странах и регионах Российской Федерации, реализован на базе Фермы Данных.

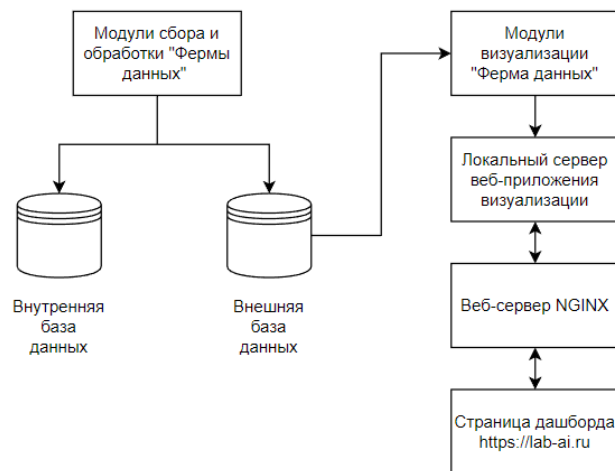
«Ферма данных» – это программно-инструментальная система, предназначенная для сбора и анализа данных из структурированных и неструктурированных открытых источников сети интернет для получения из нее информации в виде заданных исходных параметров.

В ходе реализации мы задействовали следующие источники: СтолКоронаВирус, Софол, Яндекс, группы Вконтакте оперативных штабов, Университет Джона Хопкинса, Мировая организация здравоохранения и многие другие.

- Ковид-19 | Россия
- Ковид-19 | Регионы
- Ковид-19 | Города**
- Ковид-19 | Страны
- Ковид-19 | Модель
- Осла обмен
- Скачать данные

Раздел в разработке, доступна информация по 375 городам.

Карта городов



Архитектура визуализации данных



**НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЛАБОРАТОРИЯ
ИСКУССТВЕННОГО ИНТЕЛЛЕКТА**

Левашкин Сергей Павлович

ai_lab@psuti.ru
<https://ai.psuti.ru/>
<https://lab-ai.ru/>

8(846)339-11-11 (802)

Россия, Самара, Московское Шоссе 77, 443090