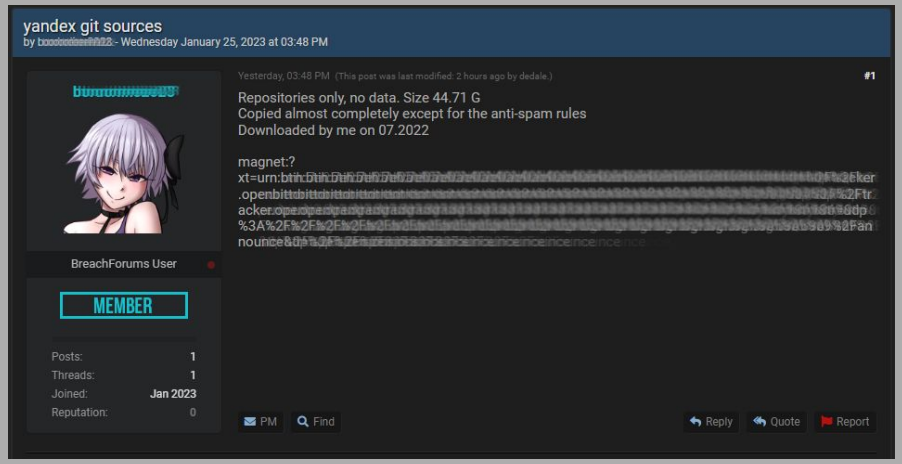
What the Yandex Leak Tells Us About How Big Tech Uses Your Data

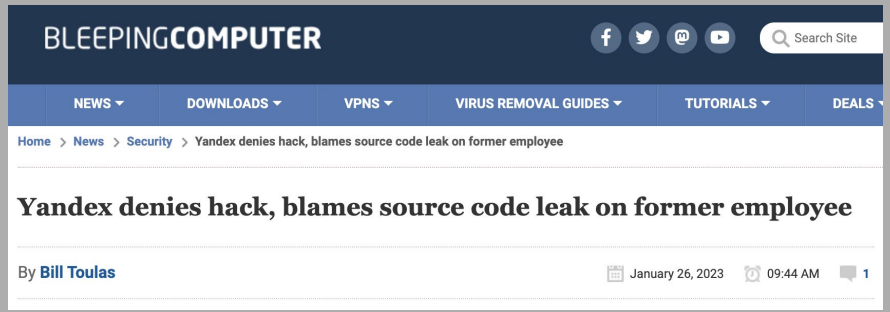
Kaileigh McCrea, Privacy Engineer @ Confiant

August 2023

What we're talking about

Yandex





Roadmap

- Background on Yandex Leak
- Dive into code:
  - What data Yandex is collecting
  - What Yandex is doing with that data
  - Who Yandex is sharing that data with
- Conclusions and wrap up
- Q&A

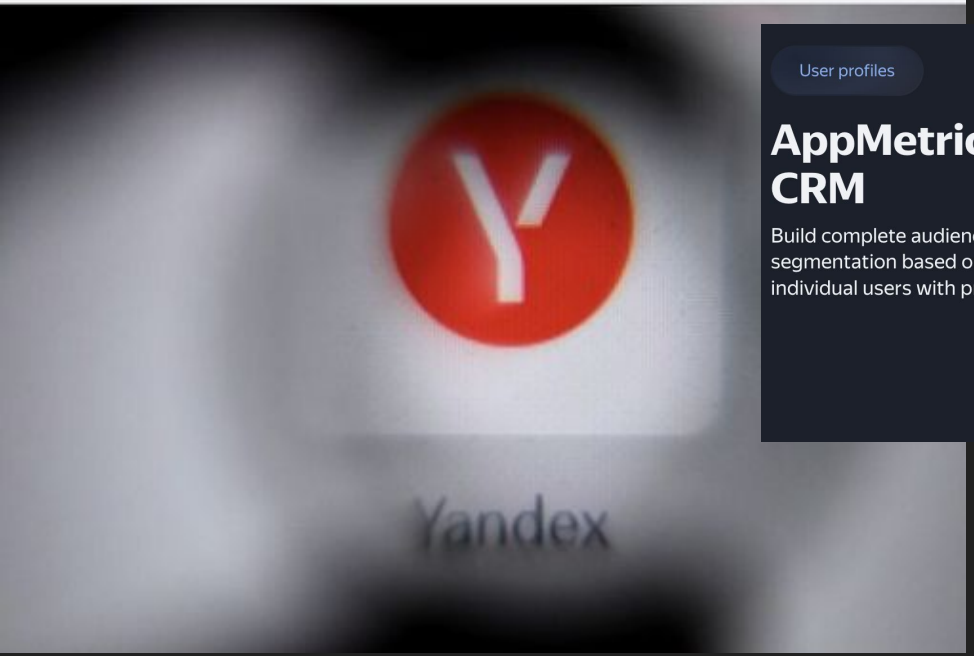
Yandex: A Drama

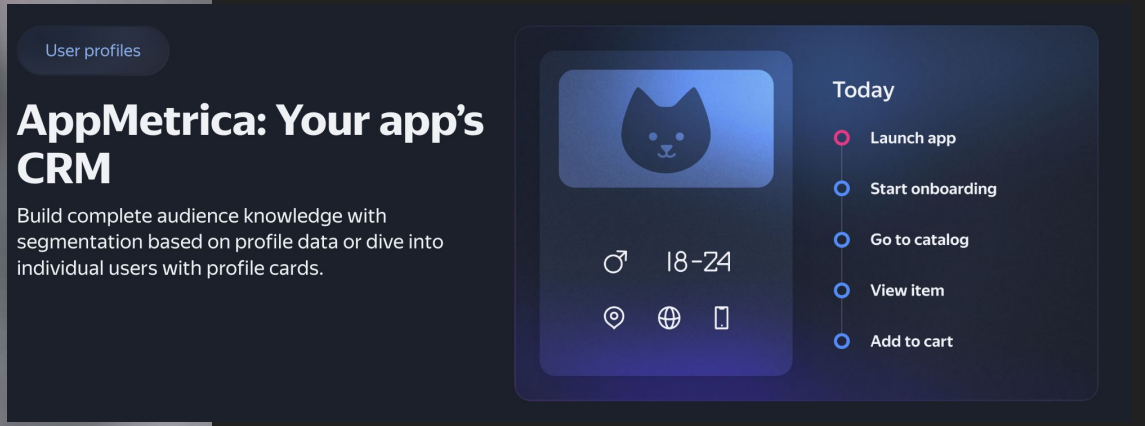
BOUNTIFUL HARVEST —

Data-harvesting code in mobile apps sends user data to “Russia’s Google”

Data from apps on Apple- and Google-powered mobile devices is sent to Russian servers.

PATRICK MCGEE, FINANCIAL TIMES - 3/29/2022, 7:18 AM





2022

Russia invades Ukraine

Alarm About Sending Data to Russian Servers

September | November | December | January | February | March | April

‘“Yandex has acknowledged its software collects “device, network and IP address” information … but it called this data “non-personalised and very limited”. It added: “Although theoretically possible, in practice it is extremely hard to identify users based solely on such information collected. Yandex definitely cannot do this.”’

– *Financial Times*

Featured Article

Russia's war hits Yandex, the 'Google of Russia'

Sources say the company is seeking a media exit as top exec hit with sanctions over propaganda charge

Natasha Lomas, Ingrid Lunden / 12:20 PM PDT • March 16, 2022

Comment

REUTERS®

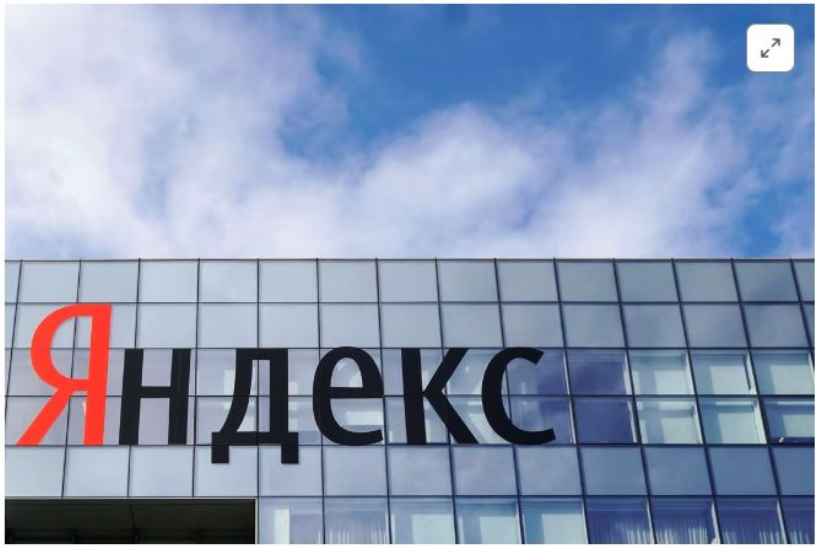
World Business Markets Sustainability Legal More

Europe

Putin, Kudrin touch on future of Yandex in late-night meeting -sources

Reuters

November 25, 2022 4:19 AM PST · Updated 8 months ago



[1/2] The logo of Russian internet group Yandex is pictured at the company's headquarter in Moscow, Russia October 4, 2018. REUTERS/Shamil Zhumatov

The Moscow Times

30 YEARS

INDEPENDENT NEWS FROM RUSSIA

NEWS UKRAINE WAR BUSINESS OPINION ARTS AND LIFE PODCASTS

Kremlin Ally Kudrin Confirms Move to Tech Giant Yandex

Dec. 5, 2022

The Moscow Times
30 YEARS
INDEPENDENT NEWS FROM RUSSIA

Russian Billionaires Line Up to Buy Yandex – Reports

May 4, 2023

MOST READ JUST IN

1 OVERNIGHT STRIKE
Russia Says Ukrainian Dron
Moscow, Crimea

2 NO PASSAGE
Russia Blocks Cargo Ship C
Traces'

3 POLITICAL PRISONER
Navalny Ally Jailed 9 Years

4 MORE MANPOWER
Russia Raises Upper-Age L
Reservists

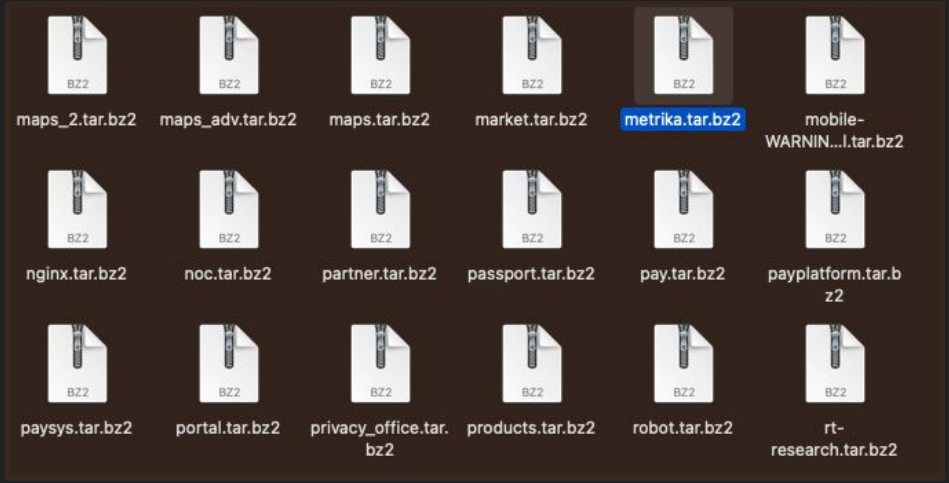
5 MONEY DRAIN
Russia Loses Record $253
Capital Flight

Roadmap

- Background on Yandex Leak
- Dive into code:
  - What data Yandex is collecting
  - What Yandex is doing with that data
  - Who Yandex is sharing that data with
- Conclusions and wrap up
- Q&A

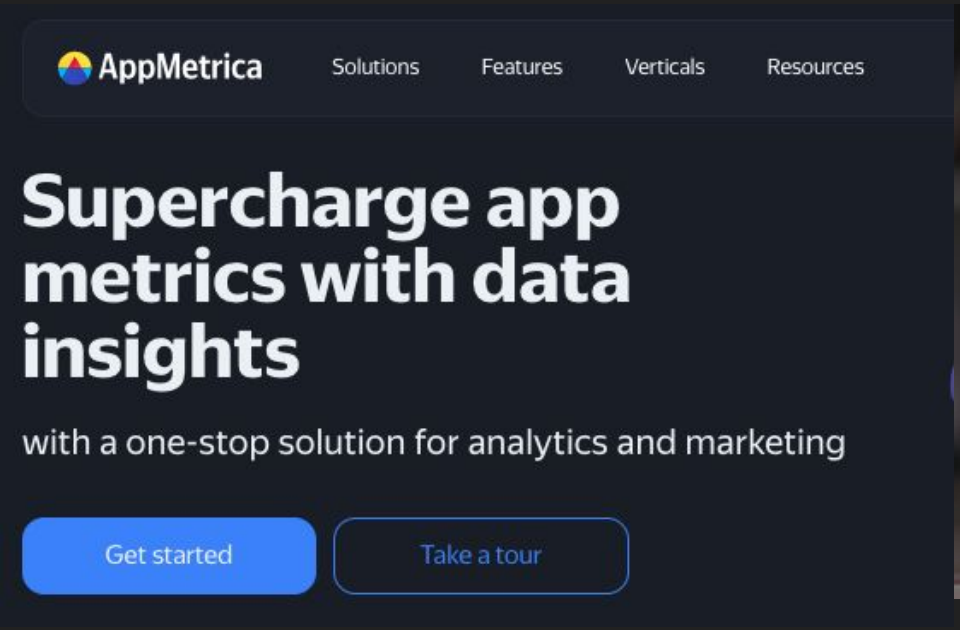
Yandex Codebase

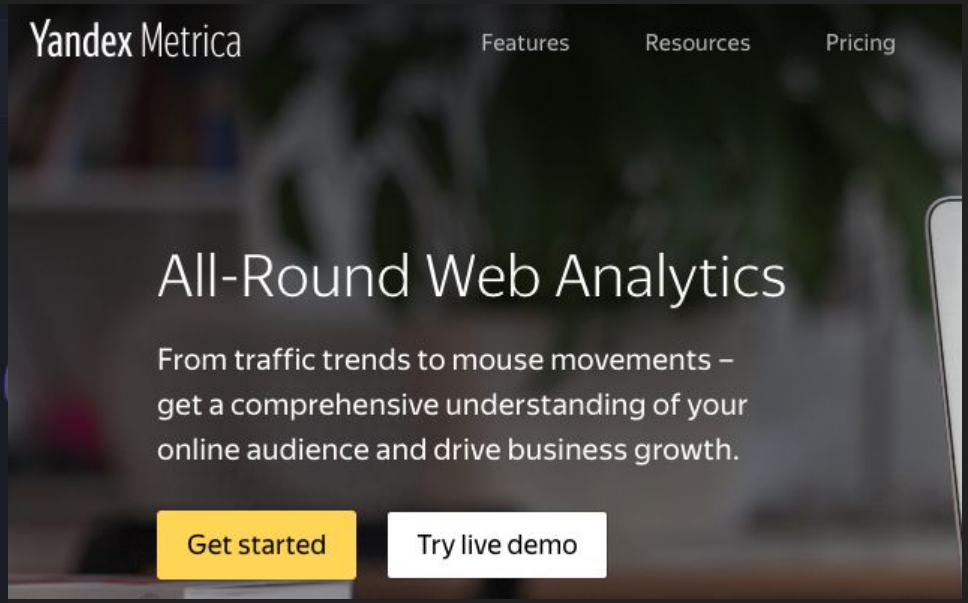
Codebase


maps_2.tar.bz2
maps_adv.tar.bz2
maps.tar.bz2
market.tar.bz2
metrika.tar.bz2
mobile-WARNIN...l.tar.bz2
nginx.tar.bz2
noc.tar.bz2
partner.tar.bz2
passport.tar.bz2
pay.tar.bz2
payplatform.tar.bz2
paysys.tar.bz2
portal.tar.bz2
privacy_office.tar.bz2
products.tar.bz2
robot.tar.bz2
rt-research.tar.bz2

Metrika

Metrika


AppMetrica
Solutions
Features
Verticals
Resources
Supercharge app metrics with data insights
with a one-stop solution for analytics and marketing
Get started
Take a tour


Yandex Metrica
Features
Resources
Pricing
All-Round Web Analytics
From traffic trends to mouse movements – get a comprehensive understanding of your online audience and drive business growth.
Get started
Try live demo

Example Raw Data Fields that AppMetrica Logs

analytics > appmetrica-location-log-anonymizer > convert_log.yql

```
insert into `//home/metrika-logs/anonym-appmetrica-location-log/1d/{table_date}`
with truncate
    select
        String::HexEncode(Digest::Blake2B(`DeviceID`, seed)) as `DeviceID`,
        String::HexEncode(Digest::Blake2B(`ADVID`, seed)) as `ADVID`,
        String::HexEncode(Digest::Blake2B(`IFA`, seed)) as `IFA`,
        String::HexEncode(Digest::Blake2B(`UUID`, seed)) as `UUID`,
        String::HexEncode(Digest::Blake2B(`AndroidID`, seed)) as `AndroidID`,
        `APIKey`,
        `AppBuildNumber`,
        `AppFramework`,
        `AppID`,
        `AppPlatform`,
        `AppVersionName`,
        `Cells_AreConnected`,
        `Cells_CellsIDs`,
        `Cells_CountriesCodes`,
        `Cells_Lacs`,
        `Cells_LastVisibleTimeOffset`,
        `Cells_OperatorsIDs`,
        `Cells_OperatorsNames`,
        `Cells_PhysicalsCellsIDs`,
        `Cells_SignalsStrengths`,
        `Cells_Types`,
        `ChargeType`,
        `ClientIP`,
        `ClientIPHash`,
        `CollectTimestamp`,
        `CollectTimestampBootOffset`,
        `CollectionMode`,
        `DeviceType`,
        `EventID`,
        `IncrementalID`,
        `IsExtraLocationEvent`,
        `IsRooted`,
        `KitBuildNumber`,
        `KitBuildType`,
        `KitVersion`,
```

Ln 117

analytics > appmetrica-location-log-anonymizer > convert_log.yql

```
        `IsExtraLocationEvent`,
        `IsRooted`,
        `KitBuildNumber`,
        `KitBuildType`,
        `KitVersion`,
        `Latitude`,
        `LatitudeLBS`,
        `LocationAltitude`,
        `LocationDirection`,
        `LocationEnabled`,
        `LocationPrecision`,
        `LocationPrecisionLBS`,
        `LocationSource`,
        `LocationSpeed`,
        `LocationTimestamp`,
        `LocationTimestampBootOffset`,
        `Longitude`,
        `LongitudeLBS`,
        `OSApiLevel`,
        `OSVersion`,
        `OperatingSystem`,
        `OriginalCollectTimestamp`,
        `OriginalLocationTimestamp`,
        `ReceiveTimestamp`,
        `RequestID`,
        `SendTimestamp`,
        `Wifi_AreConnected`,
        `Wifi_LastVisibleTimeOffset`,
        `Wifi_Macs`,
        `Wifi_SignalsStrengths`,
        `Wifi_Ssids`,
        `_logfeller_index_bucket`,
        `_logfeller_timestamp`,
        `_rest`,
        `_stbx`,
        `iso_eventtime`,
        `source_uri`,
        `subkey`,
        `timestamp`
```

Anonymized Identifiers

```
91      select
92          String::HexEncode(Digest::Blake2B(`DeviceID`, seed)) as `DeviceID`,
93          String::HexEncode(Digest::Blake2B(`ADVID`, seed)) as `ADVID`,
94          String::HexEncode(Digest::Blake2B(`IFA`, seed)) as `IFA`,
95          String::HexEncode(Digest::Blake2B(`UUID`, seed)) as `UUID`,
96          String::HexEncode(Digest::Blake2B(`AndroidID`, seed)) as `AndroidID`,
```

Location Fields

```
126     `KitVersion`,
127     `Latitude`,
128     `LatitudeLBS`,
129     `LocationAltitude`,
130     `LocationDirection`,
131     `LocationEnabled`,
132     `LocationPrecision`,
133     `LocationPrecisionLBS`,
134     `LocationSource`,
135     `LocationSpeed`,
136     `LocationTimestamp`,
137     `LocationTimestampBootOffset`,
138     `Longitude`,
139     `LongitudeLBS`,
140     `OSApiLevel`,
141     `OSVersion`,
142     `OperatingSystem`,
143     `OriginalCollectTimestamp`,
144     `OriginalLocationTimestamp`,
145     `ReceiveTimestamp`,
146     `RequestID`,
```

Wifi Fields Collected by AppMetrica

```
147         `SendTimestamp`,
148         `Wifi_AreConnected`,
149         `Wifi_LastVisibleTimeOffset`,
150         `Wifi_Macs`,
151         `Wifi_SignalsStrengths`,
152         `Wifi_Ssids`,
153         ` logfeller index bucket`,
```

Those Same Fields in Crypta

graph > fuzzy > lib > yql > export_ssid_devid_day_table.yql

```
26
27  $list_metrika_log = (
28      select coalesce(DeviceID, "") as DeviceID,
29             coalesce(OriginalDeviceID, "") as OriginalDeviceID,
30             $MakeStringList(Wifi_Macs) as Wifi_Macs,
31             $MakeStringList(Wifi_Ssids) as Wifi_Ssids,
32             $MakeIntList(Wifi_SignalsStrengths) as Wifi_SignalsStrengths,
33             $MakeIntList(Wifi_AreConnected) as Wifi_AreConnected
34      from `{source_mmetric_table}`
35      where DeviceID is not null
36  );
```

Dev Id and SSID Associated with Yandex UID

graph > fuzzy > lib > yql > export_ssid_yuids.yql

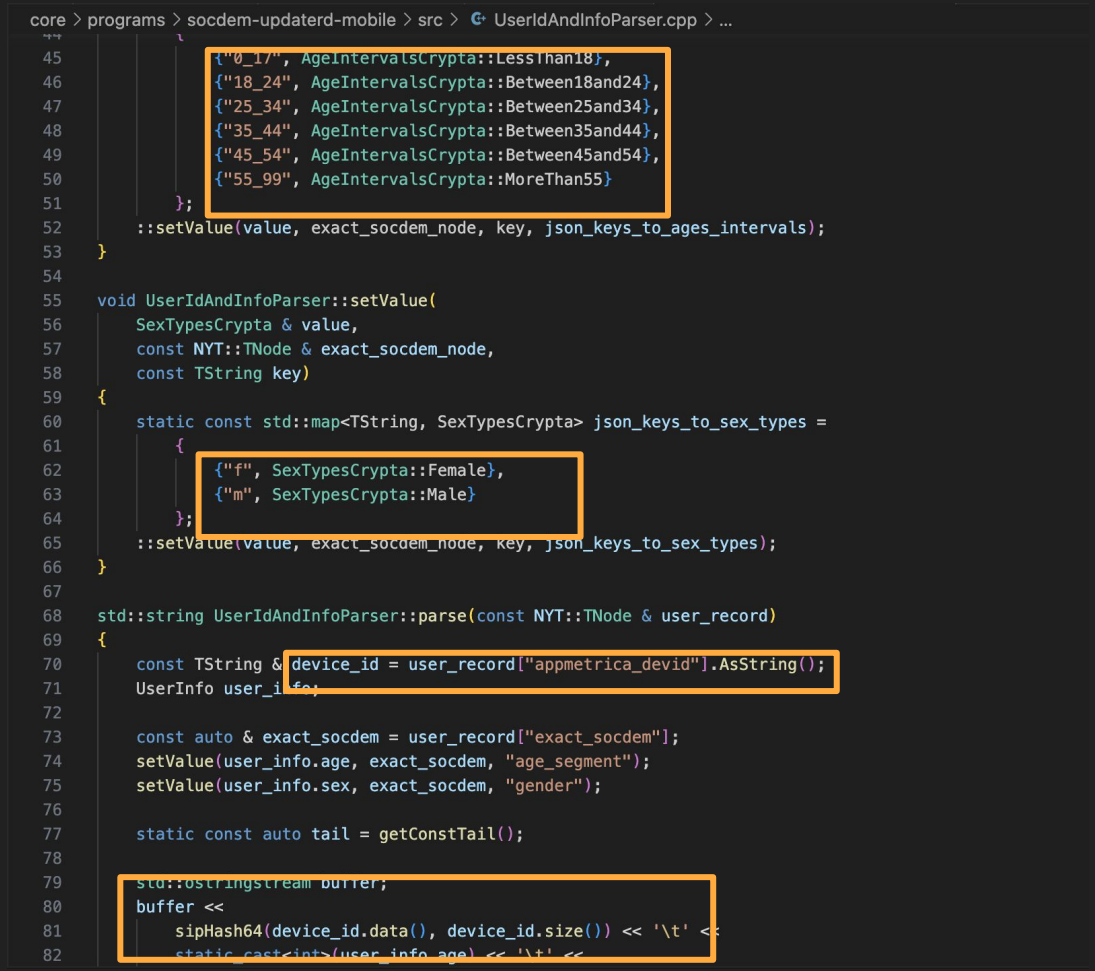
```
6   $mobile_all_table = (
7       select distinct mmetric_devid, ssid
8       from concat({sources})
9   );
10
11  $mmetric_to_devid = (
12      select mmetric_devid, devid,
13             coalesce(cast(yuid as uint64), 0) as yuid
14      from `{source_nolimit_table}`
15  );
16
```

Click Event Data Being Matched to Existing Users

core > programs > clicklogd-mobile > src > C event_indexed_pool.h > TEventIndexedPool > GetIndex<TMatchCriteria>()

```
58  private:
59      template <class TMatchCriteria>
60      TIndex<TMatchCriteria>& GetIndex() {
61          if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TAndroidId>) {
62              return AndroidId_;
63          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TAndroidIdMd5>) {
64              return AndroidIdMd5_;
65          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TAndroidIdSha1>) {
66              return AndroidIdSha1_;
67          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TDeviceIdHash>) {
68              return DeviceIdHash_;
69          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TFingerprint>) {
70              return Fingerprint_;
71          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TGoogleAid>) {
72              return GoogleAid_;
73          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TGoogleAidMd5>) {
74              return GoogleAidMd5_;
75          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TGoogleAidSha1>) {
76              return GoogleAidSha1_;
77          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TIfa>) {
78              return Ifa_;
79          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TIfaMd5>) {
80              return IfaMd5_;
81          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TIfaSha1>) {
82              return IfaSha1_;
83          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TWindowsAid>) {
84              return WindowsAid_;
85          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TWindowsAidMd5>) {
86              return WindowsAidMd5_;
87          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TWindowsAidSha1>) {
88              return WindowsAidSha1_;
89          } else if constexpr (std::is_same_v<TMatchCriteria, NMatchCriteria::TYmTrackingId>) {
90              return YmTrackingId_;
91          }
92      }
93
```

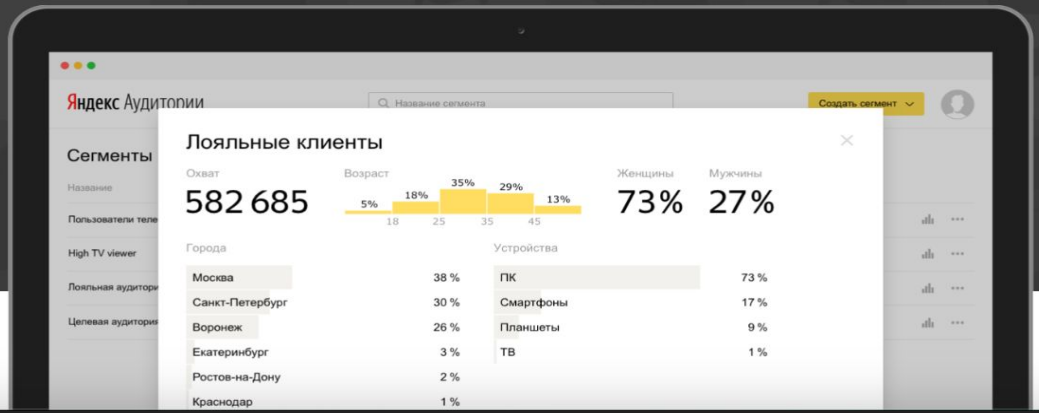
Socio-Demographic Attributes for DevID Being Updated



```
core > programs > socdem-updaterd-mobile > src > UserIdAndInfoParser.cpp > ...
45              {"0_17", AgeIntervalsCrypta::LessThan18},
46              {"18_24", AgeIntervalsCrypta::Between18and24},
47              {"25_34", AgeIntervalsCrypta::Between25and34},
48              {"35_44", AgeIntervalsCrypta::Between35and44},
49              {"45_54", AgeIntervalsCrypta::Between45and54},
50              {"55_99", AgeIntervalsCrypta::MoreThan55}
51          };
52      ::setValue(value, exact_socdem_node, key, json_keys_to_ages_intervals);
53  }
54
55  void UserIdAndInfoParser::setValue(
56      SexTypesCrypta & value,
57      const NYT::TNode & exact_socdem_node,
58      const TString key)
59  {
60      static const std::map<TString, SexTypesCrypta> json_keys_to_sex_types =
61          {
62              {"f", SexTypesCrypta::Female},
63              {"m", SexTypesCrypta::Male}
64          };
65      ::setValue(value, exact_socdem_node, key, json_keys_to_sex_types);
66  }
67
68  std::string UserIdAndInfoParser::parse(const NYT::TNode & user_record)
69  {
70      const TString & device_id = user_record["appmetrica_devid"].AsString();
71      UserInfo user_info;
72
73      const auto & exact_socdem = user_record["exact_socdem"];
74      setValue(user_info.age, exact_socdem, "age_segment");
75      setValue(user_info.sex, exact_socdem, "gender");
76
77      static const auto tail = getConstTail();
78
79      std::ostringstream buffer;
80      buffer <<
81          sipHash64(device_id.data(), device_id.size()) << '\t' <<
82          static_cast<int>(user_info.age) << '\t' <<
```

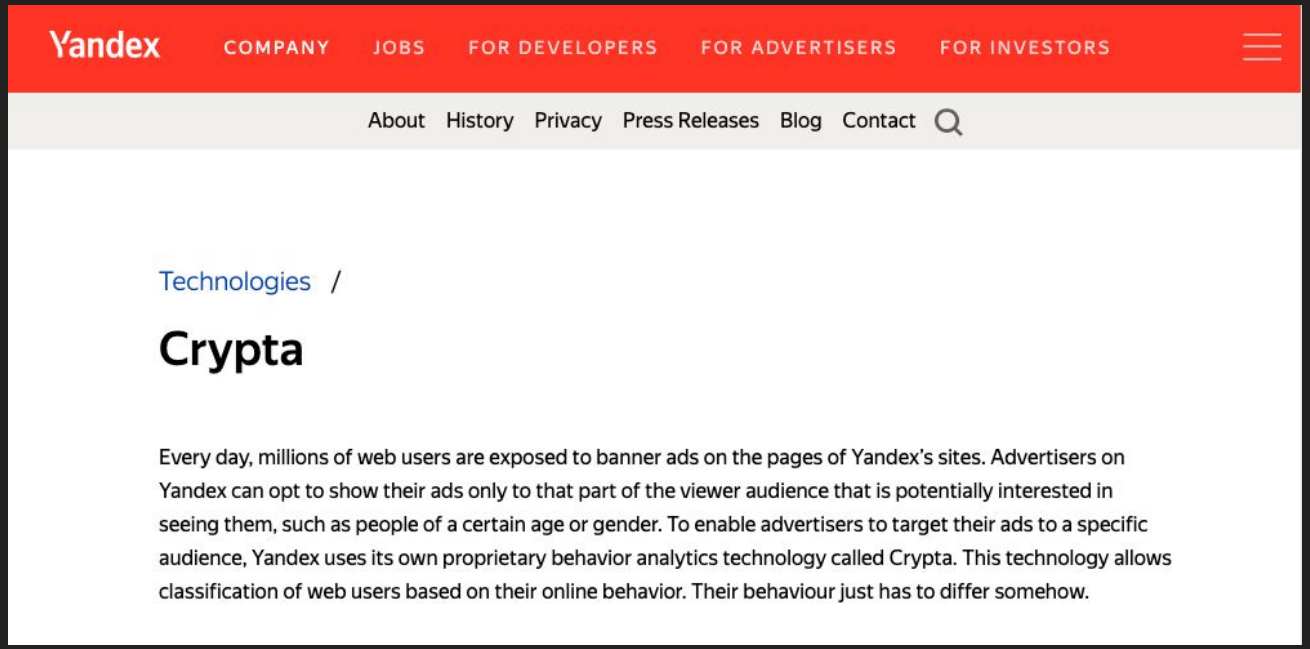
Create segments based on offline and online data

Create Segment

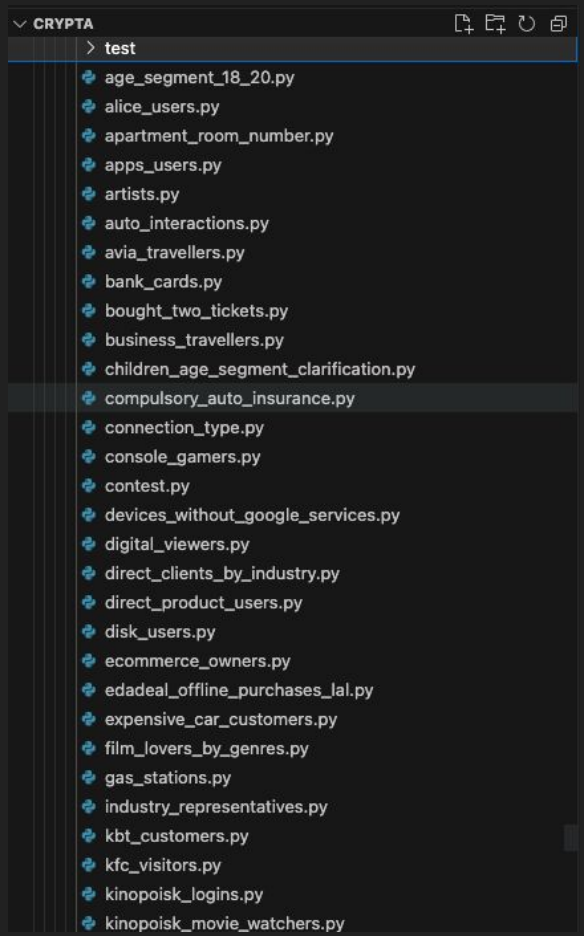


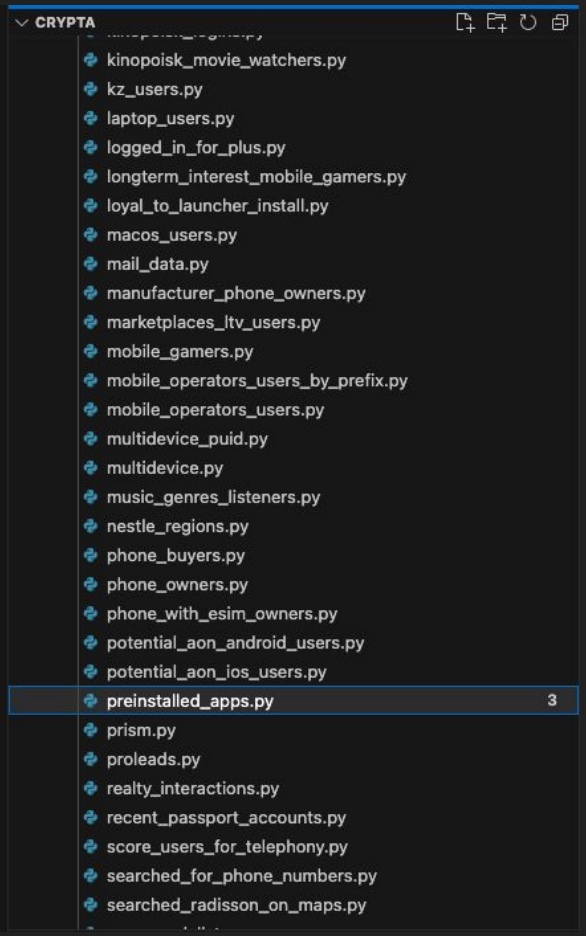
Crypta

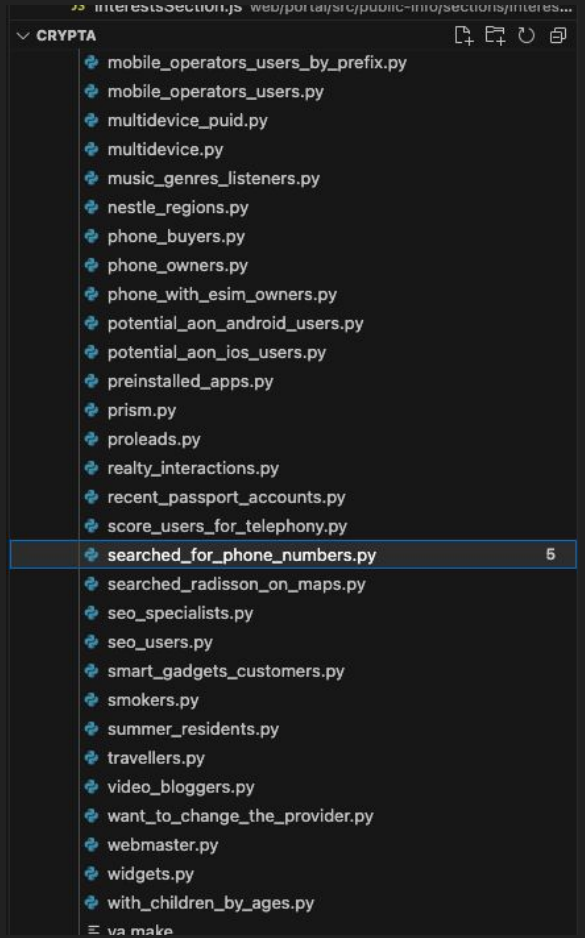
Crypta


Yandex
COMPANY
JOBS
FOR DEVELOPERS
FOR ADVERTISERS
FOR INVESTORS
About
History
Privacy
Press Releases
Blog
Contact
Technologies /
Crypta
Every day, millions of web users are exposed to banner ads on the pages of Yandex's sites. Advertisers on Yandex can opt to show their ads only to that part of the viewer audience that is potentially interested in seeing them, such as people of a certain age or gender. To enable advertisers to target their ads to a specific audience, Yandex uses its own proprietary behavior analytics technology called Crypta. This technology allows classification of web users based on their online behavior. Their behaviour just has to differ somehow.

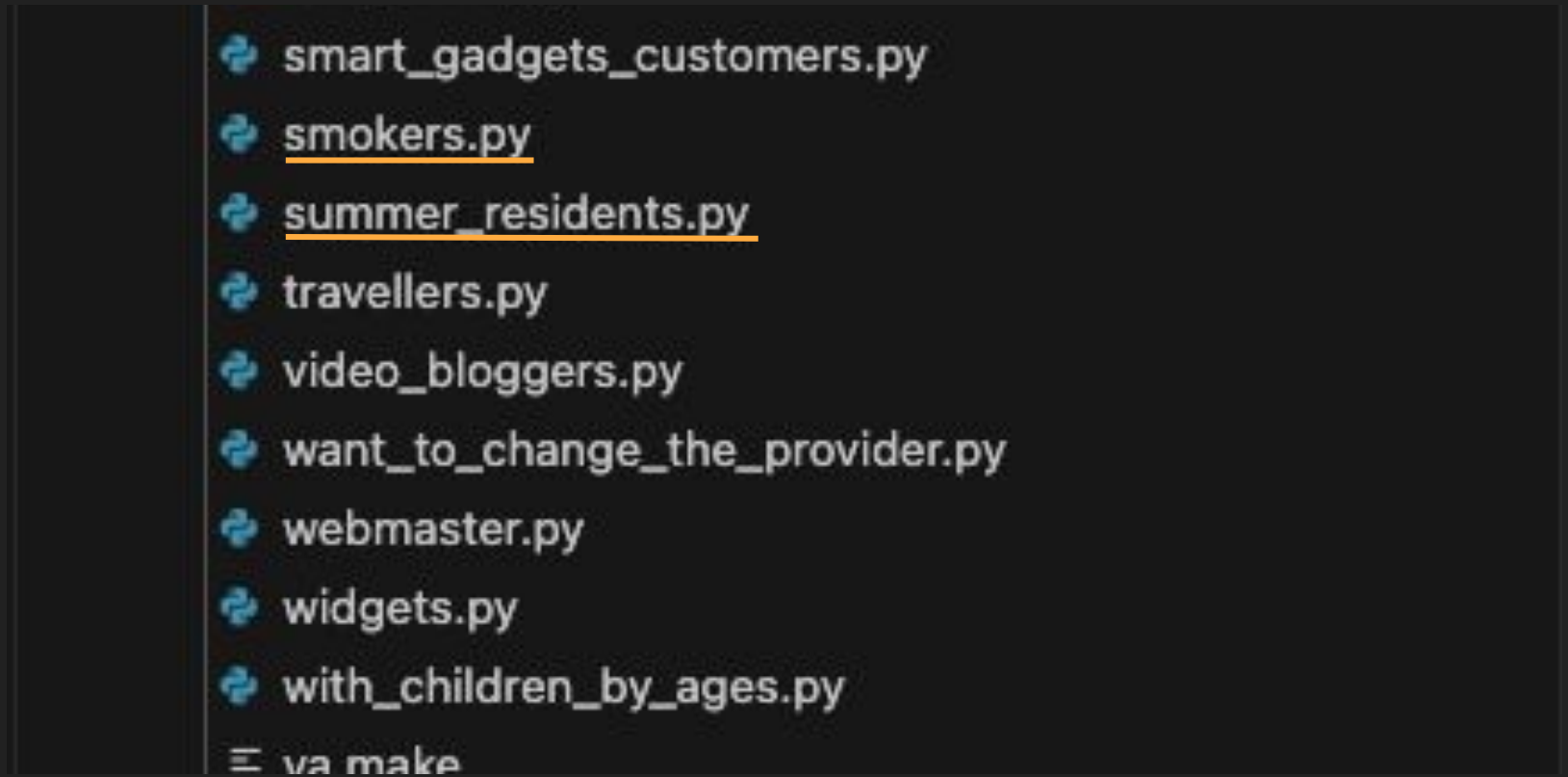
Example Segments







Example Segments



Travellers

profile > runners > segments > lib > coded_segments > travellers.py > ...

```
82
83  INSERT INTO `{output_table}` WITH TRUNCATE
84  SELECT
85      id,
86      id_type,
87      segment_name
88  FROM(
89      SELECT
90          crypta_id AS id,
91          'crypta_id' AS id_type,
92          CASE
93              WHEN Geo::RoundRegionById(region, "country").id != Geo::RoundRegionById(CAST(main_region AS Int32), "country").id THEN 'internati
94              ELSE 'domestic'
95          END AS segment_name,
96          MAX(`date`) AS last_seen,
97          MIN(`date`) AS first_seen,
98          region,
99          week_end_date,
100     FROM $travell_visits
101     GROUP BY region, main_region, crypta_id, week_end_date
102 )
103 WHERE
104     last_seen <= week_end_date AND
105     DateTime::ToDays(DateTime::MakeTimestamp($parse(last_seen)) - DateTime::MakeTimestamp($parse(first_seen))) > 0 AND
106     DateTime::ToDays(DateTime::MakeTimestamp($parse(week_end_date)) - DateTime::MakeTimestamp($parse(first_seen))) <= 7
107 GROUP BY id, id_type, segment_name
108 """
109
110
```

Mail Data

profile > runners > segments > lib > coded_segments > mail_data.py > ...

```
12
13
14  segment_query = """
15  INSERT INTO `{output_table}` WITH TRUNCATE
16  SELECT id, id_type, segment_name
17  FROM `{mail_data_table}`;
18
19  INSERT INTO `{sample_table}` WITH TRUNCATE
20  SELECT
21      yandexuid,
22      segment_name
23  FROM (
24      SELECT matching.yandexuid AS yandexuid, mail_data.segment_name AS segment_name
25      FROM `{mail_data_table}` AS mail_data
26      INNER JOIN `{indevice_yandexuid_matching}` AS matching
27      USING (id, id_type)
28  )
29  GROUP BY yandexuid, segment_name
30  """
31
32
33  class PrepareMailSampleForLalSegments(RegularSegmentBuilder):
34      keyword = 549
35      name_segment_dict = {
36          'aviaticket': 1404,
37          'boardingpass': 1405,
38          'hotel': 1406,
39      }
```

Gas Stations

profile > runners > segments > lib > coded_segments > gas_stations.py > ...

```
92  class ProcessedDeepVisitLogForGasStations(DayProcessor):
93      def requires(self):
94          return deep_visits.org_visits_deep_external_input(self.date)
95
96      def process_day(self, inputs, output_path):
97
98          self.yql.query(
99              gas_stations_query_template.format(
100                 organization_categories=config.ORGANIZATION_CATEGORIES,
101                 deep_visits=inputs.table,
102                 matching_idfa=get_matching_table('idfa', 'crypta_id'),
103                 matching_gaid=get_matching_table('gaid', 'crypta_id'),
104                 name_to_variable=',\n'.join(
105                     [u'("{}", "{}")'.format(key, value)
106                      for key, value in name_to_variable.iteritems()]
107                 ),
108                 output_table=output_path,
109             ),
110             transaction=self.transaction,
111         )
112
```

Basic Example of Household Details

graph > metrics > household > query.sql

```
DEFINE SUBQUERY $hh_size_by_yuid($title) AS
    SELECT ($title || key) AS key, COUNT(1) AS hh_size_by_yuids
    FROM $composition
    GROUP BY $size_to_range(size) AS key;
END DEFINE;

DEFINE SUBQUERY $hh_size_by_crypta_id($title, $predicat) AS
    SELECT
        ($title || key) AS key,
        COUNT(1) AS hh_size_by_crypta_id
    FROM $composition
    WHERE $predicat(size, socdems)
    GROUP BY CAST(Yson::GetLength(Yson::Lookup(data, 'crypta_ids')) AS String) AS key;
END DEFINE;

DEFINE SUBQUERY $hh_by_socdems($title, $predicat) AS
    $hh_socdem = (
        SELECT
            hhid,
            size,
            IF((Yson::LookupInt64(info, 'female') != 0), 'female', Null) AS has_female,
            IF((Yson::LookupInt64(info, 'male') != 0), 'male', Null) AS has_male,
            IF((Yson::LookupInt64(info, 'grand') != 0), 'grand', Null) AS has_old,
            IF((Yson::LookupInt64(info, 'child') != 0), 'child', Null) AS has_child
        FROM $composition
        WHERE $predicat(size, socdems)
    );

    SELECT ($title || groups) AS key, hh_c AS hh_socdem_count
    FROM (
        SELECT groups, SUM(size) AS hh_c
        FROM $hh_socdem
        GROUP BY String::JoinFromList(
            ListSort(AsList(has_female, has_male, has_old, has_child)),
            '_') AS groups
    ) WHERE groups != "";
```

Appmetrica Being Used to Pull Wifi Connection Types:

profile > runners > segments > lib > coded_segments > connection_type.py > ...

```
connection_type_query = """
INSERT INTO `{output_table}` WITH TRUNCATE
SELECT
    id,
    'mm_device_id' AS id_type,
    CASE
        WHEN types == AsSet('3g') THEN '3g'
        WHEN types == AsSet('4g') THEN '4g'
        ELSE '3g_4g'
    END AS segment_name
FROM (
    SELECT
        id,
        ToSet(AGGREGATE_LIST_DISTINCT(segment_name)) AS types
    FROM `{input_table}`
    GROUP BY id
);
"""


class ConnectionType(RegularSegmentBuilder):
    name_segment_dict = {
        '3g': (557, 17823841),
        '4g': (557, 17823853),
        '3g_4g': (557, 17823847),
    }

    number_of_days = 35

    def requires(self):
        return {
            'AppMetrica': LogProcessor(
                ProcessAppMetricaForConnectionType,
                self.date,
                self.number_of_days,
            ),
```

Appmetrica Data Being Used to Identify Users with Common SSIDS (Wifi Networks)

```python
class ImportSsidMobileMetrikaTask(BaseTask):
    date = DateParameter()
    SSID_THRESHOLD = 20
    YUID_THRESHOLD = 20
    DAYS_IN_MONTH = 7

    def requires(self):
        """
        this tasks must be done to complete this task
        """
        task_list = [
            ImportSsidMobileMetrikaDayTask(date=self.date, target_date=target_date, ssid_threshold=self.SSID_THRESHOLD)
            for target_date in days_range_back(self.date, self.DAYS_IN_MONTH)
        ]
        return task_list
```

Appmetrica Data Being Used to Identify Users with Common SSIDS (Wifi Networks)

```
def _run(self):
    self.yt.create_table_with_schema(
        self.destination, self.destination_schema, strict=True, recreate_if_exists=True
    )
    with self.yt.TempTable() as unexploded, self.yt.TempTable() as not_unique:
        self.yql.execute(self.query(unexploded), syntax_version=1)
        run_native_reduce(
            reducer_name="NCommonWifiAP::TExploder",
            source=unexploded,
            destination=not_unique,
            proxy=self.yt.proxy,
            transaction=self.yt.transaction_id,
            pool=conf.Yt.POOL,
            title="Explode yandexuids with common wifi access point",
            reduce_by=["ssid"],
        )
        yuid_pair = [conf.Constants.YUID_LEFT, conf.Constants.YUID_RIGHT]
        self.yt.run_sort(not_unique, not_unique, sort_by=yuid_pair)
        run_native_reduce(
            reducer_name="NCommonWifiAP::TUnique",
            source=not_unique,
            destination=self.destination,
            proxy=self.yt.proxy,
            transaction=self.yt.transaction_id,
            pool=conf.Yt.POOL,
            title="Make yandexuids with common wifi access point unique",
            reduce_by=yuid_pair,
        )
        self.yt.run_sort(self.destination, sort_by=yuid_pair)
    self.yt.set(self.destination + "/@generate_date", self.date.isoformat())
```

Sources

graph > fuzzy > lib > config.py > GeoPaths

```
59
60  class SourceTypes(object):
61      EMAIL_LOGIN = "EMAIL_LOGINS"
62      EMAIL_SIMILAR = "EMAIL_SIMILAR"
63      GEO_HOMEWORK = "GEO_HOMEWORK"
64      HOUSEHOLD = "HOUSEHOLD"
65      REQANS_LOG = "REQANS_LOG"
66      SSID = "SSID"
67
68
```

← Search Data

← Wifi

Yandex IDs Associated with Email

```python
class EmailPaths(object):
    ROOT = ROOT
    # Emails
    BASE = "{root}/email".format(root=ROOT)

    ALL_EMAILS_TABLE = "{base}/all_emails".format(base=BASE)
    ALL_EMAIL_LOGINS_TABLE = "{base}/all_email_logins".format(base=BASE)
    ALL_EMAILS_SORTED_BY_LOGIN = "{base}/all_email_logins.sorted_by_login".format(base=BASE)
    ALL_EMAIL_LOGINS_PAIRS_TABLE = "{base}/all_email_logins.pairs".format(base=BASE)
    ALL_EMAILS_GROUPED_BY_LOGIN = "{base}/all_email_logins.groups".format(base=BASE)
    ALL_YUID_PAIRS_FROM_EMAIL_LOGIN = "{base}/all_yuid_pairs_from_email_logins_matching".format(base=BASE)
    ALL_YUID_PAIRS_FROM_SIMILAR_EMAILS = "{base}/all_yuid_pairs_from_similar_emails".format(base=BASE)

    ALL_EMAILS_TABLE_SCHEMA = {"email": "string", "yuids": "any"}
    ALL_EMAIL_LOGINS_TABLE_SCHEMA = {"login": "string", "email": "string", "yuids": "any"}
    ALL_EMAILS_SORTED_BY_LOGIN_SCHEMA = {"login": "string", "email": "string", "yuids": "any"}
    ALL_EMAILS_GROUPED_BY_LOGIN_SCHEMA = {"login": "string", "all_emails": "any", "howmany": "uint64"}
    ALL_EMAIL_LOGINS_PAIRS_TABLE_SCHEMA = {
        "email_1": "string",
        "email_2": "string",
        "login": "string",
        "yuids_1": "any",
        "yuids_2": "any",
    }
    ALL_YUID_PAIRS_FROM_EMAIL_LOGIN_SCHEMA = {
        Constants.YUID_LEFT: ("uint64", True),
        Constants.YUID_RIGHT: ("uint64", True),
        "match": "any",
    }
    ALL_YUID_PAIRS_FROM_SIMILAR_EMAILS_SCHEMA = {
        Constants.YUID_LEFT: "uint64",
        Constants.YUID_RIGHT: "uint64",
        "email_left": "string",
        "email_right": "string",
        "fragment": "string",
    }
```

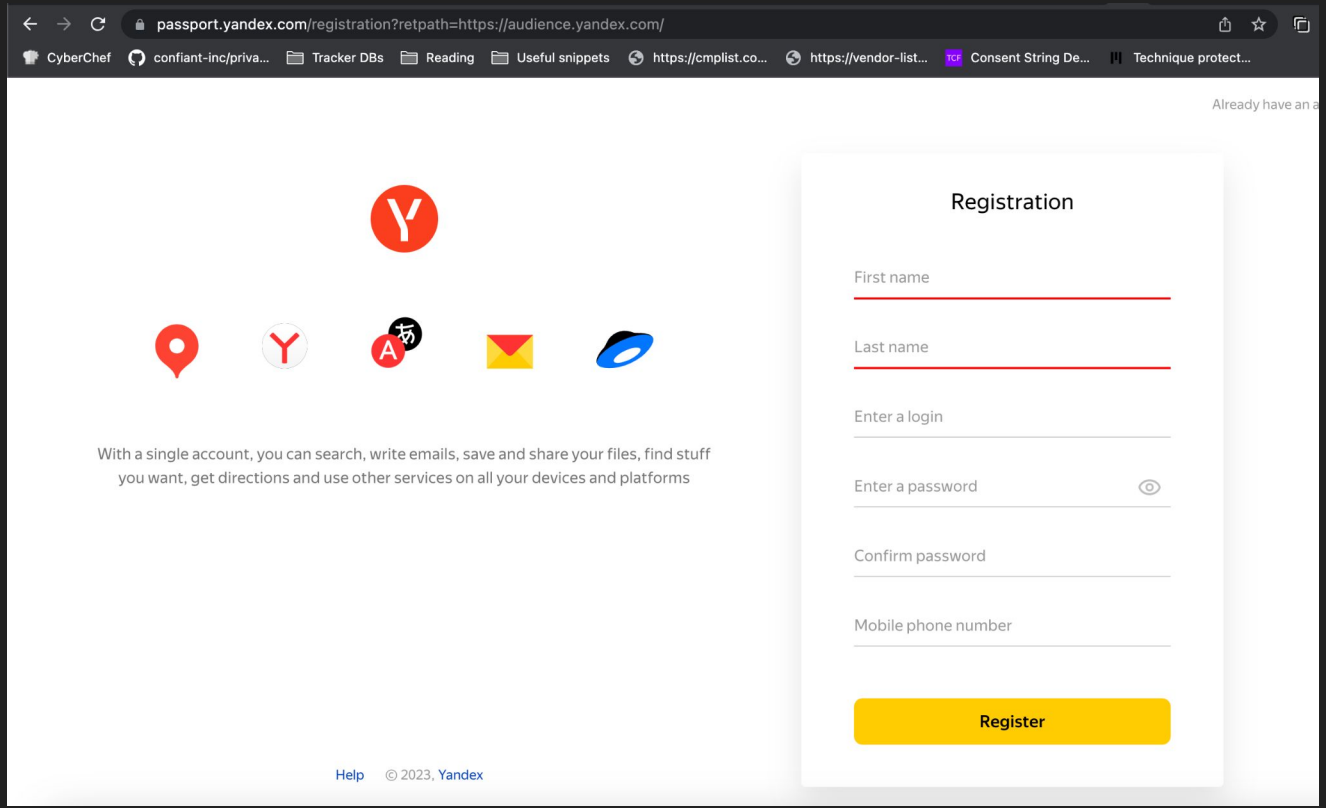
Login Data

graph > fuzzy > lib > tasks > sources > visitlog_logins > extract.py > ...

```
59        def filter_rare_logins_options(self):
60            return TFilterRareLoginsOptions(Threshold=self.threshold).SerializeToString()
61
62        @property
63        def filter_keys_options(self):
64            return TFilterKeysOptions(
65                Keywords=[
66                    "login",
67                    "user",
68                    "userid",
69                    "clientid",
70                    "uid",
71                    "email",
72                    "emailhash",
73                    "\u043b\u043e\u0433\u0438\u043d",
74                    "computerid",
75                    "cid",
76                    "suserid",
77                ]
78            ).SerializeToString()
79
```

Extracting multiple types of identifiers

Passport


passport.yandex.com/registration?retpath=https://audience.yandex.com/
CyberChef
confiant-inc/priva...
Tracker DBs
Reading
Useful snippets
https://cmplist.co...
https://vendor-list...
Consent String De...
Technique protect...
Already have an a
With a single account, you can search, write emails, save and share your files, find stuff you want, get directions and use other services on all your devices and platforms
Registration
First name
Last name
Enter a login
Enter a password
Confirm password
Mobile phone number
Register
Help
© 2023, Yandex

Passport User ID Associated with Phone

graph > data_import > passport > lib > query > passport.sql

```
33
34  $out_login_tbl = $soup_output_dir || $edge(IdType::PUID(), IdType::PHONE(), SourceType::PASSPORT_PROFILE(), LogSource::PASSPORT_PHONE_DUMP())
35  INSERT INTO $out_login_tbl WITH TRUNCATE
36  SELECT
37      id1,
38      IdType::PUID() AS id1Type,
39      id2,
40      IdType::PHONE() AS id2Type,
41      SourceType::PASSPORT_PROFILE() AS sourceType,
42      LogSource::PASSPORT_PHONE_DUMP() AS logSource,
43      ListCreate(String) AS dates
44  FROM (
45      SELECT DISTINCT puid, phone
46      FROM $log FLATTEN LIST BY phones AS phone
47  ) WHERE Identifiers::IsSignificantPhone(phone)
48  GROUP BY
49      puid AS id1,
50      phone AS id2
51  ;
52
```

Geo Graphs

Using lat/long data associated with “predicted home”, linked to Yandex UID

graph > fuzzy > lib > tasks > sources > geo > C geo_operations.h

```
61              for (; input->IsValid(); input->Next()) {
62                  const auto& row = input->GetRow();
63                  if (not IsRowValid(row)) {
64                      continue;
65                  }
66
67                  const ui64 yandexuid = FromString<ui64>(row["yandexuid"].AsString());
68                  const auto& homeCoordinates = row["predicted_home"];
69                  const auto latitude = homeCoordinates["latitude"].AsDouble();
70                  const auto longitude = homeCoordinates["longitude"].AsDouble();
71                  const auto& square = computeSquare({.Lat = latitude, .Lon = longitude}, State->radius());
72
73                  /**
74                   *    + + -
75                   *    + + -
76                   *    + - -
77                   */
78                  for (int beltOffset : {-1, 0, 1}) {
79                      for (int sqOffset : {-1, 0}) {
80                          if (beltOffset == -1 && sqOffset == 0) {
81                              continue;
82                          }
83                          const ui64 square_idx = ConvertSquareToIdx({.Belt = square.Belt + beltOffset, .Sq = square.Sq + sqOffset});
84                          TGeoSquare out;
85                          out.set_yandexuid(yandexuid);
86                          out.set_lat(latitude);
87                          out.set_lon(longitude);
88                          out.set_squareidx(square_idx);
89                          output->AddRow(out);
90                      }
91                  }
```

Geo Graphs

Then using that data to find literal neighbors within a certain radius of that home

graph > fuzzy > lib > tasks > sources > geo > C geo_operations.h

```
106
107     class TFindNeighbors: public IReducer<TTableReader<TGeoSquare>, TTableWriter<TNeighborsDistance>> {
108     public:
109         TFindNeighbors()
110                 : State()
111         {
112         }
113
114         TFindNeighbors(const TBuffer& buffer)
115                 : State(buffer)
116         {
117         }
118
119         void Do(TTableReader<TGeoSquare>* input, TTableWriter<TNeighborsDistance>* output) override {
120             const double radius = State->radius();
121             TVector<TGeoSquare> candidates;
122             for (; input->IsValid(); input->Next()) {
123                 const auto& row = input->GetRow();
124                 candidates.push_back(row);
125             }
126             for (auto i : xrange(candidates.size())) {
127                 for (auto j : xrange(i + 1, candidates.size())) {
128                     const auto& left = candidates.at(i);
129                     const auto& right = candidates.at(j);
130                     if (left.yandexuid() == right.yandexuid()) {
131                         continue;
132                     }
133                     double distance = computeDistance({.Lat = left.lat(), .Lon  = left.lon()}, {.Lat = right.lat(), .Lon = right.lon()});
134                     if (distance > radius) {
135                         continue;
136                     }
137                     TNeighborsDistance out;
138                     out.set_distance(distance);
139                     out.set_yandexuidleft(Min(left.yandexuid(), right.yandexuid()));
140                     out.set_yandexuidright(Max(left.yandexuid(), right.yandexuid()));
141                     output->AddRow(out);
142                 }
```

Appmetrica and Taxi Data Being Used to Generate Segments about Households with Children

profile > runners > segments > lib > coded_segments > with_children_by_ages.py > ...

```
441     def prepare_with_children_by_app(self, with_children_by_app_table):
442         app_segment_name_tuples = []
443
444         for segment_name, apps in SEGMENT_APP_DICT.iteritems():
445             for app in apps:
446                 app_segment_name_tuples.append('AsTuple("{}", "{}"),'.format(app, segment_name))
447
448         self.yql.query(
449             app_metrica_query.format(
450                 devid_by_app_table=self.input()['DevidByApp'].table,
451                 output_table=with_children_by_app_table,
452                 app_to_segment_name='\n'.join(app_segment_name_tuples),
453             ),
454             transaction=self.transaction,
455         )
456
457     def build_segment(self, inputs, output_path):
458         with self.yt.TempTable() as taxi_puid_table, \
459                 self.yt.TempTable() as app_metrica_table:
460             self.yt.run_map(
461                 extract_children_from_taxi,
462                 inputs['TaxiData'].table,
463                 taxi_puid_table,
464             )
465
466             self.prepare_with_children_by_app(app_metrica_table)
467
468             self.yql.query(
469                 with_children_query_template.format(
470                     metrics_table=inputs['ProcessedMetrics'].table,
471                     reqans_table=inputs['ProcessedReqans'].table,
472                     app_metrica_table=app_metrica_table,
473                     taxi_data_table=taxi_puid_table,
474                     id_to_crypta_id_table=config.VERTICES_NO_MULTI_PROFILE,
475                     crypta_id_to_hhid_table=config.HOUSEHOLD_CRYPTA_ID_TO_HHID,
476                     yandexuid_to_hhid_table=config.HOUSEHOLD_REVERSED_TABLE,
477                     hhid_to_yandexuid_table=config.HOUSEHOLD_ENRICH_TABLE,
478                     output_table=output_path,
479                 ),
480                 transaction=self.transaction,
481             )
482
```

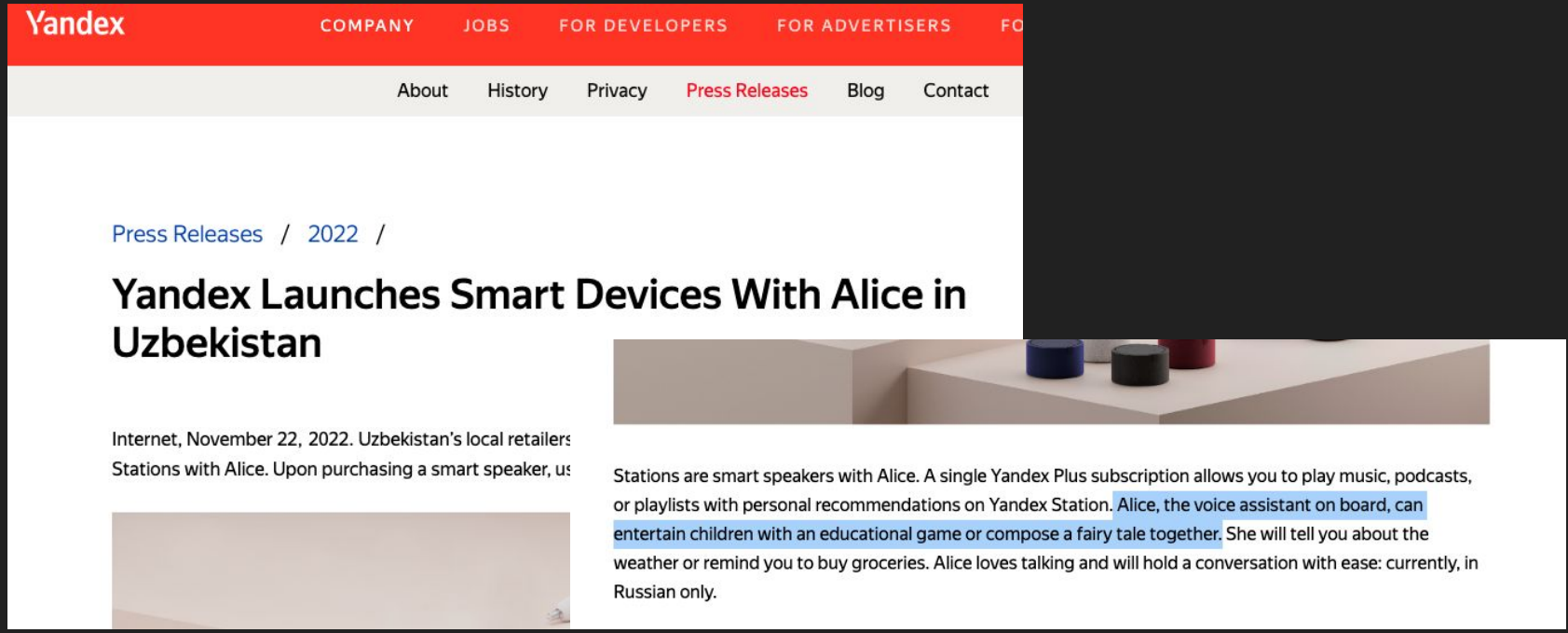
Appmetrica and Taxi Data Being Used to Generate Segments about Households with Children

```
456
457     def build_segment(self, inputs, output_path):
458         with self.yt.TempTable() as taxi_puid_table, \
459                 self.yt.TempTable() as app_metrica_table:
460             self.yt.run_map(
461                 extract_children_from_taxi,
462                 inputs['TaxiData'].table,
463                 taxi_puid_table,
464             )
465
466             self.prepare_with_children_by_app(app_metrica_table)
467
```

ID Mapping Associations

```
self.yql.query(
    with_children_query_template.format(
        metrics_table=inputs['ProcessedMetrics'].table,
        reqans_table=inputs['ProcessedReqans'].table,
        app_metrica_table=app_metrica_table,
        taxi_data_table=taxi_puid_table,
        id_to_crypta_id_table=config.VERTICES_NO_MULTI_PROFILE,
        crypta_id_to_hhid_table=config.HOUSEHOLD_CRYPTA_ID_TO_HHID,
        yandexuid_to_hhid_table=config.HOUSEHOLD_REVERSED_TABLE,
        hhid_to_yandexuid_table=config.HOUSEHOLD_ENRICH_TABLE,
        output_table=output_path,
    ),
    transaction=self.transaction,
)
```

Profiles Integrate Biometric Data, Most Likely from Smart Speakers That Use Yandex's Alice Smart Assistant



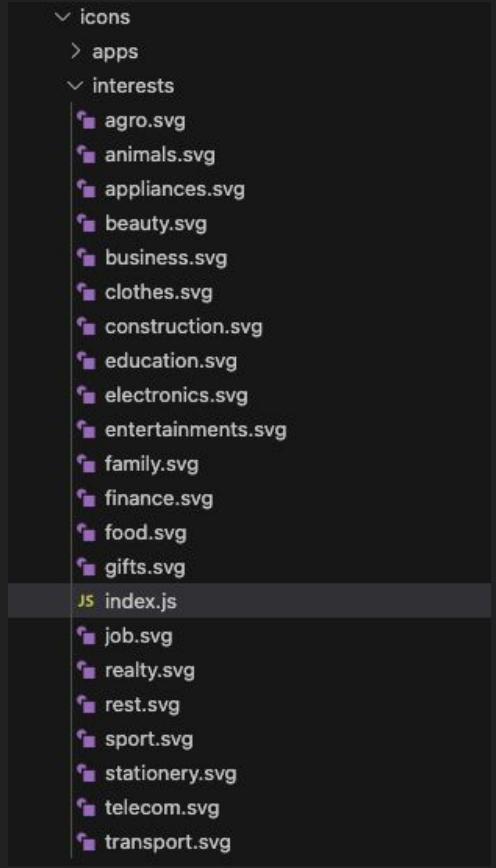
Possible Children by Voice

profile > runners > segments > lib > coded_segments > children_age_segment_clarification.py > ...

```
12
13  clarify_children_yql_template = """
14  $possible_children_by_voice = (
15      SELECT `uuid`, TableName() AS `date`, '0_12' AS segment_name
16      FROM RANGE(`{biometry_folder}`, `{biometry_first_date}`, `{biometry_last_date}`)
17      WHERE bio_child > 0.8
18  );
19
20  $possible_children_by_voice = (
21      SELECT DISTINCT `uuid`, `date`, segment_name
22      FROM $possible_children_by_voice
23  );
24
25  $possible_children_by_voice = (
26      SELECT `uuid`, segment_name
27      FROM $possible_children_by_voice
28      GROUP BY `uuid`, segment_name
29      HAVING COUNT(*) >= 2
30  );
31
32  $sources_new_age = (
33      SELECT matching.cryptaId AS cryptaId,
34          CASE
35              WHEN socdem_storage.birth_date > '{thirteenth_birthday}' THEN '0_12'
36              WHEN '{thirteenth_birthday}' >= socdem_storage.birth_date AND
37                  socdem_storage.birth_date > '{eighteenth_birthday}' THEN '13_17'
38              ELSE '18_99'
39          END AS segment_name
40      FROM `{socdem_storage_table}` AS socdem_storage
41      INNER JOIN `{id_to_crypta_id_table}` AS matching
42      ON socdem_storage.id == matching.id AND socdem_storage.id_type == matching.id_type
43      WHERE socdem_storage.birth_date is not Null
44      UNION ALL
45      SELECT matching.cryptaId AS cryptaId, biometry.segment_name AS segment_name
46      FROM $possible_children_by_voice AS biometry
47      INNER JOIN `{id_to_crypta_id_table}` AS matching
48      ON biometry.`uuid` == matching.id
49      WHERE matching.id_type == 'uuid'
50  );
```

UI for Infographics Card

```
27
28      const marriedText = convertMarriedToSingleText(exactDemographics.gender, married);
29      const incomeText = convertIncomeSegmentToText(exactDemographics.income);
30      const hasChildrenText = convertHasChildrenToText(hasChildren);
31
32      return (
33          <div className="BasicInfoGraphics">
34              <img alt="" className="BasicInfoGraphics-Image" src={images[exactDemographics.gender]}/>
35              <div className="BasicInfoGraphics-Bubble BasicInfoGraphics-Bubble_family">{marriedText}</div>
36              <div className="BasicInfoGraphics-Bubble BasicInfoGraphics-Bubble_income">{incomeText}</div>
37              <div className="BasicInfoGraphics-Bubble BasicInfoGraphics-Bubble_children">{hasChildrenText}</div>
38              <div className="BasicInfoGraphics-Interest BasicInfoGraphics-Interest_first">
39                  <div className="BasicInfoGraphics-InterestIcon"
40                       style={{ backgroundImage: `url(${interestIcons[0]})` }}/>
41              </div>
42              <div className="BasicInfoGraphics-Interest BasicInfoGraphics-Interest_second">
43                  <div className="BasicInfoGraphics-InterestIcon"
44                       style={{ backgroundImage: `url(${interestIcons[1]})` }}/>
45              </div>
46              <div className="BasicInfoGraphics-Interest BasicInfoGraphics-Interest_third">
47                  <div className="BasicInfoGraphics-InterestIcon"
48                       style={{ backgroundImage: `url(${interestIcons[2]})` }}/>
49              </div>
50          </div>
51      );
52  }
```



Search Profile by ID

web > portal > src > graph > search > JS SearchPanel.js > ...

```
112             return (
113                 <div key={"inputs-" + suffix} className="input-group">
114                     <div style={{display: showIdInput ? "block" : "none"}}>
115                         <ValueInput
116                             key="id_value"
117                             placeholder="[id value]"
118                             value={parameters["idValue" + suffix]}
119                             onChange={changeParameter("uid" + suffix)}
```

```
return (
    experiments.status !== 403 && (
        <>
            <div className="experiments-bar">
                <div className="experiments-select-uid-type">
                    <RadioButton
                        value={activeUid}
                        size="s"
                        view="default"
                        className="select-sorting"
                        onChange={(event) => selectUidType(event.target.value)}
                        options={[
                            { value: "uid", children: t("by") + " yandexuid" },
                            { value: "cryptaId", children: t("by") + " CryptaID" },
                        ]}
```

IDs Associated with Social Media Accounts

web > portal > src > public-info > sections > GraphSection > JS GraphSection.js > GraphSection >

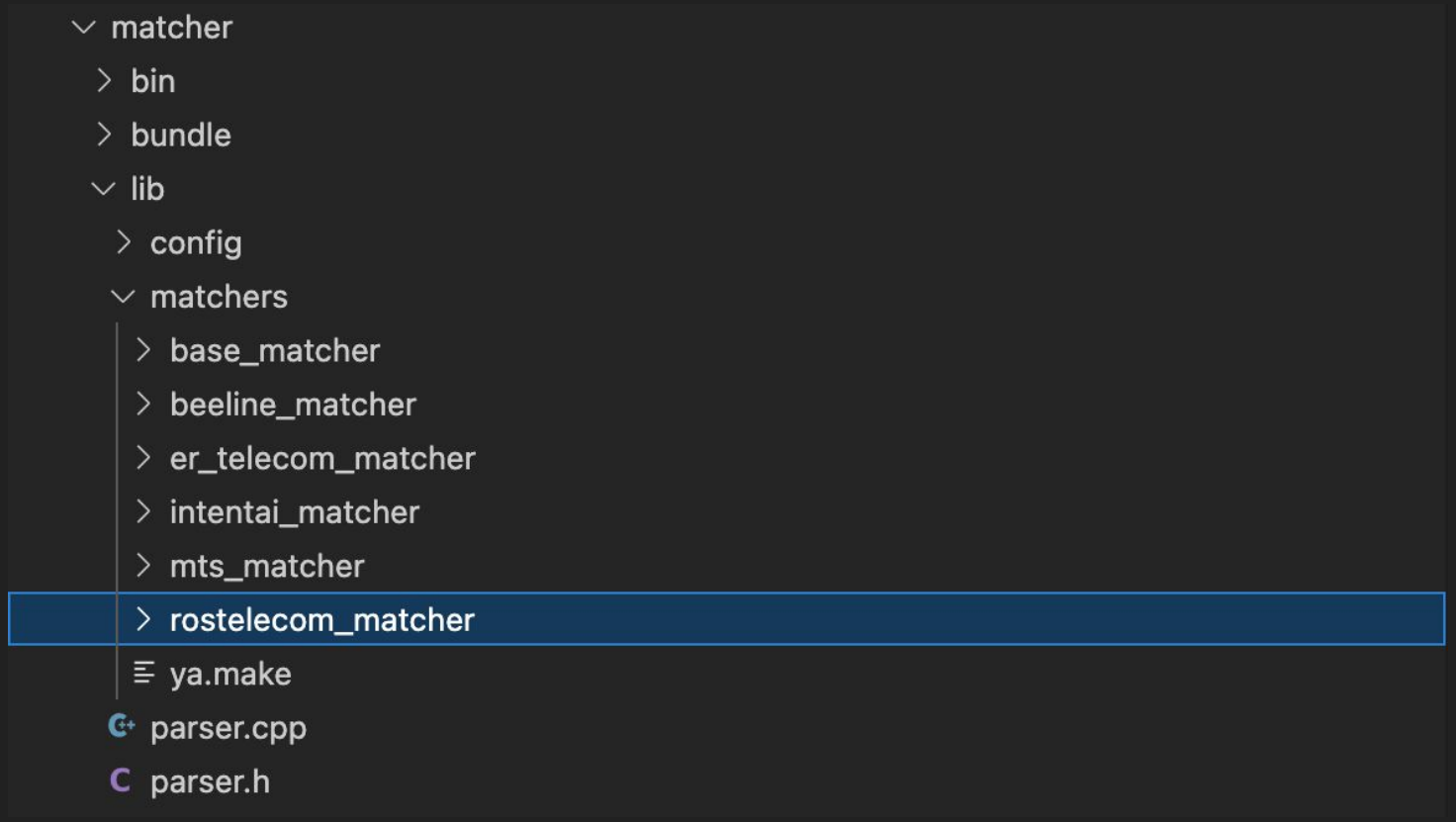
```
import React, { useEffect, useMemo, useState } from "react";
import { useSelector } from "react-redux";
import { getPublicGraph, getPublicGraphLoading } from "../../store/selectors";
import { Graph, GraphSkeleton } from "../../components/Graph/Graph";
import { Section } from "../../components/Section/Section";
import { getServiceIcon } from "../../icons/services";
import { getAppIcon } from "../../icons/apps";

import "./GraphSection.scss";

import noData from "./no-data.svg";

const IMAGE_SIZE_XS = 12;
const IMAGE_SIZE_S = 36;
const IMAGE_SIZE_M = 56;
const IMAGE_SIZE_L = 80;

const NODE_MAPPING = {
    email: {
        imageSize: IMAGE_SIZE_M,
        imageHref: "mail",
    },
    yandexuid: {
        imageSize: IMAGE_SIZE_XS,
        imageHref: "yandexuid",
    },
    idfa: {
        imageSize: IMAGE_SIZE_L,
        imageHref: "ios",
    },
    gaid: {
        imageSize: IMAGE_SIZE_L,
        imageHref: "android",
    },
    oaid: {
        imageSize: IMAGE_SIZE_L,
        imageHref: "android",
    },
    login: {
        imageSize: IMAGE_SIZE_M,
        imageHref: "key",
    },
    puid: {
        imageSize: IMAGE_SIZE_M,
        imageHref: "key",
    },
    instagram_login: {
        imageSize: IMAGE_SIZE_M,
        imageHref: "instagram"
    },
    instagram_id: {
        imageSize: IMAGE_SIZE_M,
```

web > portal > src > public-info > sections > GraphSection > JS GraphSection.js > GraphSection > useEffec

```
    instagram_id: {
        imageSize: IMAGE_SIZE_M,
        imageHref: "instagram"
    },
    fb_id: {
        imageSize: IMAGE_SIZE_M,
        imageHref: "facebook"
    },
    ok_id: {
        imageSize: IMAGE_SIZE_M,
        imageHref: "ok"
    },
    vk_id: {
        imageSize: IMAGE_SIZE_M,
        imageHref: "vk"
    },
    vk_name: {
        imageSize: IMAGE_SIZE_M,
        imageHref: "vk"
    },
    kp_id: {
        imageSize: IMAGE_SIZE_M,
        imageHref: "kinopoisk"
    },
}

function getNodeMapping(item) {
    if (item.idType === 'uuid') {
        // App
        return { imageHref: item.icon, imageSize: IMAGE_SIZE_S };
    }

    return NODE_MAPPING[item.icon] ?? { imageHref: "default", imageSize: IMAGE_SIZE_XS };
}

function getImage(item) {
    const disabled = !item.isActive;

    if (item.idType === 'uuid') {
        return getAppIcon(item.imageHref, disabled)
            .catch(() => getAppIcon("default", disabled));
    }

    return getServiceIcon(item.imageHref, disabled)
        .catch(() => getServiceIcon("default", disabled));
}
```

Matcher

Matcher


matcher
bin
bundle
lib
config
matchers
base_matcher
beeline_matcher
er_telecom_matcher
intentai_matcher
mts_matcher
rostelecom_matcher
ya.make
parser.cpp
parser.h

Rostelecom Matcher

ext_fp > matcher > lib > matchers > rostelecom_matcher > rostelecom_matcher.cpp

```cpp
void TRostelecomMatcher::AddConnection(const TFpEvent& event) {
    auto connection = MakeConnection(event);

    Stats.Count->Add("events.incoming.rostelecom.count");
    Request += TStringBuilder() << connection.Ip << '\t'
                                << connection.Port << '\t'
                                << connection.Timestamp << '\t'
                                << connection.Domain << '\n';
}

TMatches TRostelecomMatcher::GetMatches() {
    if (Request.length() == 0) {
        return TMatches();
    }
    const auto& requestId = CreateGuidAsString();
    Log->info("Rostelecom request {} body:\n{}", requestId, Request);

    NNeh::TMessage message(GetApiUrl(), "");
    Y_ENSURE(NNeh::NHttp::MakeFullRequest(message, "", Request, "text/plain"), "Failed to build request to Rostelecom API");

    Stats.Count->Add("api.calls.rostelecom.count");
    const auto& resp = MakeRequest(Client, message, TDuration::MilliSeconds(Config.GetApiCallTimeoutMs()), "Rostelecom", requestId, Log);

    return ParseResponse(resp->Data);
}

TString TRostelecomMatcher::GetApiUrl() const {
    return "post://" + Config.GetApiUrl();
}

TMatches TRostelecomMatcher::ParseResponse(const TString& response) {
    TStringInput stringInput(response);

    TString line;
    TConnection connection;
    TString extId;
    TMatches matches;

    while (stringInput.ReadLine(line)) {
        Split(TStringBuf(line), '\t', connection.Ip, connection.Port, connection.Timestamp, connection.Domain, extId);
        matches[connection] = TMatchResult{.Status = TMatchResult::EStatus::Found, .ExtId = extId};
    }

    return matches;
}
```

Rostelecom Matcher

ext_fp > matcher > lib > matchers > rostelecom_matcher > rostelecom_matcher.cpp

```
31  void TRostelecomMatcher::AddConnection(const TFpEvent& event) {
32      auto connection = MakeConnection(event);
33
34      Stats.Count->Add("events.incoming.rostelecom.count");
35      Request += TStringBuilder() << connection.Ip << '\t'
36                                  << connection.Port << '\t'
37                                  << connection.Timestamp << '\t'
38                                  << connection.Domain << '\n';
39  }
40
```

Rostelecom Matcher

```
61  TMatches TRostelecomMatcher::ParseResponse(const TString& response) {
62      TStringInput stringInput(response);
63
64      TString line;
65      TConnection connection;
66      TString extId;
67      TMatches matches;
68
69      while (stringInput.ReadLine(line)) {
70          Split(TStringBuf(line), '\t', connection.Ip, connection.Port, connection.Timestamp, connection.Domain, extId);
71          matches[connection] = TMatchResult{.Status = TMatchResult::EStatus::Found, .ExtId = extId};
72      }
73
74      return matches;
75  }
```

Test Result Data

ext_fp > matcher > bin > test > canondata > {} result.json > [ ] test_matcher.test_matcher > { } 1


```
{
    "test_matcher.test_matcher": [
        {
            "duid": 16999999761000006,
            "ext_id": "fake_ertelecom_id_for_5.3.100.0",
            "ext_source": "ertelecom",
            "hitlogid": 100506,
            "ip": "5.3.100.0",
            "log_type": "bs-watch-log",
            "logid": 0,
            "original_domain": "domain-6.ru",
            "port": 5555,
            "rtmr_timestamp": 1699999977,
            "unixtime": 1699999970,
            "user_agent": "Mozilla/5.0 (Windows NT РУС)",
            "watchid": 200000000000000006,
            "yuid": 10061699999976
        },
        {
            "duid": 16999999861000016,
            "ext_id": "mts_id_for_160.1.2.4",
            "ext_source": "mts",
            "hitlogid": 100516,
            "ip": "160.1.2.4",
            "log_type": "bs-watch-log",
            "logid": 0,
            "original_domain": "domain-16.ru",
            "port": 4444,
            "rtmr_timestamp": 1699999987,
            "unixtime": 1699999970,
            "user_agent": "Mozilla/5.0 (Windows NT РУС)",
            "watchid": 200000000000000016,
            "yuid": 10161699999986
        },
        {
            "duid": 16999999931000003,
            "ext_id": "fake_ertelecom_id_for_5.3.62.0",
            "ext_source": "ertelecom",
            "hitlogid": 100503,
            "ip": "5.3.62.0",
            "log_type": "bs-watch-log",
            "logid": 0,
            "original_domain": "domain-3.ru",
            "port": 2222,
            "rtmr_timestamp": 1699999994,
            "unixtime": 1699999990,
            "user_agent": "Mozilla/5.0 (Windows NT РУС)",
            "watchid": 200000000000000003,
            "yuid": 10031699999993
        },
```

Roadmap

- Background on Yandex Leak
- Dive into code:
  - What data Yandex is collecting
  - What Yandex is doing with that data
  - Who Yandex is sharing that data with
- Conclusions and wrap up
- Q&A

Conclusion

What they have:

- WiFi SSID
- Device information
- Location
- Search history
- Biometric voice data
- E-Mail
- Phone number
- Birth date
- App usage data
- Service usage data
- Home location
- Work location
- Movie/music history
- E-Mail data

What they can determine:

- Household composition
- Age
- Income
- Gender
- Children's ages
- Interests
- Home location
- Physical neighbors
- Travel plans/ history
- Switching providers
- Similar users
- Proximity to local businesses
- Smoker
- Music preferences

Wrap Up

- Yandex has access to a broad international reach of data and it has been evasive about what it can do with that data
- A small amount of data can say a lot when it is matched to entries from a company's other data sources and analyzed
- Yandex has code to sync some of its data with a Russian-state owned entity

Takeaways

- Anonymization is very easily undone when data gets combined with pools from other sources that may contain identifying data
- Pay attention to who runs your SDKs, what data points they collect, and where they send your user data.
- Who gets access to a company's user data when its assets are sold, the geopolitical climate changes, or a government tightens its control?

CONFIANT

Q&A

Link to Write Up:
(available August 10th)
bit.ly/455utBP

Link to Slides:
(available August 10th)
bit.ly/3qail3p