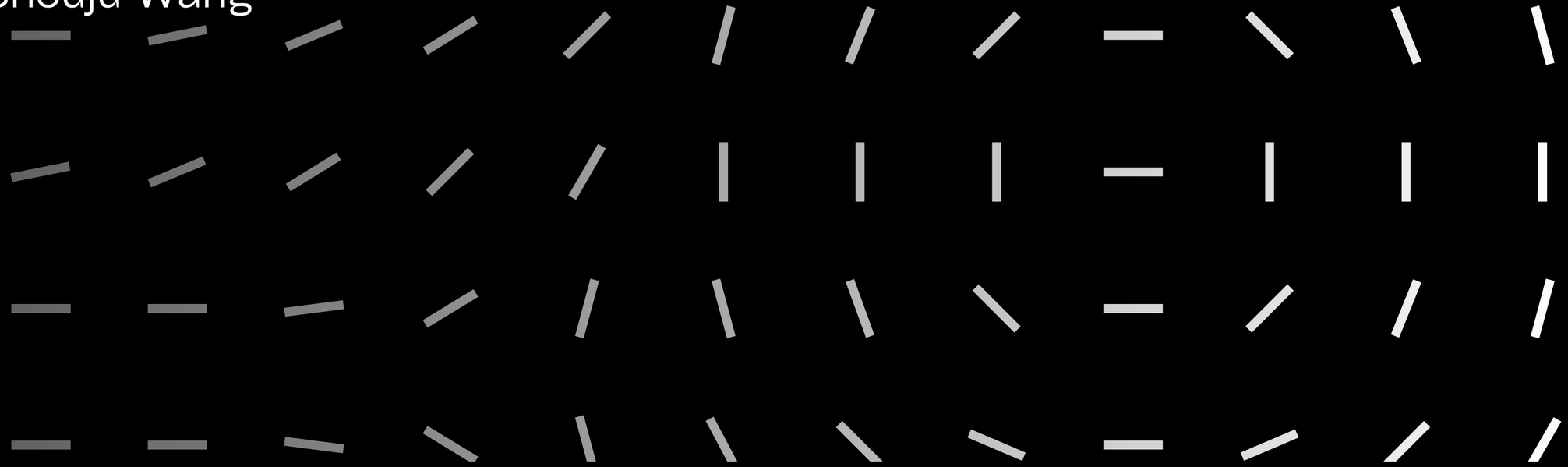
Sichang He (Steven)
Shouju Wang



Route Verification Server

A REST API server for accessing inter-domain routing information

COMPSCI 310

DUKE KUNSHAN UNIVERSITY

Instructor : Dr. Mustafa Misir

Customer : Dr. Italo Cunha

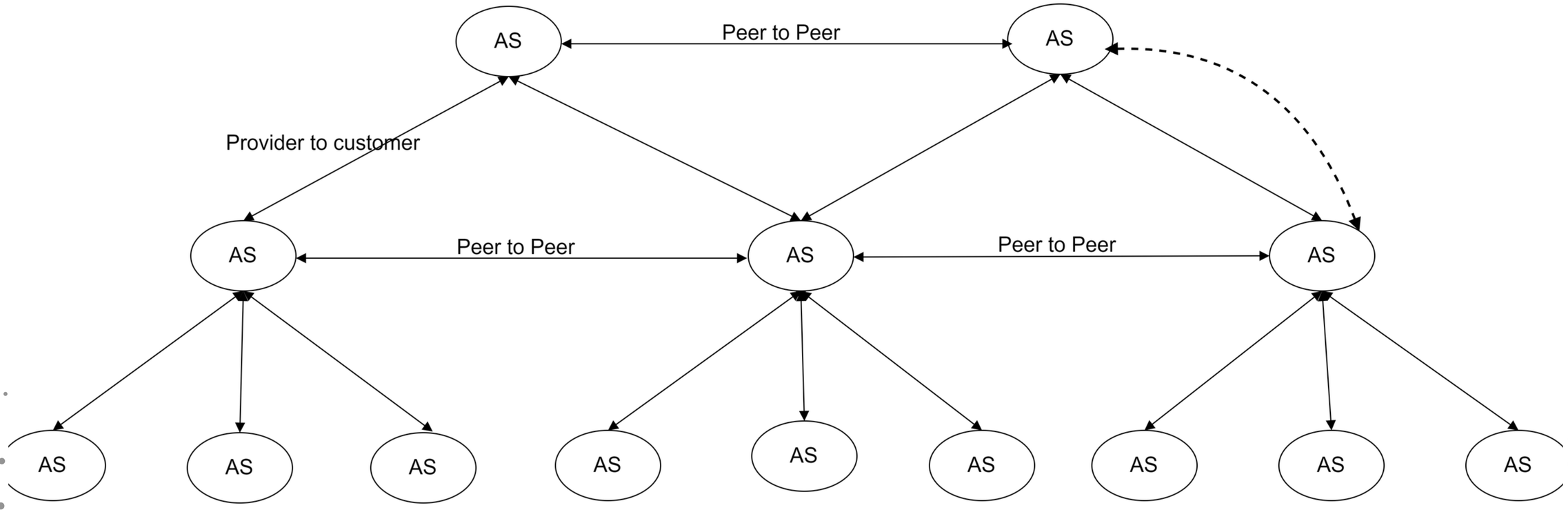
Table of content

Motivation and Objectives

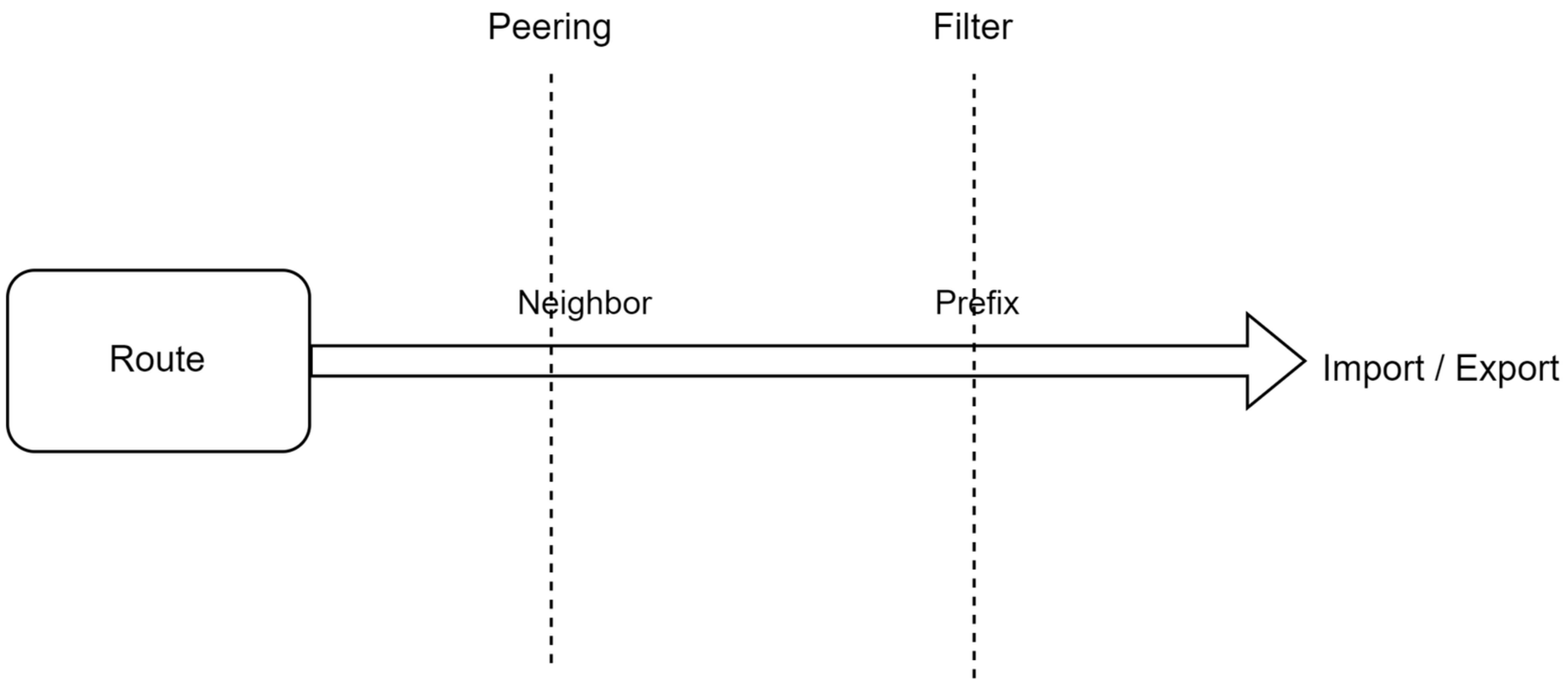
Inter-domain routing

Routing Policy Specification Language (RPSL)

Route verification report

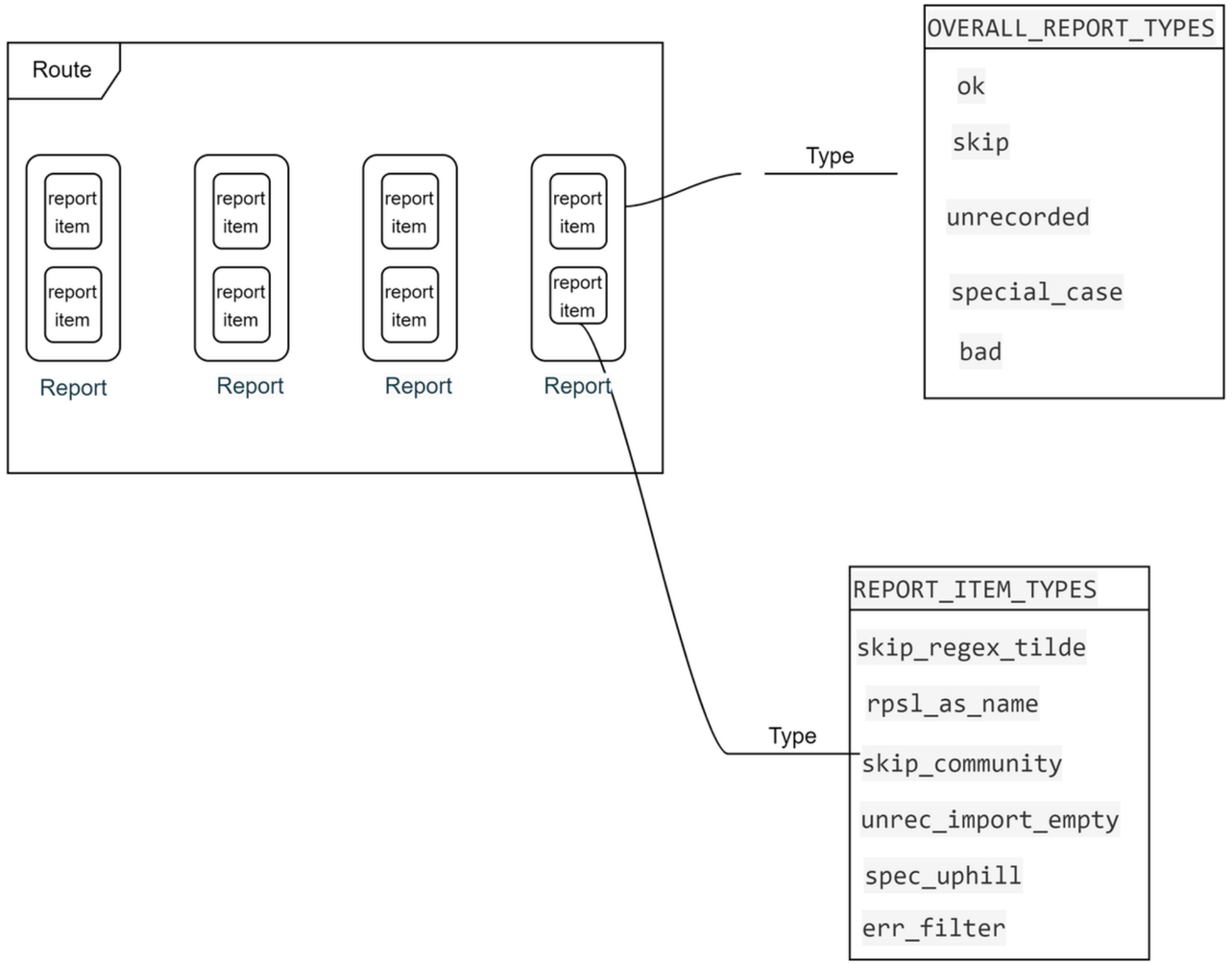


Autonomous System (AS)

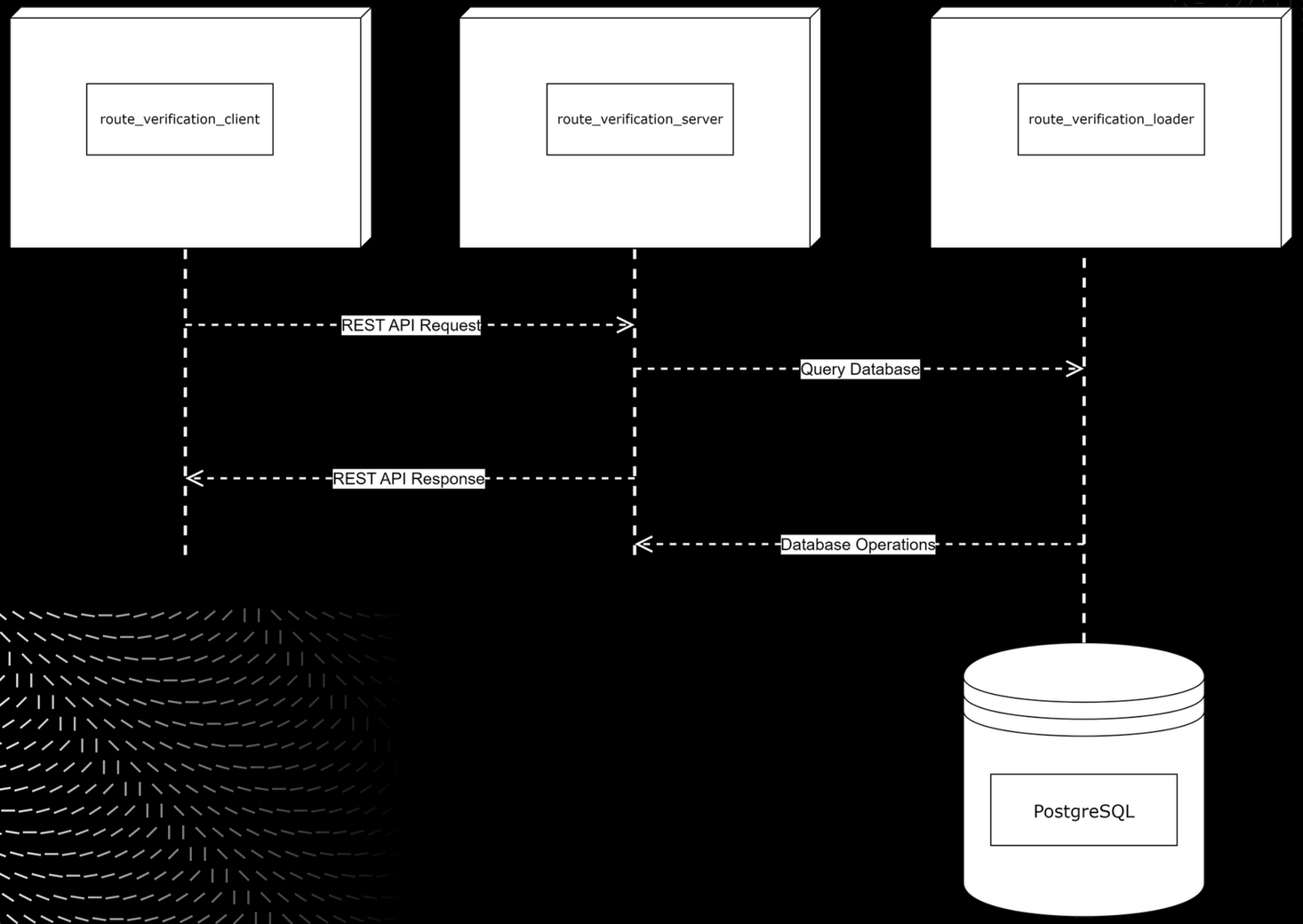


Role of RPSL

Route verification report



Project Sructure



set of Rust scripts route_verification_server_loader to load data into database

route_verification_server to store routing policies and verification reports in its internal database. It will also respond to query requests by querying the database

```rust
//! Launch Postgres and create `irv_server_test` before developing this.
use std::{env::args, fs::File, io::BufReader};

use anyhow::Result;
use encoding_rs::Encoding;
use encoding_rs_io::DecodeReaderBytesBuilder;
use log::{debug, error, warn};
use route_verification::{
    as_rel::{AsRelDb, Relationship},
    bgp::{parse_mrt, Line, QueryIr, Report, ReportItem, Verbosity},
    ir::{AddrPfxRange, AutNum, FilterSet, Ir, PeeringSet, RouteSet, RouteSetMember},
    lex::{expressions, io_wrapper_lines, lines_continued, rpsl_objects, RpslExpr},
};
use sqlx::{
    postgres::{PgPoolOptions, PgQueryResult},
    types::ipnetwork::IpNetwork,
    Pool, Postgres,
};

const ONE_MEBIBYTE: usize = 1024 * 1024;

#[tokio::main]
async fn main() -> Result<()> {
    env_logger::init();
    let pool = PgPoolOptions::new() => Pool<Postgres>, PoolOptions<Postgres>
        .connect("postgres://postgres:postgres@localhost/irv_server_test") <- (url
        .await?;

    let args: Vec<String> = args().collect();
    match arg[1].as_str() {
        "scan" => scan_db(&pool).await?,
        "load" => load_parsed(&pool).await?,
        "asrel" => as_relationship_db(&pool).await?,
        "record" => record_reports(&pool).await?,
        other => error!("Unknown command `{}`", other), => &str
    }

    Ok(())
}

async fn record_reports(pool: &Pool<Postgres>) -> Result<()> {
    let mut n_observed_route = 0; => i32
    debug!("Loading IR.");
    let db = AsRelDb::load_bz("20230701.as-rel.bz2")?; <- (path) => AsRelDb
    let parsed = Ir::pal_read("parsed_all")?; <- (directory) => Ir
    let query = QueryIr::from_ir_and_as_relationship(parsed, &db); <- (ir) => Quer

main.rs                                                                    1:1
```

```rust
    addr_pfx_range: &AddrPfxRange,
) -> sqlx::Result<()> {
    let prefix = &addr_pfx_range.address_prefix; => &IpNet
    let address_prefix: IpNetwork = IpNetwork::new(prefix.addr(), prefix.prefix_len
    ()) <- (ip) => Result<IpNetwork, IpNetworkError>
        .expect("IpNet should be valid IpNetWork"); <- (msg)

    sqlx::query!(
        "INSERT INTO route_set_contains_address_prefix(route_set_name, address_pref
        ix) VALUES ($1, $2)",
        route_set_name,
        address_prefix
    ) => Query<Postgres, PgArguments>
    .execute(pool) <- (executor) => impl Future<Output = Result<…>>
    .await?;

    Ok(())
}

async fn insert_route_set_contains_set(
    pool: &Pool<Postgres>,
    route_set_name: &str,
    contained_set_name: &str,
) -> sqlx::Result<()> {
    sqlx::query!(
        "INSERT INTO route_set_contains_set(route_set_name, contained_set) VALUES (
        $1, $2)",
        route_set_name,
        contained_set_name
    ) => Query<Postgres, PgArguments>
    .execute(pool) <- (executor) => impl Future<Output = Result<…>>
    .await?;
    Ok(())
}

fn find_rpsl_object_fields(body: &str, fields: &[&str]) -> Vec<Vec<String>> {
    let mut matches = vec![vec![]; fields.len()]; => Vec<Vec<String>>
    for RpslExpr { key, expr } in expressions(lines_continued(body.lines())) { <- (
        for (index, field) in fields.iter().enumerate() { => usize, &&str
            if key == *field {
                matches[index].push(expr);
                break;
            }
        }
    }
    matches
}

NORMAL  route_verification_server_loader/src/main.rs  < rust  Bot  818:1  main
```

route_verification_server_loader

Showcase

Detailed Query1

Query RPSL objects by name, including each specific type mentioned above.

---

returns JSON

```
def get_as_for_overall_report_type(overall_report_type: str):
    """ASes that appear in at least one report with the given
    overall_report_type and the number of reports with that overall_report_type.
    """
    if not is_valid_overall_report_type(overall_report_type):
        return (
            jsonify(
                {
                    "input_error": "Invalid overall_report_type",
                    "possible_values": OVERALL_REPORT_TYPES,
                }
            ),
            400,      Steven Hé (Sīchàng), 3 days ago • AS for report items;rename overall_ty
        )
    return execute_all(
        """
WITH filtered_exchange_report AS (
    SELECT * FROM exchange_report WHERE overall_type = %s
) SELECT aggregated_as.as_num, count(*) AS report_count FROM (
    (
        SELECT report_id, from_as AS as_num
        FROM filtered_exchange_report
        ORDER BY as_num
    ) UNION (
        SELECT report_id, to_as AS as_num
        FROM filtered_exchange_report
        ORDER BY as_num
    )
) AS aggregated_as
GROUP BY as_num
ORDER BY as_num
                """,
        overall_report_type,
```

Result

```
❯ http 'http://127.0.0.1:5000/for_overall_report_type/ok'
HTTP/1.1 200 OK
Connection: close
Content-Length: 6040
Content-Type: application/json
Date: Tue, 05 Dec 2023 12:46:56 GMT
Server: Werkzeug/2.3.7 Python/3.11.6

[
    {
        "address_prefix": "0.0.0.0/0",
        "destination_as": 34224,
        "exchange_report_time": "Sat, 02 Dec 2023 03:07:58 GMT",
        "import": false,
        "observed_route_id": 1,
        "observed_route_time": "Sat, 02 Dec 2023 03:07:58 GMT",
        "overall_type": "ok",
        "raw_line": "TABLE_DUMP2|1687212000|B|94.156.252.18|34224|0.0.0.0/0|34224 3356|IGP|94.156.252.18|0|
NAG||\n",
        "report_category": null,
        "report_numeric_content": null,
        "report_specific_case": null,
        "report_string_content": null,
        "source_as": 3356,
        "total_items": 547
    },
    {
        "address_prefix": "0.0.0.0/0",
        "destination_as": 34224,
        "exchange_report_time": "Sat, 02 Dec 2023 03:07:58 GMT",
        "import": true,
```

Detailed Query2

Query RPSL objects from the IRR by name, including each specific type mentioned above.

---

will return JSON

```python
@app.route("/for_report_item_type/<string:report_item_type>", methods=["GET"])
def get_for_report_item_type(report_item_type: str):
    """ASes, routes, reports, and report items for the given
    report_item_type."""
    if not is_valid_report_item_type(report_item_type):
        return (
            jsonify(
                {
                    "input_error": "Invalid report_item_type",
                    "possible_values": REPORT_ITEM_TYPES,
                }
            ),
            400,
        )
    return execute_all(
        # TODO: Paging from user.
        """
SELECT
    e.from_as AS source_as,
    e.to_as AS destination_as,
    e.import,
    e.overall_type,
    e.recorded_time AS exchange_report_time,        Steven Hé (Sīchàng), 2 days ago •
    r.observed_route_id,
    r.raw_line,
    text(r.address_prefix) AS address_prefix,
    r.recorded_time AS observed_route_time,
    ri.category AS report_category,
    ri.specific_case AS report_specific_case,
    ri.str_content AS report_string_content,
    ri.num_content AS report_numeric_content,
    COUNT(*) OVER () AS total_items
FROM
    exchange_report e
JOIN
    observed_route r ON e.parent_observed_route = r.observed_route_id
JOIN
    report_item ri ON e.report_id = ri.parent_report
WHERE
    ri.specific_case = %s
ORDER BY
    exchange_report_time
OFFSET
    0
LIMIT
    10
                """,
        report_item_type,
    )
```

Result

```
❯ http 'http://127.0.0.1:5000/for_report_item_type/err_peering'                                                    [150/560]
HTTP/1.1 200 OK
Connection: close
Content-Length: 7093
Content-Type: application/json
Date: Tue, 05 Dec 2023 12:48:00 GMT
Server: Werkzeug/2.3.7 Python/3.11.6

[
    {
        "address_prefix": "1.0.0.0/24",
        "destination_as": 3257,
        "exchange_report_time": "Sat, 02 Dec 2023 03:07:58 GMT",
        "import": true,
        "observed_route_id": 2,
        "observed_route_time": "Sat, 02 Dec 2023 03:07:58 GMT",
        "overall_type": "unrecorded",
        "raw_line": "TABLE_DUMP2|1687212000|B|89.149.178.10|3257|1.0.0.0/24|3257 13335|IGP|89.149.178.10|0|10|3257:4000
 3257:8794 3257:50001 3257:50110 3257:54900 3257:54901 65000:4134 65001:3320 65002:5511|NAG|13335 162.158.84.50|\n",
        "report_category": "unrecorded",
        "report_numeric_content": null,
        "report_specific_case": "err_peering",
        "report_string_content": null,
        "source_as": 13335,
        "total_items": 522237
    },
    {
        "address_prefix": "1.0.0.0/24",
        "destination_as": 3257,
        "exchange_report_time": "Sat, 02 Dec 2023 03:07:58 GMT",
        "import": true,
        "observed_route_id": 2,
        "observed_route_time": "Sat, 02 Dec 2023 03:07:58 GMT",
```

Detailed Query3

Query RPSL objects from the IRR by name, including each specific type mentioned above.

---

will return JSON type data

```
@app.route("/for_address_prefix/<string:address>/<int:prefix_length>", methods=["GET"])
def get_for_address_prefix(address: str, prefix_length: int):
    """Route, reports, and report items for the given address_prefix."""
    try:
        return execute_all(
            # TODO: Paging from user.
            # FIXME: Remove LIMIT 1.
            """
    SELECT
        e.report_id,
        e.from_as AS source_as,
        e.to_as AS destination_as,
        e.import,
        e.overall_type,
        e.recorded_time AS exchange_report_time,
        ri.report_item_id,
        ri.category AS report_category,
        ri.specific_case AS report_specific_case,
        ri.str_content AS report_string_content,
        ri.num_content AS report_numeric_content,
        COUNT(*) OVER () AS total_items
    FROM
        exchange_report e
    JOIN
        report_item ri ON e.report_id = ri.parent_report
    WHERE
        e.parent_observed_route = (SELECT observed_route_id FROM observed_route WHERE address_prefix = %s LIMIT 1)
    ORDER BY
        e.recorded_time
    OFFSET
        0
    LIMIT
        10
                """,
            f"{address}/{prefix_length}",
        )
    except InvalidTextRepresentation:
        return (
            jsonify(
                {
                    "input_error": "Invalid address or prefix_length",
                    "address": address,
                    "prefix_length": prefix_length,
                }
            ),
            400,
        )
```

Result

```
❯ http 'http://127.0.0.1:5000/for_address_prefix/1.0.0.0/24'                    [120/712]
HTTP/1.1 200 OK
Connection: close
Content-Length: 3992
Content-Type: application/json
Date: Tue, 05 Dec 2023 12:48:24 GMT
Server: Werkzeug/2.3.7 Python/3.11.6

[
    {
        "destination_as": 3257,
        "exchange_report_time": "Sat, 02 Dec 2023 03:07:58 GMT",
        "import": false,
        "overall_type": "unrecorded",
        "report_category": "unrecorded",
        "report_id": 3,
        "report_item_id": 1,
        "report_numeric_content": null,
        "report_specific_case": "unrec_export_empty",
        "report_string_content": null,
        "source_as": 13335,
        "total_items": 5845
    },
    {
        "destination_as": 3257,
        "exchange_report_time": "Sat, 02 Dec 2023 03:07:58 GMT",
        "import": true,
        "overall_type": "unrecorded",
        "report_category": "unrecorded",
        "report_id": 4,
        "report_item_id": 2,
        "report_numeric_content": 12,
        "report_specific_case": "err_remote_as_num",
```

Detailed Query4

```
@app.route("/aut_num/<int:as_num>", methods=["GET"])
def get_aut_num(as_num: int):
    return execute_one(
        """
SELECT as_num, as_name, imports, exports FROM aut_num WHERE as_num = %s
""",
        as_num,
    )
```

Query RPSL objects from the IRR by name, including each specific type mentioned above.

---

will return JSON type data

Result

```
❯ http 'http://127.0.0.1:5000/aut_num/293'          internet_route_verification_server/flask_RestAPI  main 293ms      [3/745]
HTTP/1.1 200 OK
Connection: close
Content-Length: 470
Content-Type: application/json
Date: Tue, 05 Dec 2023 12:49:07 GMT
Server: Werkzeug/2.3.7 Python/3.11.6

{
    "as_name": "ESNET",
    "as_num": 293,
    "exports": {
        "ipv4": {
            "unicast": [
                {
                    "mp_filter": {
                        "AsSet": [
                            "AS293:AS-ESNET",
                            "NoOp"
                        ]
                    },
                    "mp_peerings": [
                        {
                            "mp_peering": {
                                "remote_as": {
                                    "Single": "Any"
                                }
                            }
                        }
                    ]
                }
            ]
        }
    },
    "imports": {}
```

Detailed Query5

Query RPSL objects from the IRR by name, including each specific type mentioned above.

---

will return JSON type data

```python
@app.route("/as_for_overall_report_type/<string:overall_report_type>", methods=["GET"])
def get_as_for_overall_report_type(overall_report_type: str):
    """ASes that appear in at least one report with the given
    overall_report_type and the number of reports with that overall_report_type.
    """
    if not is_valid_overall_report_type(overall_report_type):
        return (
            jsonify(
                {
                    "input_error": "Invalid overall_report_type",
                    "possible_values": OVERALL_REPORT_TYPES,
                }
            ),
            400,
        )
    return execute_all(
        """
WITH filtered_exchange_report AS (
    SELECT * FROM exchange_report WHERE overall_type = %s
) SELECT aggregated_as.as_num, count(*) AS report_count FROM (
    (
        SELECT report_id, from_as AS as_num
        FROM filtered_exchange_report
        ORDER BY as_num
    ) UNION (
        SELECT report_id, to_as AS as_num
        FROM filtered_exchange_report
        ORDER BY as_num
    )
) AS aggregated_as
GROUP BY as_num
ORDER BY as_num
                """,
        overall_report_type,
    )
```

Result

```
❯ http 'http://127.0.0.1:5000/as_for_overall_report_type/bad'                    [143/1103]
HTTP/1.1 200 OK
Connection: close
Content-Length: 2125
Content-Type: application/json
Date: Tue, 05 Dec 2023 12:50:59 GMT
Server: Werkzeug/2.3.7 Python/3.11.6

[
    {
        "as_num": 174,
        "report_count": 2
    },
    {
        "as_num": 209,
        "report_count": 8
    },
    {
        "as_num": 1299,
        "report_count": 16
    },
    {
        "as_num": 1403,
        "report_count": 6
    },
    {
        "as_num": 2152,
        "report_count": 1
    },
    {
        "as_num": 2497,
        "report_count": 16
    },
    {
```

Detailed Query6

Query RPSL objects from the IRR by name, including each specific type mentioned above.

---

will return JSON type data

```
@app.route("/as_for_report_item_type/<string:report_item_type>", methods=["GET"])
def get_as_for_report_item_type(report_item_type: str):
    """ASes that appear in at least one report item with the given
    report_item_type and the number of report items with that report_item_type.
    """
    if not is_valid_report_item_type(report_item_type):
        return (
            jsonify(
                {
                    "input_error": "Invalid report_item_type",
                    "possible_values": REPORT_ITEM_TYPES,
                }
            ),
            400,
        )
    return execute_all(
        """
WITH filtered_report_item AS (
    SELECT *
    FROM report_item JOIN exchange_report ON parent_report = report_id
    WHERE specific_case = %s
) SELECT aggregated_as.as_num, count(*) AS report_item_count FROM (
    (
        SELECT report_item_id, from_as AS as_num
        FROM filtered_report_item
        ORDER BY as_num
    ) UNION (
        SELECT report_item_id, to_as AS as_num
        FROM filtered_report_item
        ORDER BY as_num
    )
) AS aggregated_as
GROUP BY as_num
ORDER BY as_num
                """,
        report_item_type,
    )
```

Result

```
❯ http 'http://127.0.0.1:5000/as_for_report_item_type/unrec_as_set'                [15/1151]
HTTP/1.1 200 OK
Connection: close
Content-Length: 515
Content-Type: application/json
Date: Tue, 05 Dec 2023 12:51:34 GMT
Server: Werkzeug/2.3.7 Python/3.11.6

[
    {
        "as_num": 1299,
        "report_item_count": 38
    },
    {
        "as_num": 1403,
        "report_item_count": 6
    },
    {
        "as_num": 3257,
        "report_item_count": 6
    },
    {
        "as_num": 3356,
        "report_item_count": 12
    },
    {
        "as_num": 6461,
        "report_item_count": 174
    },
    {
        "as_num": 7018,
        "report_item_count": 4
    },
    {
```

Detailed Query7

Query RPSL objects from the IRR by name, including each specific type mentioned above.

---

will return JSON type data

```
@app.route("/report_for_as/<int:as_num>", methods=["GET"])
def get_report_for_as(as_num):
    return execute_all(
        """
SELECT * FROM (
    SELECT report_id
    FROM exchange_report
    JOIN autonomous_system ON exchange_report.from_as = autonomous_system.as_num
    WHERE as_num = %s
    union
    SELECT report_id
    FROM exchange_report
    JOIN autonomous_system ON exchange_report.to_as = autonomous_system.as_num
    WHERE as_num = %s
) AS as_and_report
natural JOIN exchange_report
ORDER BY report_id
                    """,
        as_num,
        as_num,
    )
```

Result

```
ttp 'http://127.0.0.1:5000/report_for_as/293'                    internet_route_verification_server/flask_RestAPI  main 1 1[125/5803
P/1.1 200 OK
nection: close
tent-Length: 3780
tent-Type: application/json
e: Wed, 06 Dec 2023 05:08:17 GMT
ver: Werkzeug/2.3.7 Python/3.11.6

{
    "from_as": 13335,
    "import": false,
    "overall_type": "unrecorded",
    "parent_observed_route": 16,
    "recorded_time": "Sat, 02 Dec 2023 03:08:06 GMT",
    "report_id": 39,
    "to_as": 293
},
{
    "from_as": 13335,
    "import": true,
    "overall_type": "unrecorded",
    "parent_observed_route": 16,
    "recorded_time": "Sat, 02 Dec 2023 03:08:06 GMT",
    "report_id": 40,
    "to_as": 293
},
{
    "from_as": 6939,
    "import": false,
    "overall_type": "ok",
    "parent_observed_route": 50,
    "recorded_time": "Sat, 02 Dec 2023 03:08:26 GMT",
    "report_id": 239,
    "to_as": 293
},
{
    "from_as": 6939,
    "import": true,
    "overall_type": "unrecorded",
    "parent_observed_route": 50,
    "recorded_time": "Sat, 02 Dec 2023 03:08:26 GMT",
    "report_id": 240,
    "to_as": 293
},
{
    "from_as": 6939,
    "import": false,
    "overall_type": "ok",
    "parent_observed_route": 83,
```

Detailed Query8

Query RPSL objects from the IRR by name, including each specific type mentioned above.

---

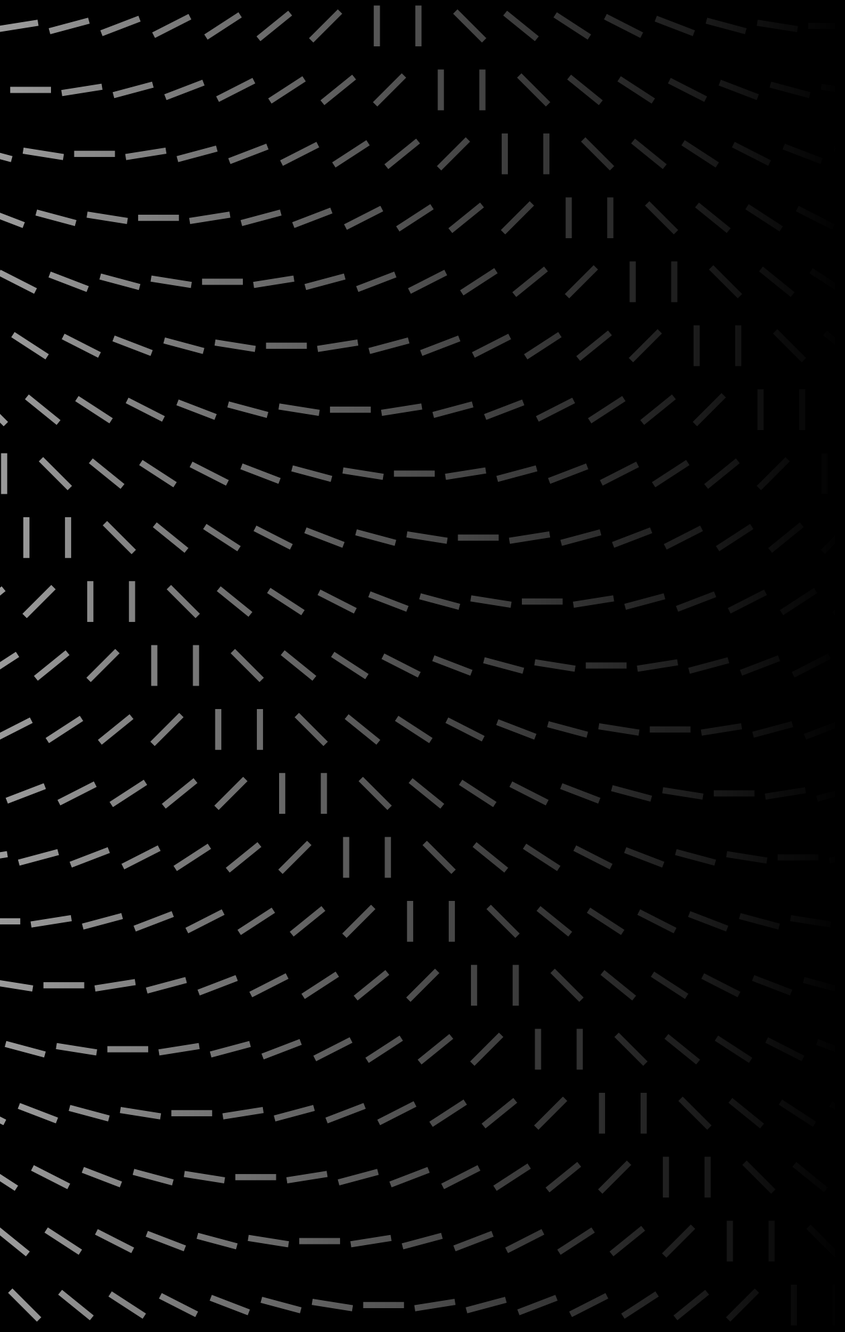
will return JSON type data

```
@app.route("/route_for_as/<int:as_num>", methods=["GET"])
def get_route_for_as(as_num):
    return execute_all(
        """
select
    e.from_as as as_num,
    o.observed_route_id,
    o.raw_line,
    text(o.address_prefix) as address_prefix,
    o.recorded_time
from
    exchange_report as e
join
    observed_route as o
on e.parent_observed_route = o.observed_route_id
where e.from_as = %s
union
select
    e.to_as as as_num,
    o.observed_route_id,
    o.raw_line,
    text(o.address_prefix) as address_prefix,
    o.recorded_time
from
    exchange_report as e
join
    observed_route as o
on e.parent_observed_route = o.observed_route_id
where e.to_as = %s
                """,
        as_num,
        as_num,
    )
```

Result

```
❯ http 'http://127.0.0.1:5000/route_for_as/293'                                internet_route_verification_server/flask_RestAPI  main 2[26/6674]
HTTP/1.1 200 OK
Connection: close
Content-Length: 2539
Content-Type: application/json
Date: Wed, 06 Dec 2023 05:09:28 GMT
Server: Werkzeug/2.3.7 Python/3.11.6

[
    {
        "address_prefix": "1.0.128.0/18",
        "as_num": 293,
        "observed_route_id": 220,
        "raw_line": "TABLE_DUMP2|1687212000|B|198.129.33.85|293|1.0.128.0/18|293 6453 38040 23969|IGP|198.129.33.85|0|710||NAG||\n",
        "recorded_time": "Sat, 02 Dec 2023 03:10:02 GMT"
    },
    {
        "address_prefix": "1.0.128.0/17",
        "as_num": 293,
        "observed_route_id": 191,
        "raw_line": "TABLE_DUMP2|1687212000|B|198.129.33.85|293|1.0.128.0/17|293 6453 38040 23969|IGP|198.129.33.85|0|710||NAG||\n",
        "recorded_time": "Sat, 02 Dec 2023 03:09:56 GMT"
    },
    {
        "address_prefix": "1.0.5.0/24",
        "as_num": 293,
        "observed_route_id": 83,
        "raw_line": "TABLE_DUMP2|1687212000|B|198.129.33.85|293|1.0.5.0/24|293 6939 7545 2764 38803|IGP|198.129.33.85|0|710||NAG||\n",
        "recorded_time": "Sat, 02 Dec 2023 03:09:01 GMT"
    },
    {
        "address_prefix": "1.0.0.0/24",
        "as_num": 293,
        "observed_route_id": 16,
        "raw_line": "TABLE_DUMP2|1687212000|B|198.129.33.85|293|1.0.0.0/24|293 13335|IGP|198.129.33.85|0|710||NAG|13335 162.158.164.1|\n",
        "recorded_time": "Sat, 02 Dec 2023 03:08:06 GMT"
    },
    {
        "address_prefix": "1.0.4.0/22",
        "as_num": 293,
        "observed_route_id": 50,
        "raw_line": "TABLE_DUMP2|1687212000|B|198.129.33.85|293|1.0.4.0/22|293 6939 7545 2764 38803|IGP|198.129.33.85|0|710||NAG||\n",
        "recorded_time": "Sat, 02 Dec 2023 03:08:25 GMT"
    },
    {
        "address_prefix": "1.0.64.0/18",
        "as_num": 293,
        "observed_route_id": 163,
        "raw_line": "TABLE_DUMP2|1687212000|B|198.129.33.85|293|1.0.64.0/18|293 2497 7670 18144|IGP|198.129.33.85|0|710||NAG|18144 219.118.225.18|\n",
        "recorded_time": "Sat, 02 Dec 2023 03:09:49 GMT"
```



Summary

Loader, PostgreSQL, REST API server

RPSL, observed route, and verification report storage

Queries for Autonomous Systems and routes

Network research and operation