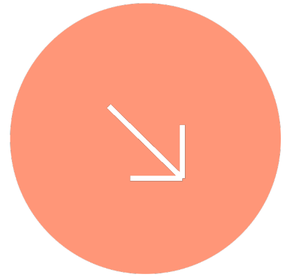
VSS <> SPREAD



We are SPREAD

Facts

Facts

Start of Operations: 01.11.2019

Headquarters: Berlin

Industry:

Automotive | Machinery | Aerospace & Defence

Management

Philipp Noll

Co-Founder / Managing Director

Robert Göbel

Co-Founder / Managing Director

SAB Board

Harald Krüger
Form. CEO of BMW Group,
Telekom & Lufthansa

Dr. Till Reuter
Former CEO of KUKA Group,
Verwaltungspräsident Müller

Sebastian Borek
Founder & CEO of Founders
Foundaton (Bertelsmann)

Charles Songhurst
Former Strategy Director
Bill Gates at Microsoft

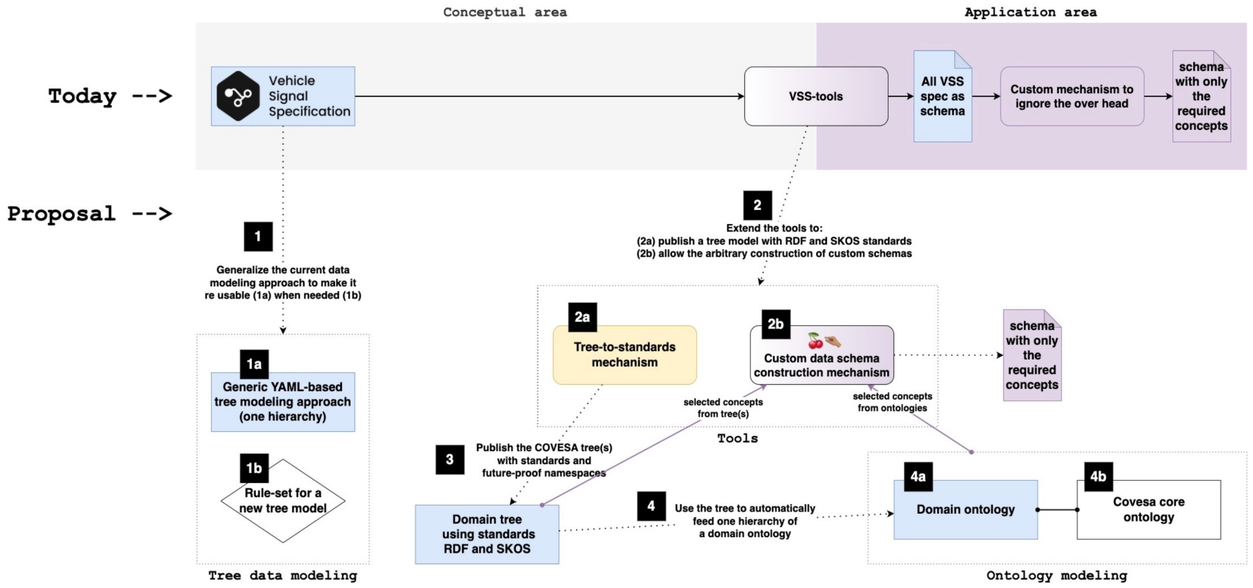
Joachim Schreiner
Exec. VP Sales & Country Leader
Salesforce Germany & Austria

Markus Ehrle
Senior VP Enterprise Sales
Salesforce Deutschland

Customers

Motivation

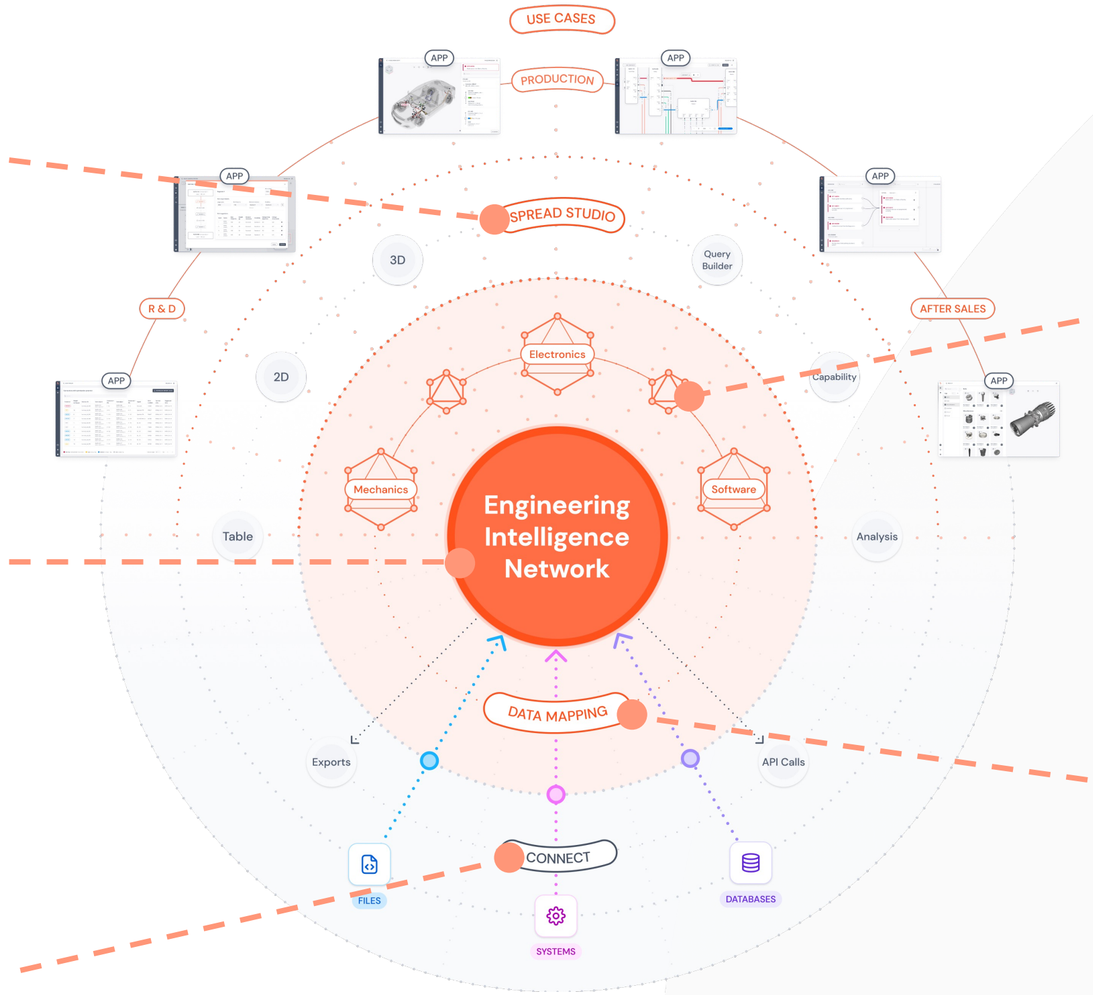
Process proposal



Domain interoperability

Our Product

SPREAD Studio is a **low code platform** with specialized widgets to create web-based applications within hours



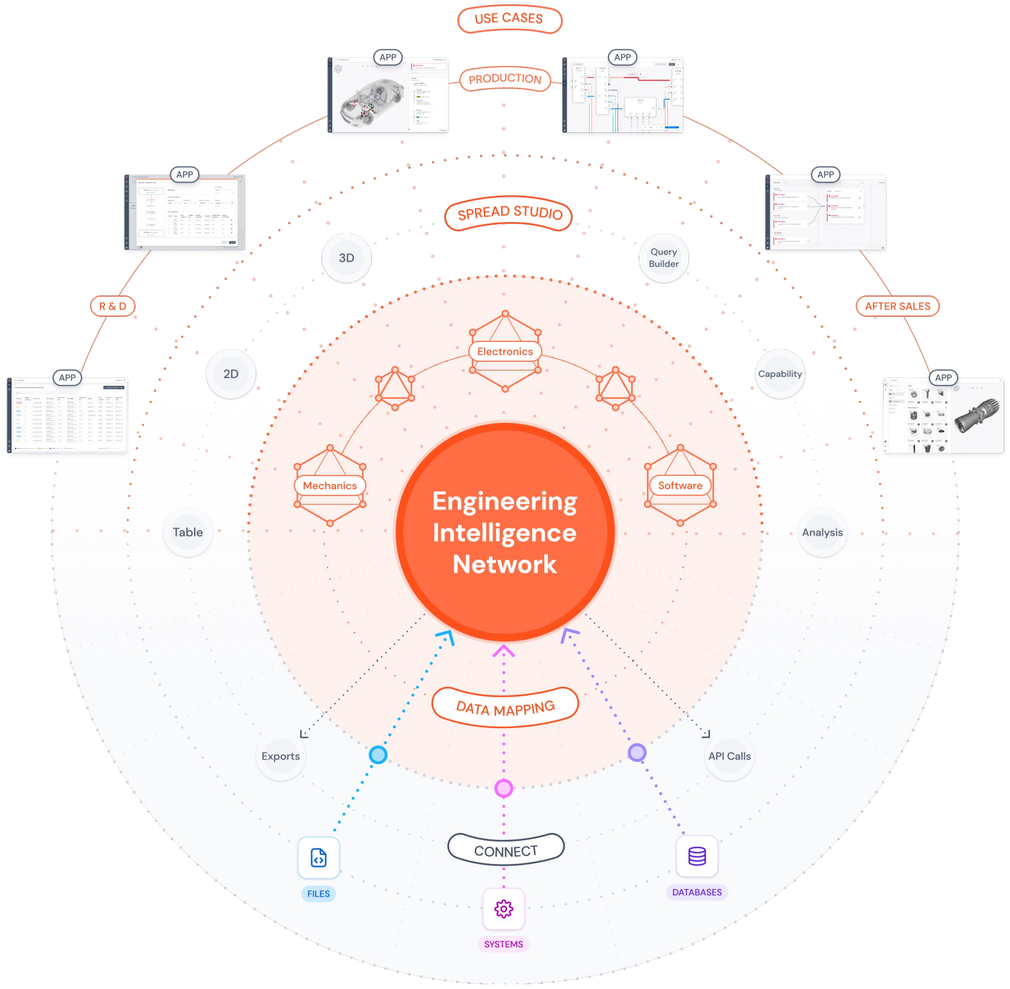
Subgraphs serve Use Case specific needs and increase **application performance** (e.g. Electric-to-Software-to-Function)

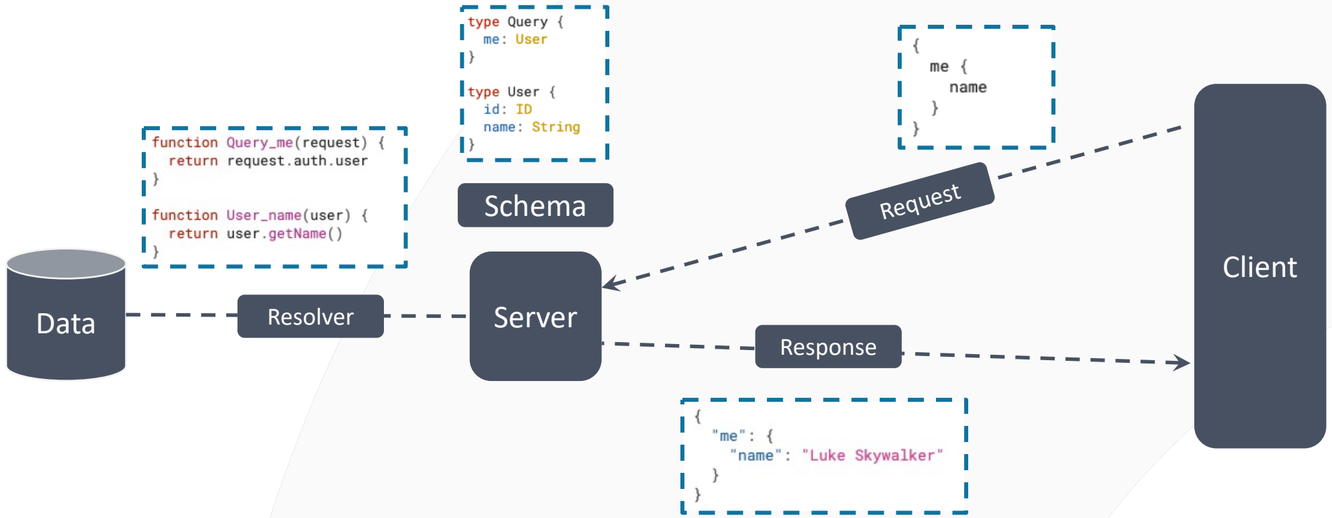
SPREAD's Engineering Intelligence Network manages **interdependencies** of **mechatronic systems.**

Generic mapper link data to SPREAD's **open and standardized** product information model

Connect source systems via **RestAPI**, authenticated **GraphQL** API, **Kafka** stream and more

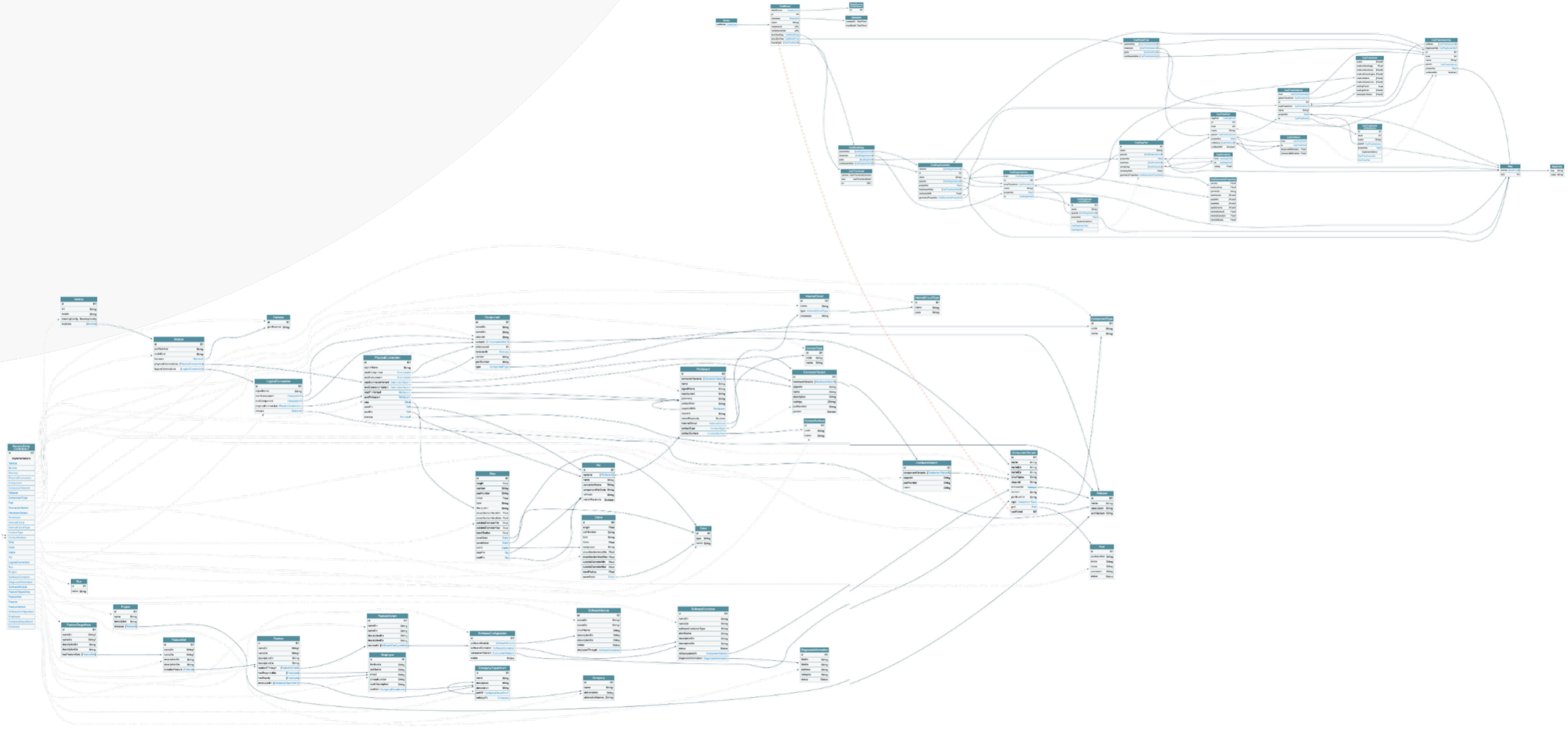
Supergraph based on GraphQL





In GraphQL, a client specifies the structure of the data it needs by defining a query, which can include multiple fields and nested objects. The server then responds with the exact data requested by the client, in the same shape as the query.

GraphQL Schema as contract



GraphQL Schema as contract

As contract...

A description of an electrical component and it's metadata

IMPLEMENTS

GenericEntity
Common fields of any entity

FIELDS

id: ID!
ID of the entity given by the interface type

nameEn: String
The name of the component in the given language @todo: filter instead of several fields.

nameDe: String
The name of the component in the given language @todo: filter instead of several fields.

objectId: String
ID for a specific version of a component. Changes, when related data changes

variants: [ComponentVariant]
Connected variants of the specific component

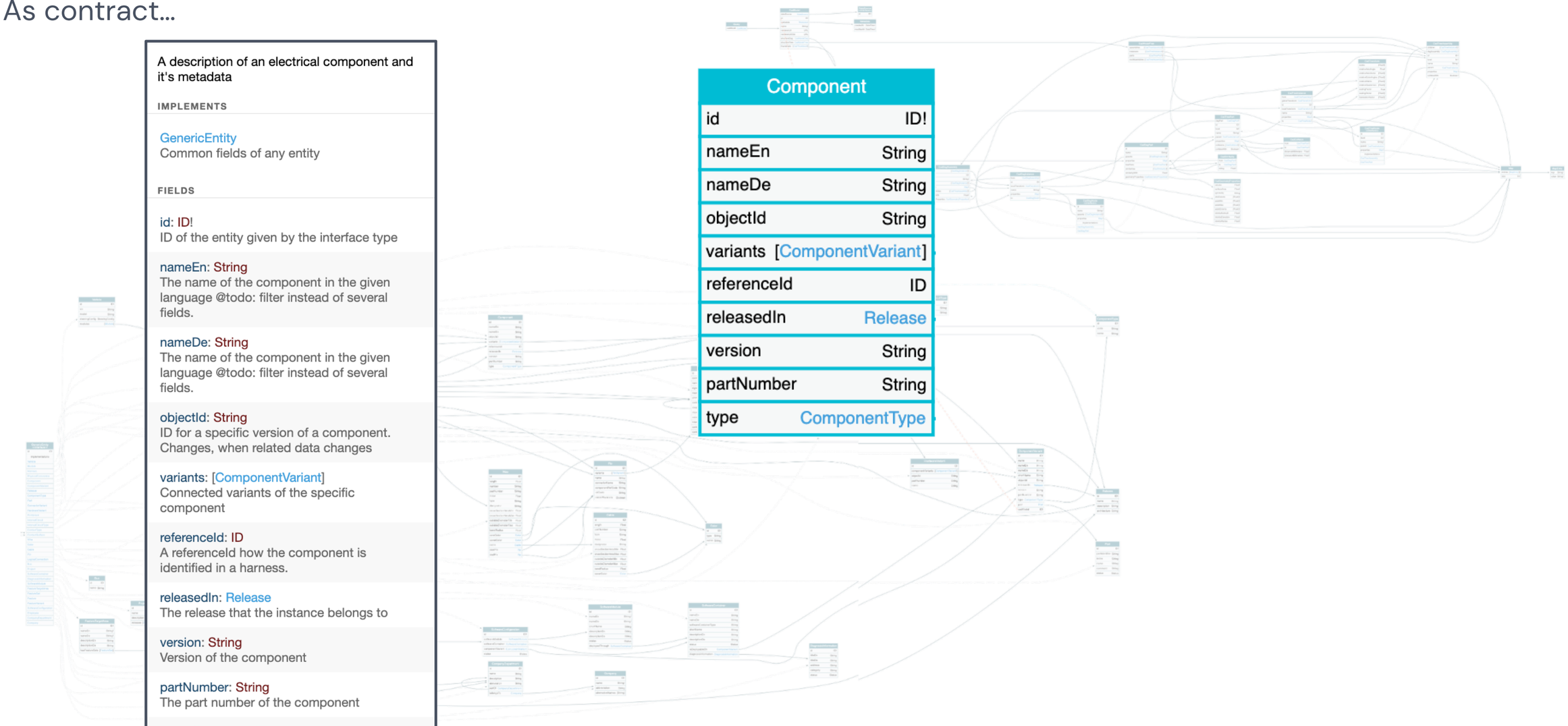
referenceId: ID
A referenceId how the component is identified in a harness.

releasedIn: Release
The release that the instance belongs to

version: String
Version of the component

partNumber: String
The part number of the component

type: ComponentType
The type of the component



GraphQL Schema as contract

As contract...

A description of an electrical component and it's metadata

IMPLEMENTS

GenericEntity
Common fields of any entity

FIELDS

id: ID!
ID of the entity given by the interface type

nameEn: String
The name of the component in the given language @todo: filter instead of several fields.

nameDe: String
The name of the component in the given language @todo: filter instead of several fields.

objectId: String
ID for a specific version of a component. Changes, when related data changes

variants: [ComponentVariant]
Connected variants of the specific component

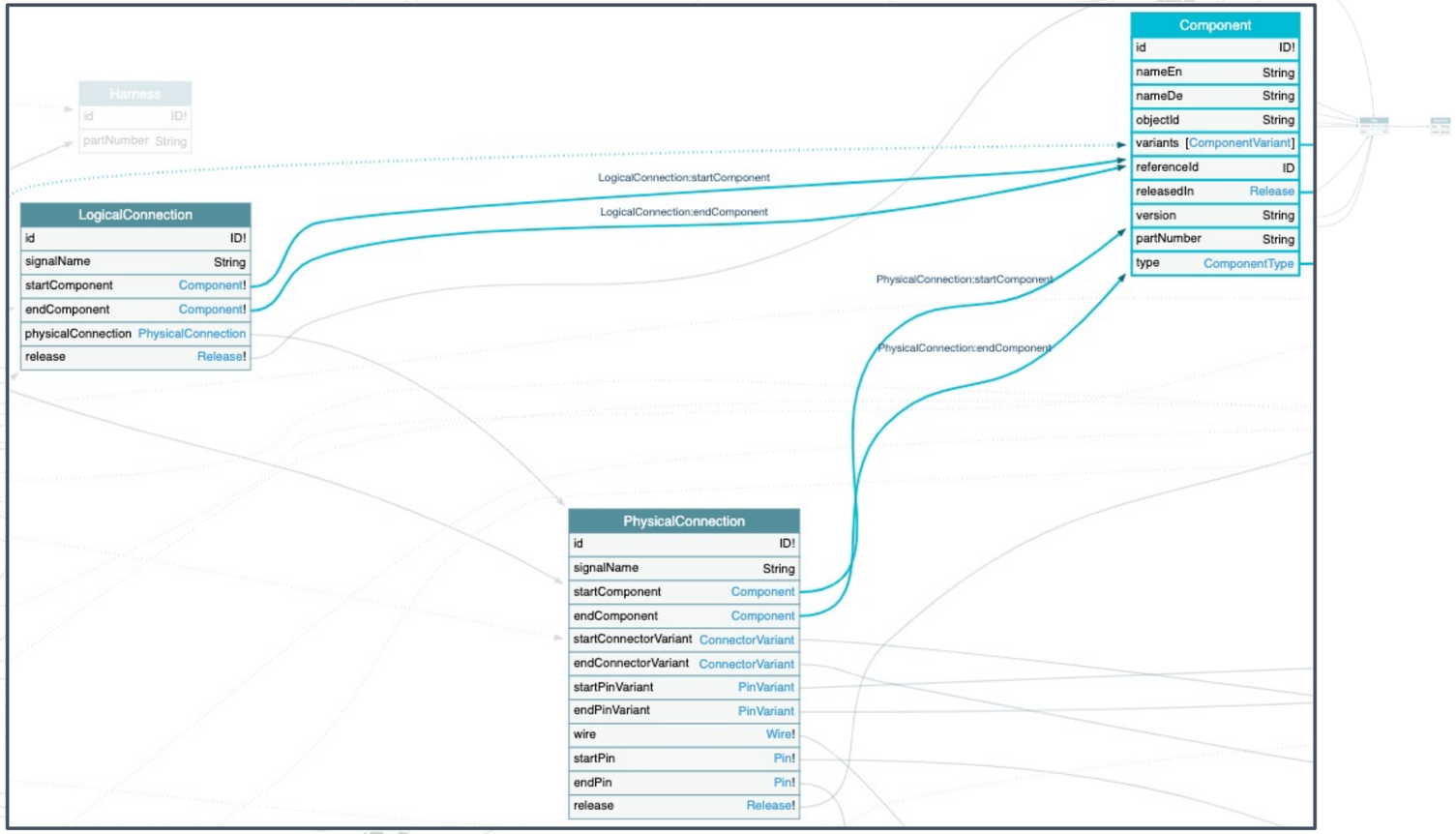
referenceId: ID
A referenceId how the component is identified in a harness.

releasedIn: Release
The release that the instance belongs to

version: String
Version of the component

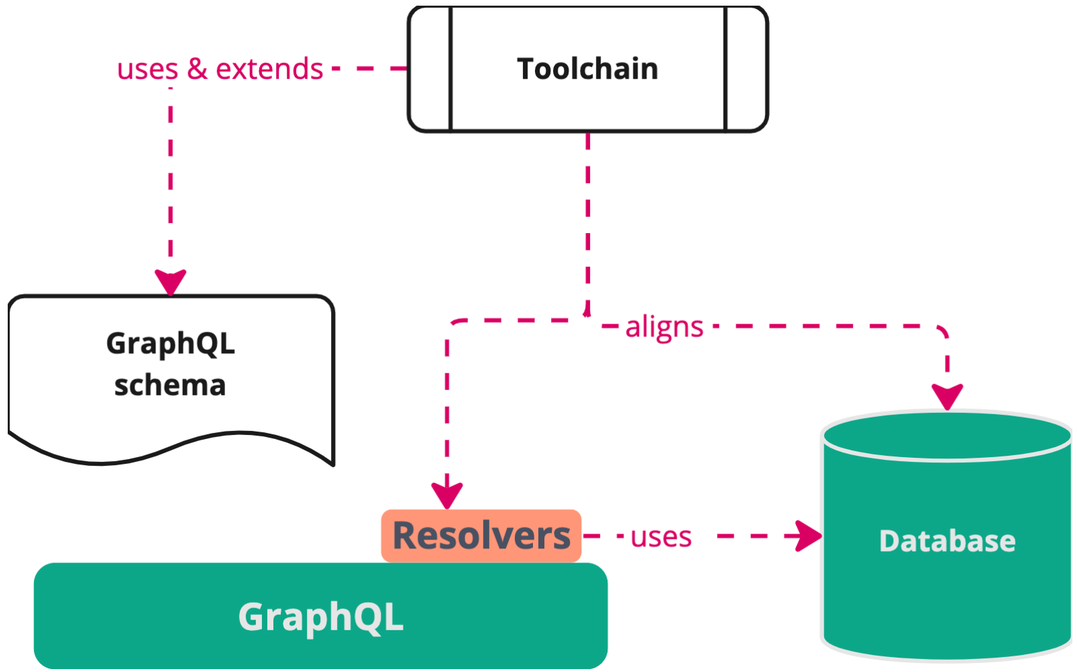
partNumber: String
The part number of the component

type: ComponentType
The type of the component



GraphQL Schema as contract

As contract between graph database ...



Start from GraphQL Schema

In our GraphQL Schema we define the entities, their attributes and how to connect them. The schema is managed, described and customer independent.

Extension

GraphQL uses `directives` for extensions. We use those to tell our toolchain the details about how to generate the graph database, generic queries/mutations and resolvers.

Generation of DB

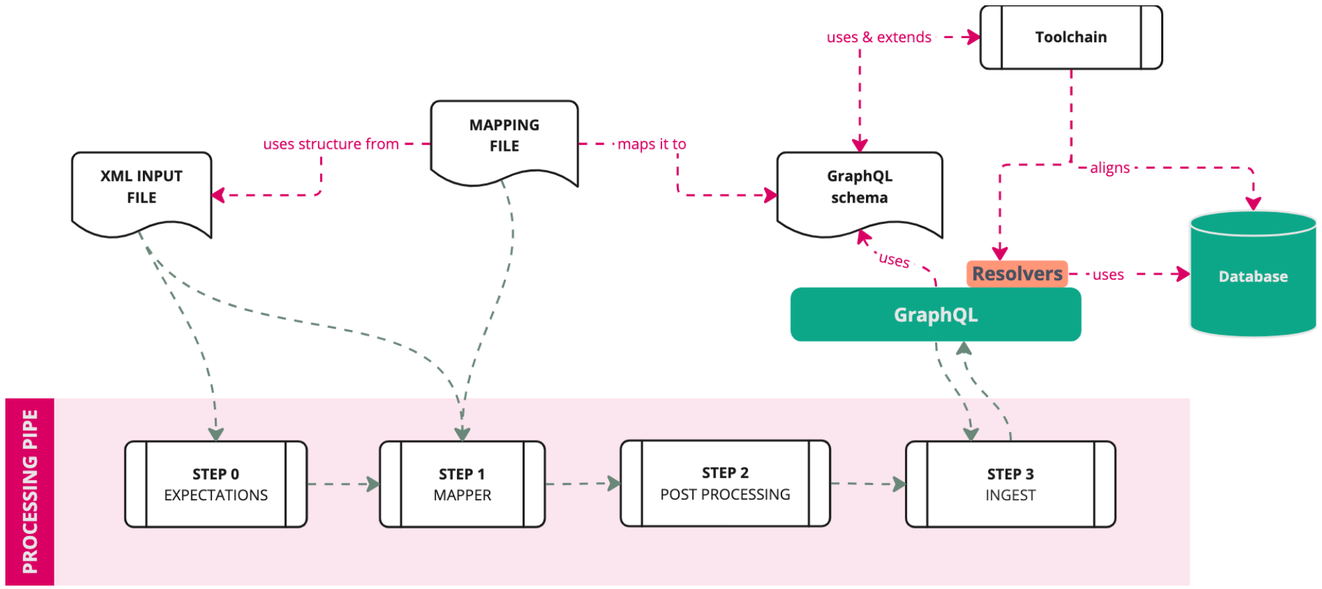
Graph DB is created based on the instructions of the schema.

Generation of resolvers

Where possible, resolvers are generated. Customization for special business queries are possible.

GraphQL Schema as contract

As contract between graph database, data ingest ...



Mapping File

Links the structure of the input file to the desired format for the GraphQL mutations. Defines the entities and their relations in a RML like way.

Step 0 & 1

Checks the input file, for expected fields and values. Uses the the mapping file for the creation of entities based on the input.

Step 2

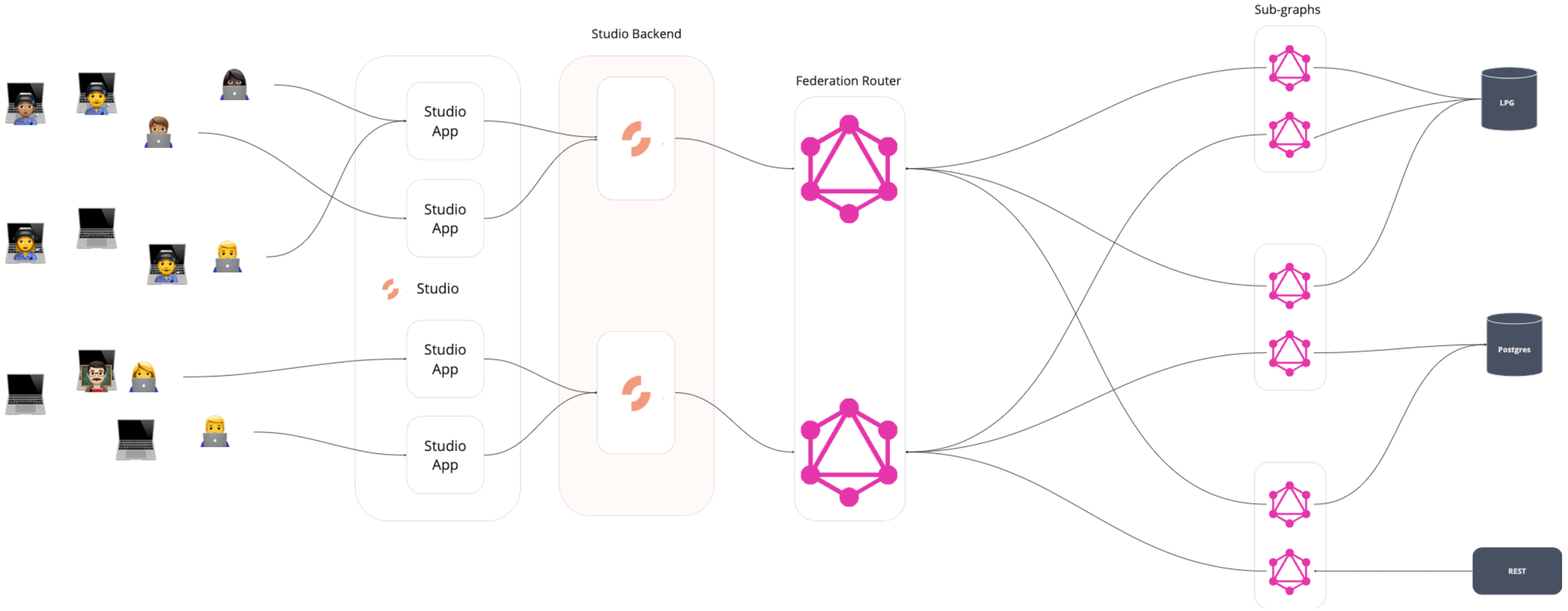
Does the postprocessing and optimization (e.g. local uniqueness of the identifier)

Step 3

Ingests the data over mutations.

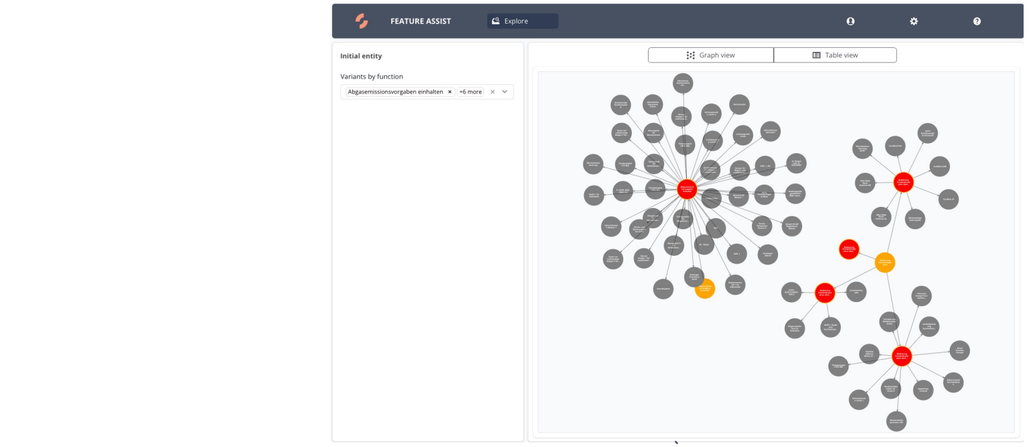
GraphQL Schema as contract

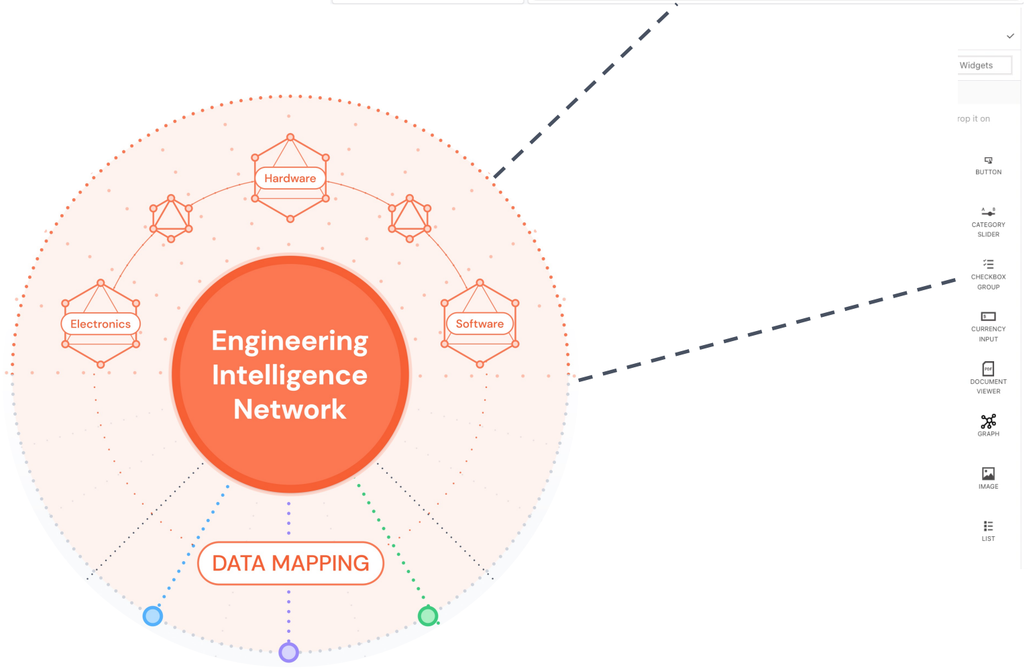
As contract between graph database, data ingest, other (customer) data sources ...

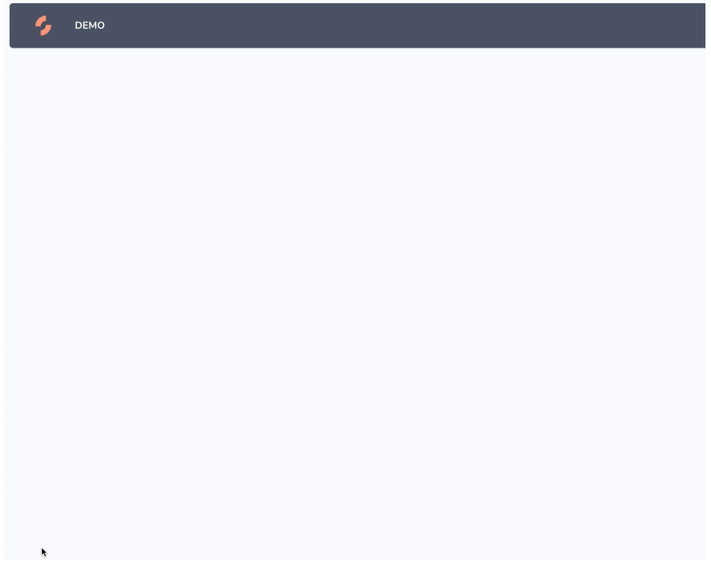


GraphQL Schema as contract

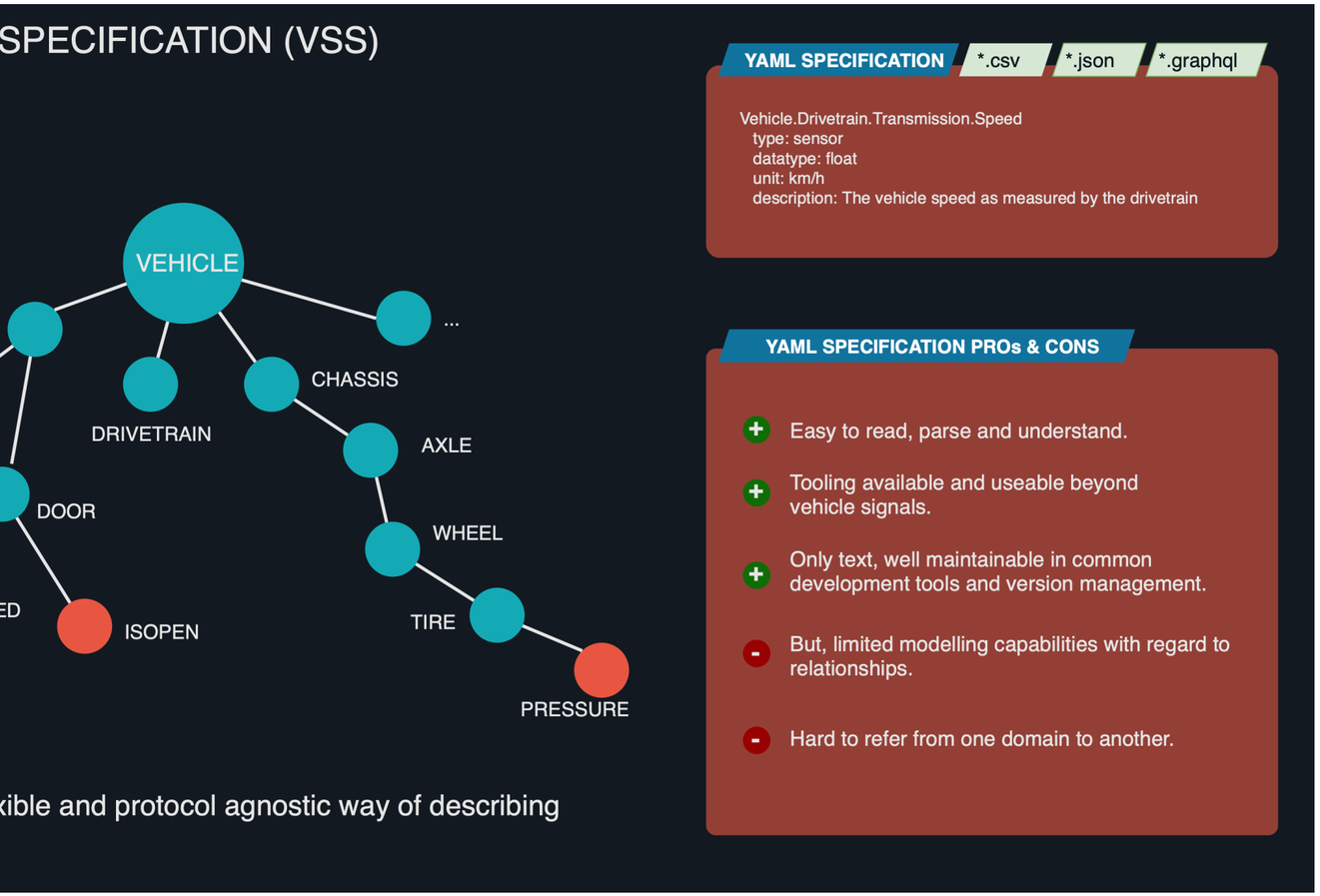
As contract between graph database, data ingest, other (customer) data sources and applications (low code)







Limits and approaches



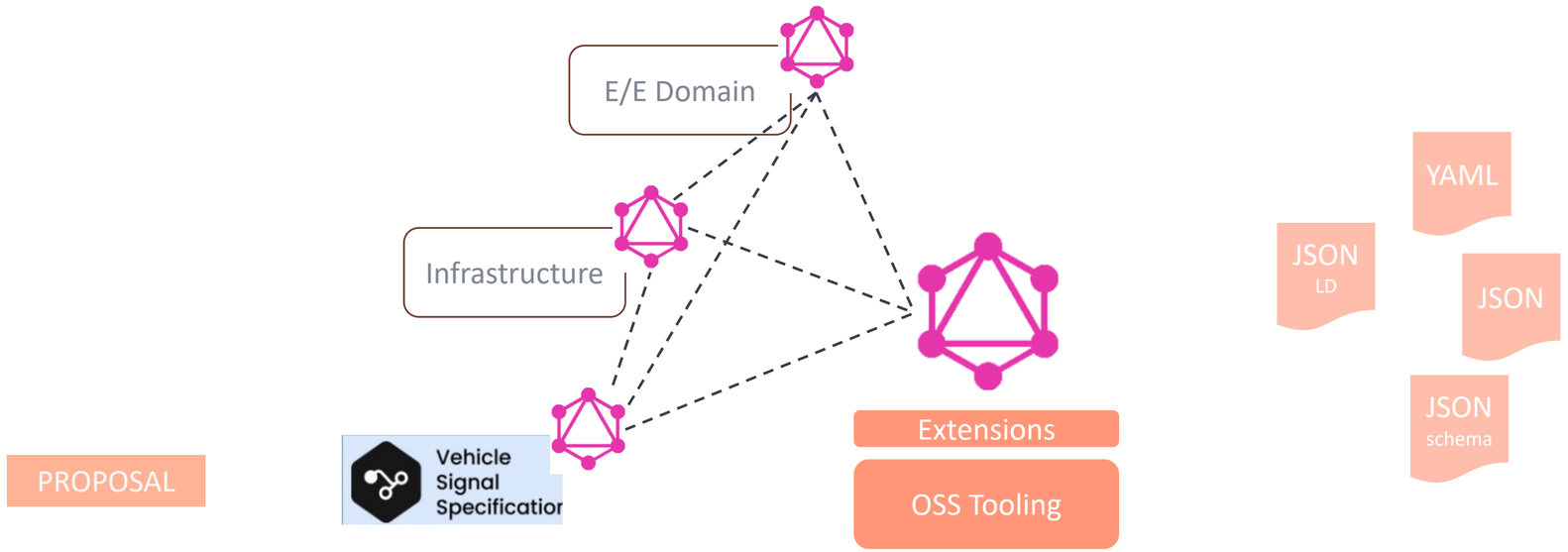
Besides tool maintenance,

- Limited modeling capabilities with regards to relationships (only parent <> child)
- Multidomain reference almost impossible

- Steep learning curve
- Deep expertise required
- Little traction in standardization

- Similar syntax
- Real graphs
- Multi-domain
- A lot of tooling available
- Query definition part of the schema language

Proposal


E/E Domain
Infrastructure
Vehicle Signal Specificatior
Extensions
OSS Tooling
PROPOSAL
YAML
JSON
LD
JSON
JSON
schema


Today -->
Conceptual area
Application area
Vehicle Signal Specification
VSS-tools
All VSS spec as schema
Custom mechanism to ignore the over head
schema with only the required concepts

We are looking forward to hearing from you

Your team from SPREAD

Philipp Noll

Co-Founder & Managing Director

philipp@spread.ai

Marius Booms

Account Executive

marius@spread.ai

+49 151 650 44441

Prinzessinnenstraße 8-14
10969 Berlin

spread.ai

info@spread.ai