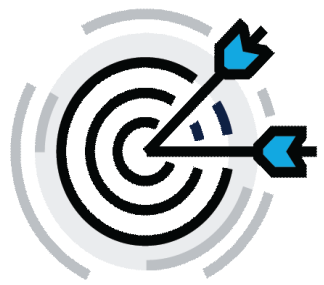
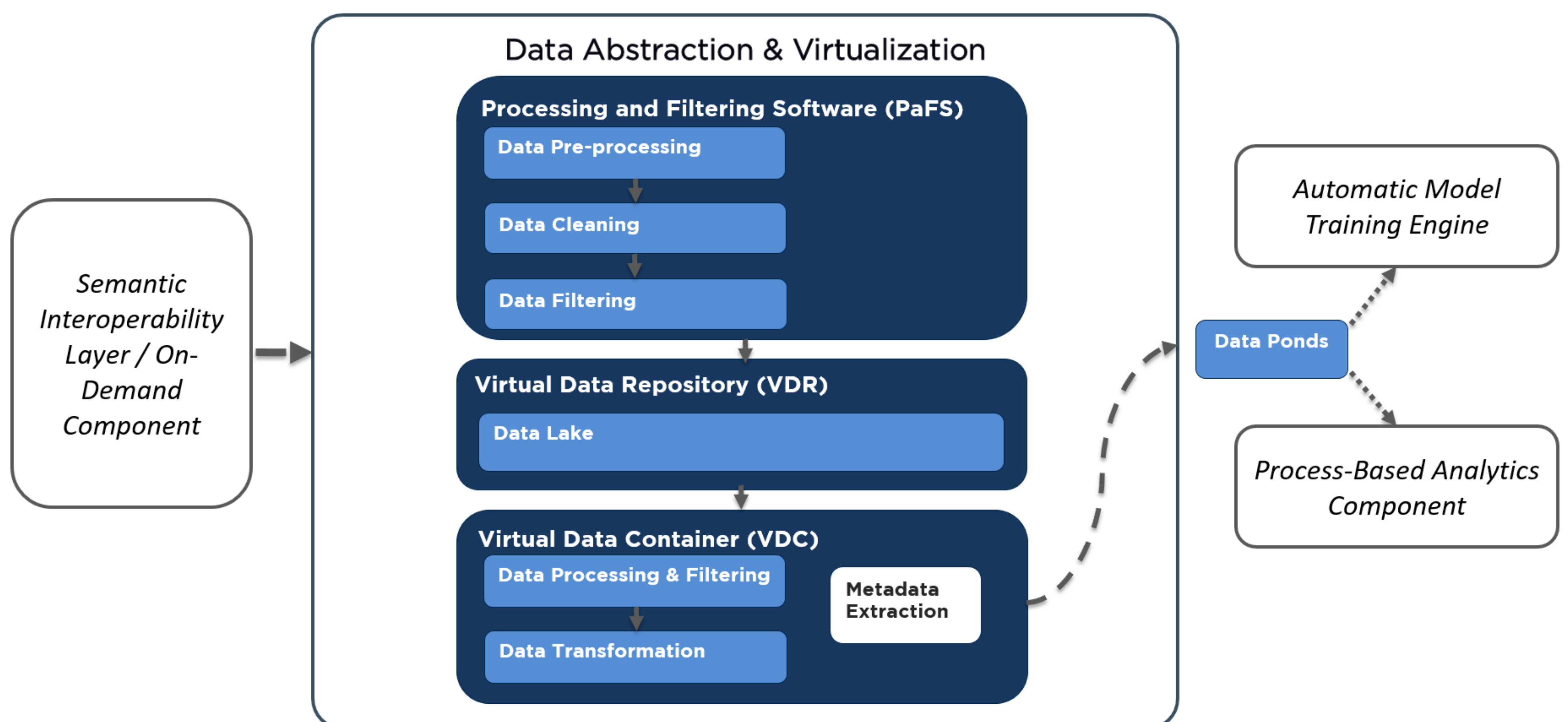


Overview



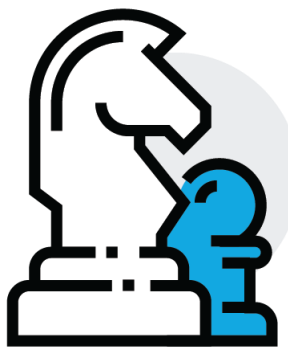
Data Abstraction and Virtualization component is responsible for correctly preparing data input from different sources inside the generic DataPorts architecture, maintaining metadata from all feeds, and finally making available the cleaned and processed datasets to any eventual client.

Component at a glance



Goals of the component

- **Data Processing Mechanisms:** Process, clean and filter the incoming data in order to enable developers building cognitive data-driven applications on top of the DataPorts Platform
- **Data Management:** Store and deliver historical Data as a Service, putting emphasis on QoD and QoS
- **Metadata Extraction:** Calculate and provide useful metadata for all the available datasets
- **Data Transformation:** Transforms the data into the requested format, such as JSON, Parquet or CSV
- **Data Services and Analytics Integration:** The component will be integrated with the Semantic Interoperability layer, the Automatic Model Training Engine and the Process-Based Analytics component of the DataPorts Platform



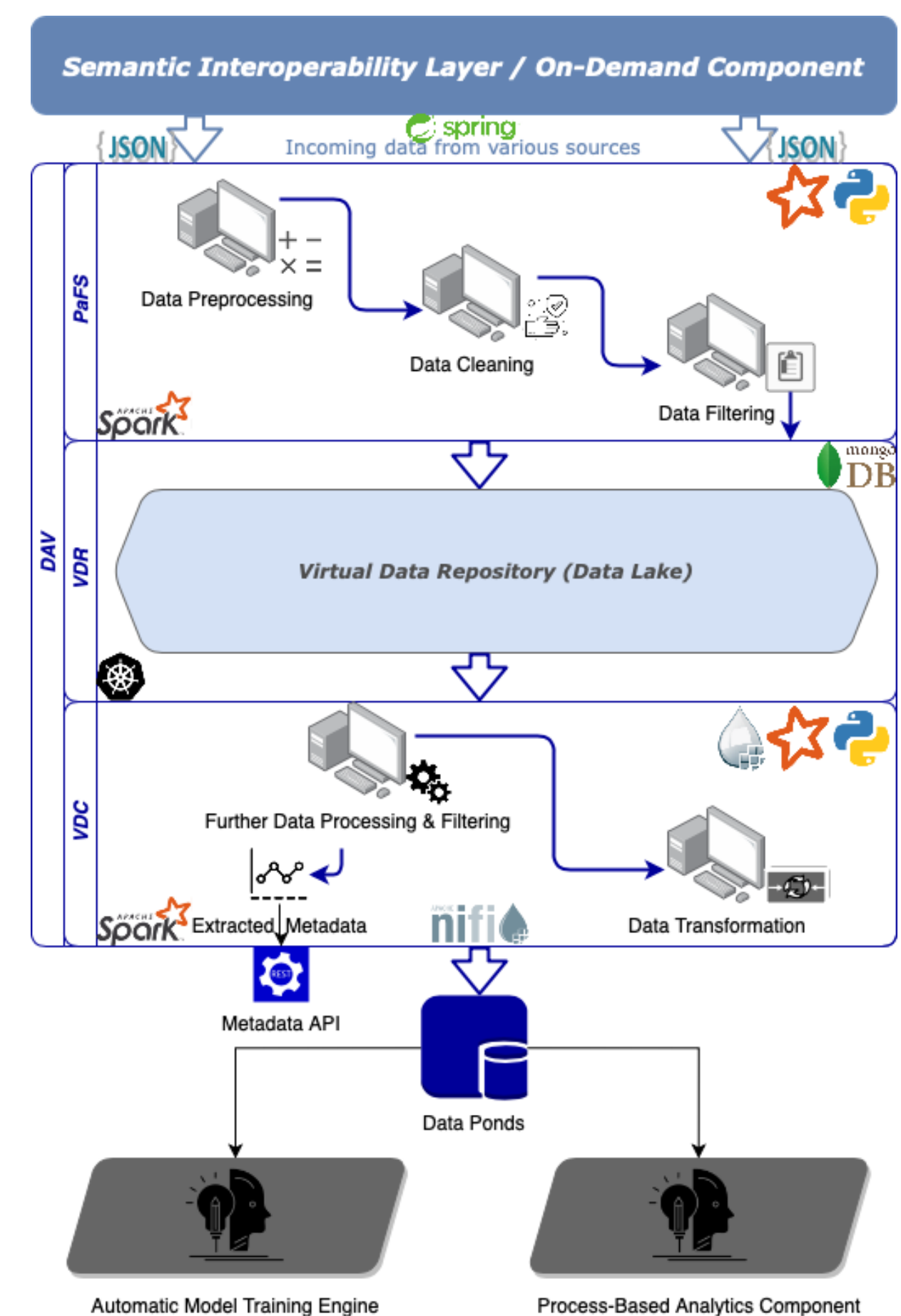
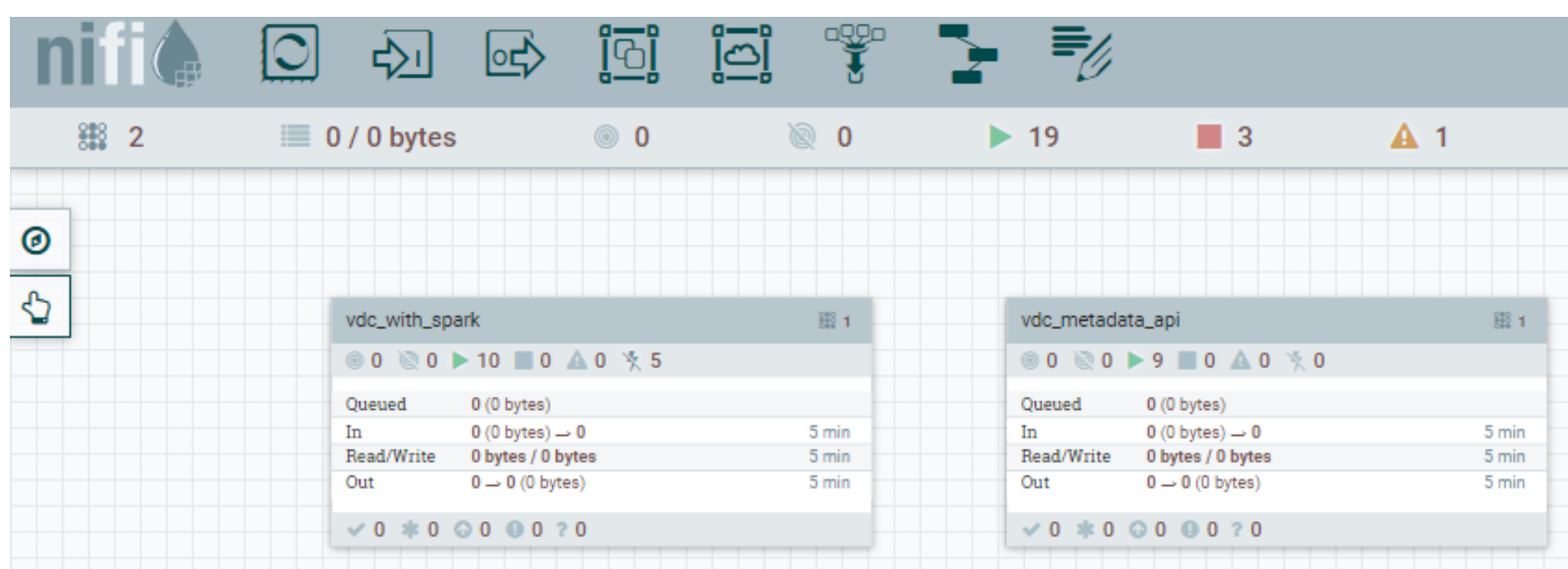
About the component

- **Pre-Processing and Filtering Software:** PaFS is responsible for the initial pre-processing, cleaning and filtering of the datasets
- **Virtual Data Repository:** VDR is the distributed infrastructure where all the pre-processed, cleaned, and filtered datasets, coming from PaFS, are saved
- **Virtual Data Container:** VDC is the interface through which communication with data recipients is achieved, for data stored in VDR to be made available



Target users

- Data Providers and Data Owners
- Data Consumers
- Application Developers
- Internal Platform Components



Use case scenarios

- Retrieve historical data
- Process, store and deliver cleaned data
- Export metadata
- Apply filtering rules to the data
- Internal components integration
- Use of the data by external apps
- Reuse of data by existing tools



Benefits

- **Interoperability:** Provide an abstraction layer between data providers and consumers
- **Less development effort:** Let the application developers just define the content and the format of the needed data and rely on the component to deliver them properly and timely
- **Scalability:** Ability to scale in a fully automated way based on the workload, aiming at optimizing resource utilization and decreasing response times
- **Ease of use and deploy:** Provide a common access layer, where the data consumers simply define queries as filtering rules in a unified format, regardless of the underlying storage technology