The BPA Lab

Architecture and Documentation of a Demonstration Factory for Business Process Automation and Analytics

Matthias Zapp

June 9, 2026

Table of contents

1. Overview **1**
- 1.1. The BPA Lab . . . 1
  - 1.1.1. BPA Lab as a research and demonstration platform . . . 1
  - 1.1.2. BPA Lab as a teaching platform . . . 1
- 1.2. Components of the BPA Lab . . . 2
- 1.3. Structure of this book . . . 3
- 1.4. Who this book is for . . . 3
- 1.5. Source code and related repositories . . . 3
- 1.6. Status and roadmap . . . 4
- 1.7. Licence and citation . . . 4

2. Business Scenario **5**
- 2.1. The implemented business scenario . . . 5
  - 2.1.1. Order management . . . 6
  - 2.1.2. Production control . . . 6
  - 2.1.3. Purchasing . . . 7
  - 2.1.4. Manufacturing . . . 7
  - 2.1.5. Shipment . . . 8
  - 2.1.6. Warehouse operations . . . 8
- 2.2. Complementary business scenario . . . 8
  - 2.2.1. Business Scenario: Custom Lead management für high-end custom bicycles 9

3. Solution Overview **13**
- 3.1. A modular, domain-driven design . . . 13
- 3.2. A simplified bird's-eye view . . . 13
- 3.3. Three architectural layers . . . 13
  - 3.3.1. Layer 1 — Controllers . . . 14
  - 3.3.2. Layer 2 — Process applications (job workers) . . . 14
  - 3.3.3. Layer 3 — BPMS workflow engine . . . 14
- 3.4. The demonstration factory implementation . . . 15

4. Software Architecture **17**
- 4.1. Requirements . . . 17
  - 4.1.1. R1 — Independent operation of components . . . 17
  - 4.1.2. R2 — Integration of external IT systems . . . 17
  - 4.1.3. R3 — Clear and distinct responsibilities . . . 17
  - 4.1.4. R4 — Easy extensibility . . . 17
  - 4.1.5. R5 — Interchangeability and independent evolution . . . 17
  - 4.1.6. R6 — Comprehensive and flexible process data capture . . . 18
  - 4.1.7. R7 — User-friendliness (operator perspective) . . . 18
  - 4.1.8. R8 — Easy installation and low infrastructure requirements (developer perspective) . . . 18

4.1.9. R9 — Robustness . . . 18
4.1.10. R10 — Maintainability and operation . . . 18
4.2. Architecture in the C4 model . . . 18
4.2.1. Context diagram . . . 18
4.2.2. Container diagram . . . 18
4.3. Architecture decisions . . . 20
4.3.1. ADR-001 — MQTT broker for job-worker–controller communication . . . 20
4.3.2. ADR-002 — Message Send/Receive Tasks instead of Events for inter-process communication . . . 20
4.3.3. ADR-003 — Messaging between processes and process applications . . . 20
4.3.4. ADR-004 — Camunda 8 self-managed with Docker Compose (Camunda 8 Core) . . . 21
4.3.5. ADR-005 — Job worker for sending emails . . . 21
4.3.6. ADR-006 — Job worker for database transactions; MySQL in a Docker container . . . 21

5. Data Architecture **23**
5.1. Overview of data sources . . . 23
5.2. The MySQL data model . . . 23
5.3. Sample and master data . . . 23
5.4. Process data and the foundation for process mining . . . 25

6. Projects **27**
6.1. Overview . . . 27
6.2. Teaching Business Process Automation in a Learning Factory (SoTL) . . . 28
6.2.1. Background and motivation . . . 28
6.2.2. Research question . . . 29
6.2.3. The teaching concept . . . 29
6.2.4. Preliminary evaluation . . . 29
6.2.5. Conclusions and consequences for teaching . . . 30
6.2.6. Funding . . . 30
6.2.7. Further reading . . . 31
6.3. System Architecture for a Modular BPA Learning Factory . . . 31
6.3.1. Contribution to the BPA Lab . . . 31
6.4. Designing and Implementing a Data Architecture for the BPA Lab . . . 31
6.4.1. Contribution to the BPA Lab . . . 32
6.5. Integrating IoT Data into Process Mining in an Industry 4.0 Environment . . . 32
6.5.1. Contribution to the BPA Lab . . . 32
6.6. Human-Centred Design of Form-Based User Interfaces in Camunda 8 . . . 33
6.6.1. Contribution to the BPA Lab . . . 33

7. Operational Instructions **35**
7.1. Purpose of this chapter . . . 35
7.2. Installing and running the solution . . . 35
7.3. Working with the productive environment of the model factory . . . 35
7.3.1. Preparations . . . 36
7.3.2. Known errors and their solution . . . 37
7.4. User guide for end-to-end process execution . . . 37
7.5. First-time configuration of the data architecture . . . 38
7.5.1. Configurations in Elasticsearch via Kibana . . . 38

7.5.2. Configurations in Grafana 41
7.6. Using the data architecture for process analysis and monitoring 46
7.6.1. Data architecture 46
7.6.2. Overview of the available data 47
7.6.3. How to use Grafana 47

Appendices **49**

A. Repositories and Components **49**
A.1. Main repositories 49
A.1.1. bpa_lab_docs 49
A.1.2. bpa_lab_demonstration_factory 49
A.1.3. bpa_lab_student_docs 50
A.2. Hardware components 50
A.3. Software components 50
A.4. Further reading 50

B. Glossary **51**

C. References **53**

1. Overview

1.1. The BPA Lab

The **Business Process Automation Lab (BPA Lab)** is an ongoing project at **TH Köln (University of Applied Sciences)** that supports research and teaching in the field of business process automation and analytics. It is run at the Faculty of Computer Science and Engineering Science and serves as a platform on which researchers and students can study and extend concepts and technologies of business process automation and analytics.

1.1.1. BPA Lab as a research and demonstration platform

As a *research platform*, the BPA Lab provides an environment in which bachelor's and master's theses as well as research projects can address questions around process automation and process analysis — for example regarding automation technologies, data architectures or process mining. The projects collected in the Projects chapter illustrate the range of work that has been carried out on this basis.

In its demonstration role, the lab shows how:

- a fictional business scenario can be represented in **process and decision models**,
- a **Business Process Management System** orchestrates human task and service in an end-to-end business processes,
- modular **job workers** (services) implement the actual work including the integration with IOT devices,
- **Internet of Things (IoT)** devices and data can be integrated into process execution,
- **process mining** can be used to analyse process executions and identify improvement opportunities.

Especially, by bringing together software and hardware in one running system, abstract concepts become more tangible.

1.1.2. BPA Lab as a teaching platform

As a *teaching platform*, the BPA Lab is used to support teaching in the field of business process management, especially business process automation and analytics.

Business process management theory, modelling notations like BPMN and DMN and business process automation technologies are often taught using abstract examples — pizza ordering, expense approvals, generic procurement workflows. These examples are accessible but rarely convey:

- the **domain complexity** of real-world processes,

- the **technical complexity** of integrating heterogeneous systems,
- the **data perspective** that is at the heart of process mining and analytics.

The BPA Lab as the learning factory aims to bridge this gap. The customer order triggers a workflow in Camunda 8, dispatches commands via MQTT to physical hardware, and eventually leads to a manufactured product while recording data. This end-to-end visibility of the process execution makes it easier for students to grasp the concepts and technologies of business process automation and analytics.

It offers lecturers a environment in which existing processes can be demonstrated, new processes be designed, implemented in a BPMS, executed against physical hardware, and analysed using the resulting data.

The didactic rationale and an early evaluation of this approach are documented in the project's first SoTL publication (Projects chapter).

1.2. Components of the BPA Lab

The BPA Lab is built around a **fictional business scenario** (see Business Scenario), which is represented by a set of process, decision, and data models. This scenario is implemented through **business process automation systems** (see Solution Overview). As part of the implementation, hardware components are also controlled.

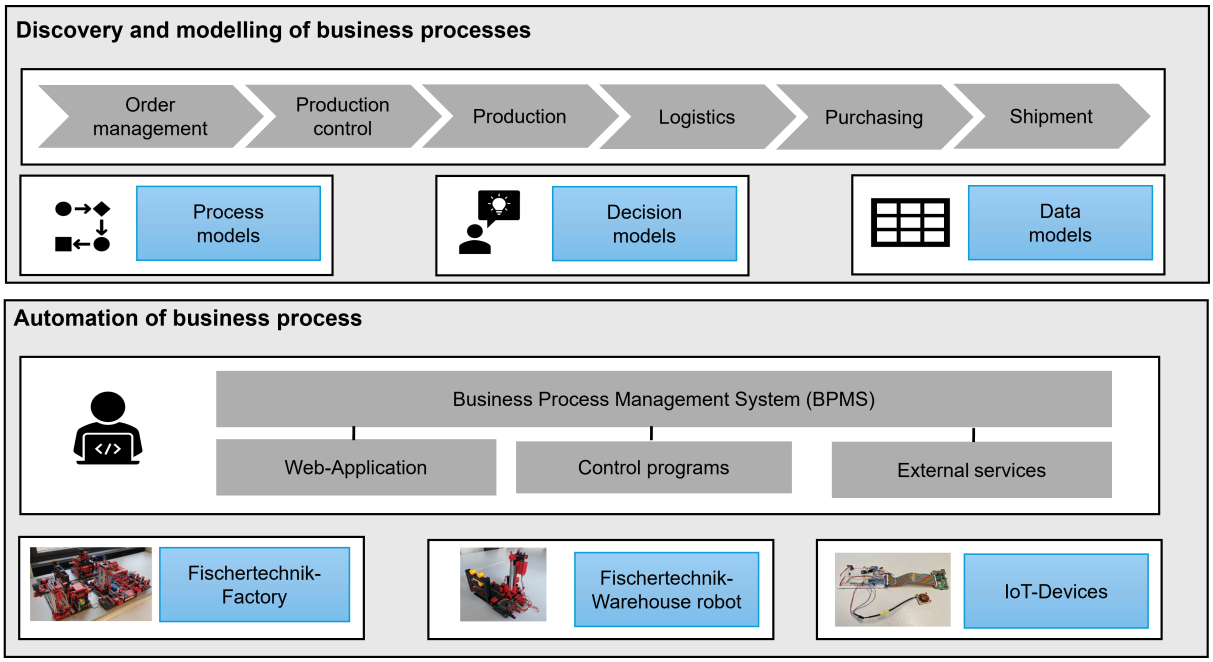


Figure 1.1.: BPA Lab — Conceptual Architecture

Building on these components, several projects have been carried out and are documented in the Projects chapter.

1.3. Structure of this book

This book s structured as follows:

1. **Business scenario:** The end-to-end bicycle ordering, manufacturing, and shipment process, broken down into its sub-processes (order management, production control, purchasing, manufacturing, shipment, warehouse operations).
2. **Solution overview:** Why a demonstration factory, and the three architectural layers — controllers, process applications, workflow engine.
3. **Software architecture:** Requirements, a C4 view of the system, and the architecture decisions taken so far.
4. **Data and data sources:** Data model, data flows, and persistence in the BPA Lab.
5. **Projects:** Research and teaching projects conducted using the BPA Lab.
6. **Technical instructions:** Operational instructions for running the demonstration factory, executing end-to-end processes, and configuring and using the data architecture.

The appendices contain:

- **Appendix A:** Overview of related components and related repositories.
- **Appendix B:** A glossary of key terms.
- **References:** All cited literature.

1.4. Who this book is for

This book is written for:

- Primary, the book is generated for **students** working on bachelor's or master's projects or any other project inside the BPA Lab.
- Other **lecturers and researchers** who are interested in teaching business process automation with a tangible, end-to-end reference system.
- **Practitioners** looking for concepts on combining BPMS, IoT hardware, and process mining.

1.5. Source code and related repositories

The implementation of the demonstration factory is maintained as a separate repository: bpa_lab_demonstration_factory.

Documentation for student component projects is available in the wiki of bpa_lab_student_docs.

1.6. Status and roadmap

> **!** Work in progress
>
> The BPA Lab — and this book — are under continuous development. Several sections are marked as *work in progress* and will be expanded in future revisions. Bug reports and suggestions are very welcome via the issue tracker.

1.7. Licence and citation

Suggested citation:

> Zapp, Matthias (2026). *The BPA Lab — Architecture and Documentation of a Demonstration Factory for Business Process Automation and Analytics.* TH Köln (University of Applied Sciences). Online: https://bpalabthcologne.github.io/bpa_lab_book/.

2. Business Scenario

In this chapter, we describe the **business scenario** that is implemented in the BPA Lab. The scenario is a fictional bicycle manufacturer that offers highly customisable bicycles to consumers. The end-to-end process includes order management, production control, purchasing, manufacturing, shipping, and warehouse operations.

In addition complementary business processes are described, which can be used for additional implementation projects, e.g. Agentic AI in business process automation.

2.1. The implemented business scenario

A bicycle manufacturer offers highly customisable bicycles to consumers.

The entire end-to-end process is orchestrated by the **order management process**, which in turn initiates purchasing, manufacturing, and shipping processes as needed.

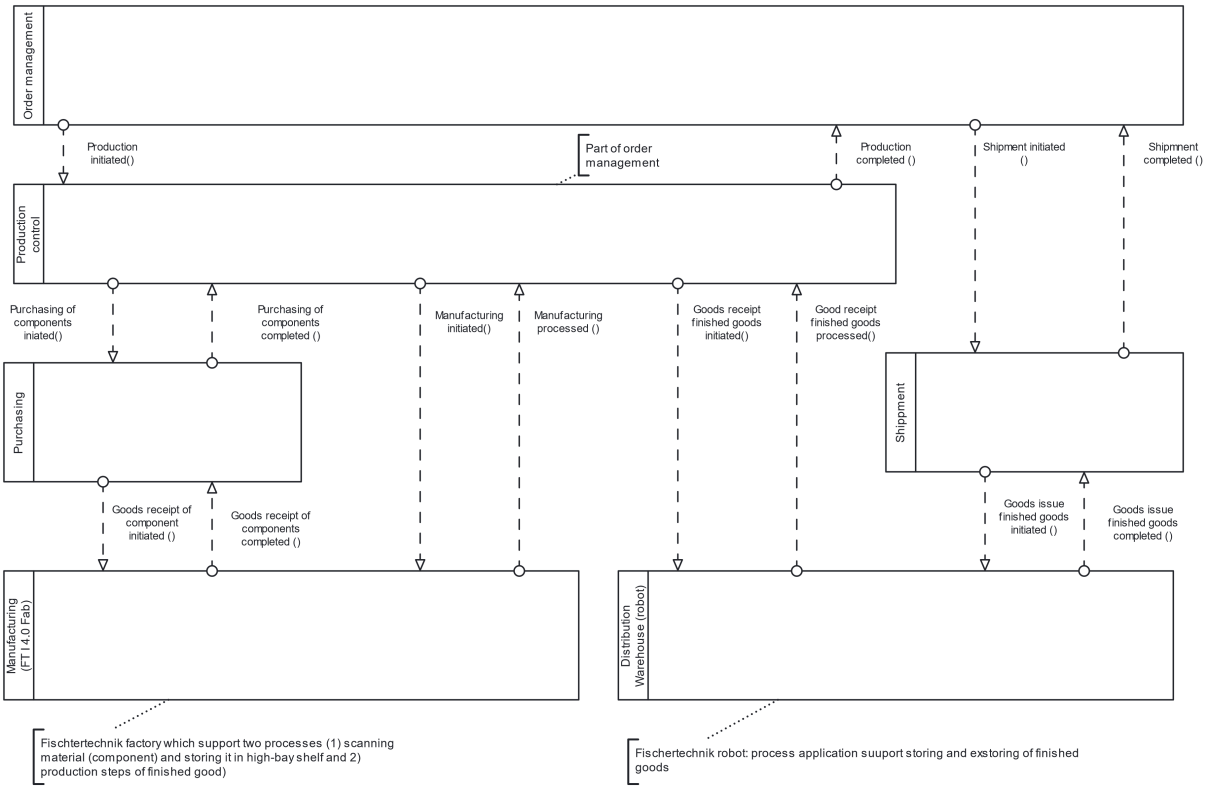


Figure 2.1.: Process landscape of the BPA Lab

The manufacturing processes use the **Fischertechnik Industry 4.0 factory** to post goods receipts of components and to perform the actual manufacturing steps. Order management and

shipping additionally use the **distribution warehouse** (a separate Fischertechnik robot) to post goods receipts and goods issues for finished products.

The following sections describe each sub-process in turn. The process models represent mainly the **happy path** of execution (to be extended).

2.1.1. Order management

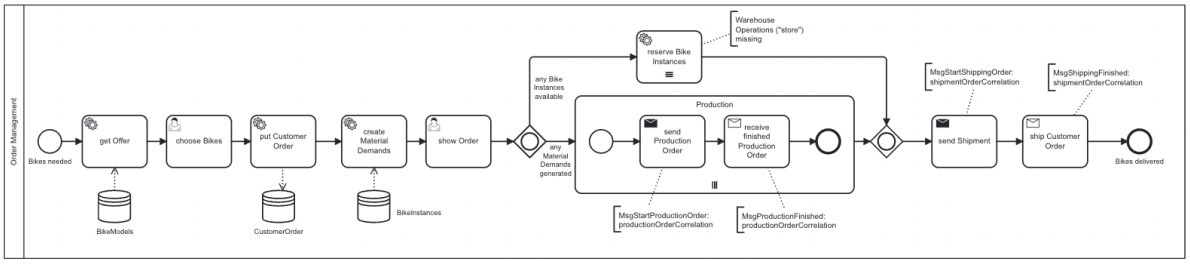


Figure 2.2.: Order management process

The order management process governs the fulfilment of customer orders.

It starts when a customer order is entered into a web form, which is pre-filled with product data (bicycle models) from the database. During execution, the **availability of the ordered products** is checked.

Two main scenarios apply:

- **All products in stock:** the shipping process is triggered directly. Once shipping completes successfully, the process is finished.
- **One or more products out of stock:** for each unavailable product, the production control process is triggered, which creates a **production order**. The production order specifies the required components.
  - If the components are also unavailable, a **purchasing process** is started first.
  - Otherwise, the **manufacturing process** is triggered immediately.

Once the missing products are manufactured, shipping is initiated.

2.1.2. Production control

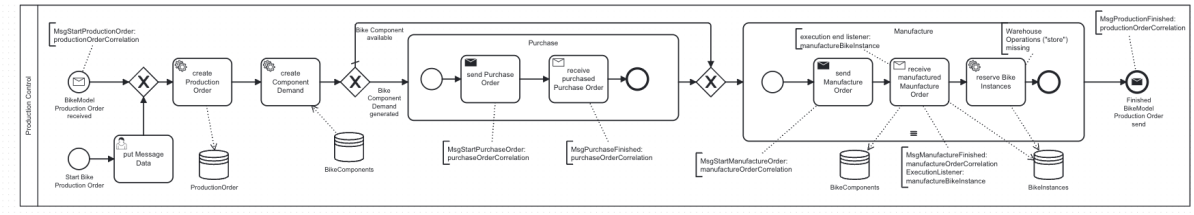


Figure 2.3.: Production control process

The production control process manages the creation and execution of **one production order for one product**. It determines the component demand and — if needed — triggers a purchasing

process to procure the missing components. Once components are available, it triggers the actual manufacturing process.

2.1.3. Purchasing

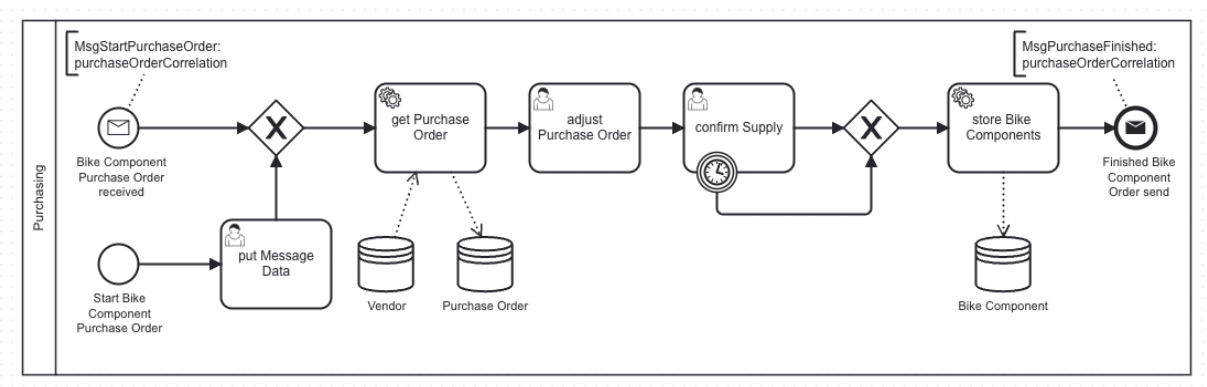


Figure 2.4.: Purchasing process

The purchasing process manages the procurement of required components. It starts whenever a component is needed and creates a **purchase order**.

The purchase order can be adjusted by the user — for example by selecting a specific vendor. Materials are then stored and stock levels are updated. Upon user confirmation (or after a configurable timeout), the stock level is increased in the inventory database.

2.1.4. Manufacturing

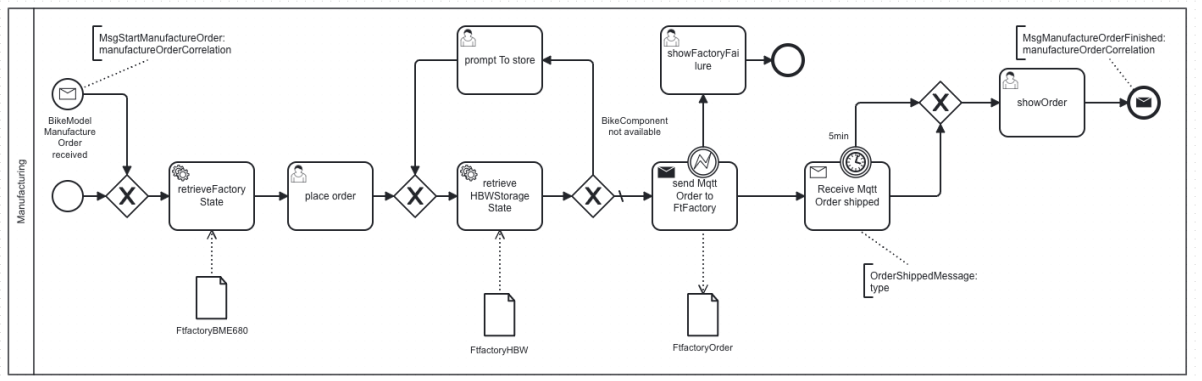


Figure 2.5.: Manufacturing process

The manufacturing process governs the **physical manufacturing simulation** on the Fischertechnik factory for **one item of one product type**.

It starts after a production order has been created by the production control process and sufficient components are available.

The process must place a manufacturing order that produces a new bicycle from the specified material. Before that, however, it must check whether a **physical workpiece** is already available

in the warehouse of the Fischertechnik factory. If not, a physical workpiece must first be transferred to the factory so that the control program can store and recognise it. Once the bicycle has been produced from the material (workpiece), the process is complete.

2.1.5. Shipment

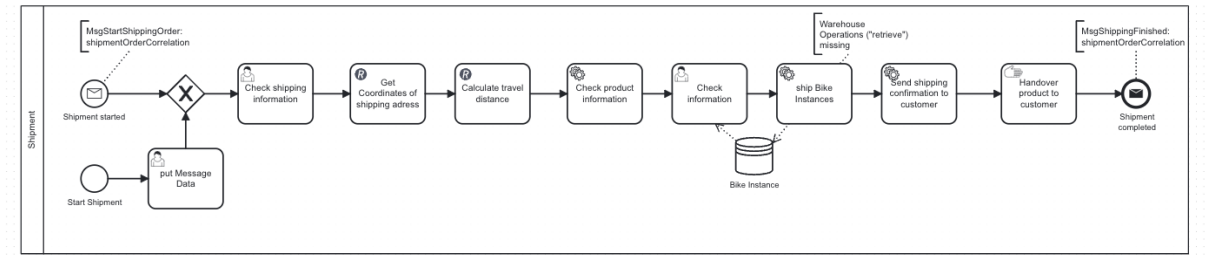


Figure 2.6.: Shipment process

The shipment process manages the dispatch of finished goods to the customer. It is **only** triggered if a product is in stock and ready to be shipped.

The flow is:

1. Shipping data is checked.
2. The distance and duration from the warehouse to the delivery address are computed — the **Openrouteservice API** is used for this.
3. The logistics sub-process retrieves the customer order from the warehouse.
4. A message is sent to the customer to inform them about the dispatch.
5. Upon handover of the order to the customer, both the shipping process and the surrounding end-to-end process instance are completed.

2.1.6. Warehouse operations

The warehouse operations process governs the flow of finished goods in a distribution warehouse — represented physically by the Fischertechnik warehouse robot.

It starts when a **transfer order** for the warehouse is received from another business process. A transfer order can be either an instruction to **store products** in the warehouse or to **retrieve products** from it.

2.2. Complementary business scenario

This section describes complementary business scenarios/processes that are not yet implemented or integrated in the current system, but have and can be used for projects.

2.2.1. Business Scenario: Custom Lead management für high-end custom bicycles

This business scenario deals with managing leads and orders for custom high-end bicycles. Remark: This description was designed to show case AI assisted process analysis. The content generation was supported by Claude AI.

2.2.1.1. IT landscape

Its IT landscape is a mix of standard software, lightweight cloud tools, and a collection of Excel files and PDF templates that sales and the workshop have built up themselves over time. There is no single system that covers the full order from inquiry to handover end-to-end. Instead, information lives across the following systems and storage locations: - Website with configurator (WordPress plus a simple form plugin): produces an email and writes a row into a Google Sheets table. - HubSpot CRM (free tier): leads and contacts maintained by sales through manual entry. - Shared Google Drive folder per project, named YYYY-NNN_LastName (e.g. 2026-042_Mueller): contains PDF quotes, scanned signatures, fitting reports, CAD exports, paint renderings, and progress photos. - Excel pricing calculator (Calculation_v17_final_NEW.xlsx): duplicated by sales for each project and filled in with the specification. - CAD system BikeCAD Pro on a single workstation used by the frame designer; geometry data leaves the system as a PDF export. - Paint configurator (browser-based, external vendor): produces PNG renderings that are downloaded individually. - DocuSign for legally binding signatures (indicative quote, binding specification, handover protocol). - Stripe for payments, reconciled against Sage 50 (accounting). - Workshop whiteboard and a laminated A3 sheet per bike acting as a physical traveler — the workshop deliberately stays largely paper-based. - WhatsApp and email as the main communication channels with the customer, occasionally supplemented by video calls. There is no customer portal. Status information reaches the customer through update emails or WhatsApp messages that the sales engineer writes.

Information Processing per Process Step (As-Is): The following description makes visible, for each step, what data is created, how it is captured, and where it is stored.

2.2.1.2. Lead Capture and Qualification

The customer fills in the web form. The plugin sends an email to sales and writes a record into a Google Sheet. These two data stores are independent and are not synchronized to HubSpot. In the morning, the sales engineer reviews the inbox, opens HubSpot, and creates a new contact and deal by taking name, email, phone, and the configurator answers from the form email. Body measurements and budget are placed into a free-text notes field — there are no structured custom properties for them. For the capacity check, the sales engineer opens a separate Excel file Workshop_Capacity_2026.xlsx that the workshop manager updates once a week. If the lead is declined, sales writes an individual email and sets the deal stage in HubSpot to “Lost”.

2.2.1.3. Discovery Call, Indicative Quote, Design Deposit

The discovery call runs on Zoom. The sales engineer takes notes in a Word document that is later exported to PDF and copied into the project folder — the project folder is only created after the design deposit is paid, so until then the notes sit on the sales engineer’s local desktop. For the indicative quote, sales open the Excel pricing calculator, duplicates it via “Save As”,

and assigns a file name with the project number. The specifications from the call are entered into the calculator cells. The calculator returns a price range; the result is transferred into a Word quote template. The Word file is exported to PDF, uploaded to DocuSign, and sent to the customer. After signing, sales download the signed PDF from DocuSign and places it into the — now newly created — project folder. The design deposit is collected through a Stripe payment link generated from the Stripe dashboard and pasted into an email. Payment receipt is verified by reviewing the Stripe dashboard daily; there is no webhook into HubSpot. Once payment is in, the deal is moved to the "Design" stage in HubSpot, a project number is pulled from a separate counter Excel file, and the Google Drive folder is created.

2.2.1.4. Bike Fitting and Gate 1 (Fit Data Approval)

If the fitting takes place in Köln, the in-house fitter uses the Retül software, exports the result as PDF, and places it in the project folder. If it happens at a partner fitter, the report arrives by email attachment in one of twelve different formats — every partner uses their own tool. Sales reviews the attachment and enters the key dimensions (saddle height, stack, reach, cleat position, shoulder width) into an Excel sheet called Fit_Data_Consolidation.xlsx so that the frame designer can work from a single consistent format. For the approval (Gate 1), sales emails the fitting PDF plus a short summary to the customer and, in parallel, to the workshop manager. Approval comes back as an email reply ("looks good, approved" or follow-up questions). The confirmation emails are filed into the project folder as .eml files.

2.2.1.5. Design Proposal and Customer Review Loops

The frame designer takes the consolidated fit data from the Excel sheet and enters it into BikeCAD. The finished geometry drawing is exported as PDF. The tubeset specification (Columbus type, wall thicknesses, braze-on parts) is maintained in a separate Word template. The component list is built in another Excel sheet that draws on a supplier price list maintained by the workshop manager. This price list is refreshed once per quarter. The paint scheme is designed in the external paint configurator; the final PNG renderings are downloaded and placed into the project folder. Sales then assembles a PDF design dossier from all these individual documents and sends it to the customer by email. The revision rounds run over email. Customer change requests ("shorten the stem by 1 cm", "matte paint, RAL 7016 instead of 7021") are collected by sales, forwarded to the designer and the paint configurator, and after rework sent back out as a new dossier. Versioning is handled through file names (Design_Dossier_v1.pdf, v2.pdf, v2_final.pdf, v2_final_CORRECTED.pdf). Tracing which version the customer last saw, or what changed between v2 and v3, relies on the email thread.

2.2.1.6. Gate 2 — Binding Specification and 40% Deposit

After customer approval, sales consolidates the approved individual documents into a final specification document (Word → PDF). This document is signed via DocuSign by both customer and managing director. The 40% deposit is collected through a Stripe link with reconciliation through the Stripe dashboard. At this point, the bill of materials for procurement and production is created. The workshop manager opens the signed specification PDF and enters the components into an Excel sheet (Master_BoM.xlsx), which is then emailed to the buyer and to production. In parallel, the workshop manager adds the project to the capacity Excel so the next discovery call can assess capacity correctly.

2.2.1.7. Status Communication and Gates 3/4 During Production

During the 8–16 weeks of production, the workshop works with the laminated A3 traveler and the whiteboard. Status lives on these physical artifacts. If sales wants to know where a project stands, they walk into the workshop or call the workshop manager. Customer updates are produced on an ad-hoc basis: when the workshop manager hits a meaningful milestone, they photograph the bike with their smartphone, send the photo by WhatsApp to sales, who then composes an update email or WhatsApp message to the customer. For Gate 3 (paint approval after primer), the paint shop photographs the bike, emails the images to sales, who forwards them on. Approval comes back by reply email or WhatsApp and is filed by sales as a screenshot or .eml file in the project folder. Gate 4 before final assembly works the same way. Change requests during production ("different saddle color after all") travel through the same channels: the customer sends a WhatsApp to sales, sales emails the workshop manager, the workshop manager writes the change onto the traveler by hand and informs the affected workshop staff verbally. Additional cost is estimated case-by-case and added to the final invoice.

2.2.1.8. Handover, Final Payment, and Project Closure

The final invoice (60%) is created in Sage 50 from the Excel calculation plus any change orders, and sent by email with a Stripe link. Payment receipt is verified through the Stripe dashboard. At the handover appointment, the handover protocol is filled out on a prepared PDF form on a tablet, signed by the customer with a finger, and filed as a PDF in the project folder. Warranty registration and scheduling of the free 90-day fit-tuning appointment are entered into HubSpot and into a Google Calendar. The project folder lives on as an archive.

3. Solution Overview

3.1. A modular, domain-driven design

A key design principle in the BPA Lab is **modularity**. The system aims for aspects of a **Domain-Driven Design (DDD)** approach so that:

- **multiple students and developers can work in parallel** on different parts of the system without stepping on each other's toes,
- **components can be reused** in student projects to build new process applications,
- **components can be replaced or upgraded** independently.

The primary criterion for splitting job workers and process models is the **business domain**:

- Order management
- Manufacturing
- Shipping
- Purchasing
- Logistics / warehouse operations

This separation aligns with the bounded contexts familiar from DDD and makes the system easier to reason about, both for teachers explaining it and for students extending it.

3.2. A simplified bird's-eye view

The architecture and implementation of the BPA Lab are under continuous development. The following diagram shows the main components of the solution .

The customer-facing entry point is a **user form** (or **web application**= for order capture. From there, work flows downwards through Camunda 8 and the job workers and eventually reaches the physical Fischertechnik components. Events generated along the way flow back upwards and are persisted in databases for later analysis.

3.3. Three architectural layers

The solution is structured into three clearly separated layers.

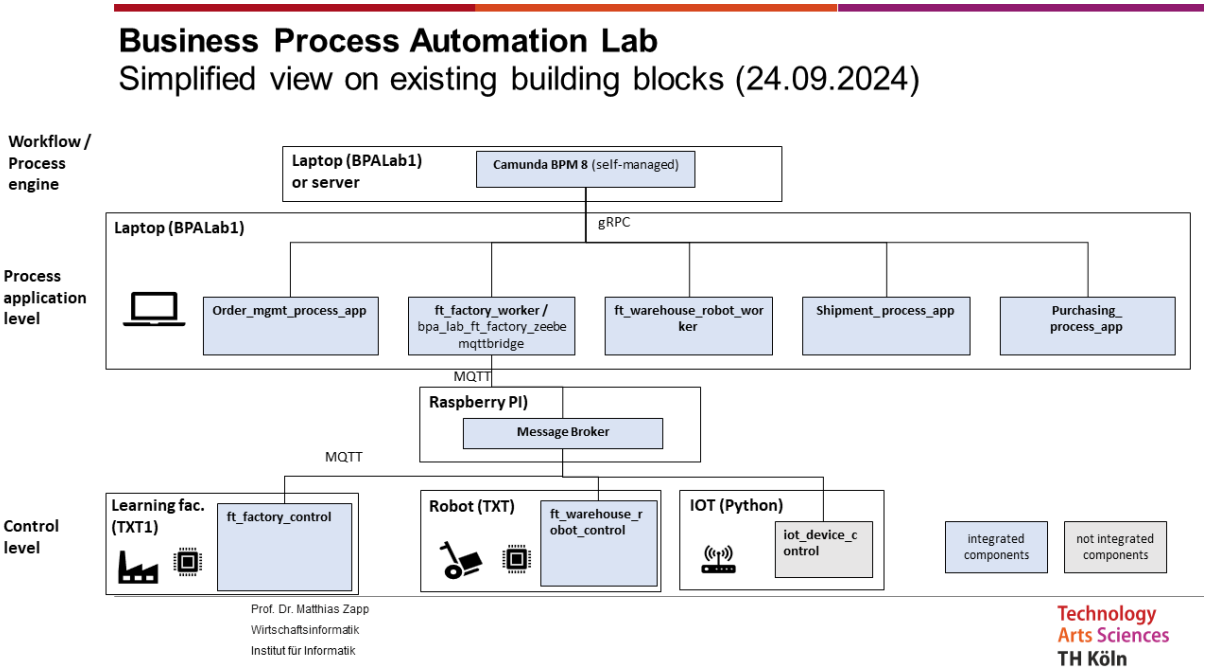


Figure 3.1.: Simplified architecture of the BPA Lab

3.3.1. Layer 1 — Controllers

Controllers are responsible for the direct control of hardware components: the Fischertechnik Industry 4.0 factory, the Fischertechnik warehouse robots, and IoT devices. They are predominantly implemented in **Python** and translate high-level commands (for example *"transport workpiece from slot A to slot B"*) into the low-level signals that move motors and read sensors.

3.3.2. Layer 2 — Process applications (job workers)

Process applications are middleware components that implement the actual business logic of individual work steps. In Camunda 8 terms, they are **job workers**: they subscribe to specific service tasks and execute them when activated by the workflow engine.

Job workers communicate:

- with the **workflow engine** via **Rest** or **gRPC** APIs,
- with the **controllers** via **MQTT** (a lightweight publish-subscribe protocol that is well established in the IoT space).

Process applications are predominantly implemented in **Java**.

3.3.3. Layer 3 — BPMS workflow engine

The central workflow engine is **Camunda 8 (self-managed)**. It executes:

- **BPMN** process models — describing how work flows through the system,
- **DMN** decision models — encoding business rules.

Camunda 8 maintains the state of each running process instance, handles incidents, and provides the data foundation for process mining and analytics.

3.4. The demonstration factory implementation

The latest version of the BPA Lab as a runnable **demonstration factory** is available in a separate repository:

github.com/BpaLabTHCologne/bpa_lab_demonstration_factory

The implementation is fully containerised with **Docker**. This makes local setup straightforward and ensures that the environment is reproducible across different machines — a key requirement when students set up the system on their own laptops.

> Getting started in practice
>
> If you are setting up the BPA Lab for the first time, follow the installation guide in (Technical Instructions).

4. Software Architecture

> **!** Status
>
> The architecture is under continuous development. Several aspects are still being discussed. This chapter captures the current state.

4.1. Requirements

The following high-level requirements were gathered at the very beginning of the BPA Lab project (originally as part of a thesis project). They served as the basis for the first software architecture and remain the reference point against which architectural choices are evaluated.

4.1.1. R1 — Independent operation of components

The individual components and processes should be able to run independently of each other. Each process must be startable on its own, without depending on a triggering instance from a higher layer.

4.1.2. R2 — Integration of external IT systems

The system must allow integration with existing IT components — for example with the web application used for order capture, or with tools that flexibly extract data for process mining and analytics.

4.1.3. R3 — Clear and distinct responsibilities

The responsibilities of individual components must be clearly defined and demarcated from one another, so that the system remains understandable as it grows.

4.1.4. R4 — Easy extensibility

The system must be easy to extend with additional, self-developed components.

4.1.5. R5 — Interchangeability and independent evolution

Components must be developable, deployable, and replaceable independently of one another.

4.1.6. R6 — Comprehensive and flexible process data capture

Process data must be comprehensively capturable. It must be stored and made flexibly available for a range of analytical use cases — in particular process mining.

4.1.7. R7 — User-friendliness (operator perspective)

The user interface must be intuitive and easy to understand, ensuring a positive user experience for those interacting with the lab during demonstrations and student sessions.

4.1.8. R8 — Easy installation and low infrastructure requirements (developer perspective)

The system must be easy to install. Students must be able to host their own instances on standard laptops without specialised infrastructure.

4.1.9. R9 — Robustness

The architecture must be robust. Failures must be minimised and reliable functionality must be ensured under a wide range of operating conditions.

4.1.10. R10 — Maintainability and operation

The system must be easy to maintain over its lifetime.

4.2. Architecture in the C4 model

The architecture is documented using the **C4 model** (Brown 2018) — a notation for describing software systems at different levels of abstraction: Context, Container, Component, and Code. Note that *containers* in the C4 sense are **not** the same as Docker containers.

4.2.1. Context diagram

The context diagram shows the BPA Lab in relation to its users and external systems.

4.2.2. Container diagram

The container diagram shows the main technical building blocks of the BPA Lab and how they interact.

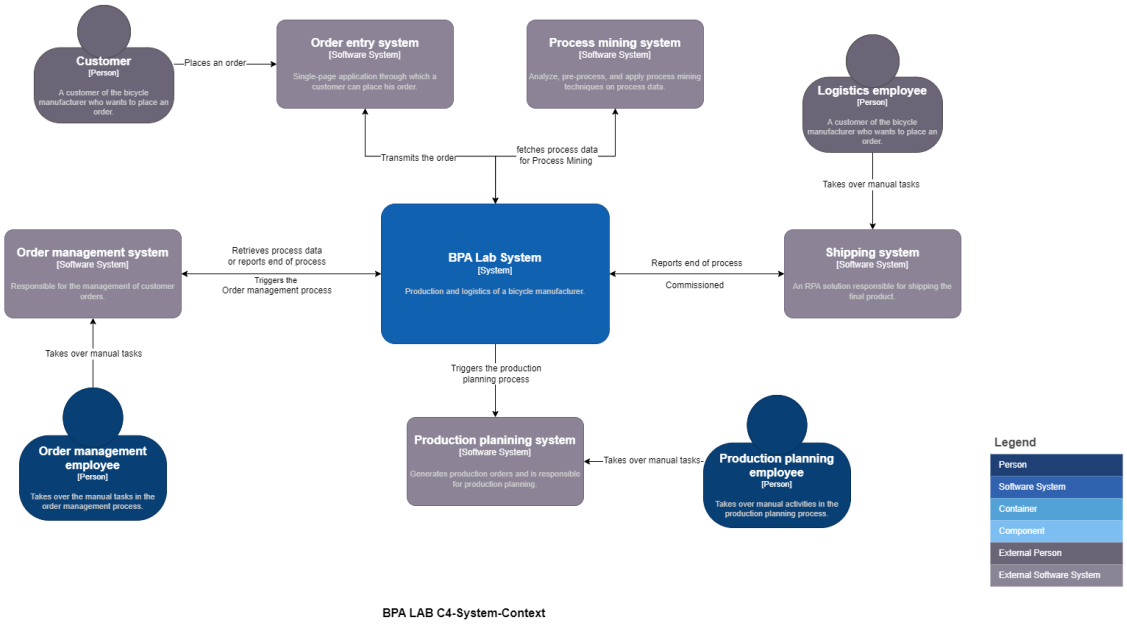


Figure 4.1.: C4 context diagram

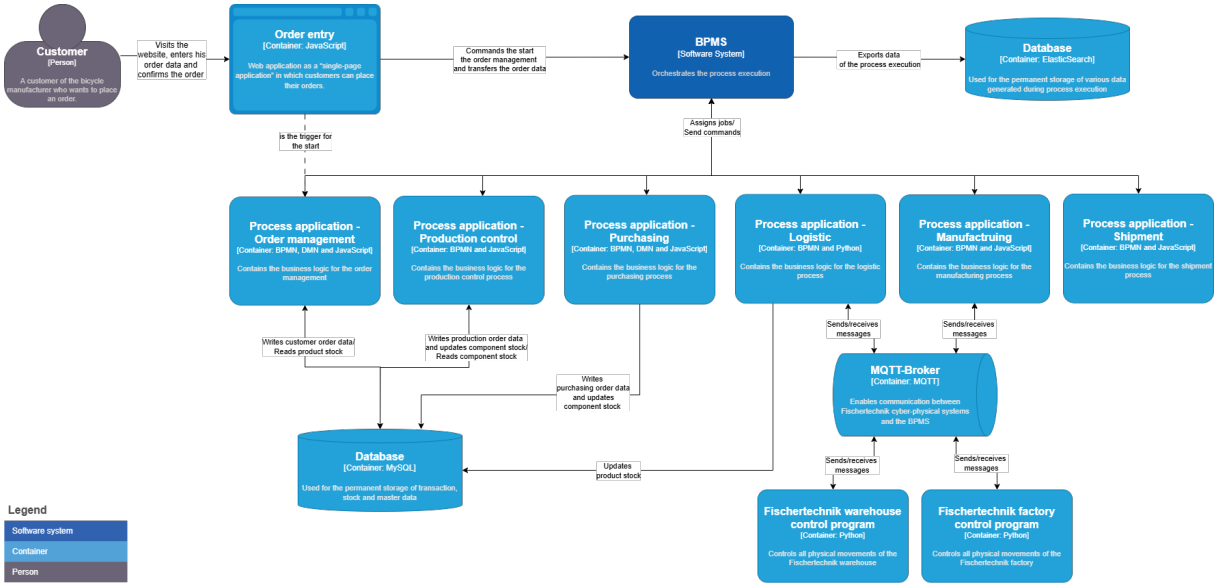


Figure 4.2.: C4 container diagram

4.3. Architecture decisions

The remainder of this chapter records selected architecture questions raised during the BPA Lab project, together with the corresponding decisions and justifications. The list is **not exhaustive** and is updated as the lab evolves.

Each decision follows a lightweight **ADR** (Architecture Decision Record) schema: context, decision, and rationale are captured succinctly.

4.3.1. ADR-001 — MQTT broker for job-worker–controller communication

Status: Decided.

Decision: A MQTT broker is used for communication between process applications (job workers) and controllers.

Rationale:

- decoupling of senders and receivers through the publish-subscribe pattern,
- MQTT is a well-established standard protocol in the IoT space,
- lightweight footprint, well suited to the hardware components in use.

4.3.2. ADR-002 — Message Send/Receive Tasks instead of Events for inter-process communication

Status: Decided.

Decision: Inter-process communication is modelled in BPMN using **Message Send Tasks** and **Message Receive Tasks**, not events.

Rationale:

- Tasks allow the use of **boundary events** — for example to handle errors raised by job workers,
- complex error paths and timeouts are easier to express.

4.3.3. ADR-003 — Messaging between processes and process applications

Status: Decided.

Decision: Communication between processes and process applications is implemented using messages as well.

Rationale:

- a simple, uniform solution,
- consistent with the communication pattern between process applications and process control.

4.3.4. ADR-004 — Camunda 8 self-managed with Docker Compose (Camunda 8 Core)

Status: Decided.

Decision: The self-managed variant of Camunda 8 is used in combination with Docker Compose and Camunda 8 Core.

Rationale:

- avoids interferences between contributors during the development phase,
- development and production-like environments are identical,
- Kubernetes is intentionally not used — the BPA Lab is not a production environment in the classical sense.

Open question: Performance with Docker and the resulting number of containers needs to be verified as the system grows.

4.3.5. ADR-005 — Job worker for sending emails

Status: Decided (reversible).

Decision: Email sending is implemented through a dedicated job worker rather than via the SendGrid connector.

Rationale:

- issues encountered with the SendGrid connector at the time of the decision,
- an existing solution was already in place in the project,
- to be re-evaluated as Camunda 8 connectors continue to evolve.

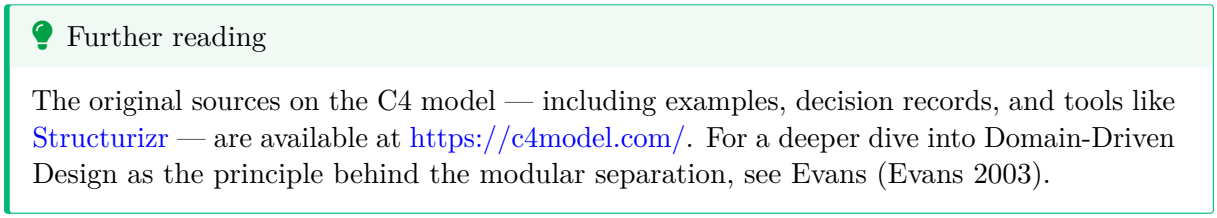
4.3.6. ADR-006 — Job worker for database transactions; MySQL in a Docker container

Status: Decided.

Decision: Database transactions are handled by a dedicated job worker; the MySQL database runs in a Docker container.

Rationale:

- existing solution available in the project,
- greater flexibility than the connectors available at the time of the decision.



5. Data Architecture

5.1. Overview of data sources

The BPA Lab integrates data from several sources — from **master data** in a relational database, via **process state** maintained by the workflow engine, to **sensor and telemetry data** from the Fischertechnik components.

The following figure gives an overview of the main data sources and the kinds of data they hold:

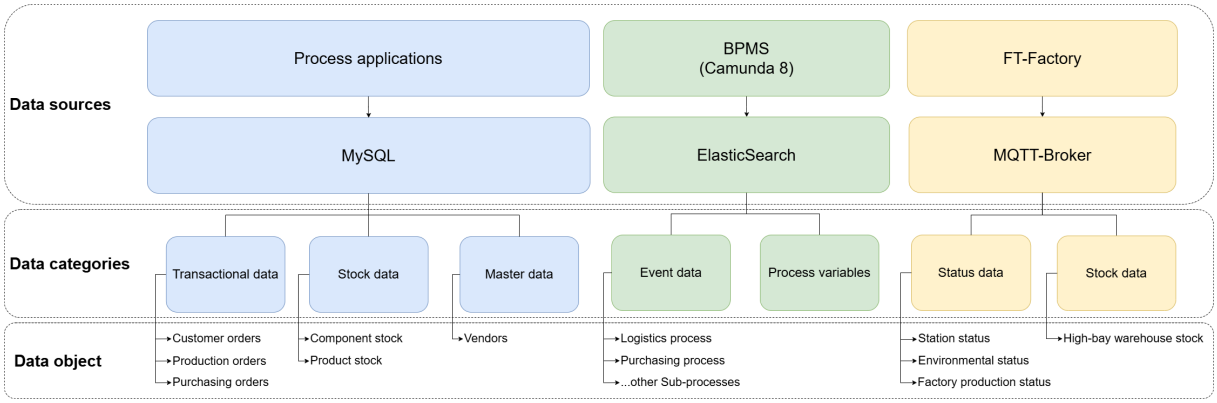


Figure 5.1.: Data overview in the BPA Lab

5.2. The MySQL data model

The following diagram shows the concrete data model of the MySQL database, including the entities and attributes of the current implementation:

The database serves as the central store for:

- **Product master data** — bicycle models and their components,
- **Customer data** and delivery addresses,
- **Stock levels** for components and finished goods,
- **Orders** — customer orders, production orders, purchase orders — together with their status history,
- **Vendor data**.

5.3. Sample and master data

Documentation and sample data of the Fischertechnik factory are available in the `Sample_data` folder of the documentation repository. This data serves as:

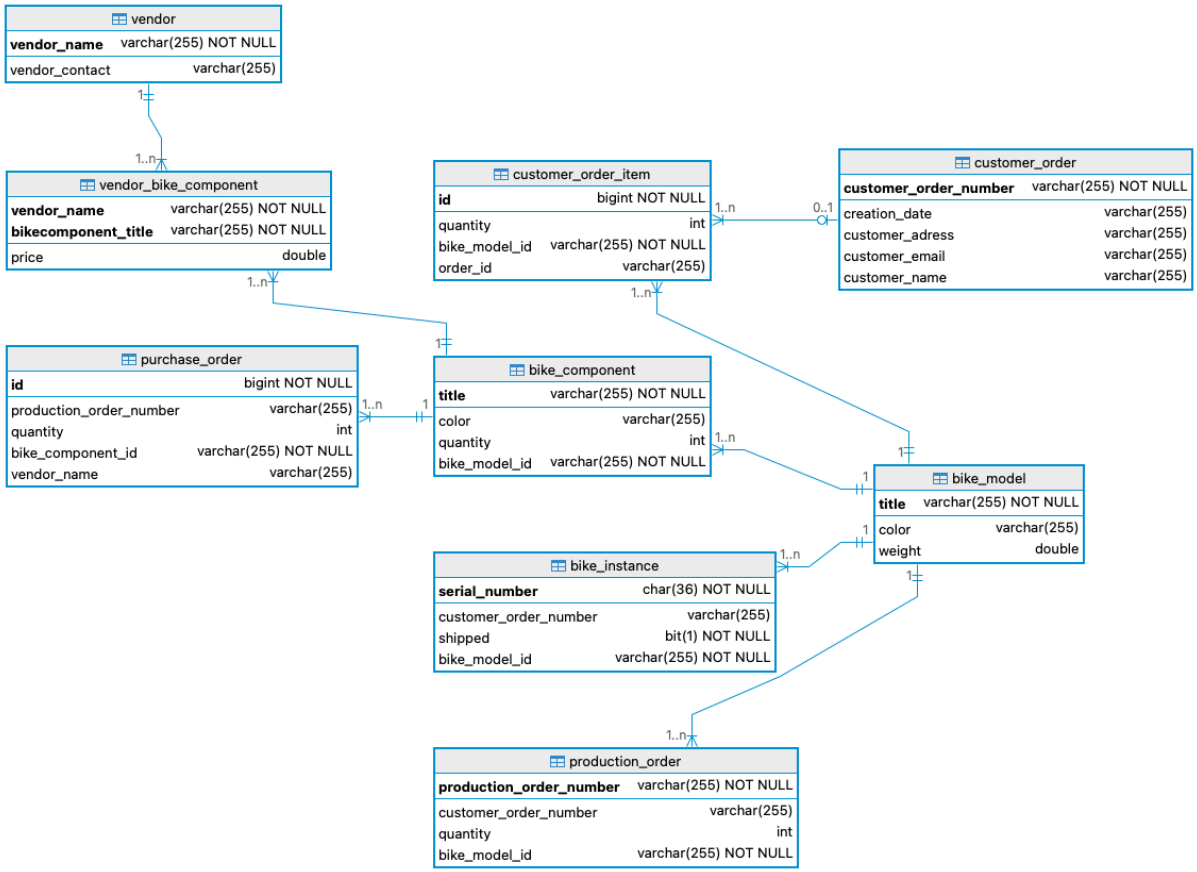


Figure 5.2.: MySQL data model

- **initial master data** when setting up a new instance of the BPA Lab,
- **test data** for developing new components,
- a **demonstration baseline** for teaching sessions and presentations.

5.4. Process data and the foundation for process mining

In addition to persistent master data, the running processes generate substantial amounts of **process data**:

- **Camunda 8** stores process instance data, variables, tokens, and lifecycle events.
- **MQTT messages** record the communication between job workers and controllers.
- **Job worker logs** capture technical operations and exceptions.

This event data is the foundation for **process mining** — a central teaching and demonstration component of the BPA Lab. Process mining allows students to:

- **discover** the actual flow of processes from event logs (and compare them to the modelled flow),
- **check conformance** between modelled and actual behaviour,
- **analyse performance** — durations, bottlenecks, rework loops.

6. Projects

> **Purpose of this chapter**
>
> The BPA Lab is not only an artefact — it is also a vehicle for **research and teaching projects**. This chapter collects projects conducted with, or around, the BPA Lab. Each entry summarises the project's motivation, approach, and main outcomes. IT is not a complete list of all projects, but rather a curated selection that illustrates the lab's potential as a research and teaching platform.

6.1. Overview

The following table gives a quick overview of the projects documented in this chapter. They are listed in the order in which they appear below.

Project	Authors	Type	Year	Contribution to the BPA Lab
Teaching BPA in a Learning Factory	Zapp, Gonzalez & Gonzalez	SoTL publication (peer-reviewed)	2025	Teaching concept and pilot evaluation of using the BPA Lab as a teaching environment for project- and problem-based learning.
System Architecture for a Modular BPA Learning Factory	J. P. Iyebi Itomb	Master's thesis	2023	Modular system architecture for the BPA Lab based on Domain-Driven Design and Camunda 8

Project	Authors	Type	Year	Contribution to the BPA Lab
Data architecture of the BPA Lab	D. Gonzalez	Master's thesis	2025	Data-virtualisation layer (Trino), IoT pipeline (Filebeat), and two Grafana dashboards for end-to-end and production-process analysis.
Integrating IoT Data into Process Mining	B. Kücük	Master's thesis	2026	Integration of workflow event data with IoT data in event logs
User form design in Camunda 8	J. Will	Bachelor's thesis	2026	BPM-form usability guidelines, a consistent form layout, and redesigned or newly created user-task forms, with usability evaluation.

6.2. Teaching Business Process Automation in a Learning Factory (SoTL)

Authors: Matthias Zapp, Saphira Gonzalez & Domenic Gonzalez (TH Köln) **Publication:** *Forschung und Innovation in der Hochschulbildung*, Nr. 25, 2025, Research Paper **Publisher:** Cologne Open Science (COS), TH Köln **DOI:** 10.57684/COS-1326 **Type:** Scholarship of Teaching and Learning (SoTL) publication (Coleman et al. 2023)

6.2.1. Background and motivation

Business Process Automation (BPA) is a socio-technical challenge that requires a diverse set of competences — **technical** (process modelling in BPMN, implementing features in a BPMS, etc.), **methodological** (planning and running automation projects), as well as **social and personal** competences for collaborating with stakeholders from technical and business backgrounds (Nerdinger et al. 2008; Schaper 2012).

To teach this combination of skills, the bachelor's programme in Business Information Systems at TH Köln uses **Project-Based and Problem-Based Learning (PBL)** (Barrows 1996; Chow 2021; Santos et al. 2023). Students work in teams of around five to six people on automation

projects using a BPMS — an approach that creates active learning and practical problem-solving, similar to PBL formats described in the literature (Bandara et al. 2010).

However, the authors observed a **persistent gap**: in regular bachelor's courses with around 100 students, project tasks must be simplified so they remain manageable. This unavoidable simplification means students often miss the experience of **integrating automation into a realistic enterprise IT landscape**. A telling classroom observation: on a final-presentation day, one student team was surprised by another team that demonstrated triggering a BPMS from a simulated company web portal — a feature mentioned in lectures, but absent from most project tasks because of the limits of a simplified setting.

6.2.2. Research question

The paper asks:

> *"To what extent can the use of a learning factory with software and hardware components improve project-oriented and problem-oriented teaching in the field of business process automation?"*

The authors draw on the well-established concept of **learning factories** from engineering education — small, realistic production environments that bridge theoretical knowledge and industrial practice (Abele et al. 2024). While most learning factories are physically large and centred on manufacturing, the BPA Lab is deliberately small-scale, with a stronger focus on **software systems and components** than on hardware.

6.2.3. The teaching concept

The proposed teaching concept extends the existing PBL format. Instead of working on isolated, simplified scenarios, student teams work on project tasks set in the **fictitious enterprise** represented by the BPA Lab — a customer-specific bicycle manufacturer with end-to-end processes for order processing, production, and shipping. The lab's process, decision, and data models together form a **"digital twin"** of this business scenario.

An illustrative example given in the paper: a project task to design and implement a **shipment process for special goods**, which requires integrating a software component of the BPA Lab dealing with shipment planning (using an external API for route planning) and integrating a **software-hardware module** — the Fischertechnik warehouse robot — to realise the connection with a warehouse management system.

Organisationally, milestone meetings with a "fictitious client" who asks critical questions and gives constructive feedback are incorporated into the existing PBL structure.

6.2.4. Preliminary evaluation

The evaluation was performed in **two phases**.

Phase 1 — Interviews with bachelor's students. Students from two previous course cycles were presented with the proposed teaching concept and asked to imagine integrating the learning factory into their project. They reacted very positively, using terms like *"very important", "inspiring", "exciting"* and *"very interesting"*. They emphasised that the lab could make BPA

"less abstract and more tangible", deepen understanding through hands-on experience, and foster transfer of learning and motivation. They also identified several requirements and concerns: clear instructions and materials, sufficient teaching support, concerns about hardware reliability, and the importance of integrating the lab thoughtfully into existing modules to avoid overload.

Phase 2 — Pilot project with master's students. A project simulation was carried out with **eight master's students** from the *Digital Sciences* programme. Compared to bachelor's students, the participants had similar BPM knowledge but stronger technical skills and software-development experience. They were asked to solve a project task and assess — from the perspective of bachelor's students — whether the concept could realistically be implemented at that level.

Overall, students viewed the concept's opportunities as **positive**: the practical experience of working with real BPM tools, the learning-by-doing approach, and collaborative work on specific challenges were all valued. At the same time, the evaluation revealed **substantial challenges**: complexity of installing and configuring the lab's components, individual problems with hardware/software/network setup, and difficulties in dealing with certain BPMS concepts. Students requested both **lab-specific documentation** and **basic background material** on the underlying technologies.

6.2.5. Conclusions and consequences for teaching

The authors arrive at a **nuanced conclusion**: the teaching concept as evaluated is **not directly applicable** to the bachelor's course at TH Köln with ~100 participants. The complexity of using the full BPA Lab — combining software development, network and hardware setup — exceeds what can reasonably be supported in a large bachelor's course.

Two practical consequences follow:

1. **For the existing bachelor's course:** the BPA Lab can serve as a **realistic project background** that is demonstrated at the start of the project. The technical project task itself, however, should only be extended carefully — in particular, **integration of hardware components requires more prior knowledge and support** than is feasible. **Pure software components** of the BPA Lab are a more realistic extension.

2. **For future courses:** the lab has potential as the basis of **new elective course formats**, which by design can cover a wider range of learning objectives — from BPM and BPA through to IoT and Cyber-Physical Systems — and allow for a higher student workload.

For BPM educators in general, the paper provides early evidence on the *opportunities*, *challenges*, and *requirements* of using a small-scale learning factory in a BPA course — an approach that is interesting but demanding, both for students and for teaching staff.

6.2.6. Funding

The underlying teaching project *"Enhancing teaching in the field of business process automation through a learning factory"* was supported by the **Foundation for Innovation in Higher Education** as part of the **REDiEE** project at TH Köln.

6.2.7. Further reading

- Full paper: 10.57684/COS-1326 (open access, CC BY-NC-ND)
- TH Köln SoTL programme: https://www.th-koeln.de/hochschule/scholarship-of-teaching-and-learning-sotl_115776.php
- Series *Forschung und Innovation in der Hochschulbildung* (FIHB), Cologne Open Science

6.3. System Architecture for a Modular BPA Learning Factory

Author: J. P. Iyebi Itomb · **Supervisor:** Prof. Dr. Matthias Zapp · **Second supervisor:** Prof. Dr. Christian Kohls **Type:** Master's thesis, Digital Sciences – Software Architecture, TH Köln · **Submitted:** October 2023

6.3.1. Contribution to the BPA Lab

This thesis developed and implemented a **modular system architecture** for the BPA Lab, enabling multiple student groups to work independently on different aspects of process automation and analysis without interfering with each other. Concretely, the work contributes:

- A **Domain-Driven Design (DDD)-based decomposition** of the BPA Lab scenario into seven business domains — Order Entry, Order Management, Logistics, Shipment, Production Execution, Purchasing, and Shopfloor — each modelled as an independent Bounded Context with its own BPMN process model.
- A **three-layer system architecture** (documented as C4 context, container, and component diagrams) comprising a control layer for IoT/hardware control, a communication layer (MQTT broker and distribution application), and a business process layer with domain-specific Camunda 8 applications.
- **Migration from Camunda 7 to Camunda 8 Self-Managed**, including a Docker Compose-based deployment and a rewrite of the Job Worker implementation to communicate with the Zeebe engine via gRPC.
- **Event-driven inter-domain communication** using BPMN message events and Camunda Webhook connectors, enabling process domains to exchange control signals (e.g., triggering production planning from order management) while remaining loosely coupled.
- **Containerised Job Workers** grouped by domain, reducing the setup effort for students to Docker Desktop only.

The evaluation confirms that the architecture effectively supports modular, independent development in teaching projects. Process analytics was identified as a direction for further development, with Camunda 8's Optimize component providing REST access to process data for future process mining integration.

6.4. Designing and Implementing a Data Architecture for the BPA Lab

Author: D. Gonzalez · **Supervisor:** Prof. Dr. Matthias Zapp · **Second supervisor:** Prof. Dr. Dietlind Zühlke **Type:** Master's thesis, Digital Sciences, TH Köln · **Submitted:**

February 2025

6.4.1. Contribution to the BPA Lab

This thesis adds the **data architecture** to the BPA Lab that was previously missing: an architecture and running implementation that makes the heterogeneous data of the lab usable for end-to-end process analysis and monitoring. Concretely, the work contributes:

- A **data architecture based on data virtualisation** using **Trino** as the central distributed query engine — heterogeneous sources (MySQL with order data, Elasticsearch with BPMS event data, FT-factory IoT data) are integrated virtually, allowing combined real-time and historical SQL queries without a separate physical data warehouse.
- An **IoT-data pipeline** that forwards MQTT messages from the Fischertechnik factory (environmental sensors, machine status, warehouse occupancy) to Elasticsearch via **Filebeat**, making them queryable alongside process data.
- **Grafana dashboards** integrated into the lab:
  - a **tactical end-to-end dashboard** with process KPIs and visualisations across the full bicycle ordering process (intended for taking process-improvement decisions),
  - an **operational production dashboard** based on the FT-factory's IoT data, visualised in real time (intended for production-floor decisions).
- A **systematic requirements catalogue** for data architectures in Industry 4.0 contexts, mapped to the BPA Lab's constraints (small footprint, single host, no high-throughput or strict security requirements). This makes the rationale for design choices traceable and adaptable for follow-up projects.

6.5. Integrating IoT Data into Process Mining in an Industry 4.0 Environment

Author: B. Kücük · **Supervisor:** Prof. Dr. Matthias Zapp · **Second supervisor:** Prof. Dr. Dietlind Zühlke **Type:** Master's thesis, Digital Sciences, TH Köln · **Submitted:** May 2026

6.5.1. Contribution to the BPA Lab

This thesis adds a **proof-of-concept integration pipeline** for IoT-enhanced process mining to the BPA Lab. It connects the workflow event data recorded by Camunda/Zeebe with the MQTT-based IoT observations produced by the Fischertechnik factory layer, making physical factory evidence accessible to process mining tools. Concretely, the work contributes:

- A **consolidated requirements catalogue** for IoT-enhanced process mining in Industry 4.0 settings, derived from literature and the BPA Lab context, covering event log readiness, semantic interpretation, timestamp validation, case correlation, and traceability.
- A **staged hybrid integration approach** that preserves a clean workflow-only event log as a baseline while selectively incorporating IoT observations — either as contextual attributes enriching existing workflow events, or as IoT-derived event rows added to the process timeline.

- **Five complementary event log artefacts**: a cleaned baseline workflow event log, a structured IoT event log, an IoT-enriched workflow event log, a derived IoT event log, and a final combined event log containing both workflow and IoT-derived events with explicit source and correlation metadata.
- A **Python-based prototype pipeline** (implemented in Jupyter notebooks) that prepares, maps, and integrates workflow and MQTT data; documented in the thesis appendix for reuse in follow-up projects.
- An **evaluation in Celonis** comparing the workflow-only baseline with the combined event log, demonstrating that IoT data can make production-floor behavior visible and reveal mismatches between workflow execution and physical factory behavior — while also quantifying the limitations introduced by timestamp uncertainty, missing shared case identifiers, and rule-based case correlation.

The evaluation confirms that IoT-enhanced process mining is feasible within the BPA Lab setting and analytically useful, but requires careful semantic interpretation and transparent handling of correlation uncertainty. The thesis builds directly on the data architecture established in the Gonzalez 2025 thesis and provides a foundation for future live-monitoring extensions of the lab.

6.6. Human-Centred Design of Form-Based User Interfaces in Camunda 8

Author: J. Will · **Supervisor:** Prof. Dr. Matthias Zapp · **Second supervisor:** Prof. Dr. Raphaela Groten **Type:** Bachelor's thesis, Business Information Systems, TH Köln · **Submitted:** January 2026

6.6.1. Contribution to the BPA Lab

This thesis delivered a **redesigned set of user-task forms** for the BPA Lab demonstration factory, replacing the previously rudimentary Camunda forms with usability-tested versions. Concretely, the work contributes:

- A **consistent layout template** — TH Köln header with form title and job role, plus colour-coded message blocks (blue for guidance, red/orange/green for error/warning/confirmation) implemented as reusable HTML views.
- **BPM-specific usability guidelines** distilled from Nielsen's heuristics, Norman's design principles, and ISO 9241-11/210 — directly applicable to future student projects using Camunda forms.
- **Redesigned or newly created forms** covering the full bicycle-manufacturer scenario, including four entirely new forms (*Confirm Supply*, *Quality Control*, *Missing Workpiece*, *Factory Failure*) that close previously empty user-task slots.
- Small **technical extensions**, e.g. a connector now retrieves warehouse coordinates dynamically rather than hard-coding them in the form.
- An **empirical baseline** from a usability test with five participants (8 forms, 40 runs, mixed-methods: think-aloud, completion time, success rate, Likert-scale questionnaire) that future revisions can be measured against.

Get involved

Companies and researchers: If you are interested in collaborating with the BPA Lab — whether as an industry partner contributing real-world scenarios, or as a researcher exploring joint projects in business process automation an analytics, or related fields — we welcome your enquiry. Please contact Prof. Dr. Matthias Zapp.
Students: If you would like to conduct your bachelor's or master's thesis, or a semester project, in the context of the BPA Lab, please reach out as well. Typical topics include extending the lab's software or hardware components, applying BPA methods and technologies, or applying process analytics to the lab's data.

7. Operational Instructions

7.1. Purpose of this chapter

This chapter provides selected **operational instructions** for running, configuring, and monitoring the BPA Lab demonstration factory. While the preceding chapters describe *what* the BPA Lab is, this chapter describes *how* to configure anbd use it. The instructions are not complete but rather focus on selected aspects that are particularly relevant for the demonstration factory.

The instructions are organised into four sections:

1. **Working with the productive environment** — how to prepare and connect the physical model factory before starting the process applications.
2. **Running an end-to-end process instance** — a step-by-step walkthrough of one full order-to-shipment process from the user perspective.
3. **First-time configuration of the data architecture** — one-time setup of Elasticsearch and Grafana that is required after an initial installation or reinstallation of the system to enable the data-driven process analysis.
4. **Using the data architecture for process analysis and monitoring** — overview of the data architecture and how to access the Grafana dashboards for data-driven process analysis.

7.2. Installing and running the solution

The instructions to install and run the BPA Lab solution (software only) are described in the README of the bpa_lab_demonstration_factory repository.

The following instructions in this chapter are building upon those.

7.3. Working with the productive environment of the model factory

The *productive environment* refers to the setup in which the process applications are connected to the **physical** Fischertechnik factory at the Gummersbach campus via the MQTT broker. The steps below should be carried out **before** the process applications (especially the manufacturing container) are started — otherwise containers may need to be restarted.

7.3.1. Preparations

1. First, switch on the power strip for the **TP-Link (WLAN router)** and the **Raspberry Pi** (which runs the MQTT broker).

2. After a few seconds, switch on the power strip for the **factory**.

3. *(Optionally)* Connect the power cable for the **warehouse robot** and switch it on via the TXT controller.

4. After a short time, check on both TXT controllers whether they have automatically connected to the local network of the BPA Lab.

   - **TXT controller of the warehouse robot:** use the touchscreen to navigate to **Settings → Network → WLAN-Setup**. The IP address `10.0.0.12` should be displayed if the warehouse robot is connected correctly.
   - **TXT controller of the factory:** use the touchscreen to navigate to **Settings → Network**. The item **Wi-Fi** should display **Connected to BPALabLocal**. If this is the case, the factory is correctly connected to the local network.

5. Load and start the program `ThKBpaLabFactory.py` on the TXT controller of the factory.

   Use the touchscreen of the TXT controller and go to **File**. Click on the folder `ThKBpaLabFactory`, then on the file `ThKBpaLabFactory.py`. Afterwards, click **Load**. The program can then be started via the red button on the touchscreen showing `ThKBpaLabFactory`.

   The program must run **continuously** during execution and should not be switched on and off. If it has to be stopped and restarted, follow the instructions under *Factory does not send MQTT messages* below.

6. Connect the workstation computer to the BPA Lab network:

   - **5a. Using the BPA Lab computer:** the computer must be **connected to the local BPA Lab network via WLAN** and must be given **internet access via a LAN connection**. Use the LAN connection box `A07`.

     The LAN connection must be activated individually for each computer. At the moment, the LAN connection only works with the BPA Lab computer.

   - **5b. Using another computer:** two WLAN adapters are recommended. One connects to the **local BPA Lab network**, the second one provides **internet access** (e.g. eduroam).

Only after these preparations are complete may the process applications be started (see the README of the demonstration-factory repository).

7.3.2. Known errors and their solution

7.3.2.1. Factory does not send MQTT messages

There may be a problem in the MQTT communication between the `FactoryMainBPALabThKoeln.py` program running on the TXT controller of the factory and the Docker container `bpa_lab_manufacturing_process`.

This error can occur if, for example, the program `FactoryMainBPALabThKoeln.py` is stopped and restarted on the TXT controller. This breaks the connection between the program and the Docker container.

> Solution
>
> **Restart the Docker container `bpa_lab_manufacturing_process`** after restarting the TXT program.

7.4. User guide for end-to-end process execution

This guide walks through a full end-to-end process instance with production. The actual scenario may vary depending on the input parameters — for example, if the requested products are already in stock, the production step may be skipped. Please also pay attention to the remarks and explanations rendered in each user form.

> Test versus productive environment
>
> The walkthrough applies to both the **test environment** (no physical factory) and the **productive environment** (connected to the physical model factory via MQTT). The most important difference is highlighted in step 4 below.

0. *(Productive environment only)* If a connection to the physical model factory via the MQTT broker is required, follow the preparations described in *Working with the productive environment of the model factory* above.

1. Start a new instance of the **`BPALabBikeFactoryOrderManagement`** process via the Camunda **Tasklist** (http://localhost:8082, *processes* view). This process orchestrates the full end-to-end scenario and triggers the other processes as needed. Other processes can also be started separately for testing purposes.

> Cleanup recommendation
>
> Regularly clean up old process instances. Use **Operate** at http://localhost:8081 to inspect and cancel running instances.

2. In the Camunda Tasklist, click on the user task **“choose bikes”**. If the task is not yet assigned, click on **“Assign to me”**. Enter the required parameters and complete the user task by clicking **“Complete Task”**.

3. Wait a few seconds, then execute the next user task (… to be continued)

> **Status of documentation**
>
> Update of user guide for the new release is required.

The end-to-end process instance should now be completed, and Operate should no longer show any running instances.

7.5. First-time configuration of the data architecture

The following configurations have to be carried out **once** for the data architecture to function correctly. They are required in the following situations:

- the system is being installed and executed on a computer for the first time,
- the Docker volumes of the system have been deleted (this also includes a complete reinstallation of Docker — note that merely deleting the images and reinstalling them does **not** require this configuration).

The configuration has two parts: Elasticsearch (via Kibana) and Grafana.

7.5.1. Configurations in Elasticsearch via Kibana

These configurations ensure that the timestamp fields of the event data recorded by Camunda are correctly mapped into the virtual tables via Trino.

1. After starting the system via Docker Compose (see the project README), Kibana becomes available at http://localhost:5601 after some time.

2. Open Kibana and enter **Index Management** in the search bar, then click on it.

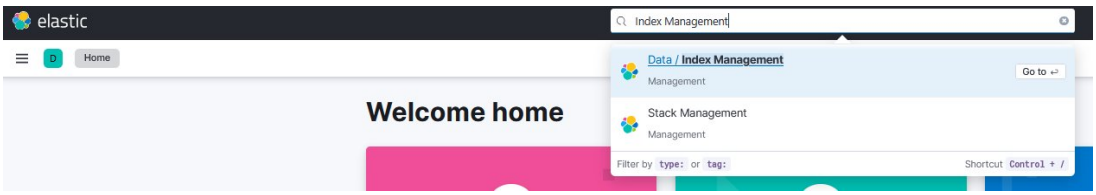


Figure 7.1.: Search for *Index Management* in Kibana

3. Click on the **Index Templates** tab.

4. Enter `operate-flownode` in the search box and click the corresponding index template.

5. A side window opens. Click on **Manage**, then on **Edit**.

6. Click on **Mappings**.

7. Enter `date` in the search box and edit the `endDate` field.

8. A side window opens. **Disable** the *Set format* toggle and click **Update**.

9. Repeat the previous step for the `startDate` field.

10. After adjusting both `endDate` and `startDate`, click **Review template**, then **Save template** to apply the changes.

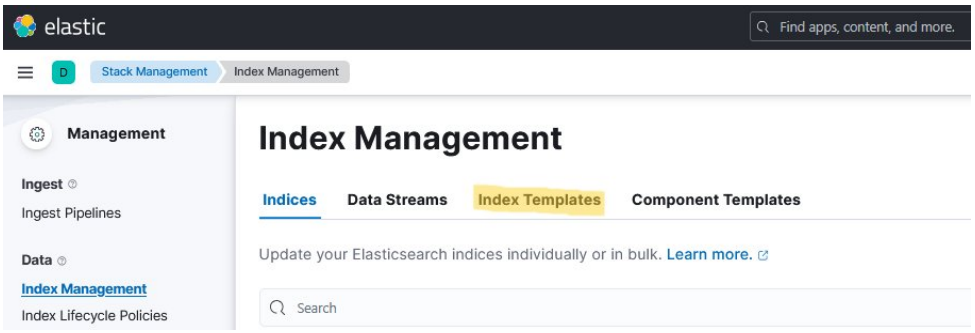


Figure 7.2.: Open the *Index Templates* tab

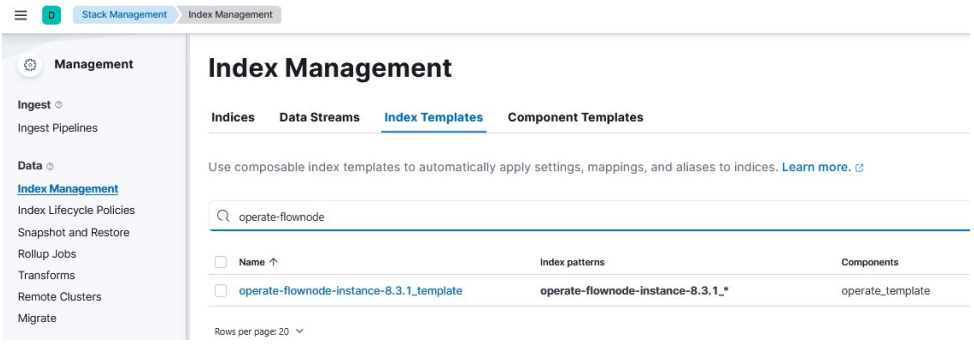


Figure 7.3.: Search for `operate-flownode`

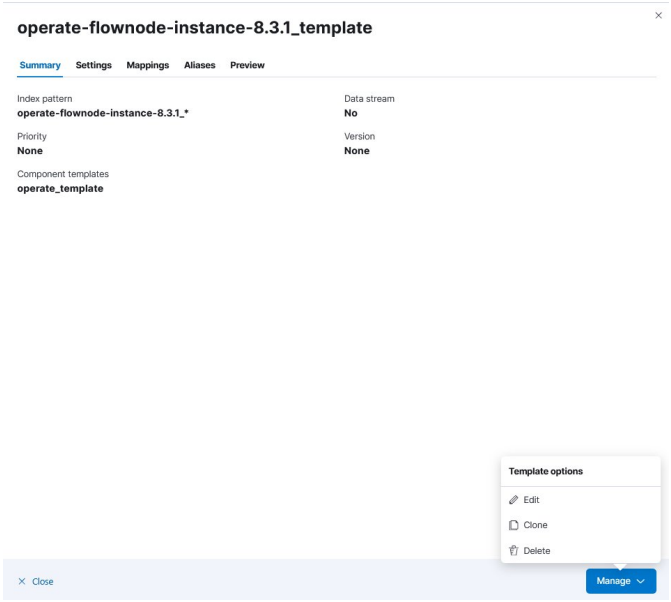


Figure 7.4.: Open the template for editing via *Manage* → *Edit*

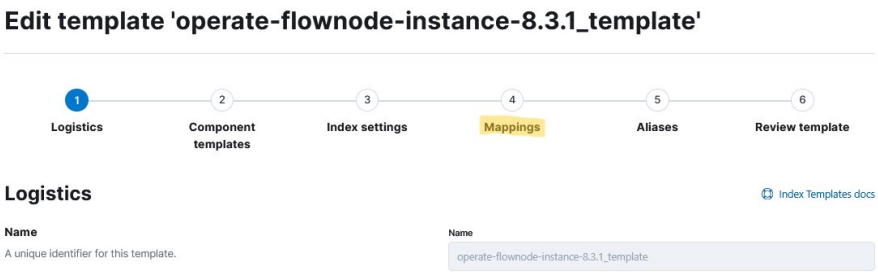


Figure 7.5.: Switch to the *Mappings* step

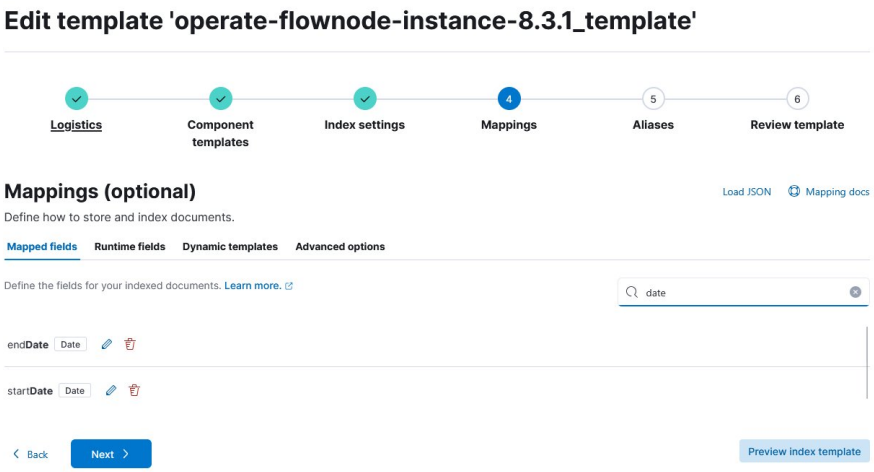


Figure 7.6.: Search for `date` and edit the date fields

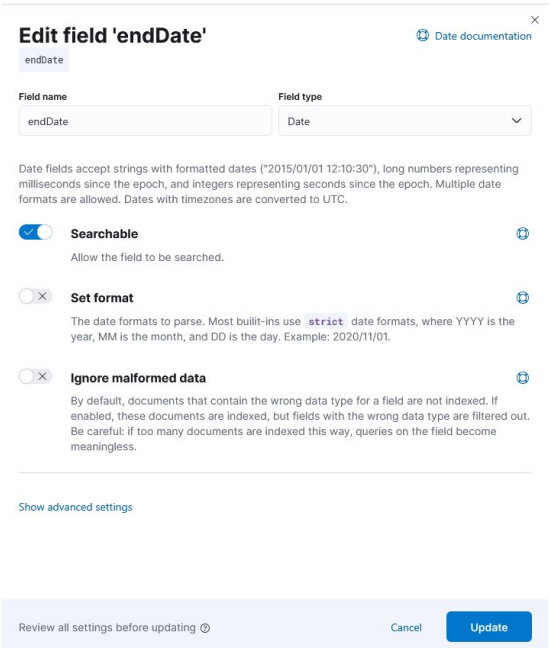


Figure 7.7.: Disable *Set format* and click *Update*

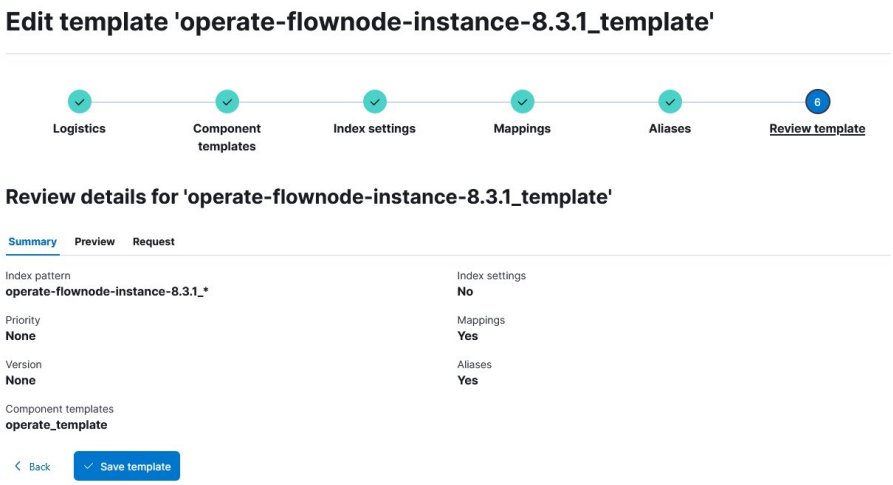


Figure 7.8.: Review and save the updated template

11. After customising the index template, the index itself must be **deleted** so that a new index is automatically created based on the updated template during operation. Click on the **Indices** tab, search for `operate-flownode`, and click on it.

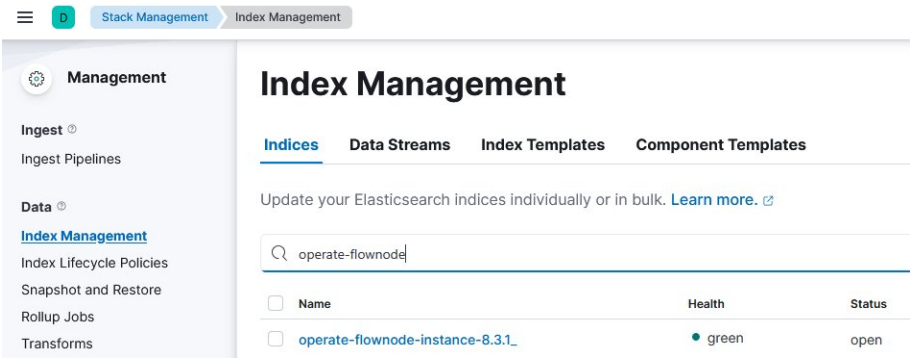


Figure 7.9.: Return to *Indices* and search for `operate-flownode`

12. A side window opens. Click on **Manage**, then on **Delete index**.

The Elasticsearch configuration is now complete. Kibana can be closed.

7.5.2. Configurations in Grafana

After configuring Elasticsearch, the remaining setup happens in Grafana so that the existing dashboards can be used.

1. Open Grafana at http://localhost:3008.

2. Log in with username **`admin`** and password **`admin`**.

3. A *Update your password* screen appears. Click **Skip** to continue without changing the password.

4. Open the side menu in the upper-left corner. Click on **Connections**, then on **Add new connection**.

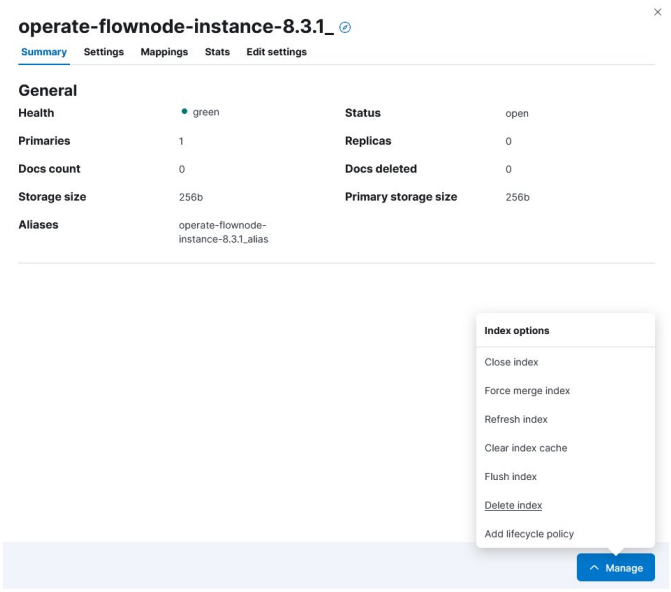


Figure 7.10.: Delete the existing `operate-flownode` index

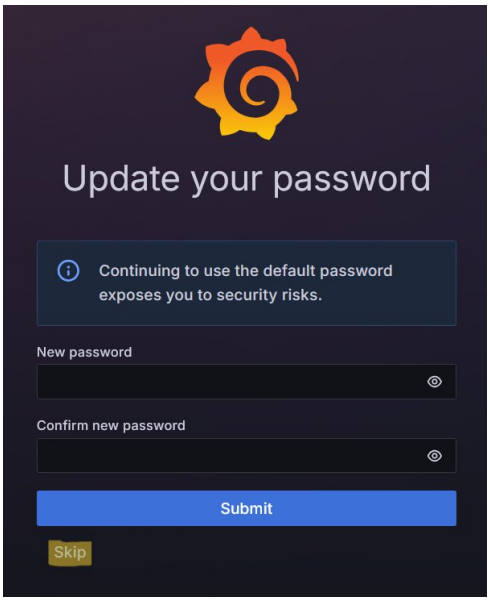


Figure 7.11.: Skip the password update

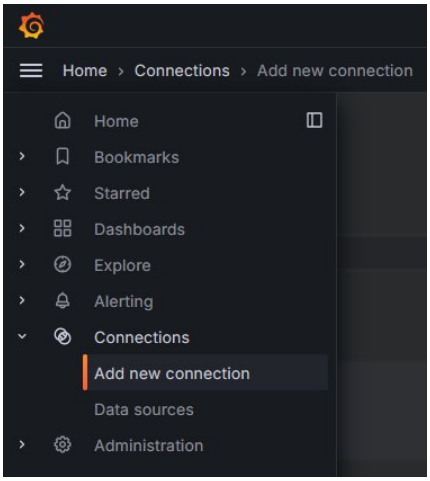


Figure 7.12.: Open *Connections → Add new connection*

5. Search for `Trino` in the search bar and click on the result.

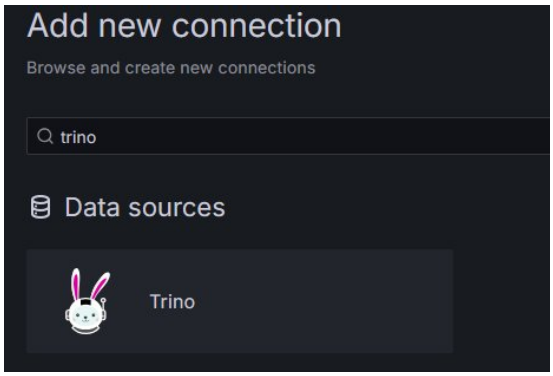


Figure 7.13.: Search for the Trino connector

6. Click **Install** in the upper-right corner.

7. After installation, open the side menu again and navigate to **Connections → Data sources**.

8. Click **Add new data source** in the upper-right corner.

9. Search for `Trino` and click on it.

10. Configure the Trino data source: set **URL** to `http://trino:8080` and click **Save & test**. The connection should be established successfully.

11. Open the side menu and click **Dashboards**.

12. Click **New → Import**.

13. Import one of the dashboards located in the `grafana_dashboards` folder of your locally cloned `bpa_lab_demonstration_factory` project. The dashboards must be imported **individually**.

14. When opening a freshly imported dashboard, no data appears at first and each panel shows an error icon. To resolve this, enter **edit mode**, hover over a panel, click the **menu (three dots)**, and select **Edit**.

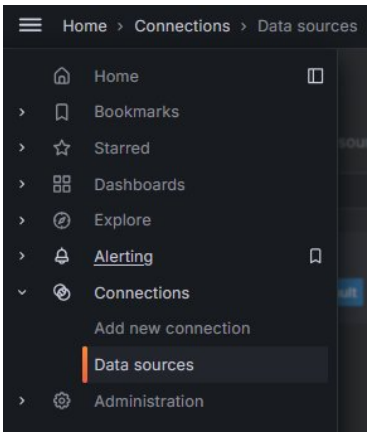


Figure 7.14.: Open *Connections* → *Data sources*

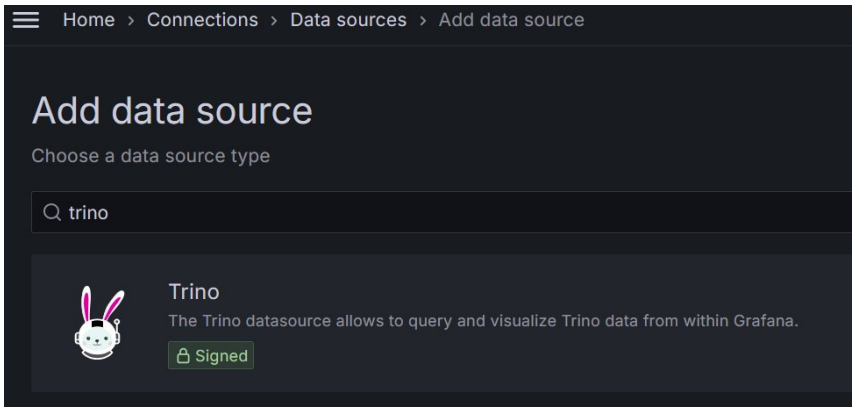


Figure 7.15.: Add a new Trino data source

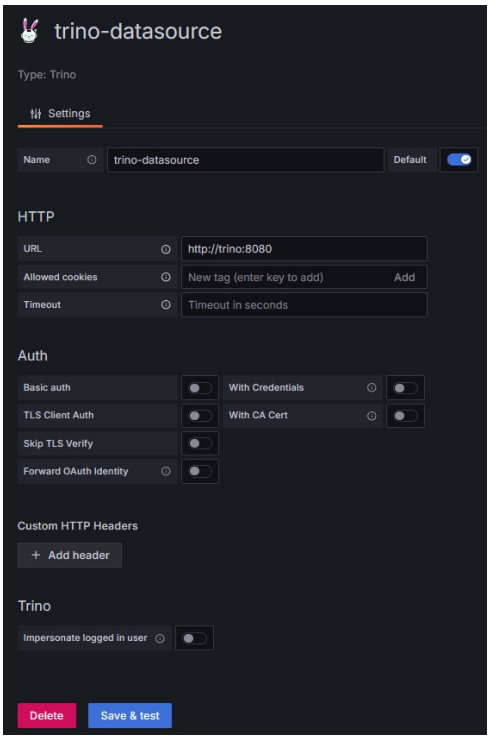


Figure 7.16.: Configure the Trino data source

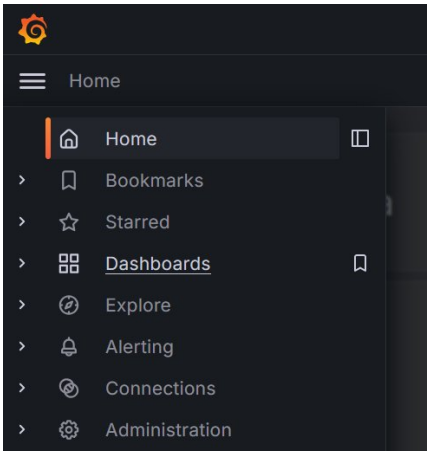


Figure 7.17.: Open *Dashboards* in the side menu

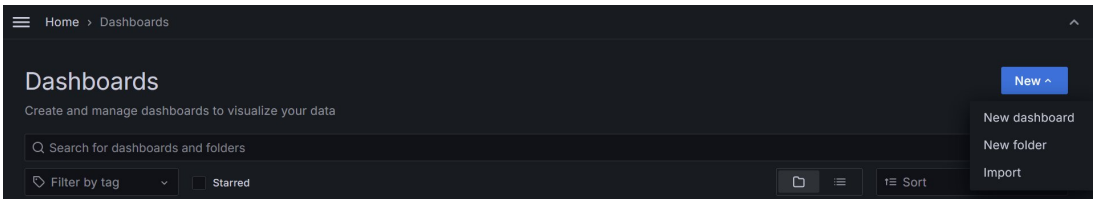


Figure 7.18.: Import a dashboard

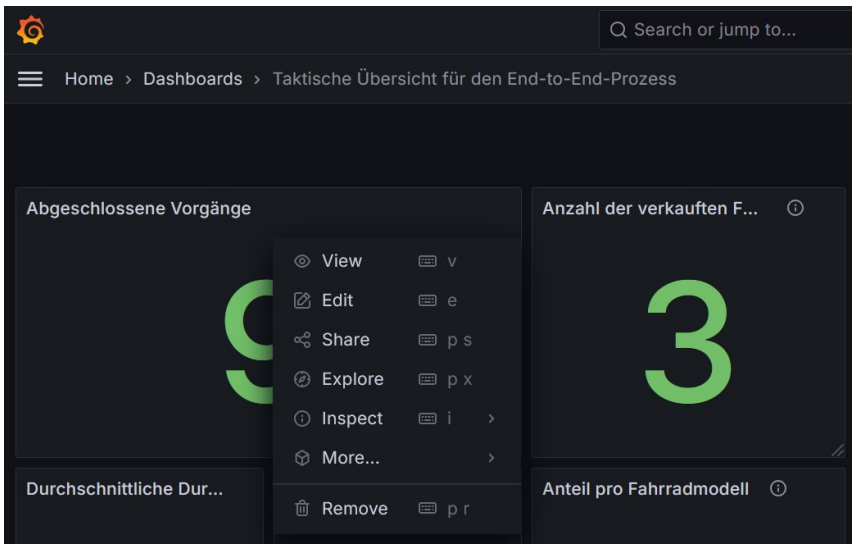


Figure 7.19.: Edit a panel via the panel menu

15. Click **Refresh** — the error icon disappears. Each panel of each dashboard must be refreshed once in this manner.

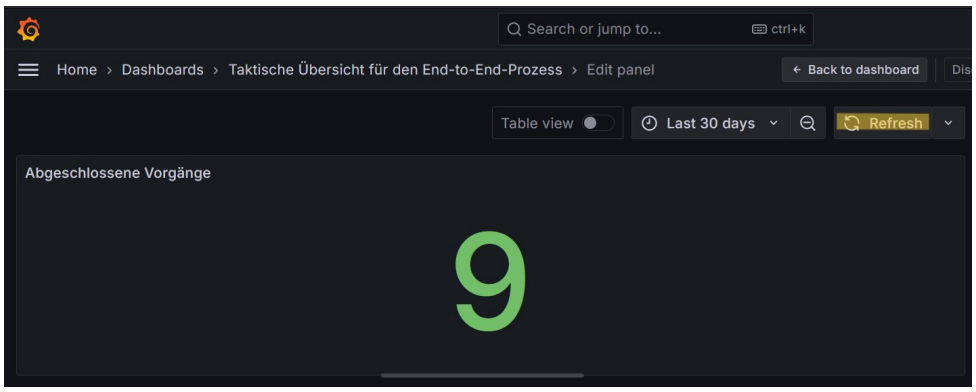


Figure 7.20.: Refresh the panel

The data architecture and dashboards should now be fully configured and ready to use.

7.6. Using the data architecture for process analysis and monitoring

7.6.1. Data architecture

The data architecture combines three source systems — process applications (MySQL), the BPMS (Elasticsearch), and the FT factory (MQTT broker) — into a unified, queryable layer based on data virtualisation with Trino. Grafana sits on top and provides the dashboards.

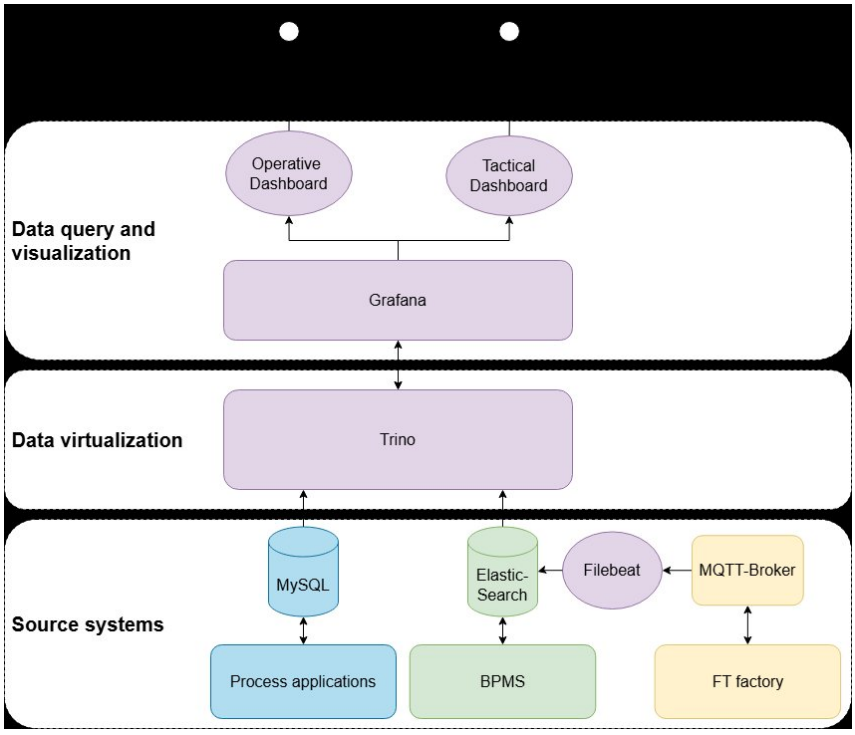


Figure 7.21.: Prototype of the BPA Lab data architecture

7.6.2. Overview of the available data

The factory publishes a variety of MQTT messages on different topics. A subset of these topics is currently subscribed to by **Filebeat** (see `filebeat/filebeat.yml`) and stored in Elasticsearch:

- `bpalab/ftfactory/i/bme680`
- `bpalab/ftfactory/i/ldr`
- `bpalab/ftfactory/f/i/order`
- `bpalab/ftfactory/f/i/state/hbw`
- `bpalab/ftfactory/f/i/state/vgr`
- `bpalab/ftfactory/f/i/state/mpo`
- `bpalab/ftfactory/f/i/state/sld`

Example logs of the *manufacturing* and *stock in* scenarios are available in the sample data folder of the `bpa_lab_docs` repository.

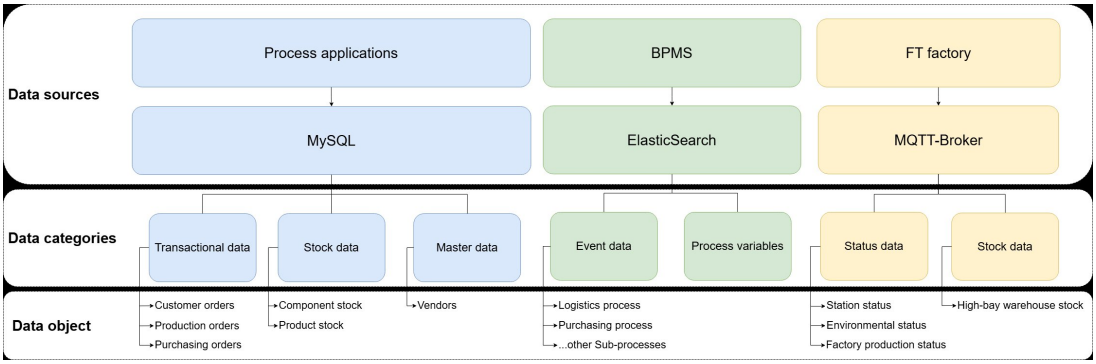


Figure 7.22.: Data sources, categories, and data objects in the BPA Lab

7.6.3. How to use Grafana

> **Warning**
>
> The instructions below assume that the *first-time configuration of the data architecture* (see previous section) has already been carried out. If not, perform that configuration first.

1. Once the Docker containers are running and have booted, open Grafana at http://localhost:3008.

2. Log in with username **`admin`** and password **`admin`**.

3. A *Update your password* screen appears. Click **Skip**.

4. Open the **Dashboards** area via the side menu. The dashboards should already be listed there. Click on any dashboard to open it.

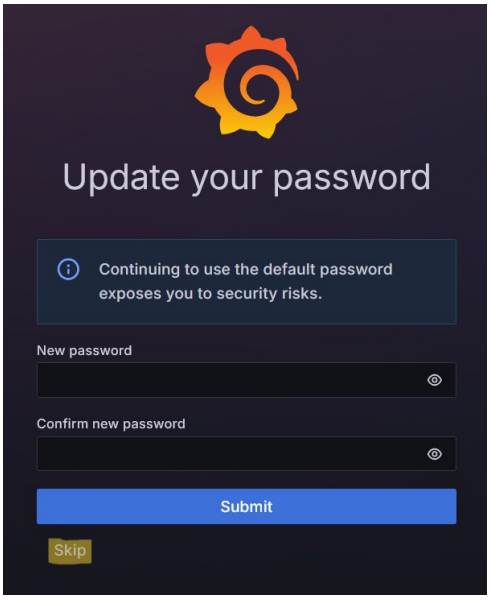


Figure 7.23.: Skip the password update

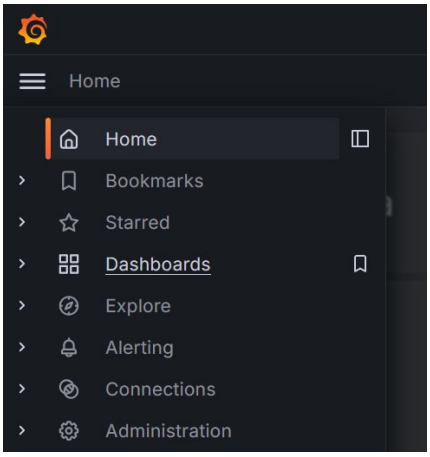


Figure 7.24.: Open the *Dashboards* area

A. Repositories and Components

This appendix gives an overview of the repositories that together make up the BPA Lab ecosystem, and lists the main hardware and software components.

A.1. Main repositories

A.1.1. bpa_lab_docs

https://github.com/BpaLabTHCologne/bpa_lab_docs

Role: former central documentation and architecture of the BPA Lab as a demonstration factory. The source of the content in this book.

Contains:

- overarching architecture diagrams,
- sample data of the Fischertechnik factory,
- BPMN process models (overview),
- C4 architecture diagrams,
- updated graphics.

A.1.2. bpa_lab_demonstration_factory

https://github.com/BpaLabTHCologne/bpa_lab_demonstration_factory

Role: current implementation of the demonstration factory. A complete, runnable setup based on Docker.

Contains:

- self-managed Camunda 8 setup,
- all job worker implementations (Java),
- BPMN and DMN models of the business processes,
- controller implementations (Python),
- docker compose for local deployment with docker.

A.1.3. bpa_lab_student_docs

https://github.com/BpaLabTHCologne/bpa_lab_student_docs

Role: documentation for student projects. Describes how individual components (e.g. the warehouse robot) can be used in self-developed process implementations.

Contains:

- per-component how-to guides,
- example projects,
- notes on setting up subsystems independently.

A.2. Hardware components

Component	Purpose	Control
Fischertechnik Industry 4.0 factory	Goods receipt and manufacturing	Python controller, MQTT
Fischertechnik warehouse robot	Storage and retrieval in the distribution warehouse	Python controller, MQTT
IoT devices	Sensor data and supplementary telemetry	Python controller, MQTT

A.3. Software components

Component	Technology	Main responsibility
Camunda 8 (self-managed)	BPMS with elasticsearch	Workflow engine and process orchestration
Job workers	mostly Java	Execution of service tasks
Controllers	Python	Hardware control
MQTT broker	MQTT broker	Asynchronous communication
MySQL	Relational database	Master data and stock data
Openrouteservice	External API	Distance and routing calculation in the shipment process

A.4. Further reading

As the BPA Lab is under continuous development, it is always worth checking the respective repository READMEs for the most current information.

B. Glossary

This appendix explains the technical terms used throughout the book.

ADR (Architecture Decision Record) Lightweight documentation format for recording architectural decisions in a structured way — context, decision, and rationale.

BPA (Business Process Automation) The use of technology to automate the execution of business processes.

BPMN (Business Process Model and Notation) A standard of the Object Management Group (OMG) for graphical modelling of business processes. Current version: 2.0.2 (Object Management Group 2014).

BPMS (Business Process Management System) A platform for modelling, executing, monitoring, and optimising business processes. The BPA Lab uses Camunda 8.

Camunda 8 A cloud-native BPMS built around the Zeebe workflow engine. The BPA Lab uses the *self-managed* variant (Camunda Services GmbH 2026).

Controller In the BPA Lab, a software component that directly controls hardware (Fischertechnik components, IoT devices). Predominantly implemented in Python.

C4 model A notation for visualising software architecture at four levels of abstraction: Context, Container, Component, Code. Introduced by Simon Brown — https://c4model.com/ (Brown 2018).

DDD (Domain-Driven Design) An approach to software development that places the business domain at the centre. The guiding principle for the modular decomposition of the BPA Lab (Evans 2003).

DMN (Decision Model and Notation) An OMG standard for modelling business decisions (Object Management Group 2024). Supported in Camunda 8 as a complement to BPMN.

End-to-end process A business process viewed from its triggering event through to its final business outcome — in the BPA Lab, from the arrival of a customer order to the delivery of a bicycle.

Fischertechnik Industry 4.0 factory A modular model of an Industry-4.0 factory by Fischertechnik. In the BPA Lab, it serves as the physical demonstration system.

gRPC A high-performance Remote Procedure Call framework originated at Google. In the BPA Lab, Camunda 8 uses gRPC to communicate with job workers.

Happy path The path through a process in which no errors, exceptions, or edge cases occur.

IoT (Internet of Things) The networking of physical devices with the internet and with each other.

Job worker In Camunda 8, an external application that picks up service tasks from the workflow and executes them. In the BPA Lab, job workers are implemented in Java.

MQTT (Message Queuing Telemetry Transport) A lightweight publish-subscribe protocol widely used in the IoT space. In the BPA Lab, MQTT is the communication protocol between job workers and controllers.

Openrouteservice A public API for routing and distance calculation based on OpenStreetMap data. Used in the BPA Lab's shipment process.

Process mining Data-driven analysis of business processes based on event logs from information systems. A central teaching and demonstration component in the BPA Lab (Aalst 2016).

Process application In the BPA Lab, a logical grouping of job workers and process models belonging to one business domain (e.g. *Order Management*).

Production order An order to manufacture a specific product. In the BPA Lab, one production order is created per product line in a customer order.

RPA (Robotic Process Automation) A technology to automate manual, repetitive tasks in existing applications via their user interface.

Workflow engine A software component that executes process models (e.g. expressed in BPMN), manages the state of process instances, and distributes work to humans or job workers.

XES (eXtensible Event Stream) An IEEE standard for event logs in process mining — a format for the exchange of process event data between systems.

C. References

Aalst, Wil M. P. van der. 2016. *Process Mining: Data Science in Action*. 2nd ed. Springer. https://doi.org/10.1007/978-3-662-49851-4.

Abele, Eberhard, Joachim Metternich, Michael Tisch, and Antonio Kreß. 2024. *Learning Factories: Featuring New Concepts, Guidelines, Worldwide Best-Practice Examples*. 2nd ed. Springer. https://doi.org/10.1007/978-3-031-46428-7.

Bandara, W., D. R. Chand, A. M. Chircu, et al. 2010. "Business Process Management Education in Academia: Status, Challenges, and Recommendations." *Communications of the Association for Information Systems* 27: 748–50. https://doi.org/10.17705/1CAIS.02741.

Barrows, Howard S. 1996. "Problem-Based Learning in Medicine and Beyond: A Brief Overview." *New Directions for Teaching and Learning* 1996 (68): 3–12. https://doi.org/10.1002/tl.37219966804.

Brown, Simon. 2018. *The C4 Model for Visualising Software Architecture*. Leanpub. https://c4model.com/.

Camunda Services GmbH. 2026. *Camunda 8 Documentation*. https://docs.camunda.io/.

Chow, W. 2021. "Teaching Business Process Management with a Flipped-Classroom and Problem-Based Learning Approach." *2021 International Conference on Engineering, Technology & Education (TALE)*. https://doi.org/10.1109/TALE52509.2021.9678885.

Coleman, K., D. Uzhegova, B. Blaher, and S. Arkoudis, eds. 2023. *The Educational Turn: Rethinking the Scholarship of Teaching and Learning in Higher Education*. Springer. https://doi.org/10.1007/978-981-19-8951-3.

Evans, Eric. 2003. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.

Nerdinger, F., G. Blickle, and N. Schaper. 2008. *Arbeits- Und Organisationspsychologie*. Springer Medizin. https://doi.org/10.1007/978-3-540-74705-5.

Object Management Group. 2014. *Business Process Model and Notation (BPMN), Version 2.0.2*. https://www.omg.org/spec/BPMN/.

Object Management Group. 2024. *Decision Model and Notation (DMN)*. https://www.omg.org/spec/DMN/.

Santos, S. C. dos, J. Vilela, and A. Vasconcelos. 2023. "Promoting Professional Competencies Through Interdisciplinary PBL: An Experience Report in Computing Higher Education." *2023*

Frontiers in Education Conference (FIE). https://doi.org/10.1109/FIE58773.2023.10343050.

Schaper, N. 2012. *Fachgutachten Zur Kompetenzorientierung in Studium Und Lehre*. Hochschulrektorenkonferenz, Projekt Nexus. https://www.hrk-nexus.de/fileadmin/redaktion/hrk-nexus/07-Downloads/07-02-Publikationen/fachgutachten_kompetenzorientierung.pdf.