Training Material - UPRC

4. Local Optimization

4.3 Setting up the Optimization Engine

Funded by
the European Union

Educational Goals of the Training

What will the user learn?

Contents

1. **Setting up the Optimization Engine**
   - 1.1 Optimization Service description
   - 1.2 Optimization Service setup
   - 1.3 Integration of Optimization Service with Digital Twins
   - 1.4 Input and Output Data Structure

1.1 Optimization Service description

Why We Use an Optimization Service?

In modern manufacturing, many problems—such as planning, scheduling, routing, or energy minimization—can be mathematically formulated as optimization problems.

These problems are:

- Complex and involve multiple constraints
- Require real-time decisions or high-frequency updates
- Involve trade-offs between cost, time, energy, or resource usage

What Is an Optimization Service?

An optimization service is a dedicated software component that receives input data (e.g., task parameters, constraints), runs an appropriate solver (e.g., heuristic, metaheuristic, IP), and returns the best solution according to a defined objective function.

This approach decouples the optimization logic from the rest of the system, offering:

- **Modularity** – reusable optimization logic
- **Scalability** – runs in the cloud or in parallel
- **Interoperability** – supports different solvers and data schemas

1.1 Optimization Service description

Relevant Applications from Training Material 4.2.1, 4.2.2

The use of optimization services has already been introduced in earlier training modules:

- **4.2.1 Pick-and-Place Optimization**
  Optimization service used to generate robot pick sequences using algorithms such as Nearest Neighbor, Q-learning, and Exact IP solvers.
- **4.2.2 Energy Optimization of Robot Movements**
  A service selects the best movement scenarios from a Digital Twin simulation using an Integer Programming model to minimize energy use within a time constraint.

These examples show how different manufacturing challenges can be tackled through flexible, solver-agnostic services, rather than hard-coded, rigid tools.

1.1 Optimization Service description
Optimization-as-a-Service (OaaS)

OaaS: A Modern Paradigm for Scalable Optimization

Optimization-as-a-Service (OaaS) is a cloud-based delivery model for solving optimization problems. Just like Software-as-a-Service (SaaS), OaaS abstracts the complexity of solving optimization problems and makes advanced solvers accessible via APIs.

Why OaaS?

- **Reusable:** Optimization models are implemented once and reused across multiple scenarios and workflows.
- **Scalable:** Solvers can scale to support many concurrent users and large datasets.
- **Decoupled:** Business applications do not need to embed optimization logic.
- **Accessible:** Solutions are exposed as simple, secure API endpoints.

1.1 Optimization Service description Optimization-as-a-Service (OaaS)

Historical Context and Evolution

- Early systems like NEOS Server and Optimization Services (OS) frameworks introduced the idea of exposing solvers over the internet.
- Evolved through cloud manufacturing, modular architectures, and simulation-based optimization platforms.
- Supported by the shift to cyber-physical systems and Industry 4.0 principles (e.g., interoperability, modularity, resilience).

Real-World Need

Industries need:

- Continuous, automated improvement
- Real-time adaptability
- Solutions that are not bound to one solver or one data schema

OaaS enables this by making optimization a plug-and-play component of industrial IT systems.

1.1 Optimization Service description
Optimization-as-a-Service (OaaS)

How OaaS Works:

1. **User Submits an Optimization Request**
   → Contains problem data, objective, and constraints (in a JSON format or other schema)
2. **The Optimization Engine Processes the Request**
   → Uses backend solvers (e.g., MILP, heuristics, metaheuristics) to compute the solution
3. **Solution is Returned to the Requesting System**
   → Result is sent back via API, stored, or visualized in connected tools

Industrial Benefits of OaaS:

- **Reusability**: One engine can serve many use cases (robot movement, logistics, energy, etc.)
- **Maintainability**: Updates or new algorithms can be added centrally
- **Cloud-ready**: Can be deployed on any infrastructure (on-prem, hybrid, or public cloud)
- **Secure and Isolated**: Handles sensitive data through APIs and encrypted channels

1.1 Optimization Service description
Optimization-as-a-Service (OaaS)

optEngine: A Real-World OaaS Implementation

Middleware Layer:

Connects front-end apps (robots, schedulers, DTs) with backend solvers

Asynchronous & Synchronous Support

- Async: Submit job, check status later
- Sync: Instant optimization for live execution

Used in MODAPTO Pilots:

- Robot task planning
- Energy-aware execution
- Scheduling alternatives

1.2 Optimization Service Setup
Service Setup Workflow

1. **User Input**
    1. JSON-based request (e.g., robot tasks, constraints, configuration ID)
    2. Sent via secure HTTPS POST to the API
2. **Job Creation**
    1. optEngine registers the optimization job
    2. Generates a **UUID** for tracking
    3. Stores metadata: user, timestamp, job route
3. **Queue Management**
    1. Job is routed to the correct optimization service via RabbitMQ
    2. Message queues decouple service execution from user requests
    3. Enables parallel and fault-tolerant execution
4. **Backend Solver Execution**
    1. Solver (e.g., MILP, metaheuristic, ML) processes the job
    2. OptimizationResult is created upon success
5. **Result Retrieval**
    1. User polls API or receives webhook callback
    2. Solution is returned in structured JSON format
    3. Includes: selected configuration, objective value, status flags

1.2 Optimization Service Setup
Service Setup

- **Technologies Used**
- **Java Spring Boot**: REST API logic
- **RabbitMQ**: Message queuing for async processing
- **PostgreSQL**: Job storage and result archiving
- **Docker**: Containerized deployment for scalability

Analysis of technologies in next slides.

- **Why This Architecture Works**
- **Asynchronous-first**: Efficient for heavy or long-running optimization
- **Schema-agnostic**: Compatible with various problem types (routing, scheduling, etc.)
- **Production-ready**: Already deployed in MODAPTO pilot use cases

1.2 Optimization Service Setup Tecnhologies

Java Spring Boot – REST API Logic

What it is:
Spring Boot is a Java-based framework designed to simplify the development of stand-alone, production-ready applications.

Role in optEngine:

- Manages all RESTful API endpoints.
- Handles job submission, status polling, and solution retrieval.
- Uses @RestController and @RequestMapping annotations for fast API setup.

Why it matters:

- Rapid development and deployment of web services.
- Strong community and robust ecosystem (security, documentation, testing).
- Integrates easily with databases and message brokers.

1.2 Optimization Service Setup Tecnhologies

RabbitMQ – Message Queuing for Async Processing

What it is:
RabbitMQ is a widely used open-source message broker that supports various messaging protocols.

Role in optEngine:

- Decouples job submission from backend processing.
- Each optimization route has its own dedicated queue.
- Supports:
  - **Job Queues**: where optimization tasks are sent.
  - **Status Queues**: for job progress updates.
  - **Result Queues**: for completed solutions.

Why it matters:

- Enables load balancing and scalable distributed processing.
- Allows asynchronous job execution, ideal for long or complex optimization tasks.
- Provides fault-tolerance (jobs persist even if services restart).

1.2 Optimization Service Setup Tecnhologies

PostgreSQL – Job Storage and Result Archiving

What it is:
PostgreSQL is a powerful open-source relational database known for reliability and performance.

Role in optEngine:

- Stores:
  - User credentials and metadata
  - Job metadata (UUID, status, timestamps)
  - Optimization results in JSON format

Why it matters:

- Ensures data persistence across job executions.
- Enables auditing, historical analysis, and traceability.
- Supports JSONB fields for storing flexible, schema-less optimization data.

1.2 Optimization Service Setup Tecnhologies

Docker – Containerized Deployment for Scalability

What it is:
Docker is a containerization platform that allows software to run reliably across different environments.

Role in optEngine:

- Encapsulates the entire optimization engine and its dependencies.
- Containers are used for:
  - The REST API backend
  - RabbitMQ message broker
  - Database service
  - Solver instances

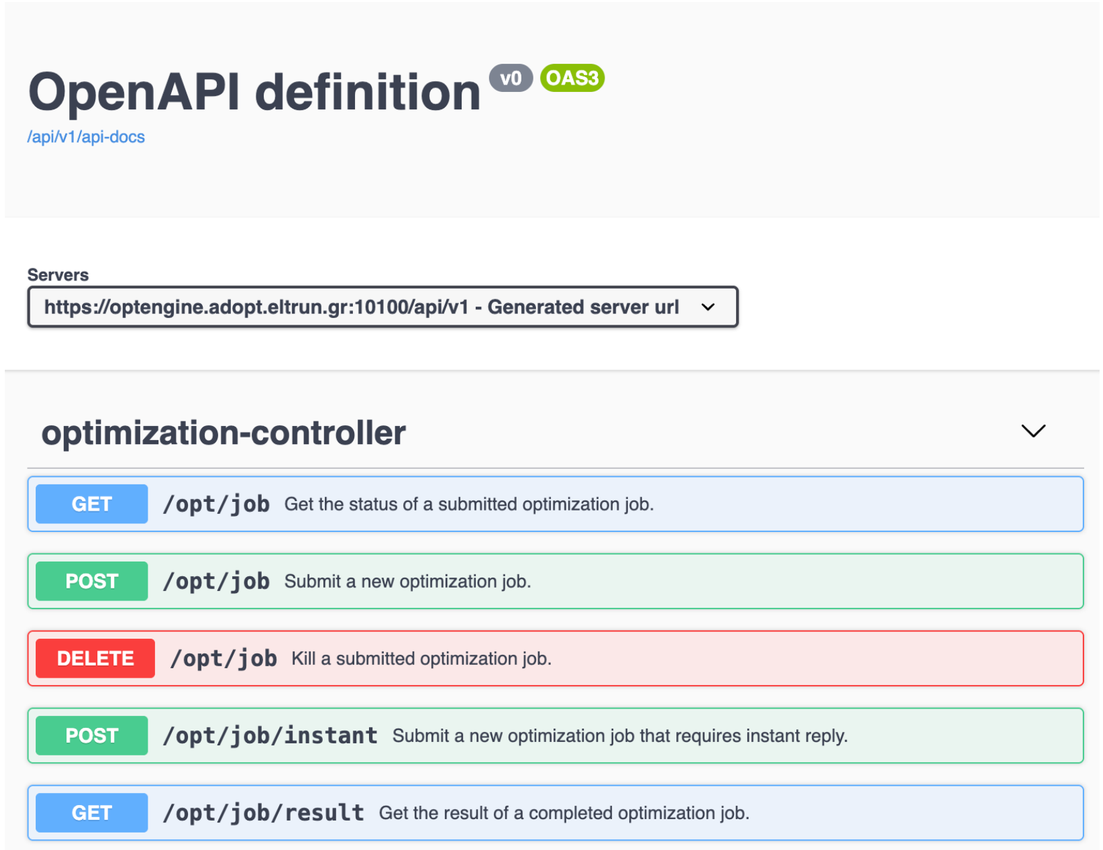
Why it matters:

- Simplifies deployment across local, cloud, or edge environments.
- Ensures consistent performance (no "it works on my machine" issues).
- Supports scalability and orchestration (e.g., using Docker Compose or Kubernetes).

1.2 Optimization Service Setup
Optimization API Overview – optEngine Swagger

What is Swagger / OpenAPI UI?

A web-based tool that documents and exposes the available endpoints of a web service, in our case the Optimization Engine (optEngine).



1.2 Optimization Service Setup
Optimization API Overview – optEngine Swagger

Submitting a New Job – /opt/job (POST)

- Used for standard optimization jobs
- Sends a JSON payload with problem data
- Returns a UUID for tracking the job

Checking Job Status – /opt/job (GET)

- Input: the job UUID
- Output: current status (PROCESSING, COMPLETE, etc.)

Retrieving Results – /opt/job/result (GET)

- Input: UUID
- Output: Full result structure with solution values (costs, configuration)

Synchronous Use – /opt/job/instant (POST)

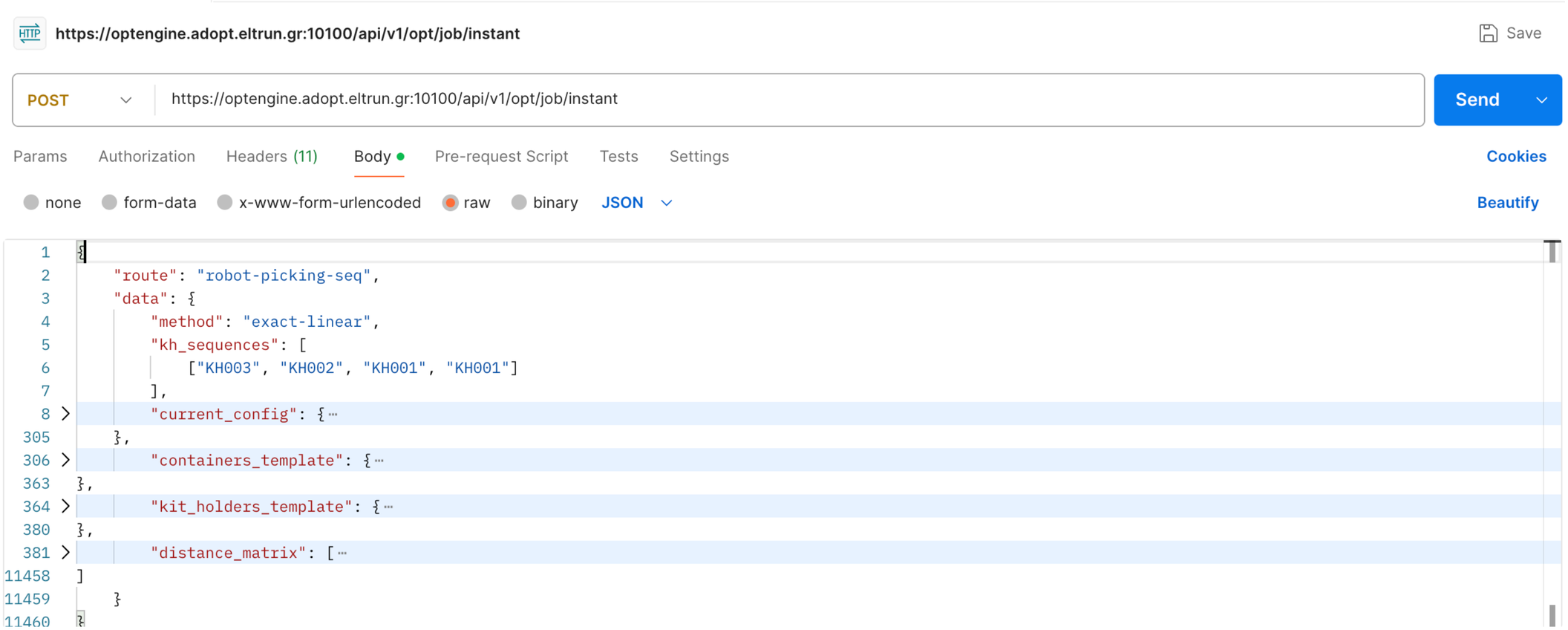
- Used when **real-time response** is needed
- Useful in simulations or production planning pipelines

Canceling a Job – /opt/job (DELETE)

- Terminates an optimization run
- Useful in case of bad input or user-side interruption

1.3 Integration of Optimization Service with Digital Twins

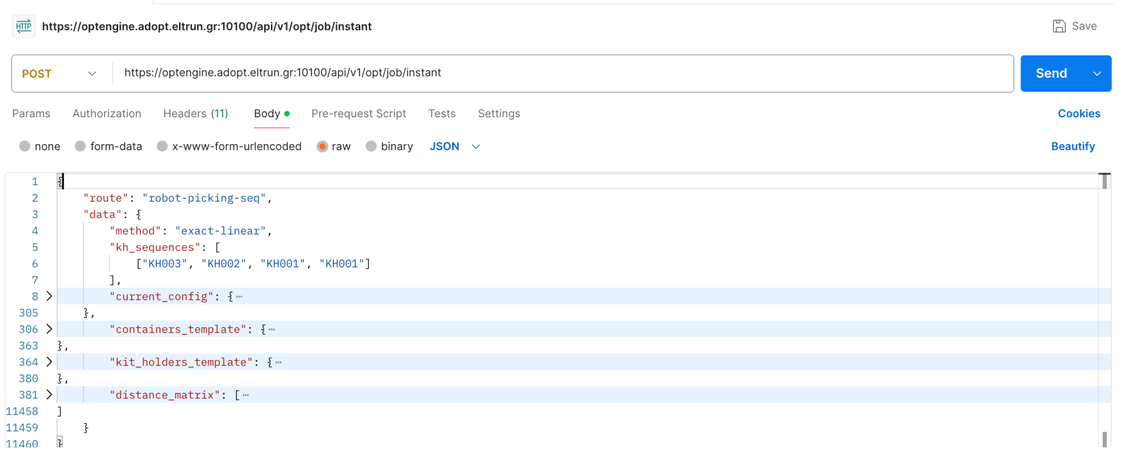
1.4 Input and Output Data Structure
Real Example — Using optEngine via Postman



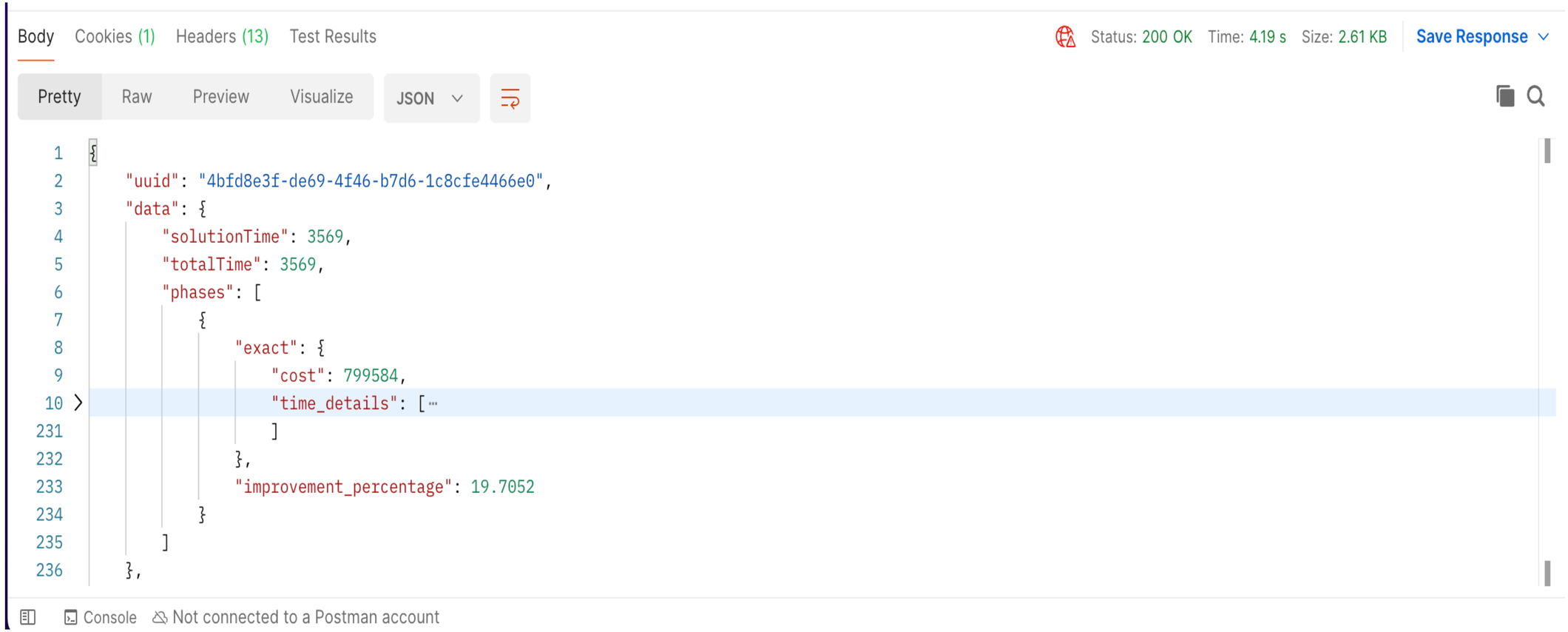
1.4 Input and Output Data Structure

Step 1: Submitting an Optimization Job
We send a **POST** request to /opt/job/instant for real-time optimization.
Route: "robot-picking-seq" indicates the use case — pick-and-place sequencing.
Method: "exact-linear" triggers a hybrid method combining the Exact and Linear approach.
Input data includes:

- kh_sequences: A list of Kit Holder (KH) sequences for the robot to follow.
- current_config, containers_template, kit_holders_template: Define the layout and expected container contents.
- distance_matrix: All travel times between robot movements (GR↔KH, etc.).

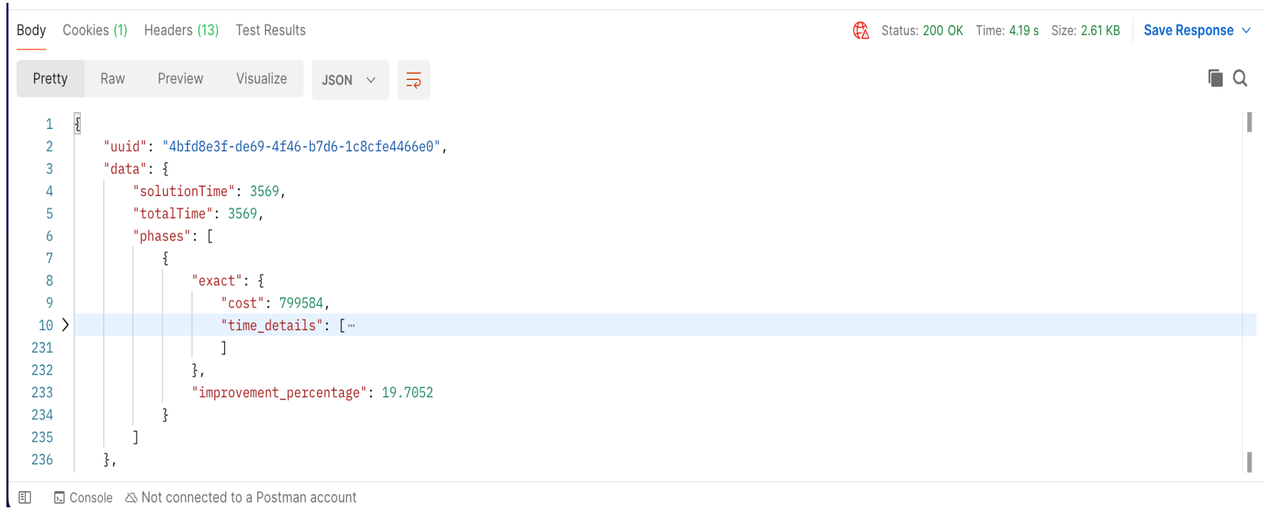


1.4 Input and Output Data Structure
Real Example — Using optEngine via Postman
Receiving the Optimization Result



1.4 Input and Output Data Structure
Real Example — Using optEngine via Postman
Receiving the Optimization Result

The result is returned instantly after computation:
- **uuid**: Unique identifier for tracking the result.
- **solutionTime**: Time (in ms) taken to compute the solution.
- **totalTime**: Total operation time across all movements.
- **phases**: Each robot movement batch with:
    - exact.cost: Total cost of the optimized path (e.g., total travel time).
    - time_details: Fine-grained movement logs.
    - improvement_percentage: Gain compared to baseline layout or method.



CONSORTIUM

meet the team

MOSAPTO

ATC
ATHENS TECHNOLOGY CENTER

Fraunhofer
IOSB

UNIVERSITÉ
DE LORRAINE

UNIVERSITY OF PIRAEUS
RESEARCH CENTER

THE E-BUSINESS RESEARCH CENTER
ELTRUN
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

EKS
InTec GmbH

BOC
www.boc-group.com

AEGIS
IT RESEARCH

LIC
LIETUVOS
INOVACIJŲ
CENTRAS

FFT

SEW
USOCOME

CRF

ILTAR
ITALBOX

Funded by
the European Union

CONTACT us

MODAPTO

modapto.eu

info@modapto.eu

MODAPTO Horizon Europe Project

@MODAPTO_eu

MODAPTO Horizon Europe Project

Funded by
the European Union