

# T Systems

## hackaTUM 2024 Connected Mobility Challenge

Garching bei München

November 22, 2024



# T-Systems at hackaTUM 2024



**Sebastian Lang**

Architect & Product Manager



**Michael Schafhauser**

Product Manager



**Michail Chatzipanagiotou**

DevOps Engineer

**T-Systems** is a part of  
**Deutsche Telekom**.  
We are big enough to  
**scale globally** and  
small enough to **care**.



**#1**  
**GERMAN  
BRAND**

**#1**  
**EUROPEAN  
BRAND**

**#9**  
**GLOBAL  
BRAND**

## About **Deutsche Telekom**

**1** of the **strongest brands** in the world

**> 50** countries

**112 B €** revenue in 2023

**291 M** customers

**> 1,000** global leading **partners**

**199,652** employees worldwide

# T-Systems as Telekom's B2B IT unit – Automotive

## Mission

- Deliver one-stop, best-in-class ICT solutions
- Extensive industry- and company-specific expertise due to taking on employees from a variety of organizations



**powered by #magenta as Deutsche Telekom subsidiary**

## #1 AUTOMOTIVE

Market Leader in Germany  
in the Automotive sector  
for over 10 years

## #1

IT Service  
provider in DACH

## > 4K

Automotive  
Experts Worldwide

## > 3K

Dealerships  
are our clients

## > 700K

Vehicles with  
OBD adapter

## 65%

of the top 20 OEMs,  
numerous multi-  
national Tier 1 supplier  
and manufacturer

## > 32M

Vehicles on  
Connected Car  
Platform

## > 5K

Automotive  
projects in 2022

## > 25 yrs

of Expertise  
in Automotive



# Automotive

Challenges and opportunities: today

Consumer  
Technology  
Regulation

Autonomous/  
automated



Connected



Electrified



Shared



Economics

Opportunity sizing and investments

Capabilities

Build-up and partnering

Outlook and predictions: 2030

>10% of cars sold will be used for driverless

**taxis fleets** (with a total of 26 mn autonomous vehicles in operation)



New cars

100%  
Electric

100%  
Connected  
and software-defined

50%  
Autonomous  
L2 / L2+



Up to ten thousand dollars

Lifetime revenues from subscriptions, upselling, and ecosystems



Net-zero

production; recycled content quotas; and modular / upgradable components





**Develop a solution to manage  
a fleet of robotaxis.**

# Goals of the challenge



Build an algorithm to commission a robotaxi fleet in the most effective/efficient way, i.e. optimize for different parameters such as kWh, utilization, revenues, etc.



Create a visualization layer to  
(1) manage the fleet and  
(2) create dashboards/reports.

# Robotaxi Fleet Management: the challenge in detail

- GIVEN:** A number of robo-taxis in the area München Zentrum  
A number of customers, expecting to use said robo-taxis to get from point A to point B
- WHERE:** A robo-taxi expects an input of a customer's information (current XY coordinates, destination XY coordinates), in order to commission it to serve said customer.
- THEN:** Develop a solution that will handle the distribution and commission order of your fleet

## Requirements:

- Your solution only knows customer's current and destination coordinates, which vehicles are ready to accept a commission, their position and other relevant metadata.
- Your solution operates autonomously based on the above data and handles the job of a coordinator (who goes where).
- Commission decisions are made based on two pillars:
  - 1) customer satisfaction (fast service, optimized routes, etc.)
  - 2) the environmental footprint of the operation (expected fuel consumption, total distance traveled, etc.)
- A UI of some sort must exist and should contain relevant information (# of vehicles currently handled, total number of commissions, errors, etc.)

# What we provide: API

- A simple API that handles the administration of mobility scenarios.

The screenshot shows a Swagger API documentation interface with three main sections: **customers**, **scenario**, and **vehicles**.

- customers**: The endpoints to get customer information and control customers
  - GET /customers/{customerId}** Get a customer
  - GET /scenarios/{scenarioId}/customers** Get all customers for a scenario
- scenario**: The endpoints to get scenario information, initialize and run a scenario
  - POST /scenario/create** Initialize a scenario
  - GET /scenarios** Get all scenarios
  - DELETE /scenarios/{scenarioId}** Delete a scenario
  - GET /scenarios/{scenarioId}** Get a scenario
- vehicles**: The endpoints to get vehicle information and control vehicles
  - GET /scenarios/{scenarioId}/vehicles** Get all vehicles for a scenario
  - GET /vehicles/{vehicleId}** Get a vehicle
  - POST /vehicles/commission/{vehicleId}/{customerId}** Commission a vehicle

# What we provide: API

- A python API that runs simulations for scenarios created with the backend API.

The screenshot shows the Swagger UI interface for the ScenarioRunner API version 1.0. The title is "ScenarioRunner API 1.0" with a "1.0" badge. Below it is the base URL "[ Base URL: / ]" and a link to "/swagger.json". A descriptive text states "A simple API for managing scenarios".

The main content is organized into sections:

- Scenarios** (Scenario operations):
  - GET** /Scenarios/get\_scenario/{scenario\_id}
  - POST** /Scenarios/initialize\_scenario
  - PUT** /Scenarios/update\_scenario/{scenario\_id}
- Runner** (Scenario runner operations):
  - POST** /Runner/launch\_scenario/{scenario\_id}

# What we provide: Administration UI

- A UI panel that lets you interact with the scenario administration backend.

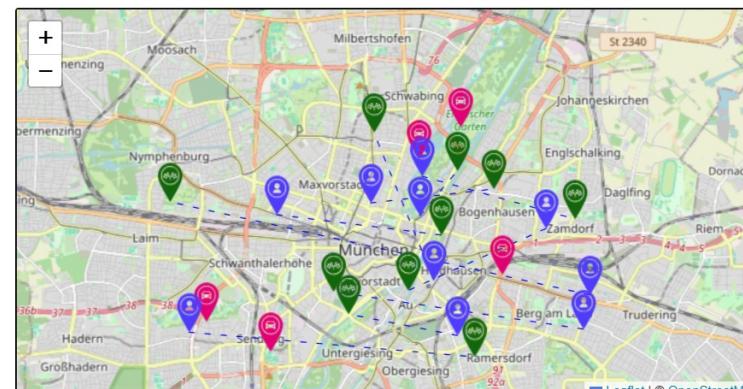
Home Scenario Administration

Number of Vehicles Number of Customers Create Scenario

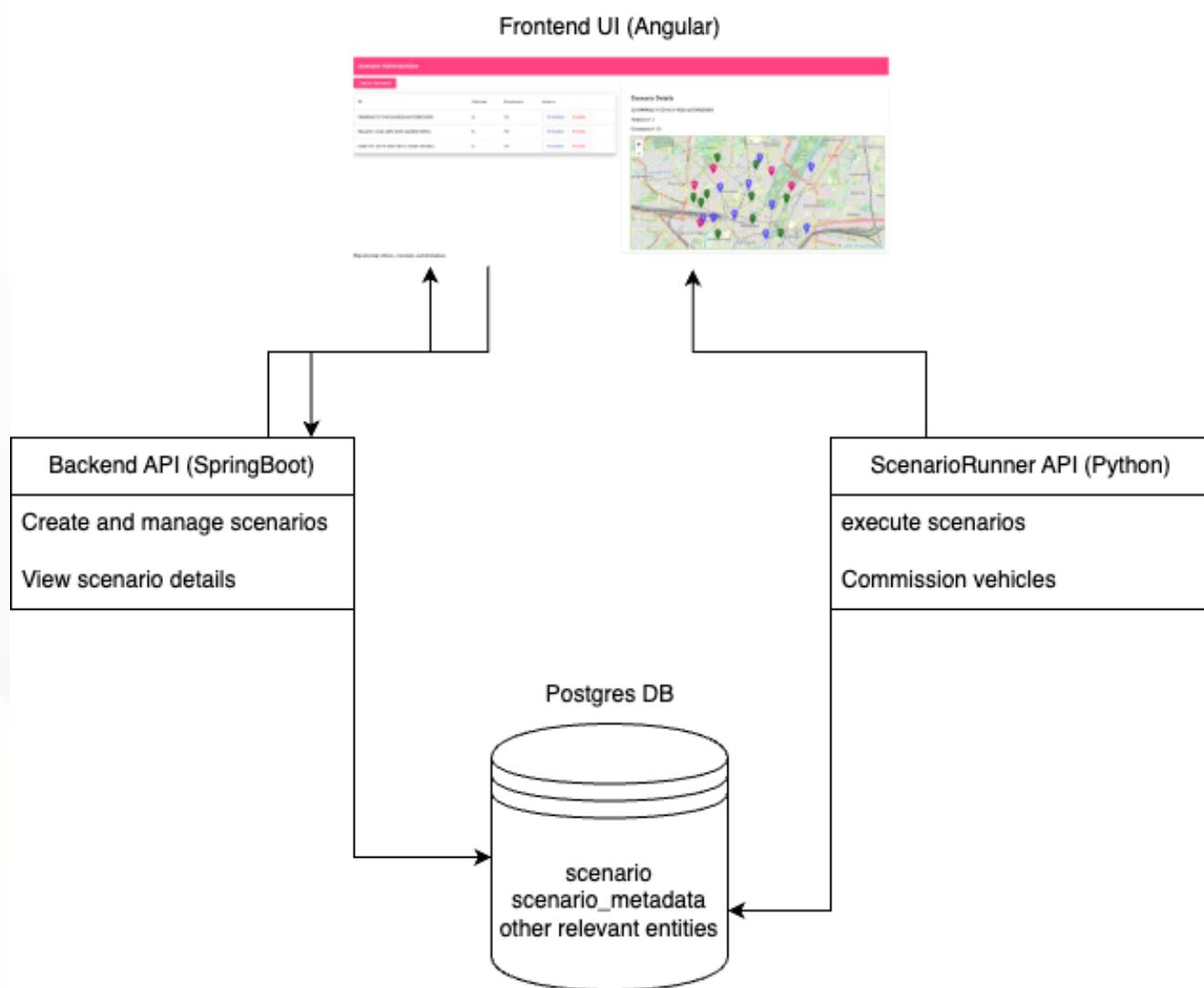
ID	Vehicles	Customers	Actions
6f68c50d-ff2c-4cd3-b6b1-ed7265919e93	5	10	Show Details Delete
9aa14373-6824-4aae-a8ed-a366ed914029	5	10	Show Details Delete
01de9299-b3d1-423b-9b2a-41c956e1da01	5	10	Show Details Delete
cabec1ee-ee27-403c-82e1-3df0fb79bef	5	10	Show Details Delete
f2a280e9-a4ed-46d2-b2cc-65b65e8cacdb	5	10	Show Details Delete
593fddae-959b-497e-8b30-2754ce13f152	50	200	Show Details Delete

Scenario Details  
ID: 6f68c50d-ff2c-4cd3-b6b1-ed7265919e93

Show Scenario Vehicles 5  
Show Scenario Customers 10



# What we provide: Architecture diagram



# Steps to use in a nutshell

- BE or UI: Create a scenario
- RUNNER: Initialize a scenario (input via JSON or provide ID to query existing scenario from DB)
- RUNNER: Launch the scenario (provide desired execution speed)
- RUNNER: Use /Scenarios/update\_scenario/{scenario\_id} endpoint to commission vehicles
- YOUR APPLICATION: Keep track of vehicle and customer parameters and extract/generate relevant data to visualize and optimize.
- BE or RUNNER: Limited scenario metadata is available after each simulation run to provide you a basic set of relevant parameters

# Details

- Docker and Docker compose is required to run the API locally
- The scenario execution happens in memory (no persistence)
- The runner supports multiple concurrent scenario executions
- The distance and time calculations are linear for simplicity
- Vehicle range, wear etc. are not simulated for simplicity. You may apply your own vehicle model in your implementation.
- The API is intended as a basic testing platform for your solution, therefore, any other open-source licensed tool that you deem as helpful may be used

# Helpful additional optional tools

- A postgres DB client (PGadmin or similar)
- A docker UI solution (Docker desktop or similar)
- Detailed code analysis tools to check app cpu/ram benchmarks

# Evaluation criteria

Your solution will be evaluated on the following criteria:

- Code efficiency (does it consume significant amount of resources to operate)
- Operational efficiency (How effectively it optimizes the commissioning process)
- Execution speed (How fast it can make decisions)
- Ability to learn/self-improve (for the AI enthusiasts)
- UI/UX experience (is it easy to deploy/operate from a no-knowledge standpoint?  
Does it look good to the eye)

# Prizes



**1<sup>st</sup> PLACE**



Meta Quest 3  
Mixed Reality VR Headset



**2<sup>nd</sup> PLACE**



DJI Mini 4K  
Drone

# Q&A