
ANYLOGIC CONFERENCE 2026 · SEPTEMBER 15 · ONLINE

TwinPort: An Optimization-Driven Port Digital Twin for Iron Ore Terminals

Powered by AnyLogic Cloud

Afonso Medina

Genoa Soluções Ltda · Brazil



Funded by FAPESP · PIPE program (Innovative Research in Small Enterprises)

Grant 23/09733-2 · “Development of a decision support tool for port task scheduling”

PI: Afonso Celso Medina · Genoa Soluções Ltda · Feb 2024 – Jul 2026 ·

bv.fapesp.br/pt/auxilios/114090

WHO WE ARE



Genoa Soluções builds simulation and optimization tools for ports and heavy logistics, from Brazil.

TwinPort is our web-based port digital twin — developed under a FAPESP PIPE research grant and validated on a real iron ore export terminal.



PIPE program — Innovative
Research in Small Enterprises
Grant 23/09733-2 · Feb 2024 – Jul
2026

Agenda

- 01 Executive summary
- 02 The business challenge
- 03 The optimize-then-simulate loop
- 04 Why AnyLogic
- 05 The model in action
- 06 Answers achieved · Q&A

Optimize the plan. Stress-test it in the cloud.

TwinPort builds the operating plan for a **25 Mtpy Brazilian iron ore export terminal** — trains, stockpiles, blending, ships — and pushes it through the **AnyLogic Cloud API** to stress-test it against real-world uncertainty.

100%

of scheduled ships served in every test scenario

-47%

reduction in estimated demurrage cost

11 / 11

ships served in the recorded October 2022 window, queue cut from 8.8 to 4.7 days

Optimize



Simulate



Decide

One loop, driven from a web browser — no simulation software installed.

A 25 Mtpy Brazilian iron ore export terminal



Configured and driven by the terminal's **actual operational records** — real train discharges, ship voyages and quality-control charts. TwinPort itself is general-purpose and configurable to other bulk terminals.

The one berth is the bottleneck the whole plan converges on — every scheduling mistake upstream ends up as ship queue time.

Four pressures, one schedule



Train arrivals

Schedules set by mine and railway — largely outside the terminal's control.



Stockyard capacity

Three yards, finite positions — every pile placed constrains the next one.



Blending contracts

Chemical specs with tight tolerances — Fe, SiO₂, Al₂O₃, P — per shipment.



Ship queues

Every extra day at anchor is demurrage owed to the shipowner.



Demurrage is where every planning mistake becomes a dollar figure — it is the KPI this whole project attacks.

Every decision is coupled — optimizing is not enough, you have to simulate.

An optimize-then-simulate decision loop



The planner closes the loop from the TwinPort web UI

Accept the plan, adjust preferences, or re-optimize — all from the browser.



Also on the platform: an AI support assistant (Claude), grounded in TwinPort's own docs — pilot in progress

The optimizer writes the whole playbook

- 1 **Which train discharges where** — car dumper, route and destination pile for every arrival
- 2 **How piles are formed and blended** — a MILP stage keeps every shipment inside its chemical spec
- 3 **When each ship berths and loads** — sequencing the single berth to cut queue and demurrage



Berth occupancy and the operations timeline on one shared axis (TwinPort, Outputs > Routes). Search: the SAIL engine's metaheuristic over discharge and berthing decisions, with an exact MILP for the blending sub-problem — one slide, no math, happy to go deeper in Q&A.

From plan to cloud: one JSON, one API call

THE CONTRACT

```
{  
  "export_info": { ... },  
  "ships": [...], "trains": [...],  
  "stocks": [...], "yards": [...],  
  "equipments": [...], "routes": [...],  
  "car_dumpers": [...],  
  "maintenance": [...],  
  "quality_model": { ... }  
}
```

The optimizer's full plan (scenarioJson), plus a second JSON with the stable port setup (portJson). The same model also reads them as local files in development — one codebase, no forks.

HOW TWINPORT TALKS TO ANYLOGIC CLOUD — PYTHON



```
from anylogiccloudclient.client.cloud_client import CloudClient  
  
client = CloudClient(API_KEY, CLOUD_INSTANCE)  
model = client.get_model_by_name("TwinPort")  
version = client.get_latest_model_version(model)  
  
inputs = client.create_default_inputs(version)  
inputs.set_input("scenarioJson", plan_json) # the plan  
inputs.set_input("portJson", port_config)   # port setup  
  
run = client.create_simulation(inputs)  
run.execute()  
outputs = run.get_outputs()
```



Built with the official `anylogiccloudclient` package, wired up straight from AnyLogic's Cloud API help and examples — a new plan is just a new API call, no model re-upload.

The plan is deterministic. The port is not.

Simulation is the stress test a plan passes before anyone commits ships to it — inside a commercial web product. AnyLogic won on four counts:

1 The Cloud API — the enabler

Simulation as a service: TwinPort's Python backend drives runs through the official `anylogiccloudclient` library, and planners watch the animation in the browser. No local install, no per-seat license.

2 Real Java inside the model

The model parses the plan JSON itself — the same code runs locally in development and on the cloud in production. One model, no forks.

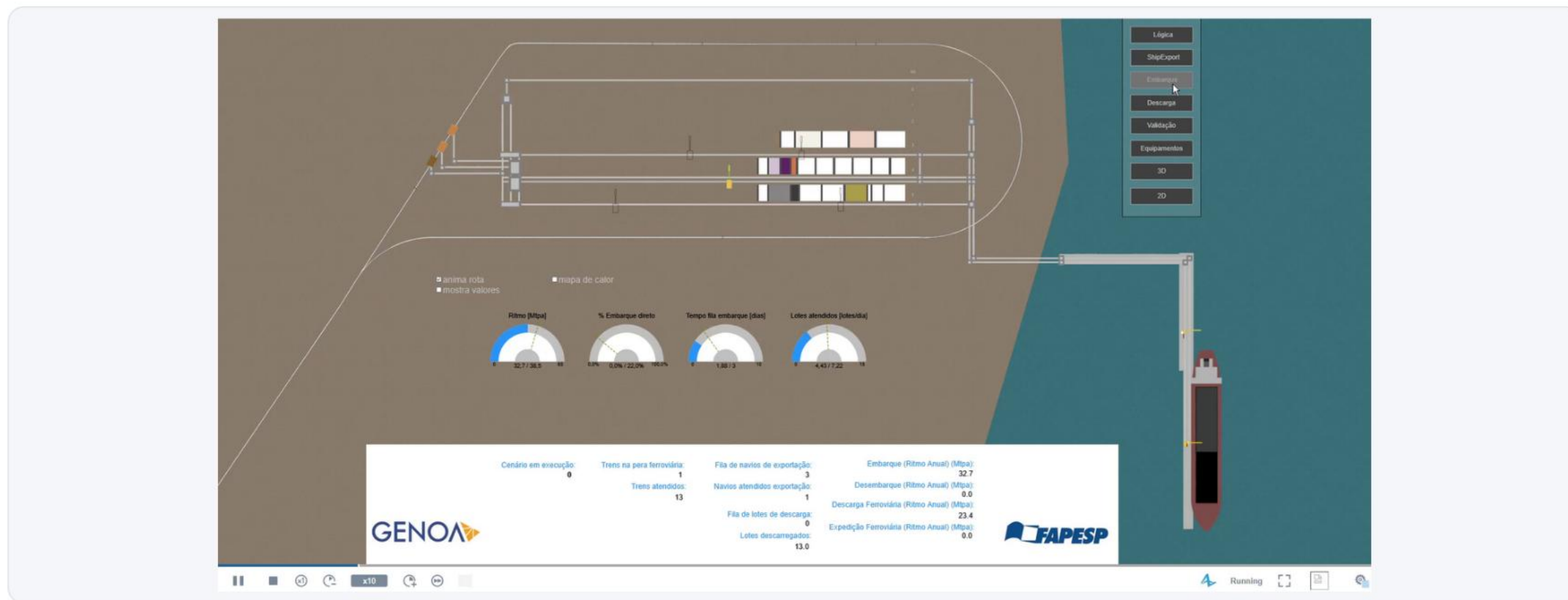
3 Multimethod + material handling

Discrete events for the train→yard→ship flow, failure distributions and per-origin quality sampling — all in one model.

4 Native 2D/3D animation

The same run that validates the plan convinces the executive — no separate visualization work.

The terminal, rebuilt in AnyLogic



3 stockyards · 17 stockpiles

Car dumpers

Stacker-reclaimers

Ship loaders

Single berth

 Built entirely from the two JSON inputs

Three ways reality gets injected



Equipment failures

Every car dumper, stacker-reclaimer and ship loader has its own **MTBF / MTTR distributions**. Failures interrupt operations mid-task, exactly as they do on the ground.

source: equipment records



Maintenance windows

Scheduled stops per equipment are read straight from the plan JSON — the simulation respects the **same calendar the optimizer planned around**.

source: maintenance plan



Ore quality variability

Each train's chemistry is sampled from a **multivariate normal fitted to the terminal's real quality-control charts** — 36 mine origins, 10 chemical attributes, correlations preserved.

source: 2023 QC charts

A real plan, running

TwinPort

Scenarios

Scenario

Port

Optimization

Simulation

Results

Docs

afonso.medina@genoads.com

afonso.medina@genoads.com.br

Settings

Logout

TwinPort v1.0.0

WORKSPACE · SCENARIOS

Scenarios

View and manage scenarios from the database.

Table: Scenarios

Search scenarios...

Refresh Table

Add Scenario

Upload

Export CSV

ID	User	Scenario Name	Description	Port	Creation Date	Start Date	End Date	Optimized	Edit	Duplicate	Delete	Select
1	Sistema	Default Scenario	Default Scenario for development proposes. Can't be edited or modified.	Porto Padrão (Sistema)	2025-11-22 18:12:00	2022-10-04	2022-10-28	Sim	Edit	Duplicate	Delete	Select

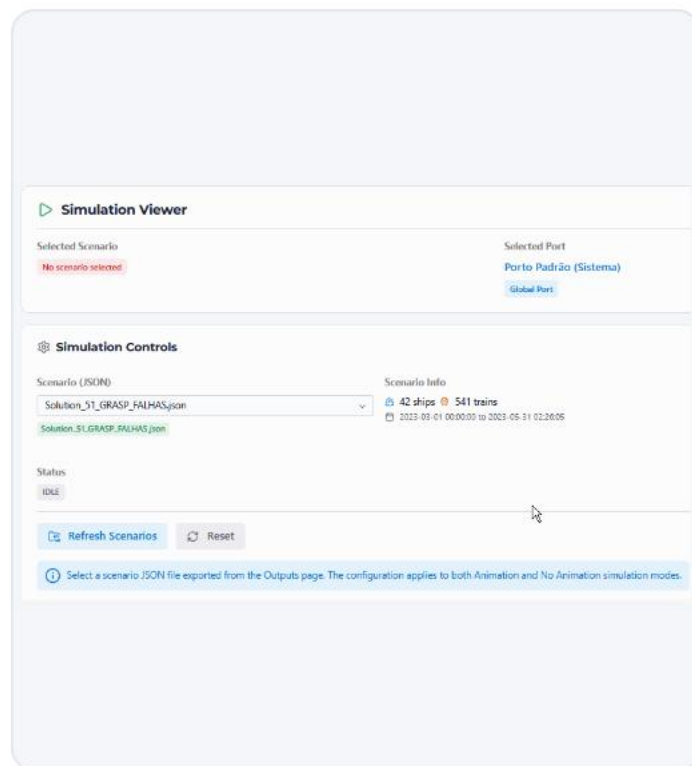
System Logs

Clear Logs

No logs available. Upload data to see logs here.

From the planner's seat: one button

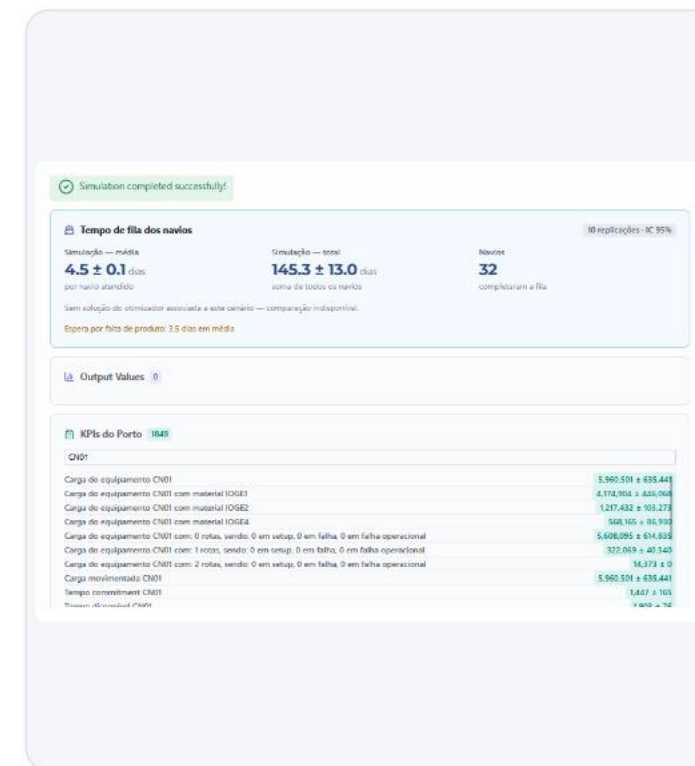
01 Pick a plan



02 Run in the cloud



03 Explore the results



Nothing installed, nothing configured on the planner's machine — **the simulation lives behind a button in the web app.**

How do we know the plans are any good?

✓ We rebuilt the terminal's recorded October 2022 operation inside TwinPort and ranked our plans against it — same ships, same trains, same time window.

✓ **Closed ship voyages** — cargo, queue time and stay per vessel, from the terminal's records

✓ **Per-train discharge logs** — which yard received each train, including direct-to-ship loading

✓ **One window, ship by ship** — the 11 vessels served in October 2022, same arrivals, same trains

Solution Ranking

[Refresh](#)

All solutions for the current scenario, ranked by demurrage cost (lowest = best).

#	Solution	Method	Ships	Ship Cargo (t)	Train Cargo (t)	ED Cargo (t)	Avg Queue (days)	Demurrage (USD)	Estadia (days)	Ship Load %
1	Solution_40_SAIL-Fast	loaded	11	1991455	2119077	103160	4.65	665636	68.21	100
2	Solution_38_SAIL-Fast		11	1991455	2119077	73061	7.07	1010972	93.87	100
3	Real 2022 Baseline		11	1910445	0	0	8.8	1258553	115.73	100

Compare tab: all plans for the scenario ranked by demurrage cost; the real baseline is the last row.

What did the terminal gain?

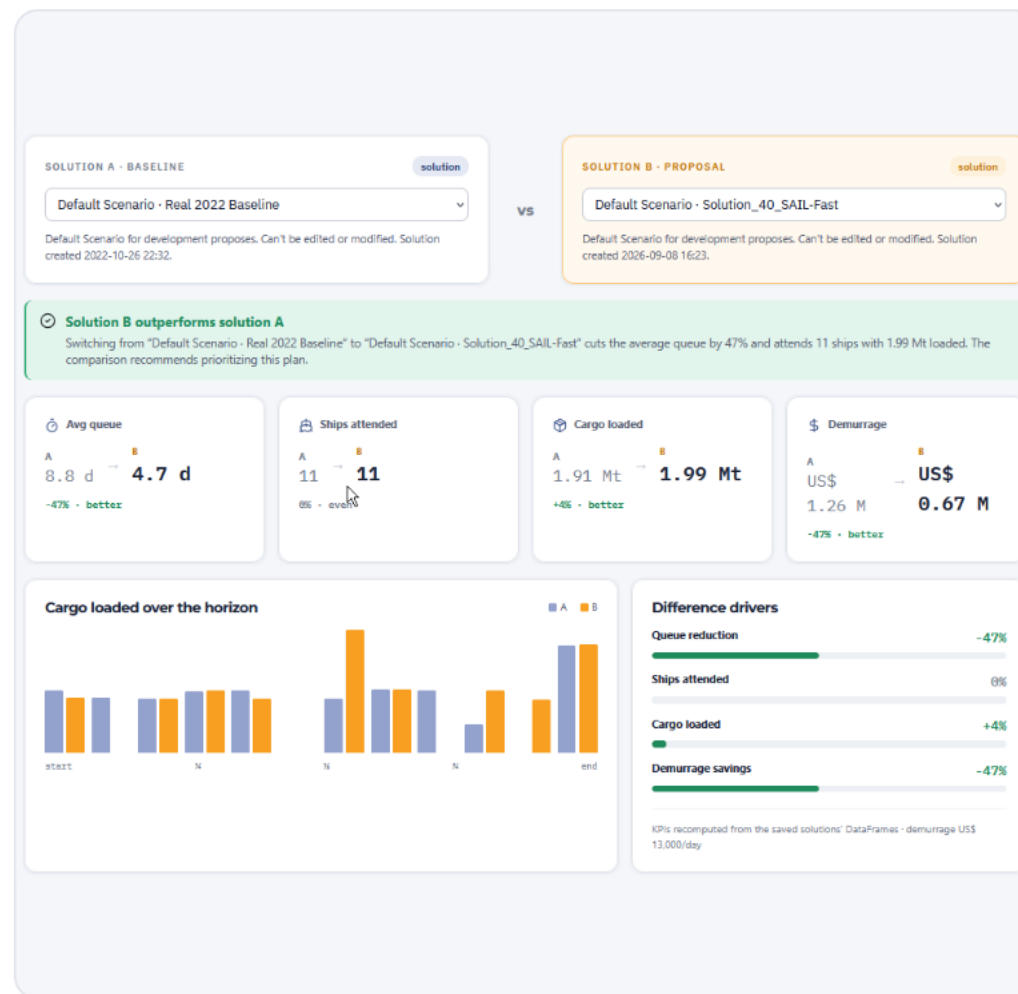
Could every scheduled ship be served? **Yes – all of them**

How much demurrage could be avoided? **47% of the cost**

How much shorter would the queue be? **8.8 → 4.7 days on average**

Does the plan survive reality? **Stress-tested in the cloud**

Versus the terminal's recorded October 2022 operation, 11 ships, demurrage at US\$ 13,000/day. Demurrage figures are model estimates on real data.



What changed beyond the numbers?



From spreadsheet to auditable decisions

Every plan is stored, ranked and reproducible — scheduling stops being one planner's private spreadsheet.



Zero-install adoption

Simulation runs on AnyLogic Cloud — the planning team adopted a browser tab, not a software rollout.



Confidence to negotiate

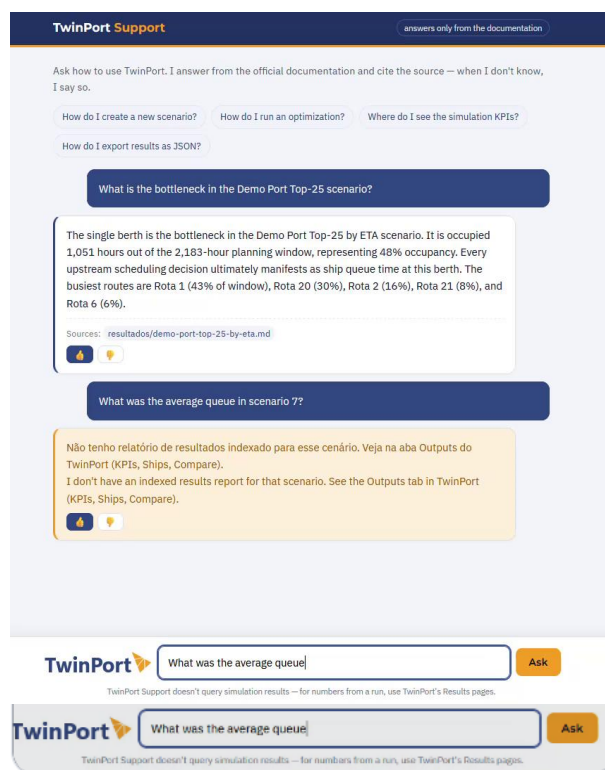
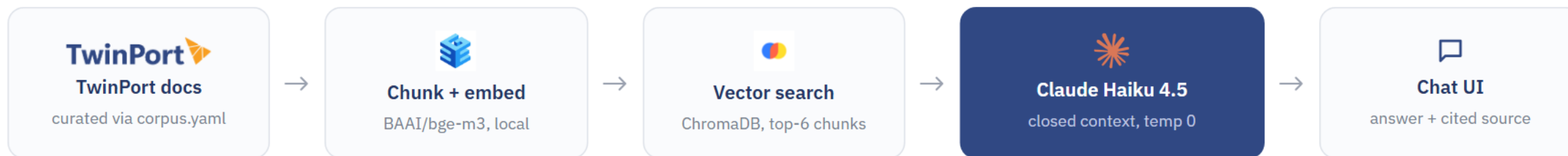
A plan stress-tested under failures and quality drift backs harder conversations about laydays and berthing windows.



A platform, not a one-off study

Terminal layout, equipment and products are configuration — the same product fits other bulk terminals.

An AI support assistant, grounded in the documentation



The documentation answers, not the model. Claude never draws on its own knowledge: it only assembles the retrieved TwinPort passages into a reply, in the user's language, and cites the file they came from. No passage, no answer.

Refuses without a source
No retrieved context, no answer — the question is logged instead of guessed at.

Temperature 0, closed context
Output constrained to what was retrieved — no free-form model knowledge.

Citations required
Every answer names the exact TwinPort doc file it came from.

Scope-aware
Simulation results and optimizer internals are explicitly out of scope.

Status: internal pilot — validating whether documentation RAG is enough before it becomes a Genoa product feature.

Three lessons from embedding cloud simulation in a product

1 Make the JSON the contract

A stable, explicit schema between optimizer and simulator let the two sides evolve independently — and made cloud vs. local execution a non-issue.

2 Tolerate model versions in the cloud

Guard every new input parameter so an older uploaded model still runs — a product cannot break because the cloud model lags one release behind.

3 Validate against real records, publicly

The real-operation baseline turned our KPIs from a claim into a comparison — it is the single most persuasive artifact in every client conversation.

Thank you

QUESTIONS?

Afonso Medina

afonso.medina@genoads.com.br

Genoa Soluções Ltda · Brazil



Funded by FAPESP · PIPE program (Innovative Research in Small Enterprises)

Grant 23/09733-2 · “Development of a decision support tool for port task scheduling”

PI: Afonso Celso Medina · Genoa Soluções Ltda · Feb 2024 – Jul 2026 ·

bv.fapesp.br/pt/auxilios/114090