



# OAR Rust Redux

---

Février 2026

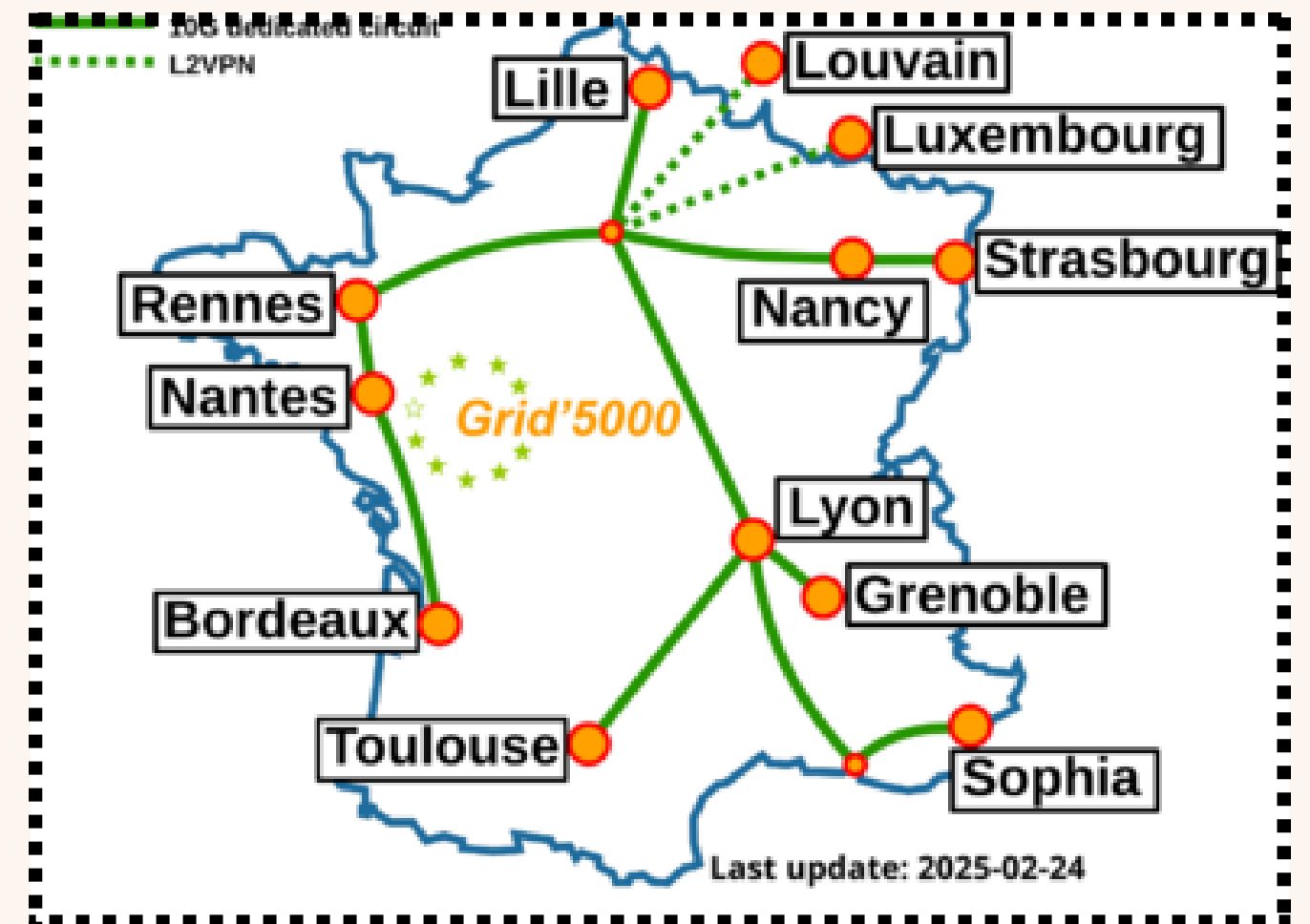
 Annexes

Salmane Amine  
Er-Rami Moataz  
Amessegher Aymane  
Memil Dila

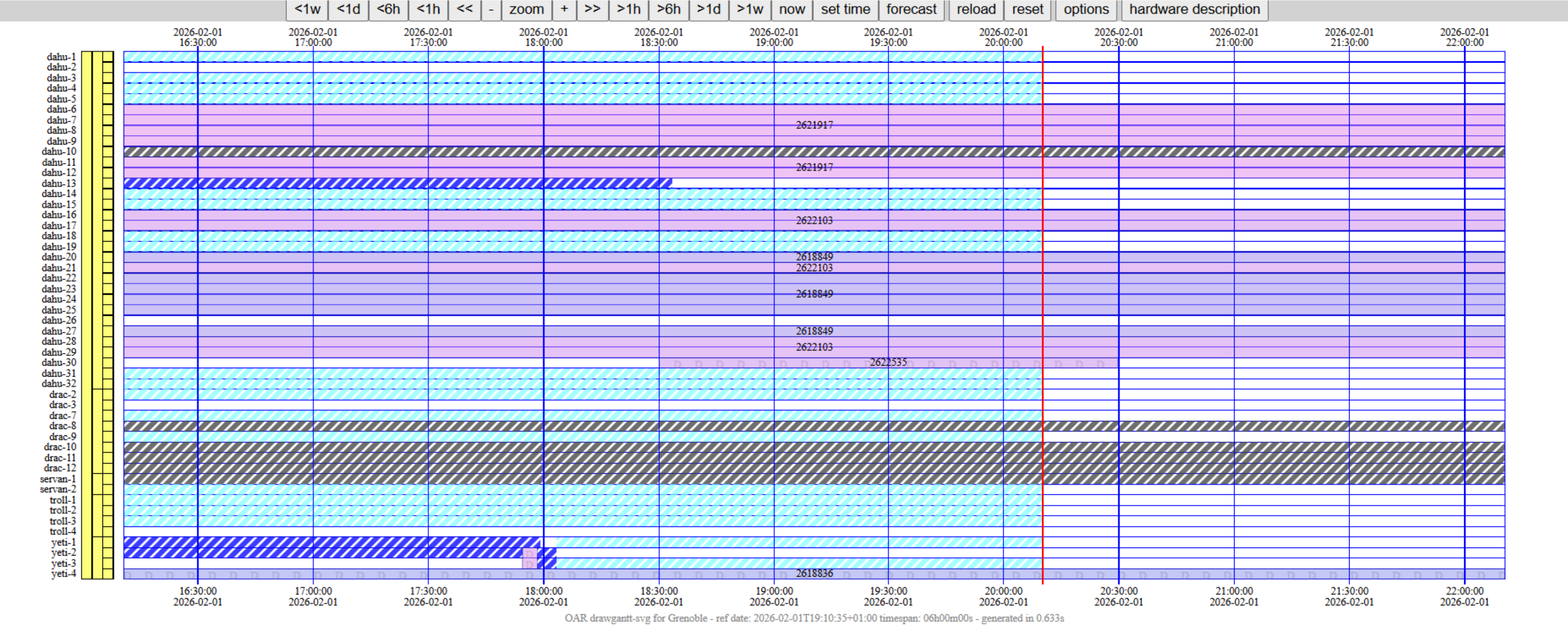
# Contexte et objectif du projet

- **Epurer l'interface graphique actuelle**
  - Développer une interface similaire aux diagramme de gant sur Grid5000
  - Visualiser des statistiques
- **Améliorer l'intégration avec OAR grâce à la mise en place d'une API développée en Rust.**
  - Architecture modulaire et découplage fonctionnel
  - Alignement métier
  - Fiabilité du contrat d'interface
  - Sécurisation transverse et gestion des accès

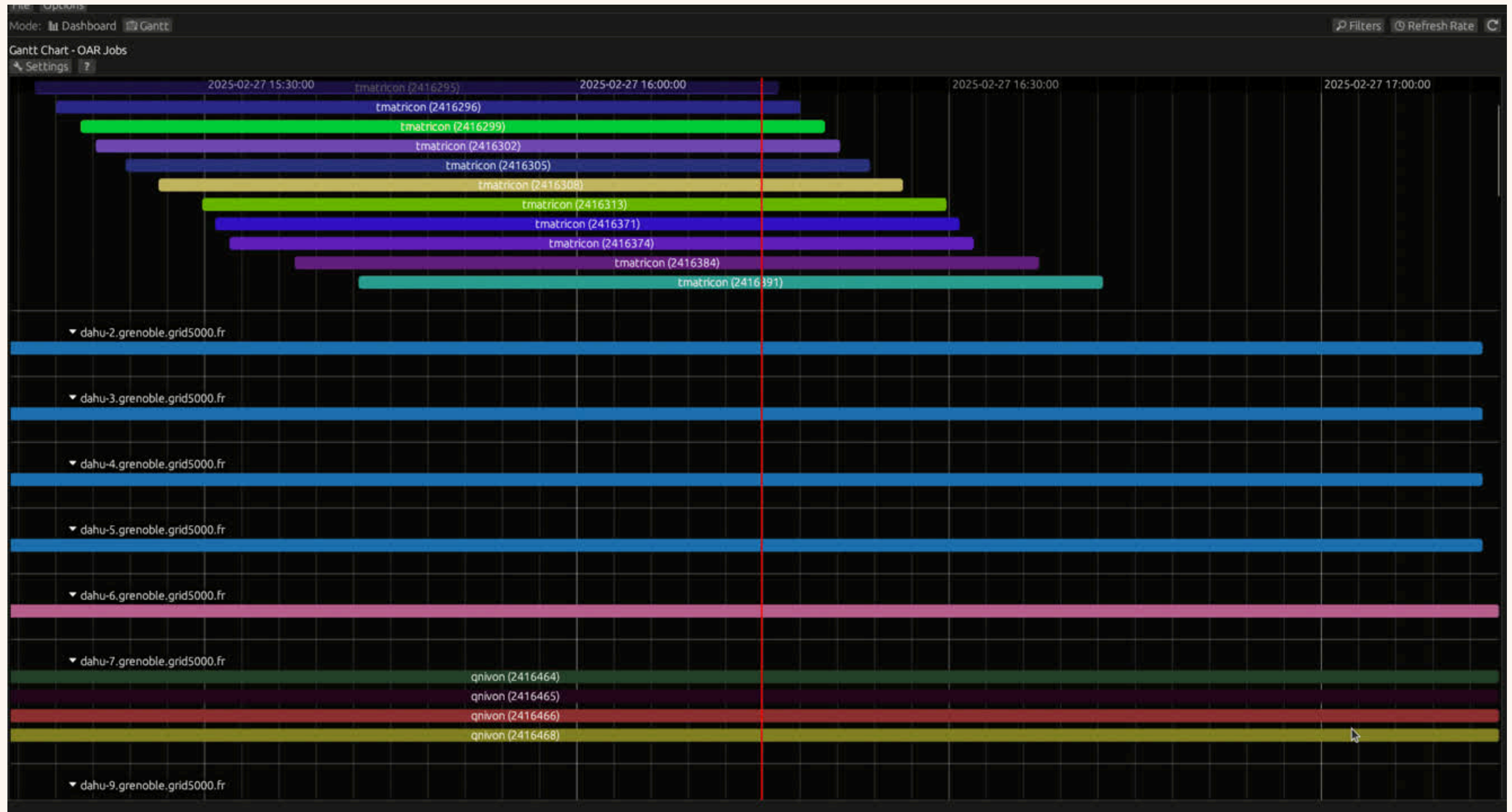
Carte réseaux Grid5000



# Diagramme de Gantt (Grid5000)



# Goard Rust 2025



Source : <https://github.com/Goard-Rust/.github/blob/main/profile/goard.gif>

# Composition de notre équipe

**Amine  
Salmane**

**API  
Chef de Projet**

**Moataz  
Er-rami**

**API**

**Aymane  
Amessegher**

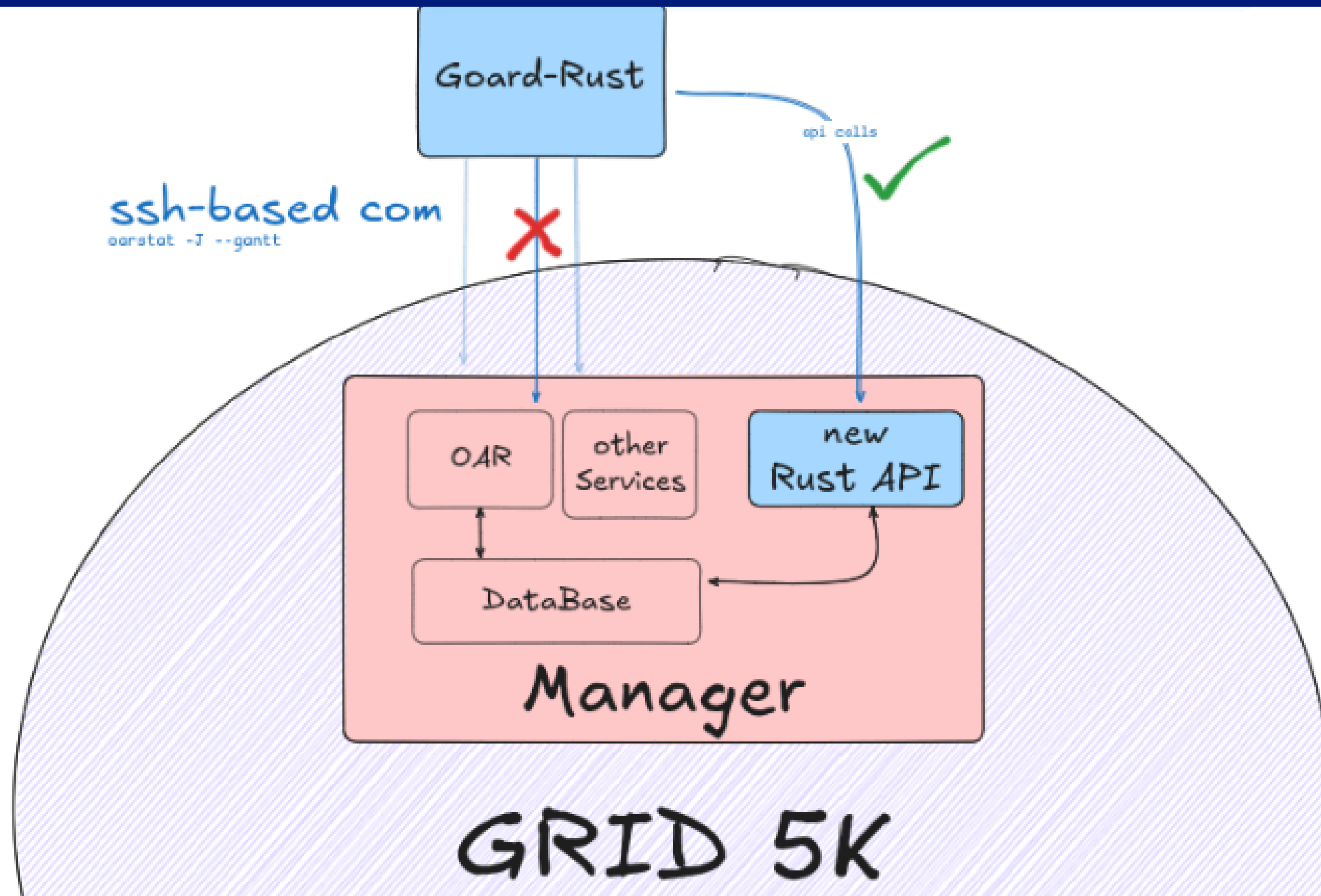
**UI/UX**

**Dila  
Memil**

**UI/UX**



# L'architecture du système



# Les technologies à utiliser

## API

Tokio/AXUM 

Aide : Open Api Swagger/Scalar

## GUI (UI)

Librairie : Egui 

## Common

Cargo

Docker

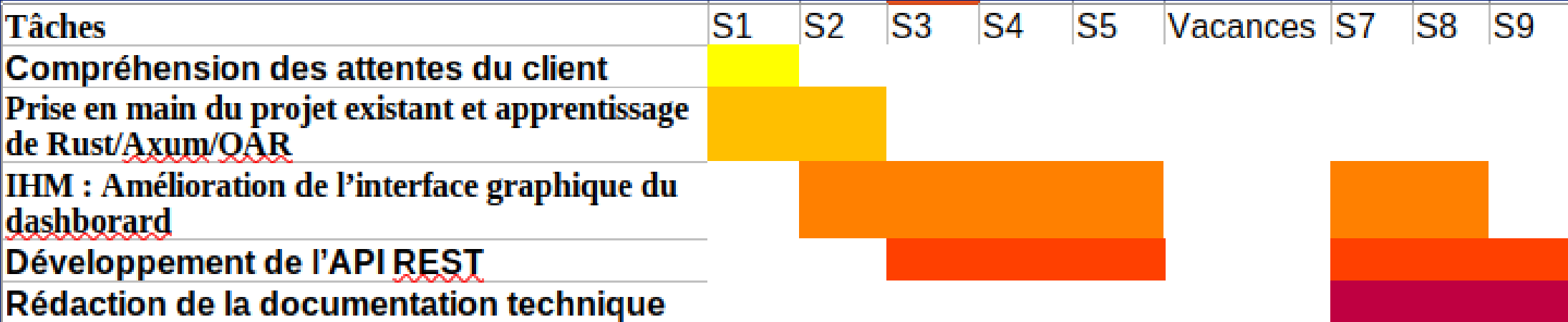
Github actions

Discord

# Plan de travail

Ticket	Solution	Etat	Solution	Ticket	Etat
Architecture	DDD / Multi-crates	✓	-Réorganiser le code pour garantir une maintenabilité.	Architecture	X
Metier	Modeliser la bdd a partir de la doc/code OAR3		-Ajout de la barre hiérarchique (site,cluster,host)	Metier	✓
Confiance	Doc-as-Code	✓		Design	
Sécurité	Middleware OAuth / OpenID Connect	X	-Améliorer la lisibilité pour un résultat compact		

## Diagramme de Gantt





# Difficultés

- Apprentissage Rust
- Manque de ressource (langage émergent)



**Merci pour votre  
Attention !**

# ANNEXE



[Code source](#)

# Les enjeux :

OAR gère l'allocation de ressources HPC pour des milliers de jobs quotidiens dans les clusters de calcul. Migration progressive vers Rust pour gains en performance, sécurité mémoire et concurrence. Les centres de calcul exigent fiabilité 24/7 et réactivité (<100ms pour requêtes API).

# Solutions actuelles & insuffisances :

Version Python actuelle : lente pour traiter gros volumes, consommation mémoire élevée, difficultés de maintenance. Dashboard 2024-2025 créé mais API REST manquante.

# Besoins client :

- API REST performante (remplacement Python)
- Dashboard opérationnel avec Gantt temps réel
- Maintien compatibilité ascendante et compacité

# Défis techniques :

- Interfaçage Rust <-> base existante OAR
- Architecture asynchrone (Axum/Tokio)
- Parsing/validation requêtes complexes

# Approche solution :

Technique : Axum pour API REST, architecture modulaire (handlers → services → data), POC existant comme base  
Organisationnelle : tests unitaires/intégration continus .

## Cadrage initial :

- Durée : 9 semaines
- Livrables : architecture documentée, API fonctionnelle, dashboard enrichi
- Budget temps : ~200h
- Risques : complexité intégration existant, difficulté d'apprentissage Rust



Tableau comparatif Framework

Framework	Points forts	Limites
axum	<ul style="list-style-type: none"> <li>- Ergonomie moderne &amp; modulaire</li> <li>- Très bon écosystème async</li> <li>- Facile à étendre</li> </ul>	<ul style="list-style-type: none"> <li>- Peut être verbeux avec traits avancés</li> </ul>
actix-web	<ul style="list-style-type: none"> <li>- Très hautes performances</li> <li>- Bon pour API à fort trafic</li> <li>- Large support middleware</li> </ul>	<ul style="list-style-type: none"> <li>- API plus complexe</li> <li>- Mémoire plus importante</li> </ul>
rocket	<ul style="list-style-type: none"> <li>- API très conviviale</li> <li>- Routing macro intuitif</li> <li>- Bon pour prototypes</li> </ul>	<ul style="list-style-type: none"> <li>- Moins performant</li> <li>- Moins adapté à gros services API</li> </ul>
warp	<ul style="list-style-type: none"> <li>- Style fonctionnel &amp; composable</li> <li>- Bon pour APIs légers</li> </ul>	<ul style="list-style-type: none"> <li>- Moins populaire</li> <li>- Architecture un peu différente</li> </ul>

Q Search

Introduction

Users

Get user by IDGET

Models

OAS 3.1.0

OAR 3 API

Download OpenAPI Document

description

Server

http://0.0.0.0:3000

Client Libraries

ShellRubyNode.jsPHPPythonMore

Curl Shell

Users

OPERATIONS

GET /users/{id}

Get user by ID

Retrieve a detailed user profile by their unique UUID from the database.

Path Parameters

idApiKeyLocation: uuidrequiredThe unique UUID of the user

Responses

200

404User not found in the system

GET /users/{id}Shell Curl

1curl 'http://0.0.0.0:3000/users/{id}'

Test Request

200404500Show Schema

{email: "string",id: "123e4567-e89b-12d3-a456-426614174000",username: "string"}

Open API Client

Powered by Scalar



idApiKeyLocation: uuidrequired  
The unique UUID of the user

- 200
- 404User not found in the system

GET /users/{id}Shell Curl

1curl 'http://0.0.0.0:3000/users/{id}'

Test Request

200404500Show Schema

{  
 "email": "string",  
 "id": "123e4567-e89b-12d3-a456-426614174000",  
 "username": "string"  
}

OAR 3 API

OAS 3.1

[/api.json](#)

description

Users

^

GET

/users/{id}

Get user by ID

^

Retrieve a detailed user profile by their unique UUID from the database.

Parameters

Try it out

Name	Description
<div><div>id</div><div>★ required</div><div>string(\$uuid)</div><div>(path)</div></div>	<div>The unique UUID of the user</div> <div><div>id</div></div>

Responses

Code	Description	Links
200	<div>Media type</div> <div><div>application/json</div></div> <div>Controls Accept header.</div> <div>Example Value   Schema</div> <div><div>{   "email": "string",   "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",   "username": "string" }</div></div>	No links
404	User not found in the system	No links
500	Internal server error - something went wrong on our end	No links

Plan de Travail

Tâches	S1	S2	S3	S4	S5	Vacances	S7	S8	S9
Compréhension des attentes du client									
Prise en main du projet existant et apprentissage de Rust/Axum/OAR									
IHM : Amélioration de l'interface graphique du dashborard									
Développement de l'API REST									
Rédaction de la documentation technique									

S3	Ajout barre hièrarchie cluster	Choix Architecture + Generation doc
S4	réorganiser le code (maintenabilité) prototype (Stats) + implémentation	Finaliser schema bdd
S5	Ajout de filtre pour la visualisation	Implementation JWT

# État actuel de l'UI

