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

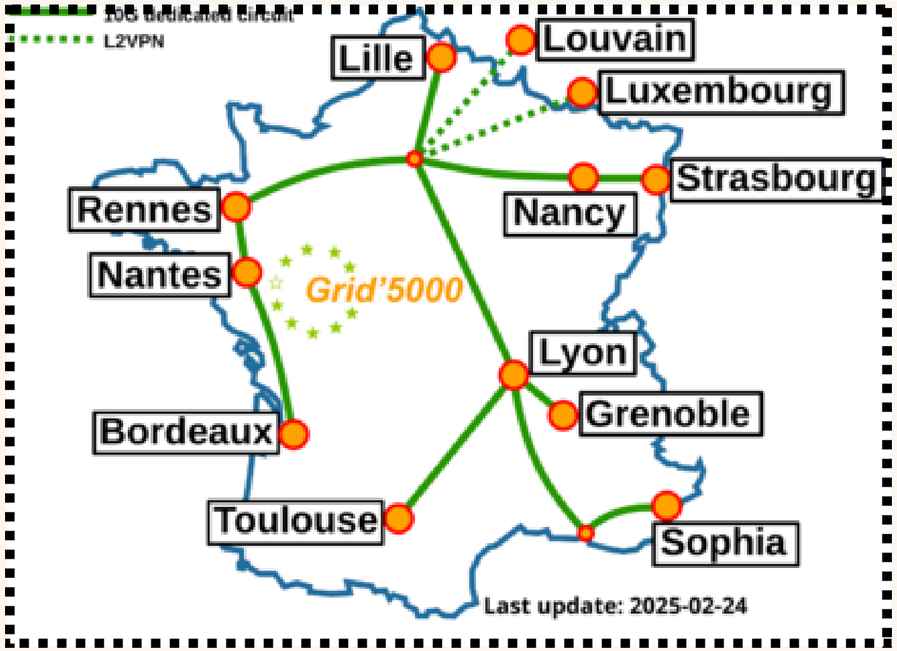
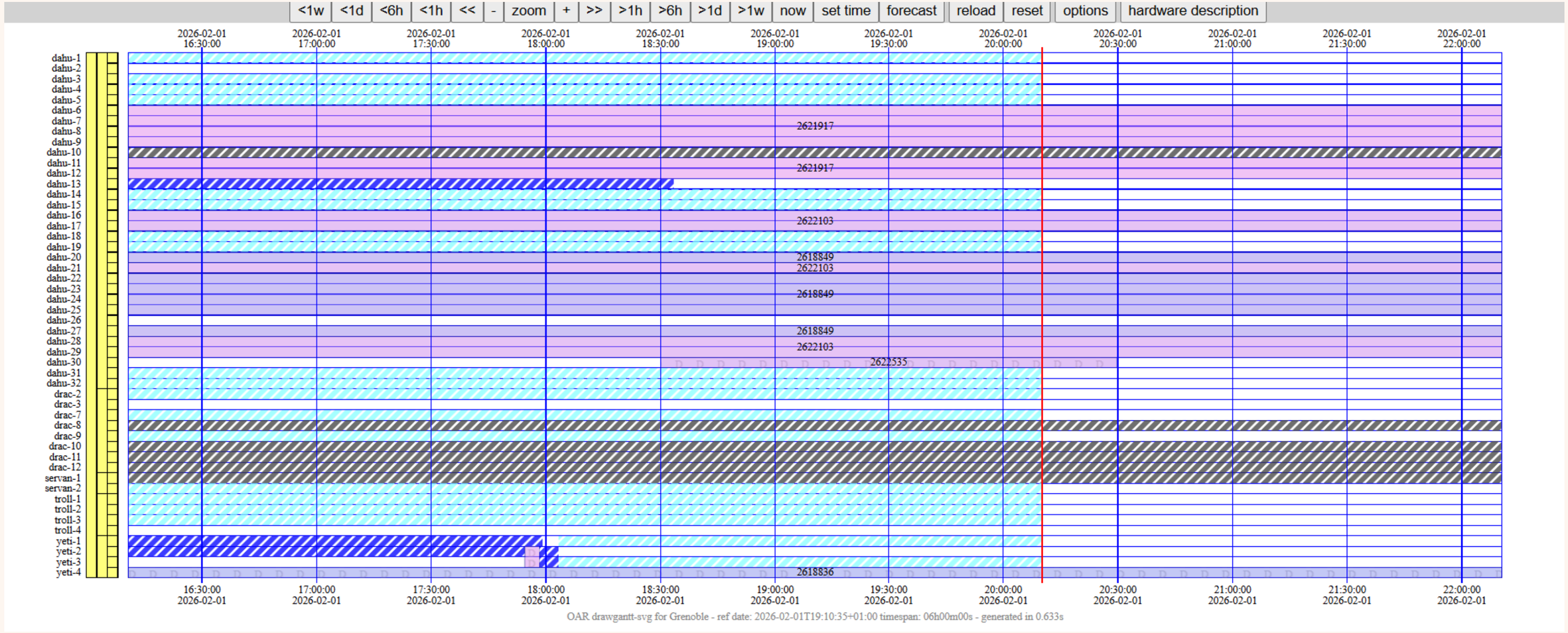
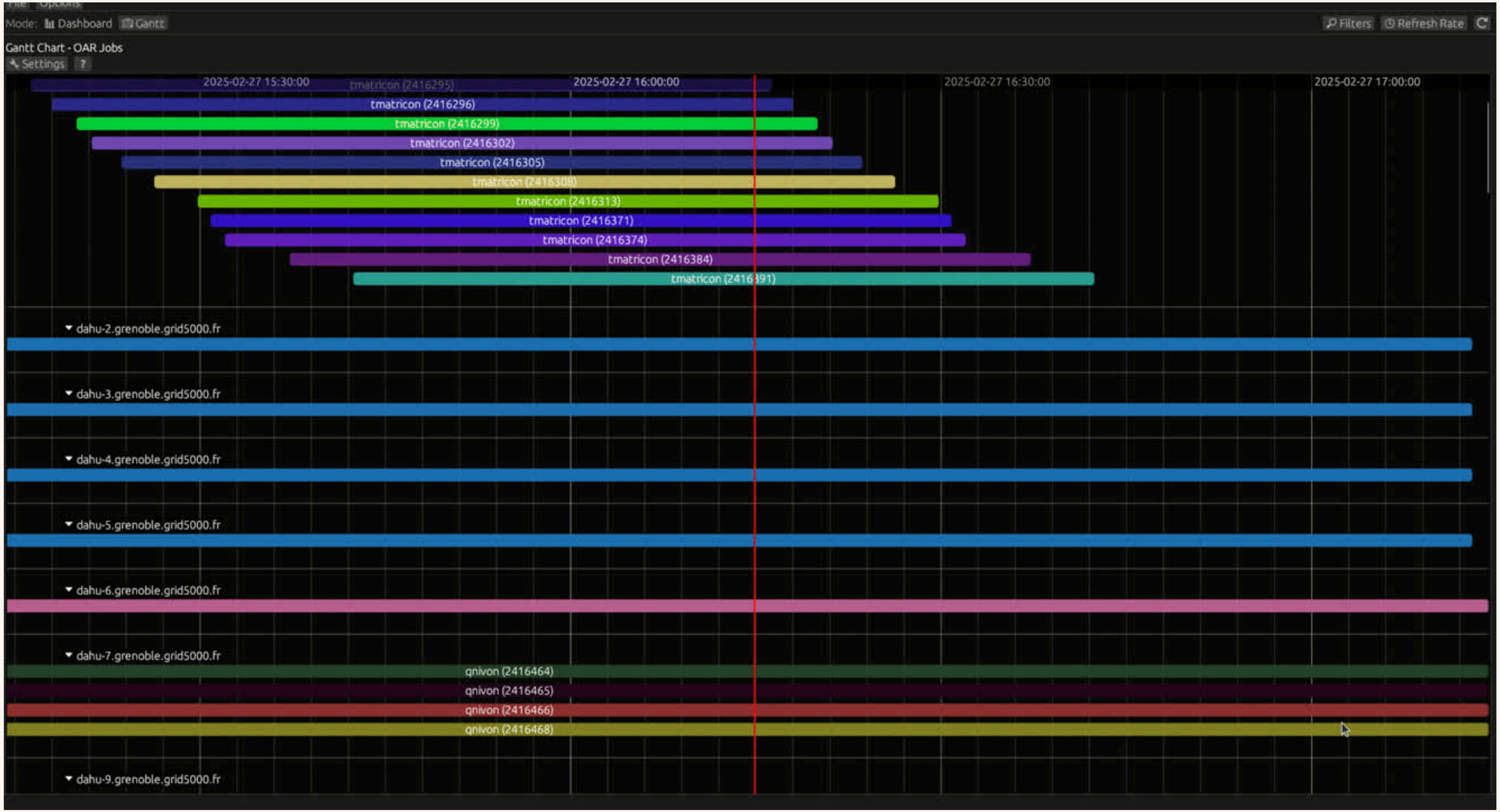


Diagramme de Gantt (Grid5000)



Goard Rust 2025



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

Composition de notre équipe

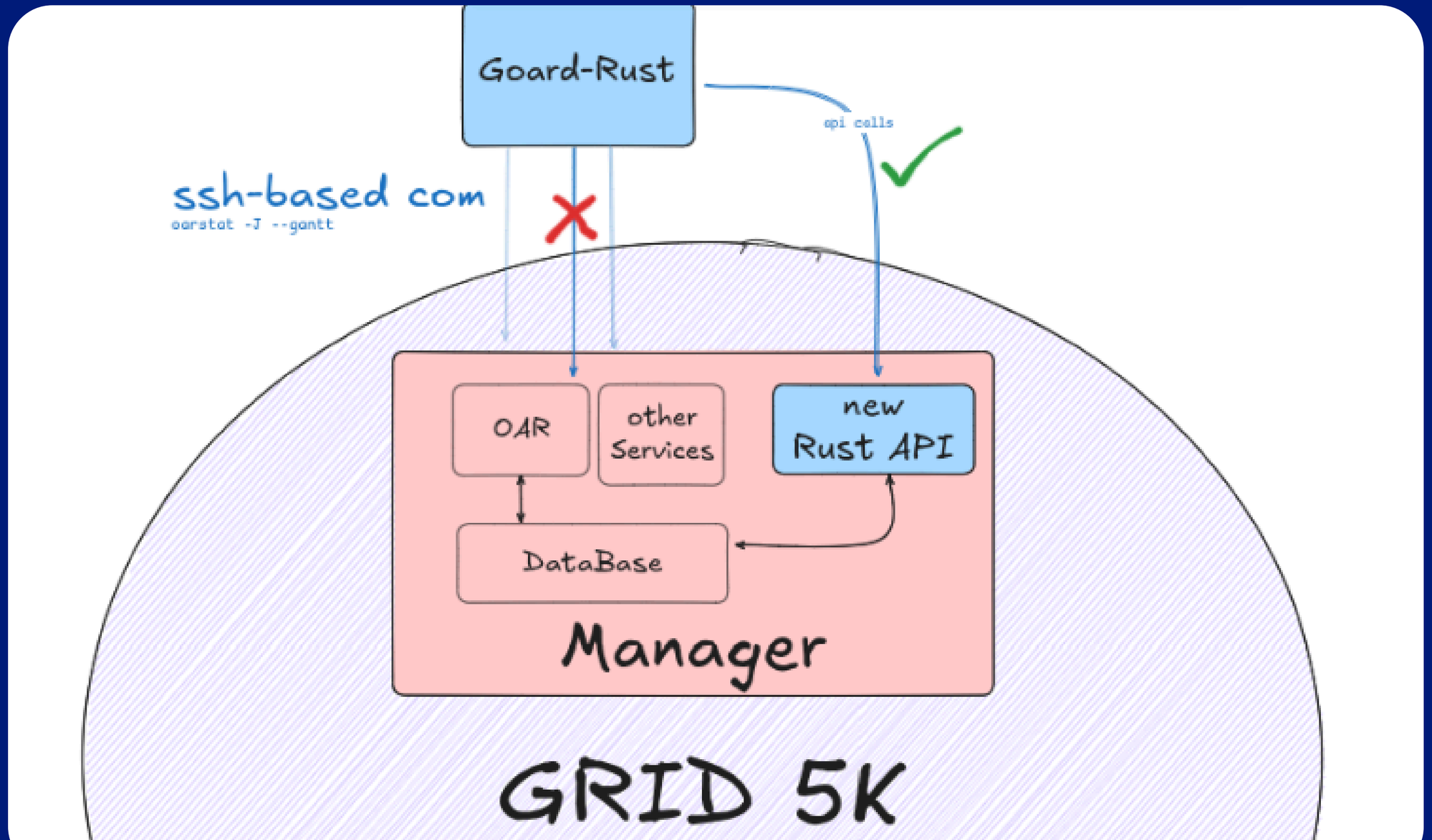
Amine Salmane
API Chef de Projet

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

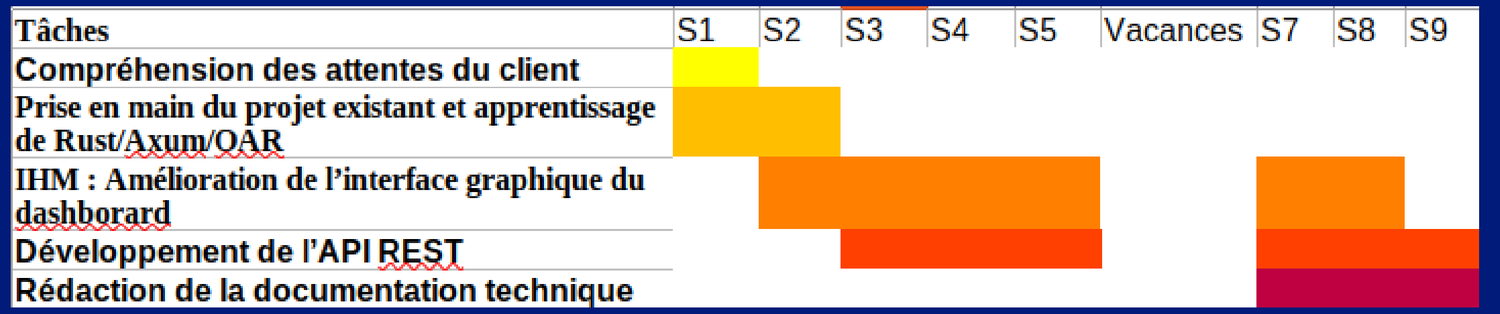
- Cargo
- Docker
- Github actions
- Discord

Plan de travail

Ticket	Solution	Etat
Architecture	DDD / Multi-crates	✓
Metier	Modeliser la bdd a partir de la doc/code OAR3	
Confiance	Doc-as-Code	✓
Sécurité	Middleware OAuth / OpenID Connect	X

Solution	Ticket	Etat
-Réorganiser le code pour garantir une maintenabilité.	**Architecture**	X
-Ajout de la barre hiérarchique (site,cluster,host)	**Metier**	✓
-Améliorer la lisibilité pour un résultat compact	**Design**	

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 :

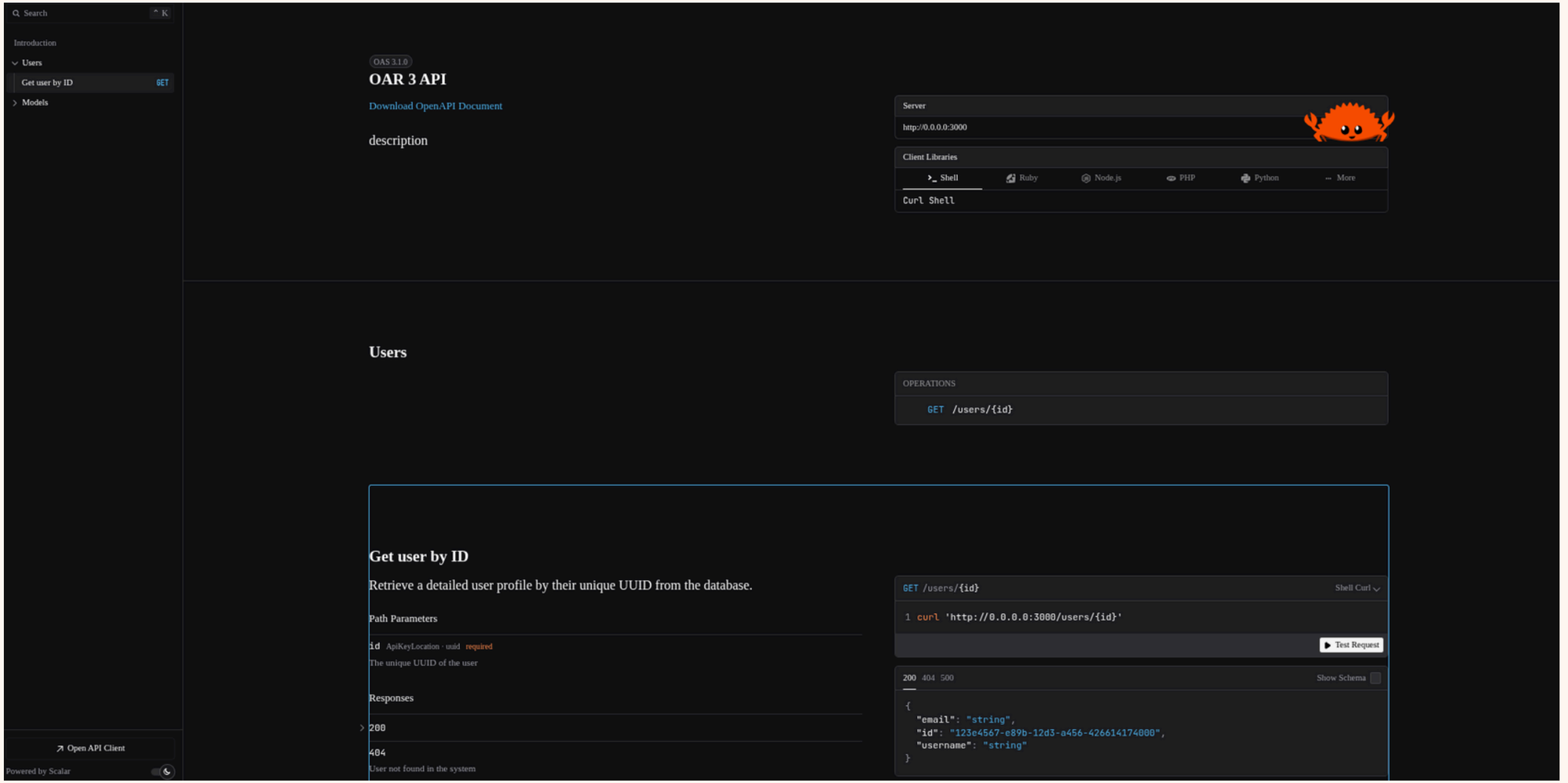
Technique : Axum pour API REST, architecture modulaire (handlers → services → data), POC existant comme base
Organisationnelle : tests unitaires/intégration continus .

Cadrage initial :

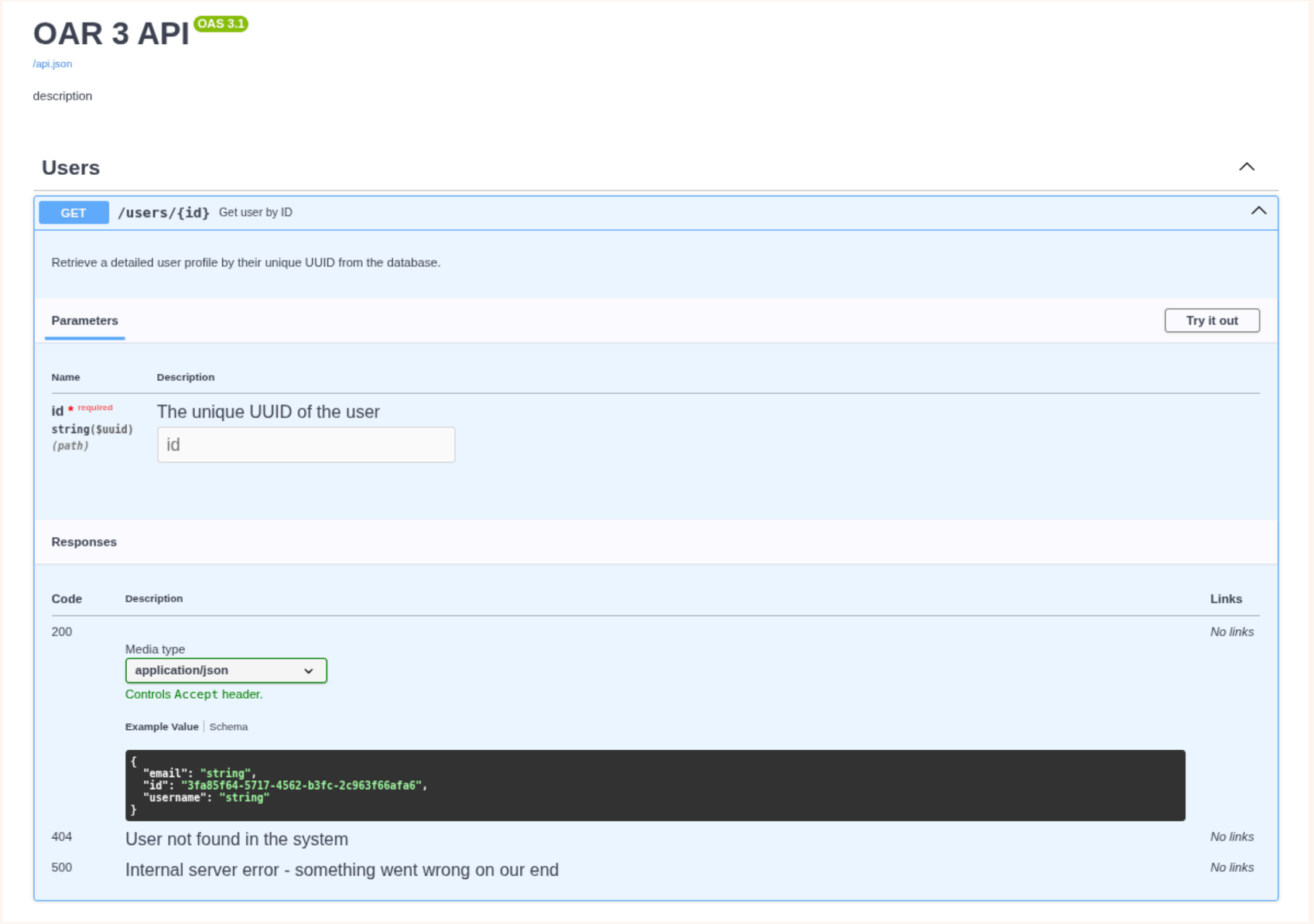
- Durée : 9 semaines
- Livrables : architecture documentée, API fonctionnelle, dashboard enrichi
- Budget temps : ~200h
- Risques : complexité intégration existant, difficulté d’apprentissage Rust

Tableau comparatif Framework

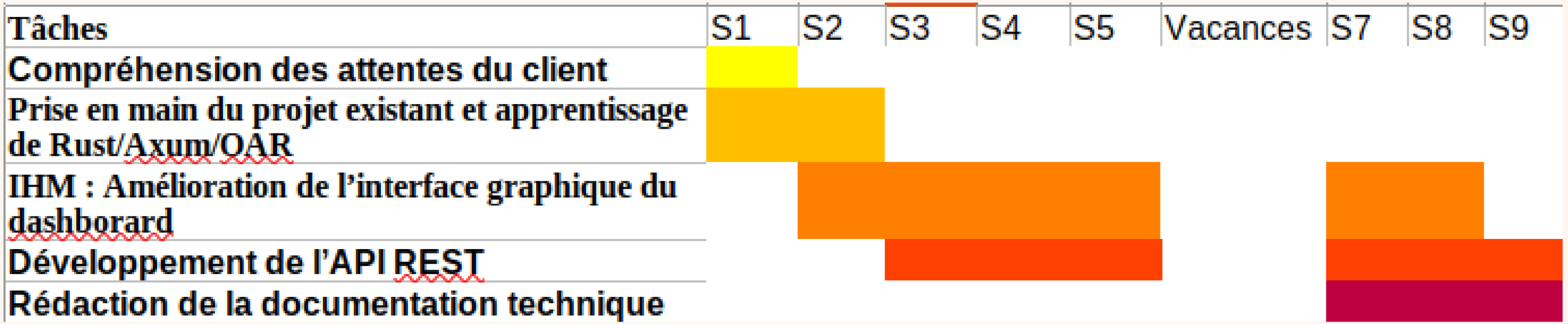
Framework	Points forts	Limites
axum	- Ergonomie moderne & modulaire - Très bon écosystème async - Facile à étendre	- Peut être verbeux avec traits avancés
actix-web	- Très hautes performances - Bon pour API à fort trafic - Large support middleware	- API plus complexe - Mémoire plus importante
rocket	- API très conviviale - Routing macro intuitif - Bon pour prototypes	- Moins performant - Moins adapté à gros services API
warp	- Style fonctionnel & composable - Bon pour APIs légers	- Moins populaire - Architecture un peu différente


Search ⌃ K
Introduction
Users
Get user by ID GET
Models
Open API Client
Powered by Scalar
OAS 3.1.0
OAR 3 API
Download OpenAPI Document
description
Server
http://0.0.0.0:3000
Client Libraries
Shell Ruby Node.js PHP Python More
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
id ApiKeyLocation · uuid required
The unique UUID of the user
Responses
200
404
User not found in the system
GET /users/{id}
Shell Curl
1 curl 'http://0.0.0.0:3000/users/{id}'
Test Request
200 404 500
Show Schema
{
"email": "string",
"id": "123e4567-e89b-12d3-a456-426614174000",
"username": "string"
}

Swagger


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
id
required
string($uuid)
(path)
The unique UUID of the user
id
Responses
Code
Description
Links
200
No links
Media type
application/json
Controls Accept header.
Example Value
Schema
{
"email": "string",
"id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
"username": "string"
}
404
User not found in the system
No links
500
Internal server error - something went wrong on our end
No links

Plan de Travail



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