SOUTENANCE FINALE

16 - Dashboard en technologie egui/rust pour plateforme HPC

Groupe 9

Porteur de projet : Olivier RICHARD

14/03/2025

EQUIPE DE PROJET

Rémi DEL MEDICO

Amaury GOGUILLOT

Romain MIRAS

Alexandre ARLE

COMPÉTENCE

C1. Réaliser des solutions numériques

C1.1 En utilisant des méthodes et des outils adaptés

C1.2 En respectant la réglementation (RGPD, propriété intellectuelle, contrat) et les recommandations (bonne pratique, sécurité, etc.)

C1.3 En optimisant l'utilisation des ressources matérielles et énergétiques

C1.4 En collaborant efficacement

C1.5 En respectant les spécifications et normes techniques

C1.6 En justifiant ses choix de conception

C1.7 En s'adaptant à un environnement technologique et stratégique en constante évolution

CONTEXTE

OAR

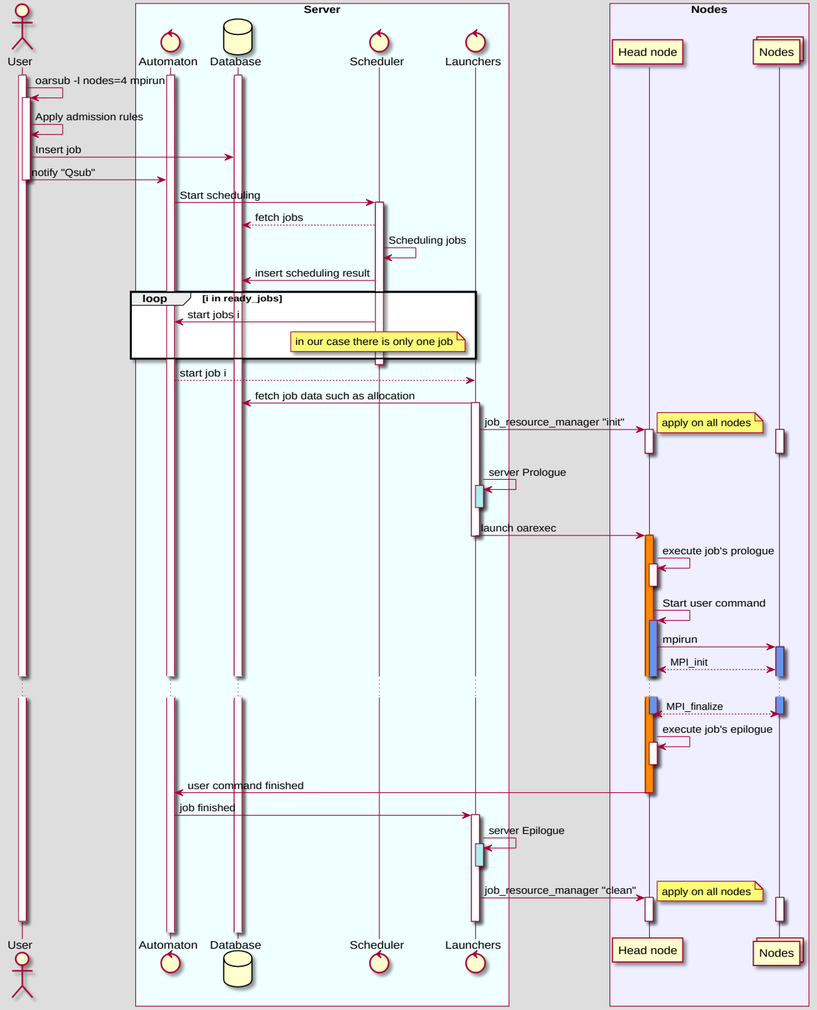
Gestionnaire de ressources et d'ordonnancement de tâches pour clusters HPC

- Outils en ligne de commande (CLI)
- API REST (Python depuis Oar3)

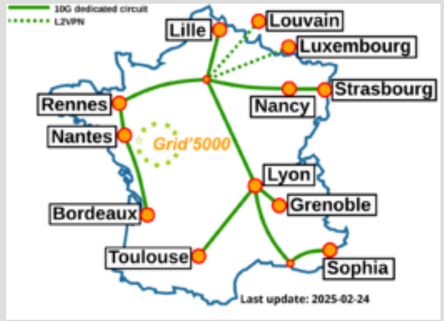
DataMove

Data Aware Large Scale Computing

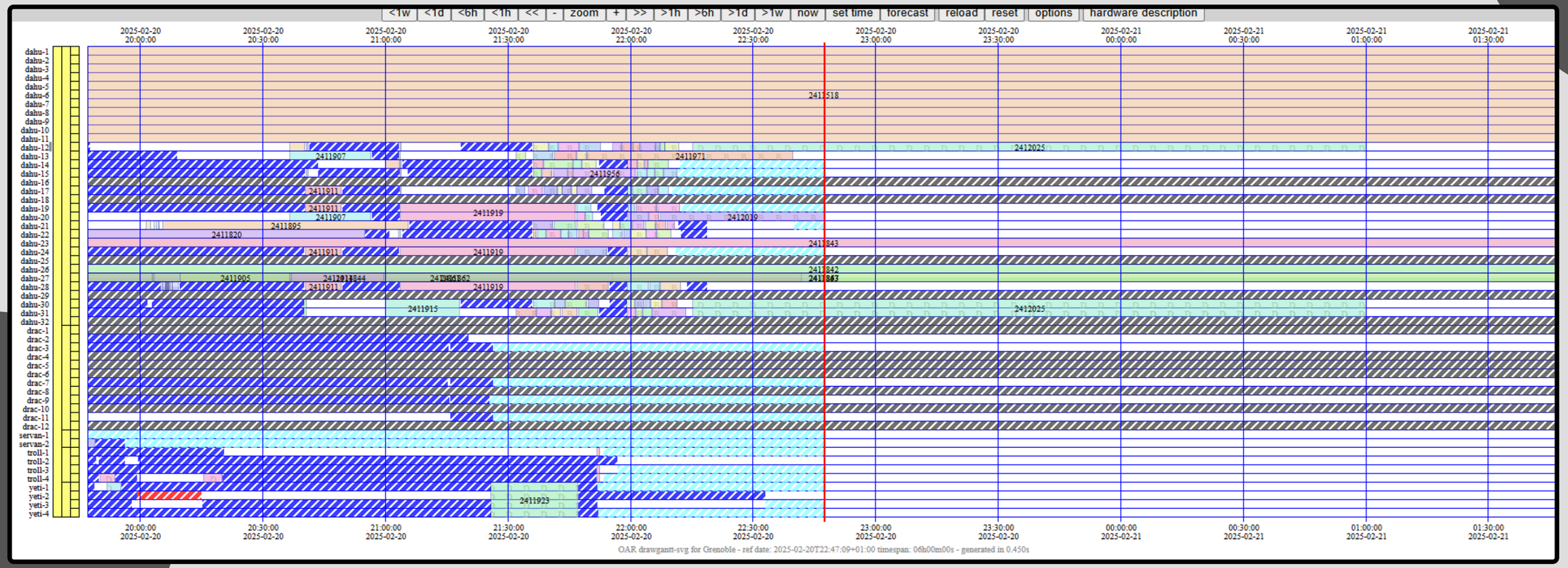
Équipe de recherche DATAMOVE (LIG-INRIA)



CONTEXTE



Plateforme Grid’5000



Interface Gantt de monitoring pour la plateforme Grid’5000

CAHIER DES CHARGES

Fonctionnalités essentielles

- Affichage des tâches en **temps réel** dans une vue **Gantt** interactive avec **navigation temporelle**
- **Filtrage** des tâches
- Affichage **détaillé** des propriétés d'une tâche
- **Aggrégation** des tâches selon leurs propriétés

C1.5 En respectant les spécifications et normes techniques

CAHIER DES CHARGES

Contraintes techniques

- Développement en **Rust**
- Utilisation de la bibliothèque **Egui**
- Communication avec l'**API REST** d'**OAR**
- Support **multi-plateforme** (Linux, macOS, Windows)

Livrables attendus

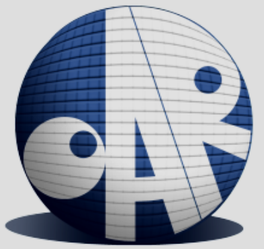
- Application exécutable **nativement**
- Application exécutable dans un **navigateur web**
- Documentation **technique** et **manuel d’installation**
- Code source documenté sous **GitHub**

C1.5 En respectant les spécifications et normes techniques

TECHNOLOGIES

Technologies employées

- Rust et Cargo
- Egui + Eframe (gui instantanée)
- Puffin (flamegraph)
- OAR



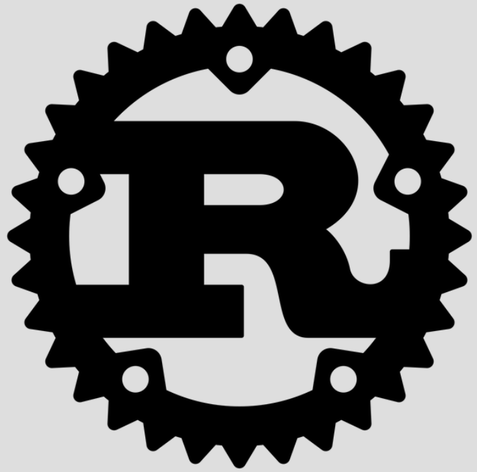
C1.1 En utilisant des méthodes et des outils adaptés

C1.7 En s’adaptant à un environnement technologique et stratégique en constante évolution

TECHNOLOGIES

Rust

- Langage système **sécurisé**, **performant** et **concurrent**
- Garantit une **détection** précoce des **erreurs**



Cargo

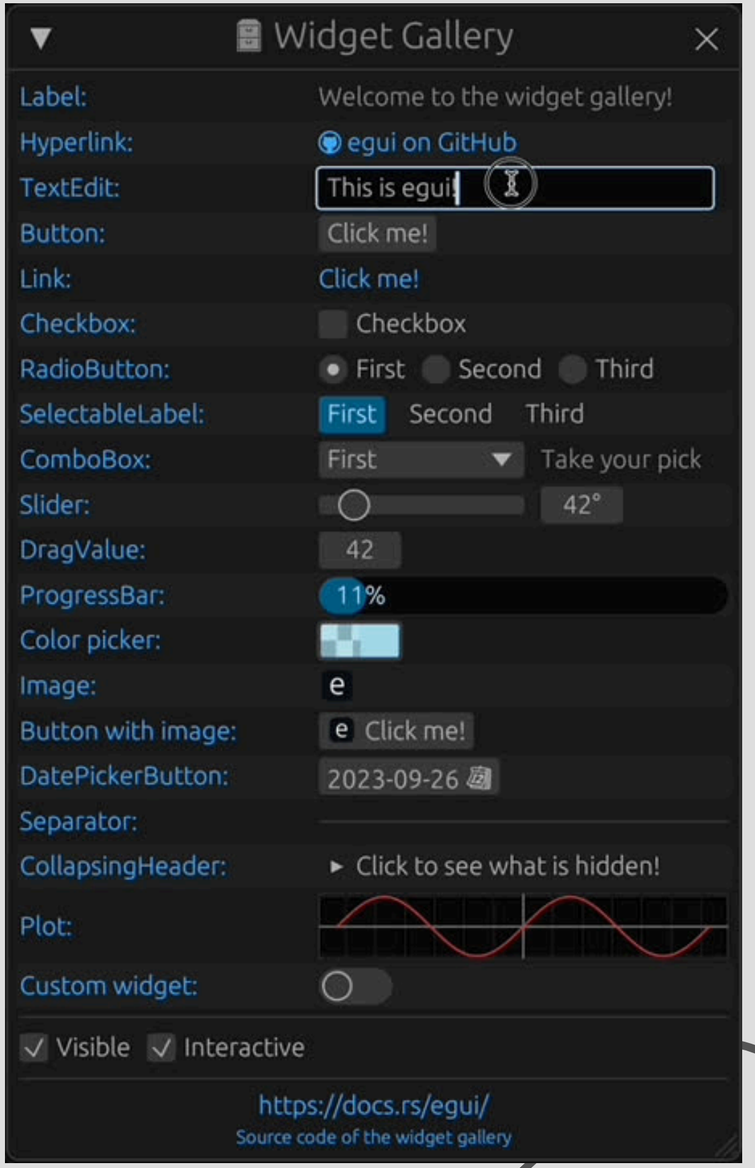
- **Gestionnaire de paquets** dépendances
- Compilation, tests et déploiement via Cargo.toml

C1.3 En optimisant l'utilisation des ressources matérielles et énergétiques

TECHNOLOGIES

Egui & Eframe

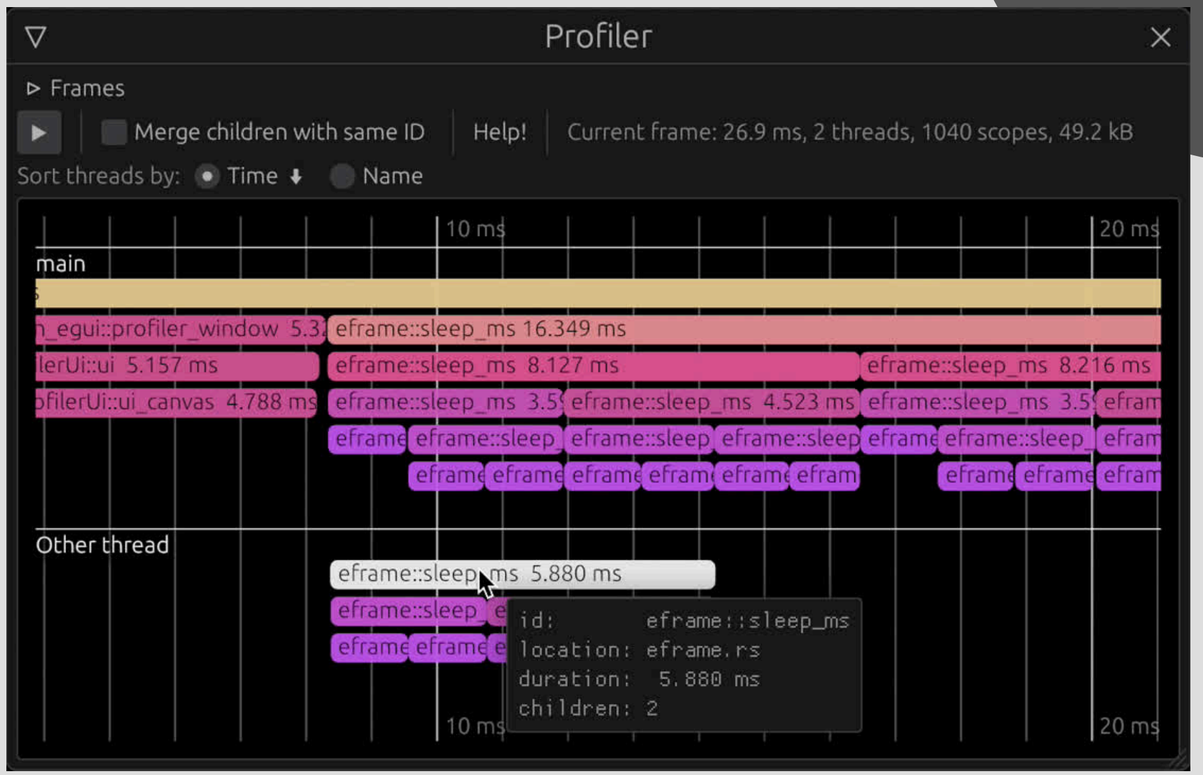
- **Bibliothèque** d'interface graphique immédiate
- **Simplicité** d'utilisation
- **Optimisée** pour le rendu rapide
- Multi-plateforme : **Application Lourde** et **Web** (via WebAssembly)



TECHNOLOGIES

Puffin

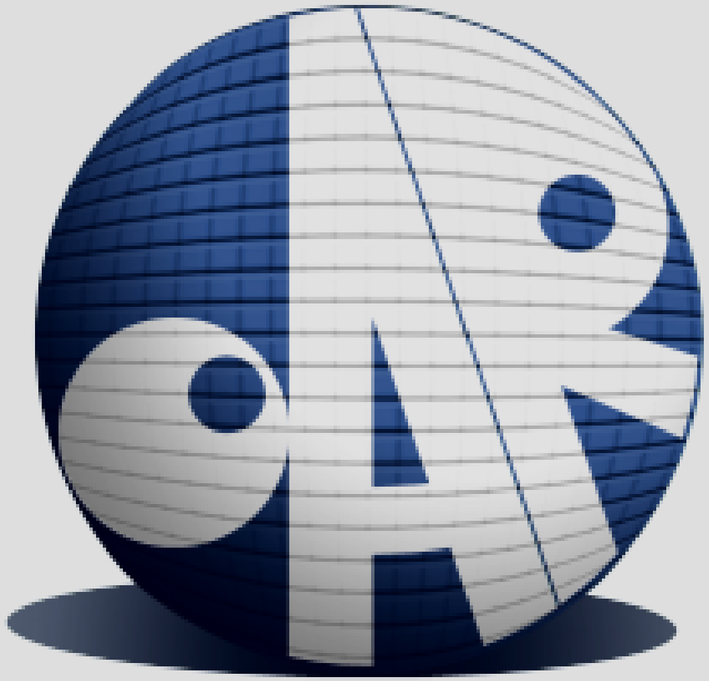
- Profileur de performance **open source**
- Basé sur **Egui Rust**
- Gestion Temporelle **dynamique**
- **Optimisée** pour le rendu rapide
- Base de code pour le Gantt



TECHNOLOGIES

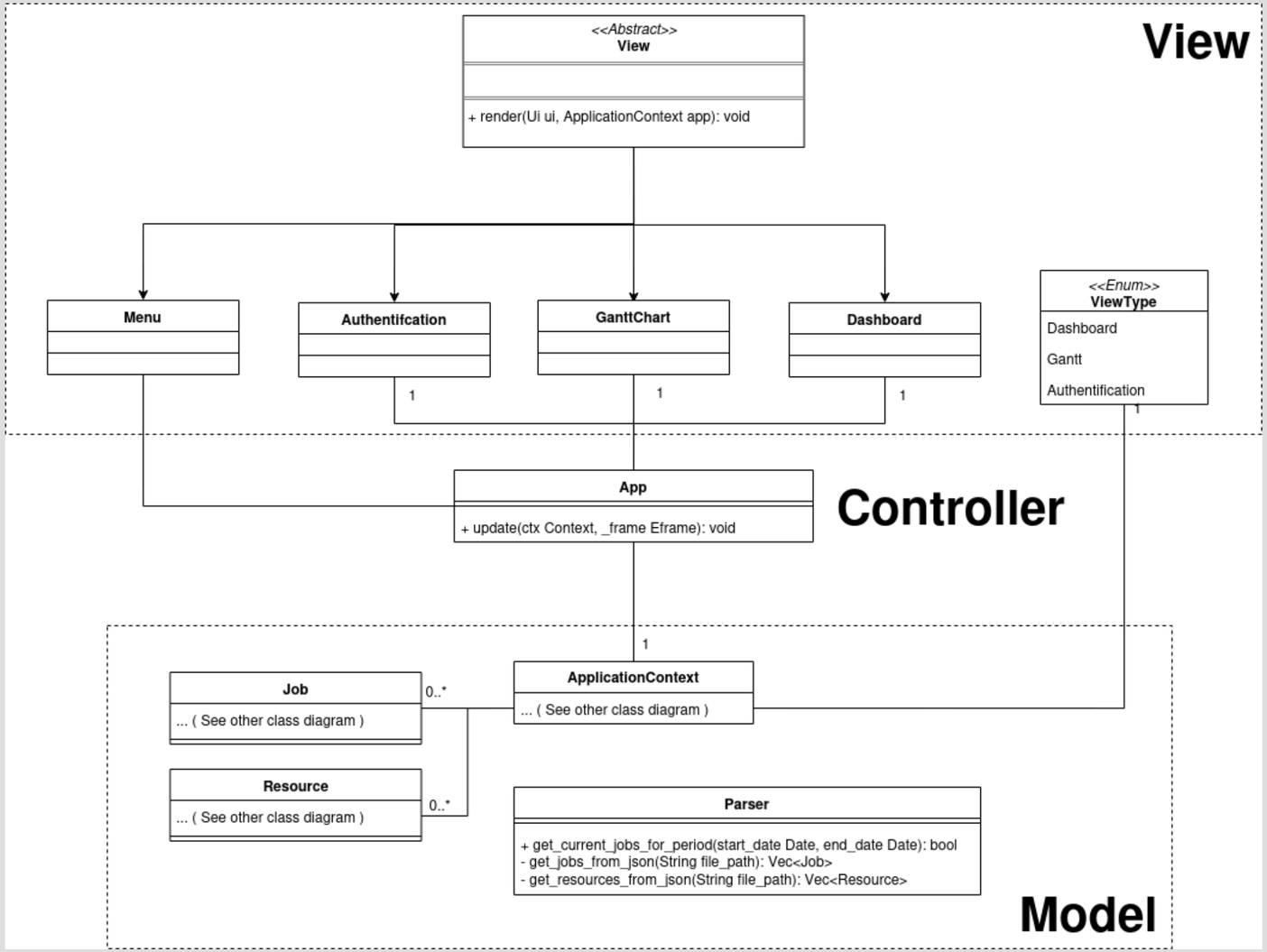
OAR

- **Gestionnaire** de ressources
- **Ordonnanceur** de tâches
- Pour des **clusters** de calcul haute performance
- Permet de récupérer l'**état des ressources**
- **Surveiller** les jobs



ARCHITECTURE

MVC



ARCHITECTURE

Modèle

- Gestion des **jobs OAR** et leurs états
- Stockage des informations sur les **ressources** du cluster
- Gestion des données de **configuration** et des **paramètres utilisateur**
- Communication en **SSH** avec l'instance OAR pour récupérer les données
- **Traitement** et **transformation** des données (filtrage, tri, etc.)

ARCHITECTURE

Vue

- Diagramme de **Gantt** interactif utilisant Egui et basé sur Puffin
- **Tableau de bord** détaillé sur la liste des jobs visualisés
- **Filtres** et contrôles d'interface
- **Rendu graphique** de l'état des ressources
- **Menu** de navigation
- Page d'**authentification** prévue en cas de besoin

ARCHITECTURE

Contrôleur

- Intercepte les **actions utilisateur** provenant de l'interface
- **Met à jour** le modèle en conséquence
- Déclenche le **rafraîchissement** de la vue avec les nouvelles données
- Gère la **logique de navigation** et les **interactions**

RÉALISATIONS

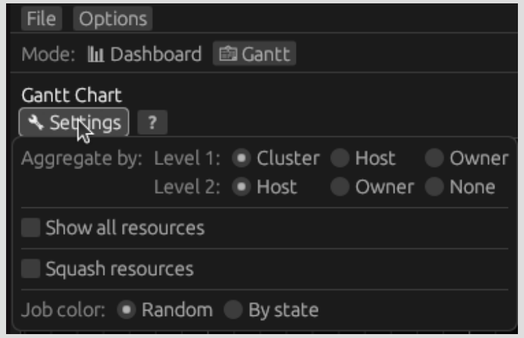
Communication avec OAR

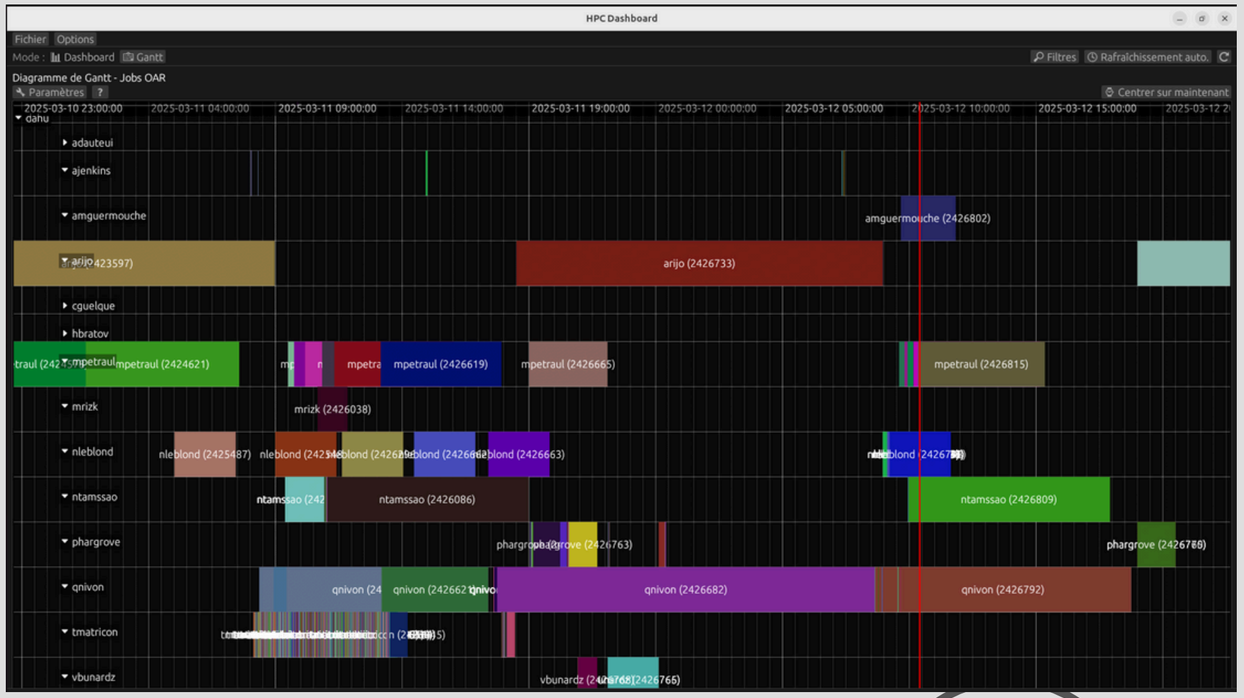
- Parser JSON : bibliothèque *serde*
- Modèles de données
- Cache intelligent
- Gestion d'erreurs robuste

RÉALISATIONS

Diagramme de Gantt

- Framework Puffin
- Navigation temporelle
- Sélection interactive
- Regroupement des ressources
- Agrégation des données

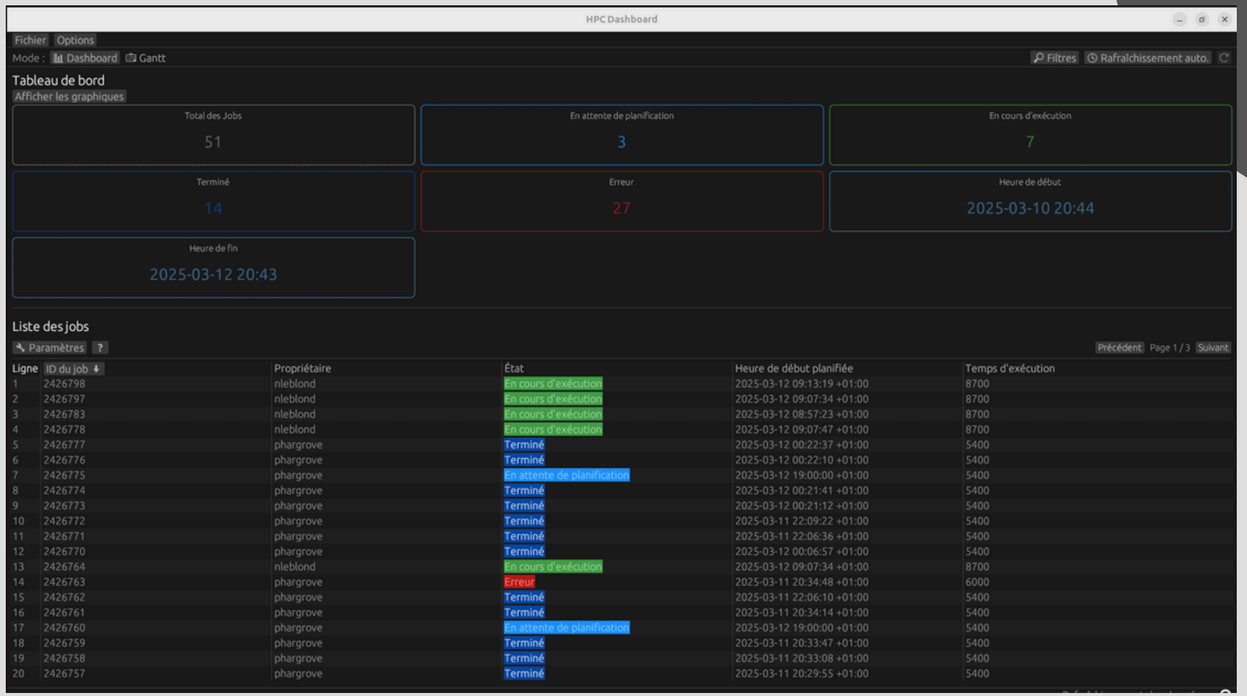




RÉALISATIONS

Tableau de bord détaillé

- Liste des jobs
- Informations détaillées
- Métriques en temps réel
- Graphiques du système



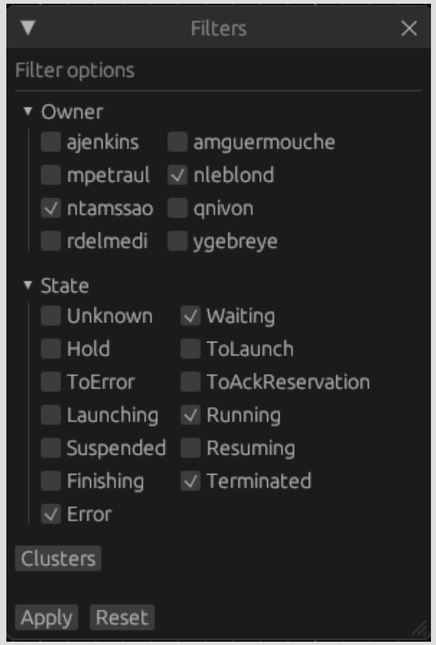
RÉALISATIONS

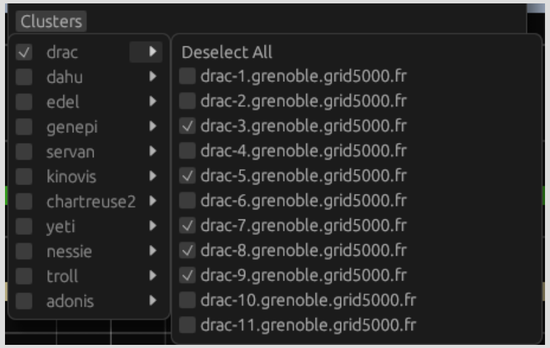
Fenêtre de détails des jobs

- Propriétés du job
- Historique des temporelles
- Resources associées

Filtrage et recherche avancés

Par utilisateur, état, ressources, date





RÉALISATIONS

Interface utilisateur adaptée

- Mode compact / étendu
- Thèmes clair / sombre
- Persistance des préférences
- Intégration i18n
- Personnalisation de la taille de la police
- Multi-plateforme
- Mockup de données pour la version Web

RGPD

Règlement Général sur la Protection des Données

- Traitement des données personnelles
- Les licences des logiciels tiers, droits d’auteur
- Notre propre logiciel est sous licence GPL 2.1
- Respect des contrats et obligations légales
- Sécurité informatique et cybersécurité

C1.2 En respectant la réglementation (RGPD, propriété intellectuelle, contrat) et les recommandations (bonne pratique, sécurité, etc.)

GESTION DE PROJET

Organisation de l'équipe

Rôle	Mission	Membre
Chef de projet	Suivi projet, répartition des tâches, communications avec le porteur du projet	Rémi DEL MEDICO
Git master	Mise en place et maintien du répertoire Git	Alexandre ARLE
Développeur	Conception et de l'implémentation des fonctionnalités ainsi que de la correction des bugs	Romain MIRAS, Amaury GOGUILLOT, Alexandre ARLE, Rémi DEL MEDICO

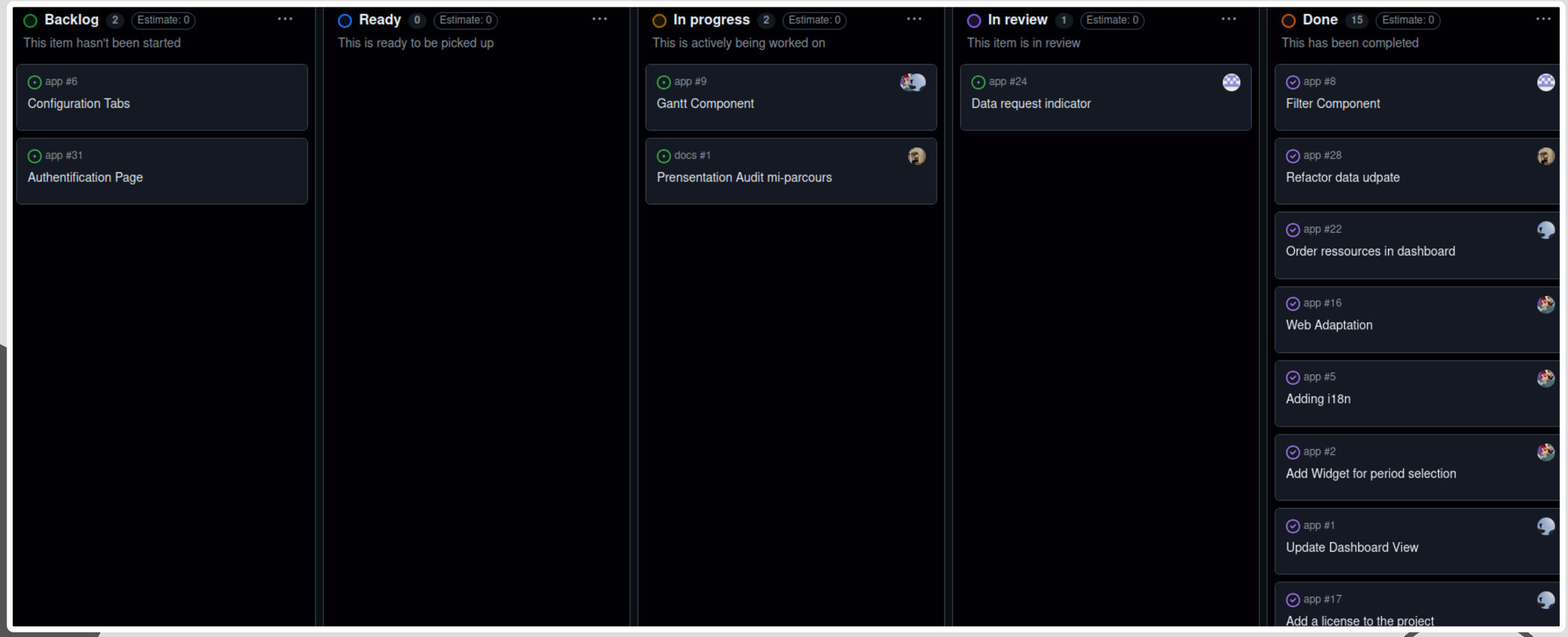
GESTION DE PROJET

Méthodologie Agile & Sprints

	JANVIER	FEVRIER				MARS	
	du 27 au 31	du 3 au 7	du 10 au 14	du 17 au 21	du 24 au 28	du 3 au 7	du 10 au 14
Phase 1 : Analyse							
Analyse des besoins							
Formation Rust/Egui							
Préparation de l'environnement de développement							
Phase 2 : Conception							
Diagramme d'architecture							
Diagramme de séquence							
Phase 3 - Sprint 1 : Développement Interface							
Implémentation fonctionnelle							
Phase 4 - Sprint 2 : Développement Dynamicité							
Implémentation des requêtes dynamique							
Lien requêtes dynamique - déplacement gantt							
Phase 5 : Documentation - Test							
Rédaction de la documentation technique							
Rédaction du manuel utilisateur							
Réalisation des tests							

ORGANISATION

Gestion des Tâches avec un Système de Tickets



C1.4 En collaborant efficacement

ORGANISATION

Suivi du Projet et Communication

- **Réunion physique** => discussion sur les avancées, les éventuels obstacles et les décisions à prendre.
- **Échange par e-mail** => Faire un point sur l’avancement et lever d’éventuelles questions.

C1.7 En s’adaptant à un environnement technologique et stratégique en constante évolution

OUTILS

Gestion de projet et collaboration

- GitHub
  - Repositories : dépôts app et docs
  - Issues a
  - Pull Requests
  - Projects : tableau kanban
- Discord : hub de communication
  - Canaux thématiques
  - Partage de fichiers

C1.1 En utilisant des méthodes et des outils adaptés

C1.4 En collaborant efficacement

OUTILS

Environnement de développement

- Visual Studio Code
  - rust-analyzer
  - Copilot

Documentation

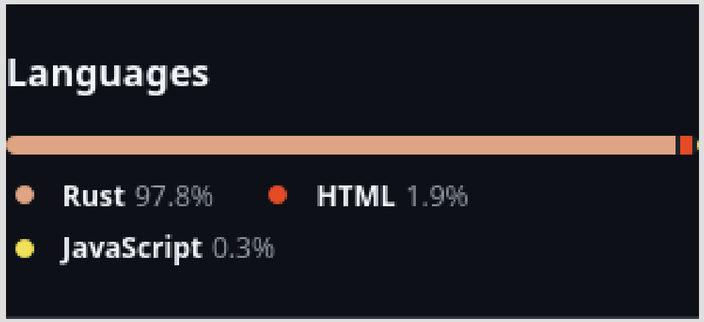
Génie logiciel - diagrammes UML

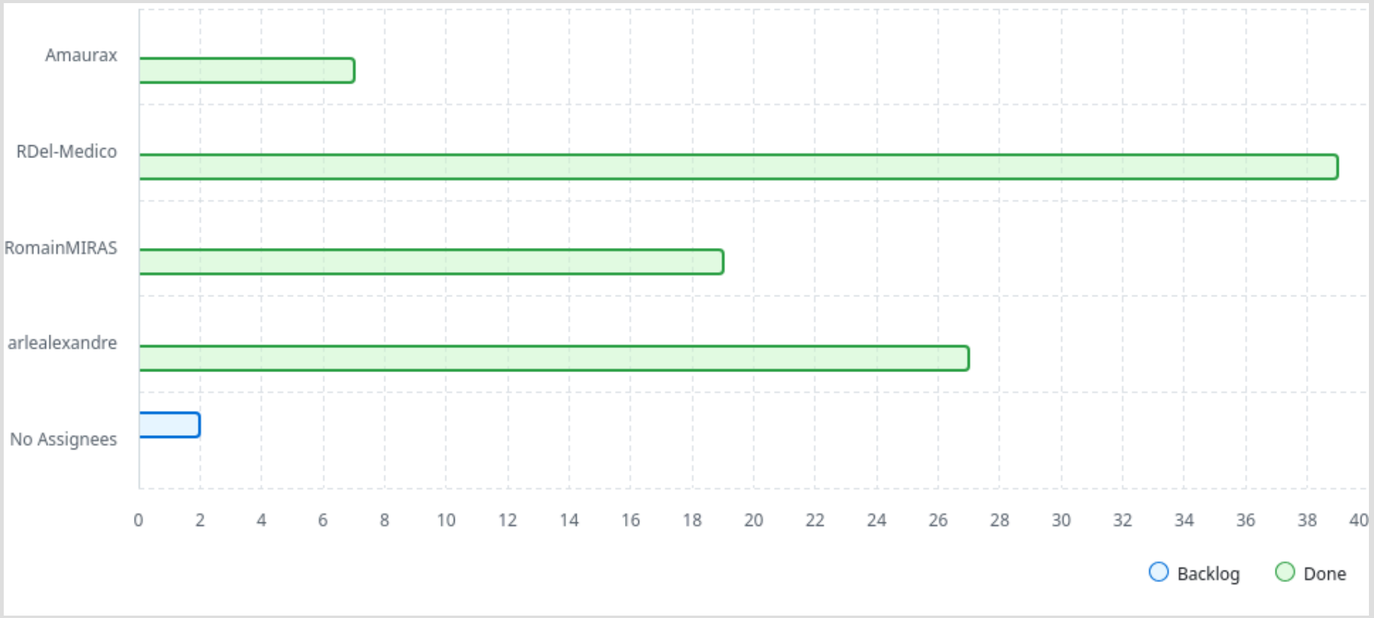
- Diagrammes:
  - contexte
  - cas d'utilisation
  - classes
  - séquence
- Vues :
  - haut niveau
  - détaillées
  - système

C1.6 En justifiant ses choix de conception

METRIQUES LOGICIELLES

- Total lignes de code : **6 450 lignes**
- Total GitHub issues complétées : **84 issues**
- Total commits : **177 commits**
- Total merged pull requests : **94 pull requests**





DÉMONSTRATION

Démo Web

http://79.92.83.218

CONCLUSION

Résultat Obtenu

- Tableau de bord / diagramme de Gantt
- Une interface ergonomique
- Une intégration réussie avec OAR via SSH et JSON
- Une gestion des filtres / agrégation
- Un code Rust robuste et documenté
- Une version web (WebAssembly)

CONCLUSION

Améliorations et Évolutions Futures

- Personnalisation avancée de l’interface
- Prise en charge d’autres gestionnaires que OAR
- Ajout de graphique / option de visualisation
- Utilisation de l’API OAR
- Ajout de la gestion des jobs directement via l'application

Merci de votre attention