+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Mis à jour le 27/06/2025

S'inscrire

Formation Async Rust

3 jours (21 heures)

Présentation

Notre formation Async Rust vous permettra de concevoir des applications hautement concurrentes, non bloquantes et performantes en tirant parti du modèle asynchrone propre à Rust. À travers un parcours complet, vous apprendrez à écrire, structurer et déboguer du code async sûr et efficace.

Vous découvrirez en profondeur le fonctionnement des Future, l'usage d'async/await, les patterns de concurrence (spawn, join, select), ainsi que la gestion fine des I/O asynchrones (fichiers, sockets, HTTP). Chaque notion est appuyée par des ateliers pratiques, vous permettant d'appliquer immédiatement les concepts étudiés.

La formation vous guidera également dans l'utilisation avancée de Tokio : tâches supervisées, cancellation, timeouts, synchronisation asynchrone, et communication inter-tâches avec mpsc ou broadcast. Vous apprendrez à structurer vos applications asynchrones pour éviter les pièges classiques comme les deadlocks ou les attentes infinies.

Vous serez initié aux tests et au débogage d'applications async avec les outils les plus récents et l'instrumentation dynamique.

Comme pour toutes nos formations, elle s'appuie sur les dernières évolutions de l'écosystème Rust async

Objectifs

- Comprendre le fonctionnement des Future en Rust et le modèle de programmation asynchrone sans runtime intégré
- Savoir utiliser async/await pour structurer des fonctions non bloquantes et concurrentes
- Maîtriser l'utilisation des runtimes Tokio et async-std pour exécuter et orchestrer des tâches asynchrones
- Mettre en œuvre des patterns de concurrence efficaces avec spawn, join!, select! et les canaux de communication

- Intégrer des opérations I/O asynchrones (fichiers, réseau, HTTP, base de données) dans une application Rust
- Gérer les erreurs, les timeouts et la cancellation dans un contexte async multi-tâches
- Structurer et tester des applications Rust asynchrones robustes avec tokio::test et les outils de tracing
- Concevoir un projet full-async sécurisé, testable et performant, en appliquant les bonnes pratiques du langage et de son écosystème async

Public visé

- Développeurs Rust
- Architectes logiciels
- Développeurs back-end

Pré-requis

- Maîtriser les fondamentaux du langage Rust

Programme de la formation Async Rust

Introduction à l’asynchronisme en Rust

- Différences entre concurrence, parallélisme et asynchronisme
- Comparaison avec les modèles async de Python, JavaScript, Go
- Futures : qu’est-ce qu’une Future ?
- Le modèle poll et Pin, Waker, Context
- Pas de GC, pas de scheduler intégré ? implication sur le design

Syntaxe de base : async / await

- Retour implicite de impl Future<Output = T>
- Pas d’appel direct : besoin de .await
- Blocage non-bloquant : .await rend la main au runtime
- Possibilité d’attendre plusieurs futures en séquence ou en parallèle
- Création de blocs async temporaires
- Imbrication de blocs dans une logique plus complexe

Utilisation des runtimes

- Architecture de Tokio : scheduler, workers, event loop
- Utilisation de #[tokio::main] et tokio::spawn
- Choix des features de compilation
- API calquée sur la stdlib
- Moins performant que Tokio, mais plus lisible pour débuter
- Cas d’usage pour chaque runtime

Programmation concurrente avec async

- tokio::spawn, join!, select!
- Hiérarchie et supervision des tâches
- tokio::sync::mpsc, broadcast, oneshot
- Pattern producteur / consommateur
- Mutex et RwLock async
- Sémaphores et barrières

Gestion des I/O asynchrones

- tokio::fs, tokio::net
- Utilisation de reqwest et hyper
- Pattern async/await avec client HTTP
- sqlx, sea-orm, surrealdb
- Connexions, transactions, pooling asynchrone

Patterns avancés et bonnes pratiques

- batchs d’APIs, timeouts, cancellation
- tokio::time::timeout, Abortable
- Gérer les tâches longues ou bloquées
- Deadlocks async
- .await dans les locks
- Send et Sync dans les Futures

Debugging et testing d’applications async

- Limitations de dbg! et println! dans les tâches concurrentes
- tokio-console, tracing, instrument
- #[tokio::test] et #[async_std::test]
- Mock des services asynchrones
- Coverage, profiling, benchmarks

Cas pratiques et projets fil rouge

- Routing asynchrone
- Gestion des requêtes parallèles
- Traitement async avec timeouts
- Traitement d’événements avec backpressure
- Retry et gestion d’erreurs
- Backend simple
- SQLx + Axum/Tonic + Tokio

Aller plus loin

- Implémenter son propre Future
- Quand c’est utile ?
- Pourquoi async fn dans les traits est limité
- Usage de async-trait ou dyn Trait avec Box<dyn Future>
- Async avec WASM
- Appels réseau en WebAssembly via futures

Sociétés concernées

Cette formation s’adresse à la fois aux particuliers ainsi qu’aux entreprises, petites ou grandes, souhaitant former ses équipes à une nouvelle technologie informatique avancée ou bien à acquérir des connaissances métiers spécifiques ou des méthodes modernes.

Positionnement à l'entrée en formation

Le positionnement à l'entrée en formation respecte les critères qualité Qualiopi. Dès son inscription définitive, l'apprenant reçoit un questionnaire d’auto-évaluation nous permettant d’apprécier son niveau estimé sur différents types de technologies, ses attentes et objectifs personnels quant à la formation à venir, dans les limites imposées par le format sélectionné. Ce questionnaire nous permet également d’anticiper certaines difficultés de connexion ou de sécurité interne en entreprise (intraentreprise ou classe virtuelle) qui pourraient être problématiques pour le suivi et le bon déroulement de la session de formation.

Méthodes pédagogiques

Stage Pratique : 60% Pratique, 40% Théorie. Support de la formation distribué au format numérique à tous les participants.

Organisation

Le cours alterne les apports théoriques du formateur soutenus par des exemples et des séances de réflexions, et de travail en groupe.

Validation

À la fin de la session, un questionnaire à choix multiples permet de vérifier l’acquisition correcte des compétences.

Sanction

Une attestation sera remise à chaque stagiaire qui aura suivi la totalité de la formation.