

Composant Serveur API REST pour WinDev®

Léger.

Rapide.

Gratuit & Open Source.

Sans serveur IIS/Apache/...

Sans licence Serveur WebDev®.

Pas de SAAS, pas d'abonnement.

Windows 32/64 & Linux

Présentation

NEW : LightREST V3

La Documentation



Téléchargements

Un coup de main ?

Vous trouverez ci-dessous les questions habituellement posées au sujet de LightREST (techniques, fonctionnelles, comparatives, marketing, DSI, aspects juridiques, licence, ...)

S'il vous reste des interrogations, n'hésitez pas à utiliser le [formulaire de contact](#).



Questions techniques



LightREST nécessite-t-il IIS, Apache ou Nginx ?

Non. LightREST embarque son propre serveur HTTP/HTTPS (moteur GO) et fonctionne de façon autonome, sans IIS/Apache/Serveur WebDev®.

L'exécutable WinDev® écoute directement le port HTTP, et reçoit les requêtes directement sans aucun intermédiaire chronophage.

Il peut toutefois être placé derrière un reverse proxy (Nginx/Traefik/HAProxy) si nécessaire.



LightREST fonctionne-t-il sous Linux ?

Oui. LightREST fonctionne non seulement sous **Linux 64 bits** (c'est le cas de nombreux serveurs LightREST déployés), mais aussi sous **Windows 32/64 bits** (binaires natif GO compilés spécifiquement pour chaque OS).



LightREST est-il multi-thread / concurrent ?

Oui. Le moteur GO gère la concurrence nativement (goroutines).

LightREST permet :

- Traitement **simultané** de multiples requêtes

- **Limitation** de charge via

`MaxRequests`

- Gestion fine des **timeouts**

(`ReadTimeout` , `WriteTimeout` ,
`IdleTimeout` , `RequestTimeout`).



Comment sont gérés les timeouts ?

LightREST distingue plusieurs niveaux :

- **ReadTimeout** : lecture socket

- **WriteTimeout** : écriture réponse

- **IdleTimeout** : inactivité keep-alive, permet de laisser la connexion ouverte entre 2 requêtes et d'éviter une renégociation (coûteuse en SSL).

- **RequestTimeout** : exécution logique de la route

Cela permet un contrôle fin des performances et de la robustesse.



LightREST est-il sécurisé ?

LightREST propose plusieurs briques :

- **HTTPS** (certificat fourni, auto-signé ou Let's Encrypt)
- **Hooks** serveur et route
- Limitation du nombre de **requêtes simultanées**
- **Chiffrement** d'identifiants (`CipherID`) pour éviter le pillage des données
- **Gestion centralisée** des erreurs

La sécurité globale dépend aussi de la stratégie d'authentification et de déploiement.



Peut-on utiliser LightREST derrière un load balancer ?

Oui. LightREST est **stateless par défaut** : il peut être placé derrière un load balancer (Nginx, HAProxy, ELB, Application Gateway...).



Puis-je utiliser LightREST comme serveur back-end pour mon application WinDev Mobile® ?

Oui. LightREST est parfaitement adapté comme serveur back-end pour une application WinDev Mobile®. Il permet :

- d'exposer des services REST consommables en HTTP/JSON
- de ne pas exposer publiquement un serveur HFSQL
- de centraliser la logique métier côté serveur
- de sécuriser les échanges via HTTPS
- de gérer la montée en charge indépendamment des clients mobiles

Cette architecture permet de découpler totalement l'application mobile du moteur métier, tout en conservant l'écosystème WinDev® côté serveur.



Pourquoi le moteur LightREST est-il développé en GO et non directement en WinDev® ?

Le moteur LightREST est développé en GO pour bénéficier de mécanismes systèmes avancés particulièrement adaptés aux serveurs HTTP haute performance.

Créé par Google en 2009, GO est devenu un standard de facto pour les infrastructures cloud et les architectures distribuées. Sa large adoption

industrielle, son évolution continue et son écosystème mature garantissent une forte stabilité et une excellente visibilité à long terme.

Il apporte notamment :

- Les **goroutines** : gestion native de la concurrence à très faible coût.
- Les **channels** : communication sûre et structurée entre routines.
- Les primitives de synchronisation du package **sync** : **Mutex**, **RWMutex**, **WaitGroup**...
- Les variables atomiques via **sync/atomic** pour garantir la cohérence sans verrou global.
- Un **modèle mémoire** clair et sécurisé.
- Une compilation native produisant un **binaire autonome**, exécutable directement, sans interprétation chronophage d'un **script**, sans machine virtuelle, et sans runtime externe.

WinDev® est quand à lui excellent pour la logique métier et l'accès aux données. GO est particulièrement adapté pour le moteur réseau, la gestion fine des sockets, la performance concurrente et la robustesse système.

LightREST combine ainsi le meilleur des deux mondes :

- WinDev® pour le métier
- GO pour l'infrastructure serveur



LightREST est-il adapté aux environnements à forte charge ?

Oui. Grâce au moteur GO et à sa gestion efficace de la concurrence, LightREST peut traiter simultanément un grand nombre de requêtes.

Les mécanismes disponibles incluent :

- limitation configurable du nombre de requêtes (`MaxRequests`)
- timeouts multi-niveaux
- gestion concurrente via `goroutines`
- synchronisation fine via `sync` et `atomic`

Les performances réelles dépendront principalement de la logique métier exécutée côté WinDev® et de l'infrastructure de déploiement.



Questions fonctionnelles



À quoi sert LightREST concrètement ?

LightREST permet de transformer une application WinDev® en **API REST** :

- Exposition de routes HTTP/REST
- Réponses JSON/TXT
- Authentification personnalisée
- Monitoring et logs
- CORS et headers globaux



Peut-on créer des routes dynamiques ?

Oui. Les routes peuvent inclure des paramètres (ex: `/client/{id}`), des méthodes HTTP distinctes (GET/POST/PUT/DELETE...), et des timeouts spécifiques par route. Les routes sont définies dans le code (pas de paramétrage fastidieux dans un outil tiers).



LightREST gère-t-il le CORS ?

Oui. On peut définir des headers globaux et des headers spécifiques aux réponses `OPTIONS`, qui sont ajoutés automatiquement dans les réponses REST.



Peut-on ajouter une logique avant/après chaque requête ?

Oui via les **Hooks** :

- Niveau serveur
- Niveau route
- Interception d'événements (connexion, received, before_handler, after_handler, result, warning, error, panic, ...).

 LightREST vs WebDev®**Pourquoi utiliser LightREST plutôt que WebDev® ?**

Pour un usage **API pur**, LightREST apporte :

- Pas de serveur WebDev® requis
- Pas de licence serveur WebDev®, pas d'abonnement SAAS
- Pas de dépendance IIS
- Binaire autonome, déploiement rapide (simplement un exécutable ou un service à lancer)
- Moteur GO optimisé pour le HTTP/REST
- Définition des routes REST directement dans le code, à côté des fonctions métier. Maintenance simplifiée et fiabilisée.

**LightREST remplace-t-il WebDev® ?**

Non. WebDev® est un framework web complet (pages, sessions, état). LightREST est un moteur **REST pur**, léger et orienté API.

 LightREST peut être utilisé parallèlement à WebDev® pour prendre en charge l'exécution optimisée d'API REST, WebDev® assurant son rôle de serveur d'applications Web.



Les performances sont-elles meilleures que WebDev ?

Dans un usage API pur, généralement oui : moins de couches intermédiaires, pas de moteur page/session, serveur GO efficace.

Les performances exactes dépendent du métier exécuté côté WinDev® et du déploiement.



Peut-on migrer progressivement depuis WebDev® ?

Oui. On peut conserver WebDev® pour certaines APIs et exposer progressivement des routes via LightREST, sans big-bang.



Juridique & licence MIT / open source



LightREST est-il open source ?

Oui. LightREST est distribué sous licence **MIT**.

Le code distribué inclut le moteur LightREST en GO et le composant WinDev® multi-OS.



Que permet la licence MIT ?

La licence MIT permet sans aucune limite :

- Usage commercial
- Modification
- Intégration dans des projets

propriétaires

- Redistribution

Seule obligation : **conserver l'avis de copyright/licence.**



Dois-je publier mon code si j'utilise LightREST ?

Non. La licence MIT n'impose pas la publication du code source de l'application.



LightREST est-il gratuit ?

Oui pour l'usage du moteur : pas d'abonnement, pas de licence serveur, pas de limitation artificielle.

Un support professionnel ou de la formation peuvent être proposés en option par CODE LINE.

 Puis-je intégrer LightREST dans un logiciel commercial sans ouvrir mon code ?

Oui. La licence MIT autorise l'usage commercial sans obligation de publication du code source.

Vous pouvez :

- intégrer LightREST dans un logiciel propriétaire
- modifier le moteur
- redistribuer votre solution

La seule obligation est de conserver la mention de licence MIT.

 Questions marketing & positionnement **Pourquoi LightREST est-il différent des autres frameworks REST ?**

LightREST n'est pas un framework générique : il est conçu pour l'écosystème WinDev® et l'exposition d'API HTTP/JSON, avec un déploiement simple et autonome (sans IIS/Apache/Serveur WebDev®).

 À qui s'adresse LightREST ?

Aux équipes WinDev® et aux organisations qui veulent :

- Exposer des API modernes
- Découpler front/back
- Intégrer mobile/web/partenaires
- Moderniser sans réécriture brutale

 LightREST est-il adapté aux projets critiques ?

Oui : moteur GO robuste, déploiement contrôlé, timeouts et limitations, hooks et gestion d'erreurs. Comme toute brique, il doit être intégré dans une stratégie sécurité/monitoring adaptée.

 LightREST est-il pérenne ?

Oui : licence MIT, code auditable, pas de dépendance imposée à un serveur applicatif tiers, déploiement autonome.

 Pourquoi éviter une solution SaaS ?

LightREST se déploie **chez vous** : pas d'abonnement, pas de dépendance cloud obligatoire, contrôle total sur les données et l'exploitation.

**LightREST permet-il d'ouvrir mon SI vers l'extérieur ?**

Oui : exposition d'API sécurisées pour partenaires, applications mobiles, front web moderne, automatisations, etc.

**LightREST est-il compatible avec une stratégie API-first ?**

Oui : séparation claire entre UI et métier, standard HTTP, intégration facile avec des frontends modernes et des systèmes tiers.

**Le point de vue des DSI / Architectes IT****LightREST respecte-t-il les standards HTTP ?**

Oui : méthodes HTTP standard, codes HTTP cohérents, headers configurables, support HTTPS.



Peut-on l'intégrer dans une architecture existante ?

Oui : reverse proxy, load balancer, cluster, conteneurisation (Docker), orchestration (Kubernetes) - selon votre contexte.



Quelles garanties de sécurité ?

HTTPS (certificat/auto-signé/Let's Encrypt), hooks, timeouts, limitation de charge, gestion d'erreurs.
L'authentification/autorisation se branche via vos mécanismes (hooks, headers, tokens...).



LightREST supporte-t-il la scalabilité horizontale ?

Oui : architecture stateless par défaut → duplication d'instances derrière un load balancer.



Quel impact sur la maintenance ?

Faible : binaire autonome, moins de dépendances serveur, déploiement simple, configuration claire.

 Quelles dépendances logicielles ?

Aucune dépendance serveur obligatoire : pas d'IIS, pas d'Apache, pas de runtime externe à installer côté serveur (hors vos choix : proxy, monitoring...).

 RGPD : compatible ?

Oui : LightREST n'impose aucun stockage. La conformité dépend des données que vous traitez et de votre implémentation (logs, consentements, durées de conservation...).

 Peut-on auditer le code ?

Oui : licence MIT, code inspectable et modifiable.

 Quel est le modèle économique ?

Moteur gratuit (MIT).
Support/accompagnement possibles en option (conseil, intégration, architecture, durcissement, etc.).



Pourquoi ne pas utiliser .NET Web API / Java / Node ?

Ces stacks sont très valables, mais dans un SI WinDev® existant, LightREST permet de **capitaliser sur le métier et le patrimoine applicatif** sans réécriture massive et sans double maintenance prolongée.



Et si le projet s'arrête ?

La licence MIT permet de continuer à utiliser, maintenir et adapter le code sans dépendance contractuelle.



LightREST peut-il être utilisé comme couche d'API centralisée pour plusieurs applications (Desktop, Mobile, Web) ?

Oui. LightREST permet d'exposer une couche d'API centralisée et mutualisée.

Une même instance peut servir :

- une application WinDev® Desktop
- une application WinDev Mobile®
- un front web moderne (React, Vue, Angular...)
- des partenaires externes

Cela favorise une architecture orientée services et réduit la duplication de logique métier.



Le choix de GO améliore-t-il la stabilité en production ?

Oui. Go est reconnu pour sa stabilité et sa robustesse dans les environnements serveurs.

Son modèle de concurrence maîtrisé, l'absence de gestion manuelle de mémoire, et la simplicité du binaire déployé réduisent considérablement les risques :

- pas de dépendance IIS
- pas de serveur applicatif tiers
- pas de runtime lourd à maintenir
- comportement déterministe sous forte charge

Cela en fait un choix pertinent pour un moteur REST industriel.



Quelle solution en cas de besoin d'assistance technique ?

CODE LINE peut accompagner les équipes de développeurs dans la mise en œuvre et l'optimisation de LightREST.

Les prestations peuvent inclure :

- Audit d'architecture et recommandations techniques
- Aide à la conception d'API REST robustes
- Formation des développeurs (hooks, gestion des timeouts, sécurité, bonnes pratiques)
- Optimisation des performances et gestion de la concurrence
- Assistance au déploiement (reverse proxy, HTTPS, load balancing, Docker)
- Revue de code et durcissement sécurité

L'objectif est de sécuriser les choix techniques, d'accélérer la montée en compétence des équipes et de garantir une mise en production maîtrisée.



Moderniser WinDev® : REST plutôt que réécriture

 Pourquoi ne pas réécrire entièrement l'application dans une autre technologie ?

Une réécriture complète est souvent longue, coûteuse et risquée.

Transformer l'existant en **serveur REST** permet de moderniser progressivement tout en conservant le cœur métier et en capitalisant sur le patrimoine de fonctions métier.

 Quels sont les risques d'une réécriture complète ?

Principaux risques :

- **Perte de logique métier** (règles implicites, cas limites, corrections accumulées)
- **Explosion des coûts** (nouvelles compétences, double maintenance)
- **Allongement des délais** (recette, migration, conduite du changement)
- **Risque projet élevé** (retards, dépassements, abandon)



Quels sont les avantages d'une transformation REST ?

Exposer le métier via HTTP/JSON

permet :

- Conservation du cœur WinDev®
- Découplage UI / backend
- Ouverture vers

web/mobile/partenaires

- Modernisation progressive sans rupture



Est-ce une modernisation réelle ou un simple "pansement" ?

C'est une modernisation structurelle : en REST, le backend devient un service consommable par n'importe quel front (React/Vue/Angular, mobile, partenaires). Cela prépare une architecture API-first.



Peut-on migrer progressivement vers une autre technologie ?

Oui. L'approche REST permet une migration **module par module** (ou microservice par microservice), au lieu d'un big-bang.



Quel ROI comparé à une réécriture ?

Transformer via REST réduit les délais de mise en production, limite les risques et valorise l'existant. La réécriture est souvent un pari ; la transformation REST est une stratégie maîtrisée.

© CODE LINE 2018-2026

