Durée :	4 jours soit 28 heures	Référence :	IF-LANG-GO

Public visé :

- Développeurs (Back-end, Fullstack, Système) souhaitant monter en compétence sur un langage moderne.
- Architectes logiciels et Ops/SRE souhaitant comprendre ou maintenir des outils comme Docker ou Kubernetes (écrits en Go).

Pré-requis :

- Maîtrise d'au moins un langage de programmation structuré ou objet (Java, C/C++, Python, PHP, etc.).
- Familiarité avec l'usage de la ligne de commande (Terminal).
- Des notions sur les protocoles HTTP et le format JSON sont un plus.

Objectifs pédagogiques :

À l'issue de cette formation, le stagiaire sera capable de :

- **Installer et configurer** un environnement de développement Go standard.
- **Appliquer les spécificités syntaxiques** (typage statique, structures, pointeurs) pour écrire un code robuste.
- **Concevoir des interfaces** pour créer des architectures découplées et testables.
- **Implémenter la concurrence** via les Goroutines et les Channels en évitant les "race conditions".
- **Développer et tester une API REST** conforme aux standards du marché.
- **Packager et optimiser** une application Go pour un déploiement en production.

Compétences acquises à l'issue de la formation :

- Installer et configurer un environnement de développement Go standard.
- Appliquer les spécificités syntaxiques (typage statique, structures, pointeurs) pour écrire un code robuste.
- Concevoir des interfaces pour créer des architectures découplées et testables.
- Implémenter la concurrence via les Goroutines et les Channels en évitant les "race conditions".
- Développer et tester une API REST conforme aux standards du marché.
- Packager et optimiser une application Go pour un déploiement en production.

Modalités pédagogiques :

Session dispensée en présentiel ou téléprésentiel, selon la modalité inter-entreprises ou intra-entreprises sur mesure. La formation est animée par un(e) formateur(trice) durant toute la durée de la session et présentant une suite de modules théoriques clôturés par des ateliers pratiques validant l'acquisition des connaissances. Les ateliers peuvent être accompagnés de Quizz. L'animateur(trice) présente la partie théorique à l'aide de support de présentation, d'animation réalisée sur un environnement de démonstration. En présentiel comme en téléprésentiel, l'animateur(trice) accompagne les participants durant la réalisation des ateliers.

Note relative au programme : Les ateliers, démonstrations, travaux pratiques et scénarios présentés dans ce programme sont fournis à titre d'exemples illustratifs et ne sont pas exhaustifs. Afin de garantir une adéquation permanente avec les évolutions technologiques, le contenu pédagogique ainsi que les environnements de travaux pratiques (TP) sont susceptibles d'être adaptés, ajustés ou mis à jour au cours du temps.

Moyens et supports pédagogiques :

Cadre présentiel

Salles de formation équipées et accessibles aux personnes à mobilité réduite.

- Un poste de travail par participant
- Un support de cours numérique ou papier (au choix)
- Un bloc-notes + stylo
- Vidéoprojection
- Connexion Internet

- Accès extranet pour partage de documents et émargement électronique

Cadre téléprésentiel

Session dispensée via notre solution iClassRoom s'appuyant sur Microsoft Teams.

- Un compte Office 365 par participant
- Un poste virtuel par participant
- Un support numérique (PDF ou Web)
- Accès extranet pour partage de documents et émargement électronique

Modalités d'évaluation et de suivi :

Avant

Afin de valider le choix d'un programme de formation, une évaluation des prérequis est réalisée à l'aide d'un questionnaire en ligne ou lors d'un échange avec le formateur(trice), qui valide la base de connaissances nécessaires.

Les stagiaires disposent également, depuis leur extranet, d'un **outil d'auto-positionnement** leur permettant d'évaluer leur niveau de maîtrise des compétences visées par la formation. Cette auto-évaluation est réalisée avant le démarrage de la formation à l'aide d'un curseur de positionnement allant du niveau le plus faible au niveau le plus élevé.

Pendant

Après chaque module théorique, un ou plusieurs ateliers pratiques permettent de valider progressivement l'acquisition des connaissances. Un quiz peut accompagner les ateliers pratiques afin de mesurer la compréhension des notions abordées.

Après

À l'issue de la formation, les stagiaires réalisent une nouvelle **auto-évaluation** sur les mêmes compétences que celles évaluées avant la formation. Cette comparaison permet de mesurer leur perception de la progression réalisée et de visualiser les compétences développées au cours de la formation.

Lorsque le programme de formation le prévoit, un examen de certification est organisé conformément aux modalités définies par l'éditeur ou le centre de certification (TOSA, Pearson VUE, ENI, PeopleCert…).

Le formateur réalise également une **évaluation individuelle des compétences** de chaque stagiaire sur l'ensemble des compétences visées par la formation. Pour chacune d'elles, il indique si la compétence est :

- Acquise ;
- En cours d'acquisition ;
- Non acquise.

Lorsque la compétence est évaluée comme **en cours d'acquisition** ou **non acquise**, le formateur formule des observations précisant les difficultés rencontrées ainsi que les actions ou recommandations permettant au stagiaire de poursuivre sa montée en compétences.

Enfin

Un questionnaire de satisfaction permet au participant d'évaluer la qualité de la prestation de formation.

Informations sur l'accessibilité :

Nos formations sont, dans la mesure du possible, conçues pour être accessibles à toutes et à tous. Afin de garantir les meilleures conditions d'accueil et d'apprentissage pour les personnes en situation de handicap, nous vous invitons à contacter notre référente handicap certifiée :
Céline SOLATGES – 05 61 34 39 80 – csolatges@iform.fr
Nous vous remercions de bien vouloir nous communiquer toute information utile à ce sujet en amont de la formation, afin de mettre en place les adaptations nécessaires et d'assurer un accompagnement optimal.

Pour en savoir plus sur les dispositifs d'accompagnement existants, vous pouvez consulter les sites suivants :

- AGEFIPH
- FIPHFP
- MDPH de votre département

Description / Contenu

Module 1 : Philosophie et Fondamentaux de Go

- **Introduction :** Genèse chez Google, pourquoi Go ? (simplicité vs performance).
- **Environnement :** Installation, gestion des modules (go mod), structure d'un projet.
- **Syntaxe de base :** Variables, constantes, inférence de type et typage strict.
- **Structures de contrôle :** if, switch, et l'unique boucle for.
- **Fonctions :** Retours multiples, paramètres nommés, fonctions anonymes et closures.
- **TP :** Création d'un utilitaire CLI pour manipuler des données d'entrée.

Module 2 : Types de données et Gestion de la mémoire

- **Collections :** Tableaux (arrays), Slices (mécanisme interne, capacité/longueur) et Maps.
- **Pointeurs :** Comprendre le passage par valeur vs référence sans la complexité du C.
- **Structures :** Définition, composition (à la place de l'héritage) et tags de structure (JSON).
- **Méthodes :** Associer des comportements aux types.
- **TP :** Modélisation d'un système de gestion de stocks avec structures imbriquées.

Module 3 : Interfaces, Erreurs et Programmation Défensive

- **Interfaces :** L'implémentation implicite (le "Duck Typing" statique).
- **Gestion des erreurs :** La philosophie Go (pas d'exceptions), le type error.
- **Contrôle de flux avancé :** defer, panic et recover.
- **Assertions de type :** Switch de type et interfaces vides.
- **TP :** Création d'un système de logging multi-support (Fichier/Console) via les interfaces.

Module 4 : Maîtrise de la Concurrence (Le cœur de Go)

- **Goroutines :** Parallélisme léger et ordonnanceur de Go.
- **Channels :** Communication et synchronisation entre processus.
- **Patterns de concurrence :** Select, WaitGroups et Mutex (package sync).
- **Détection de bugs :** Utilisation du "Race Detector".
- **TP :** Développement d'un agrégateur de données asynchrone (Scraper de données en parallèle).

Module 5 : Tooling, Tests et Qualité de code

- **Tests unitaires :** Le package testing, benchmarks et exemples.
- **Qualité :** Formatage (gofmt), peluchage (go vet, golangci-lint).
- **Documentation :** Générer une doc propre avec godoc.
- **Profilage :** Introduction à l'analyse de performance (CPU/Mémoire).
- **TP :** Mise en place d'une suite de tests avec couverture de code (Code Coverage).

Module 6 : Vers la Production : API REST et Déploiement

- **Bibliothèque standard :** net/http pour créer un serveur web.
- **Frameworks :** Introduction à l'écosystème (ex: Gin ou Echo).
- **Persistance :** Interaction avec une base de données SQL (PostgreSQL) et notions d'ORM (GORM).
- **Packaging :** Compilation statique et création de Dockerfiles optimisés (images distroless/scratch).
- **TP final :** Développement complet d'une API REST de gestion de tâches, connectée à une base de données et conteneurisée.