# Introduction

En cette période où de nombreux retours d'expérience sur des projets à succès ont placé Go sur le devant de la scène, ce livre vise à renseigner le développeur sur les valeurs, les raisons d'être, les avantages, mais aussi les limites de Go, aussi bien sur le plan du langage que de la plateforme, et plus généralement de son écosystème. Au-delà de vous apprendre à écrire du Go, notre objectif est de vous donner une compréhension intelligente de pourquoi écrire du Go et comment mettre à profit ses points forts en vue de l'utiliser de manière pertinente sur des projets réels ; et de savoir aussi quand ne pas l'utiliser. Ce livre s'adresse aux développeurs ayant déjà acquis de l'expérience sur d'autres langages, et nous partons du présupposé que vous avez déjà codé des projets entiers, aussi modestes soient-ils. Ainsi, pour faciliter la compréhension des choix et compromis de Go, nous n'hésiterons pas à le comparer aux choix et approches d'autres langages.

Pour atteindre cet objectif, nous tenterons d'abord de répondre à *pourquoi* avant de nous intéresser à *comment*. Cette démarche s'illustre par deux groupes de chapitres proposant respectivement :

- de toujours répondre à *pourquoi* avant de répondre à *comment* en analysant en profondeur les choix que l'équipe Go a faits pour mieux comprendre les cas d'utilisation qu'ils avaient en tête, en expliquant les raisons qui ont mené à leurs choix et les conséquences de ces derniers dans l'utilisation quotidienne de Go. Notamment, nous passerons en revue différents problèmes que les langages de programmation résolvent habituellement (gestion de la mémoire, gestion des erreurs, etc.) et examinerons les choix faits par les autres langages et celui fait explicitement par Go ;
- de vous présenter, pour mettre le bon pied à l'étrier, une série de cas pratiques à complexité croissante (d'abord un algorithme simple, puis le programme lit les données dans des fichiers, plus tard il les expose via une interface HTTP, etc.) en suivant le chemin de réflexion d'un développeur Go alors qu'il résout le problème, et le code résultant de ce travail.

Un des intérêts forts de Go, celui qui a charmé l'auteur lors de sa propre découverte, est qu'il ambitionne d'offrir le meilleur des trois mondes entre portabilité du code, performance et productivité du développeur, trois axes selon lesquels la plupart des langages de l'industrie demandent de très forts compromis. Go proposant une approche plutôt efficace dans la résolution de ce compromis, nous pensons sincèrement que son apprentissage est susceptible de profiter à une majorité des ingénieurs logiciels, qu'ils aient l'intention de se spécialiser sur le langage Go ou non.

Notre objectif est que ce livre vous en fournisse une compréhension pertinente, et nous espérons que les informations qu'il vous apportera vous permettront de l'utiliser de manière immédiatement efficace, dès votre premier projet.

## Votre premier programme, sur le Go playground

Parlons-en justement de votre premier projet ! La courbe d'apprentissage de Go est habituellement considérée comme plutôt abordable, alors mettons un premier orteil dans l'écriture d'un programme Go dès maintenant.

Go propose un *playground* en ligne qui permet de rapidement tester ses idées et la syntaxe du langage directement dans son navigateur, sans avoir besoin d'installer quoi que ce soit. Rendez-vous sur https://play.golang.org, et un simple *Hello World* se présentera par défaut, que vous pouvez exécuter en cliquant sur RUN.

```
package main

import "fmt"

func main() {
  fmt.Println("Hello, playground")
}
```

```
Hello, playground
```

Tout code en Go doit appartenir à un package, le point d'entrée étant la fonction sans argument ni type de retour `main`, du package `main`. Le package `fmt` ici importé permet d'exposer la fonction `Println` pour afficher le *Hello World*, mais contient aussi une quantité d'autres méthodes de formatage.

Et voilà ! Vous avez exécuté votre premier programme en Go ! Nous nous plongerons dans des exemples autrement plus réalistes et avancés dans la deuxième partie de ce livre, les études de cas.