

Table des matières

À propos des auteurs	x
Introduction	1
Appréhender Go	3
1. Genèse de Go	4
1.1. 45 minutes pour repenser le développement logiciel	5
1.2. Une solution aux architectures modernes	6
2. Pourquoi Go ?	8
2.1. Compromis performance vs. portabilité	9
Le problème	9
Certaines solutions	9
L'approche de Go	10
2.2. Compromis performance vs. gestion de mémoire	11
Le problème	11
Certaines solutions	11
L'approche de Go	13
2.3. Compromis performance (typesafety) vs. simplicité	14
Le problème	14
Certaines solutions	15
L'approche de Go	16
2.4. Objets et gestion d'état	17
Le problème	17
Certaines solutions	17
L'approche de Go	18
2.5. Gestion d'erreurs et exceptions	22
Le problème	22
Certaines solutions	22
L'approche de Go	23
2.6. Parallélisme et concurrence	24
Le problème	24
Certaines solutions	25
L'approche de Go	26
3. Limites et avenir de Go	28
3.1. Gestion de dépendances	28

Le problème et certaines solutions	28
L'approche de Go et les problèmes qu'elle pose	30
3.2. Programmation générique	31
3.3. Ramasse-miettes	32
3.4. Qu'en est-il de la compilation sur de très grandes quantités de code ?	32
4. Projets pertinents	34
4.1. Programmation système	34
4.2. Systèmes embarqués	35
4.3. Micro-services	36
4.4. Scripts parallélisables	39
4.5. Projets non pertinents	40
Études de cas	41
5. Hello World, sur votre machine	42
5.1. Objectif de cette étude de cas	42
5.2. Mise en place optionnelle du GOPATH	43
5.3. Installer le compilateur Go	44
5.4. Installer un éditeur de code pour Go	44
5.5. Et pour finir, exécutons le programme	46
6. Un programme simple	47
6.1. Objectif de cette étude de cas	47
6.2. Premières lignes	48
6.3. Structures de données	49
6.4. Autres types qui vont nous être utiles	50
6.5. Faire le compte des votes	51
6.6. En déduire le gagnant	52
6.7. La fonction principale	54
6.8. Création de la donnée	54
6.9. Utilisons le tout	55
6.10. Tout le code en résumé	56
7. Lecture dans les fichiers	59
7.1. Objectif de cette étude de cas	59
7.2. Un nouveau package : <i>model</i>	61
7.3. Les structures de données du package <i>model</i>	62
La structure <i>FromFiles</i> pour orchestrer le modèle	62
Les autres structures, contenant les données elles-mêmes	63

7.4. Lecture dans les fichiers	65
7.5. La fonction <i>ComputeRound</i>	69
7.6. La fonction <i>Winner</i>	70
7.7. Pour lier le tout : notre fonction <i>main</i>	73
8. Concurrence	75
8.1. Objectif de cette étude de cas	75
8.2. Avant de se lancer	77
8.3. Première étape : goroutines	77
8.4. Attendre que les goroutines aient toutes terminées : <i>waitGroup</i>	80
8.5. Ne pas utiliser une même ressource simultanément : <i>mutex</i>	83
8.6. Remonter les erreurs des goroutines vers le fil d'exécution principal	85
9. Persistance	89
9.1. Objectif de cette étude de cas	89
9.2. Préparer les structures de données	90
9.3. Le fichier <i>fromMongo.go</i>	92
9.4. Modifier la fonction <i>main</i>	95
9.5. Une requête optimisée	97
10. Tests unitaires	100
10.1. Objectif de cette étude de cas	100
10.2. Commençons par un test qui réussit toujours	101
10.3. Le test le plus simple : 2+2 font-ils vraiment 4 ?	103
10.4. Un test simple : un seul cas de test	104
10.5. Plusieurs cas dans le même test : les tests orientés tableau	106
10.6. Un cas plus avancé, avec injection de dépendance	109
Définir et instancier la structure des cas de tests	109
Injecter un <i>model.Reader</i> sans effet de bord	111
Comparer des valeurs d'erreurs ?	111
Enfin, le test lui-même	112
Avons-nous couvert tous les cas ?	113
11. Un service HTTP	116
11.1. Objectif de cette étude de cas	116
11.2. Afficher le gagnant de l'élection	118
Créer les répertoires et fichiers des contrôleurs	118
Rédiger la fonction de contrôleur	119
Mettre en place notre chemin HTTP	120
11.3. Afficher le nombre de votes	121

11.4. Enregistrer un nouveau vote	123
Changement dans le modèle	123
Le contrôleur	124
12. Déploiement du projet	127
12.1. Objectif de cette étude de cas	127
12.2. Les concepts Go utiles au déploiement	128
12.3. Principales architectures de déploiement	129
Directement sur le système hôte	130
Dans un micro-container	132
Derrière un serveur HTTP (comme nginx)	133
En utilisant une plateforme PaaS	135
12.4. Et en cas de soucis ?	136
Le mot de la fin	137
Lexique	138
Liste des illustrations	145
Index	146