|  |  |  |
|---|---|---|
| 1. | Record Nr. | UNINA9910148748303321 |
|  | Autore | Ryer Mat |
|  | Titolo | Go programming blueprints : build real-world, production-ready solutions in Go using cutting-edge technology and techniques / / Mat Ryer |
|  | Pubbl/distr/stampa | Birmingham, England ; ; Mumbai, India : , : Packt Publishing, , 2016<br>©2016 |
|  | Edizione | [Second edition.] |
|  | Descrizione fisica | 1 online resource (385 pages) : illustrations |
|  | Disciplina | 005.133 |
|  | Soggetti | Go (Computer program language)<br>Computer programming |
|  | Lingua di pubblicazione | Inglese |
|  | Formato | Materiale a stampa |
|  | Livello bibliografico | Monografia |
|  | Note generali | Includes index. |
|  | Nota di contenuto | Cover -- Credits -- About the Author -- Acknowledgments -- About the Reviewer -- www.PacktPub.com -- Table of Contents -- Preface -- Chapter 1: Chat Application with Web Sockets -- A simple web server -- Separating views from logic using templates -- Doing things once -- Using your own handlers -- Properly building and executing Go programs -- Modeling a chat room and clients on the server -- Modeling the client -- Modeling a room -- Concurrency programming using idiomatic Go -- Turning a room into an HTTP handler -- Using helper functions to remove complexity -- Creating and using rooms -- Building an HTML and JavaScript chat client -- Getting more out of templates -- Tracing code to get a look under the hood -- Writing a package using TDD -- Interfaces -- Unit tests -- Red-green testing -- Implementing the interface -- Unexported types being returned to users -- Using our new trace package -- Making tracing optional -- Clean package APIs -- Summary -- Chapter 2: Adding User Accounts -- Handlers all the way down -- Making a pretty social sign-in page -- Endpoints with dynamic paths -- Getting started with OAuth2 -- Open source OAuth2 packages -- Tell the authorization providers about your app -- Implementing external logging in -- Logging in -- Handling the response from the provider -- Presenting the user data -- Augmenting messages with additional data -- Summary -- Chapter 3: Three Ways |

to Implement Profile Pictures -- Avatars from the OAuth2 server -- Getting the avatar URL -- Transmitting the avatar URL -- Adding the avatar to the user interface -- Logging out -- Making things prettier -- Implementing Gravatar -- Abstracting the avatar URL process -- The auth service and the avatar's implementation -- Using an implementation -- The Gravatar implementation -- Uploading an avatar picture -- User identification -- An upload form.
Handling the upload -- Serving the images -- The Avatar implementation for local files -- Supporting different file types -- Refactoring and optimizing our code -- Replacing concrete types with interfaces -- Changing interfaces in a test-driven way -- Fixing the existing implementations -- Global variables versus fields -- Implementing our new design -- Tidying up and testing -- Combining all three implementations -- Summary -- Chapter 4: Command-Line Tools to Find Domain Names -- Pipe design for command-line tools -- Five simple programs -- Sprinkle -- Domainify -- Coolify -- Synonyms -- Using environment variables for configuration -- Consuming a web API -- Getting domain suggestions -- Available -- Composing all five programs -- One program to rule them all -- Summary -- Chapter 5: Building Distributed Systems and Working with Flexible Data -- The system design -- The database design -- Installing the environment -- Introducing NSQ -- NSQ driver for Go -- Introducing MongoDB -- MongoDB driver for Go -- Starting the environment -- Reading votes from Twitter -- Authorization with Twitter -- Extracting the connection -- Reading environment variables -- Reading from MongoDB -- Reading from Twitter -- Signal channels -- Publishing to NSQ -- Gracefully starting and stopping programs -- Testing -- Counting votes -- Connecting to the database -- Consuming messages in NSQ -- Keeping the database updated -- Responding to Ctrl + C -- Running our solution -- Summary -- Chapter 6: Exposing Data and Functionality through a RESTful Data Web Service API -- RESTful API design -- Sharing data between handlers -- Context keys -- Wrapping handler functions -- API keys -- Cross-origin resource sharing -- Injecting dependencies -- Responding -- Understanding the request -- Serving our API with one function -- Using handler function wrappers -- Handling endpoints.
Using tags to add metadata to structs -- Many operations with a single handler -- Reading polls -- Creating a poll -- Deleting a poll -- CORS support -- Testing our API using curl -- A web client that consumes the API -- Index page showing a list of polls -- Creating a new poll -- Showing the details of a poll -- Running the solution -- Summary -- Chapter 7: Random Recommendations Web Service -- The project overview -- Project design specifics -- Representing data in code -- Public views of Go structs -- Generating random recommendations -- The Google Places API key -- Enumerators in Go -- Test-driven enumerator -- Querying the Google Places API -- Building recommendations -- Handlers that use query parameters -- CORS -- Testing our API -- Web application -- Summary -- Chapter 8: Filesystem Backup -- Solution design -- The project structure -- The backup package -- Considering obvious interfaces first -- Testing interfaces by implementing them -- Has the filesystem changed? -- Checking for changes and initiating a backup -- Hardcoding is OK for a short while -- The user command-line tool -- Persisting small data -- Parsing arguments -- Listing the paths -- String representations for your own types -- Adding paths -- Removing paths -- Using our new tool -- The daemon backup tool -- Duplicated structures -- Caching data -- Infinite loops -- Updating filedb records -- Testing our solution -- Summary -- Chapter 9: Building a Q&amp -- A Application

for Google App Engine -- The Google App Engine SDK for Go -- Creating your application -- App Engine applications are Go packages -- The app.yaml file -- Running simple applications locally -- Deploying simple applications to Google App Engine -- Modules in Google App Engine -- Specifying modules -- Routing to modules with dispatch.yaml -- Google Cloud Datastore -- Denormalizing data -- Entities and data access.
Keys in Google Cloud Datastore -- Putting data into Google Cloud Datastore -- Reading data from Google Cloud Datastore -- Google App Engine users -- Embedding denormalized data -- Transactions in Google Cloud Datastore -- Using transactions to maintain counters -- Avoiding early abstraction -- Querying in Google Cloud Datastore -- Votes -- Indexing -- Embedding a different view of entities -- Casting a vote -- Accessing parents via datastore.Key -- Line of sight in code -- Exposing data operations over HTTP -- Optional features with type assertions -- Response helpers -- Parsing path parameters -- Exposing functionality via an HTTP API -- HTTP routing in Go -- Context in Google App Engine -- Decoding key strings -- Using query parameters -- Anonymous structs for request data -- Writing self-similar code -- Validation methods that return an error -- Mapping the router handlers -- Running apps with multiple modules -- Testing locally -- Using the admin console -- Automatically generated indexes -- Deploying apps with multiple modules -- Summary -- Chapter 10: Micro-services in Go with the Go kit Framework -- Introducing gRPC -- Protocol buffers -- Installing protocol buffers -- Protocol buffers language -- Generating Go code -- Building the service -- Starting with tests -- Constructors in Go -- Hashing and validating passwords with bcrypt -- Modeling method calls with requests and responses -- Endpoints in Go kit -- Making endpoints for service methods -- Different levels of error -- Wrapping endpoints into a Service implementation -- An HTTP server in Go kit -- A gRPC server in Go kit -- Translating from protocol buffer types to our types -- Creating a server command -- Using Go kit endpoints -- Running the HTTP server -- Running the gRPC server -- Preventing a main function from terminating immediately -- Consuming the service over HTTP.
Building a gRPC client -- A command-line tool to consume the service -- Parsing arguments in CLIs -- Maintaining good line of sight by extracting case bodies -- Installing tools from the Go source code -- Rate limiting with service middleware -- Middleware in Go kit -- Manually testing the rate limiter -- Graceful rate limiting -- Summary -- Chapter 11: Deploying Go Applications Using Docker -- Using Docker locally -- Installing Docker tools -- Dockerfile -- Building Go binaries for different architectures -- Building a Docker image -- Running a Docker image locally -- Inspecting Docker processes -- Stopping a Docker instance -- Deploying Docker images -- Deploying to Docker Hub -- Deploying to Digital Ocean -- Creating a droplet -- Accessing the droplet's console -- Pulling Docker images -- Running Docker images in the cloud -- Accessing Docker images in the cloud -- Summary -- Appendix: Good Practices for a Stable Go Environment -- Installing Go -- Configuring Go -- Getting GOPATH right -- Go tools -- Cleaning up, building, and running tests on save -- Integrated developer environments -- Sublime Text 3 -- Visual Studio Code -- Summary -- Index.

Sommario/riassunto

Build real-world, production-ready solutions in Go using cutting-edge technology and techniques About This Book Get up to date with Go and write code capable of delivering massive world-class scale performance and availability Learn to apply the nuances of the Go language, and get to know the open source community that surrounds it to implement a

wide range of start-up quality projects Write interesting and clever but simple code, and learn skills and techniques that are directly transferrable to your own projects Who This Book Is For If you are familiar with Go and are want to put your knowledge to work, then this is the book for you. Go programming knowledge is a must. What You Will Learn Build quirky and fun projects from scratch while exploring patterns, practices, and techniques, as well as a range of different technologies Create websites and data services capable of massive scale using Go’s net/http package, exploring RESTful patterns as well as low-latency WebSocket APIs Interact with a variety of remote web services to consume capabilities ranging from authentication and authorization to a fully functioning thesaurus Develop high-quality command-line tools that utilize the powerful shell capabilities and perform well using Go’s in-built concurrency mechanisms Build microservices for larger organizations using the Go Kit library Implement a modern document database as well as high-throughput messaging queue technology to put together an architecture that is truly ready to scale Write concurrent programs and gracefully manage the execution of them and communication by smartly using channels Get a feel for app deployment using Docker and Google App Engine In Detail Go is the language of the Internet age, and the latest version of Go comes with major architectural changes. Implementation of the language, runtime, and libraries has changed significantly. The compiler and runtime are now written entirely in Go. The garbage collector is now concurrent and provides dramatically lower pause times by running in parallel with other Go routines when possible. This book will show you how to leverage all the latest features and much more. This book shows you how to build powerful systems and drops you into real-world situations. You will learn to develop high-quality command-line tools that utilize the powerful shell capabilities and perform well using Go’s in-built concurrency mechanisms. Scale, performance, and high availability lie at the heart ...