POLSKO-JAPOŃSKA
AKADEMIA TECHNIK
KOMPUTEROWYCH

PAKIETY, EKSPORTY, GO MOD

Go modules

- Moduł jest to zbiór pakietów/zależności wchodzących w skład projektu
- Jego definicja znajduje się w pliku `go.mod`

Tworzenie nowego modułu

1. Tworzymy folder dla naszego projektu oraz przechodzimy do jego lokalizacji

2.
```
$ mkdir pjatk_project
$ cd pjatk_project
```

2. Korzystając z narzędzia `go mod init` tworzymy plik `go.mod` jako argument przekazując nazwę modułu

```
$ go mod init pjatk_project
```

W folderze projektu został wygenerowany plik `go.mod`

```
pjatk_project/
└── go.mod
```

```
module pjatk_project

go 1.19
```

W aktualnej formie plik zawiera informacje: - nazwa modułu - wersja Go

Dodawanie zewnętrznych zależności do projektu

- Chcemy aby nasz program po uruchomieniu wyświetlał unikalny identyfikator UUID
- W tym celu wykorzystamy bibliotekę dostarczaną przez Google https://github.com/google/uuid

Instalujemy zależność z pomocą narzędzia `go get`

```
$ go get github.com/google/uuid
```

```
module pjatk_project

go 1.19

require github.com/google/uuid v1.3.0 // indirect
```

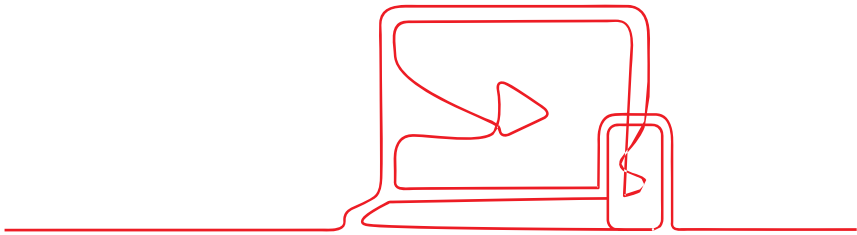
- w pliku `go.mod` została dodana nowa sekcja `require` opisująca dodaną zależność oraz jej wersję
- oznaczenie `//indirect` informuje o tym, że zainstalowana zależność nie jest jawnie wykorzystana w naszym kodzie

Dodawanie zewnętrznych zależności do projektu

- W projekcie pojawił się również kolejny plik `go.sum`

```
pjatk_project/
├── go.mod
└── go.sum

github.com/google/uuid v1.3.0 h1:t6JiXgmwXMjEs8VusXIJk2BXHsn+wx8BZdTaoZ5fu7I=
github.com/google/uuid v1.3.0/go.mod h1:TIyPZe4MgqvfeYDBFedMoGGpEw/LqOeaOT+nhxU+yHo=
```

- Plik ten zawiera informacje takie jak nazwa, wersja oraz hash zależności.



Dodawanie zewnętrznych zależności do projektu

Teraz możemy wykorzystać bibliotekę UUID [playground]

```
package main

import (
    "fmt"
    "github.com/google/uuid"
)

func main() {
    uniqueToken, _ := uuid.NewRandom()
    fmt.Printf("Twój unikalny token to: %s", uniqueToken)
}

$ go run main.go

Twój unikalny token to: 674c2952-0704-42f2-86de-7f542ab8d576
```

Zmiany zależności

W trakcie pisania programu możemy chcieć zmodyfikować, lub usunąć niektóre biblioteki. Aby zachować spójność zależności w plikach `go.mod` i `go.sum` Go dostarcza operację `mod tidy` która doda, zmodyfikuje lub usunie je automatycznie.

Przykładowo chcemy by nasz unikalny token był generowany przez bibliotekę *https://github.com/thanhpk/randstr*

```
$ go get github.com/thanhpk/randstr
```

po zainstalowaniu zależności dokonujemy zmian w kodzie playground

```go
package main

import (
    "fmt"
    "github.com/thanhpk/randstr"
)

func main() {
    uniqueToken := randstr.String(16)
    fmt.Printf("Twój unikalny token to: %s", uniqueToken)
}
```

Zmiany zależności

wykonujemy polecenie

```
$ go mod tidy
```

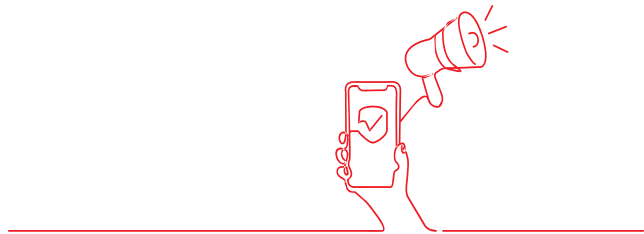
W jego wyniku pliki zostały zaktualizowane i nieużywana zależność UUID została usunięta

```
module pjatk_project

go 1.19

require github.com/thanhpk/randstr v1.0.4
github.com/thanhpk/randstr v1.0.4 h1:IN78qu/bR+My+gHCvMEXhR/i5oriVHcTB/BJJIRTsNo=
github.com/thanhpk/randstr v1.0.4/go.mod
h1:M/H2P1eNLZzlDwAzpkkkUvoyNNMbzRGhESZuEQk3r0U=
```

Tworzenie własnych pakietów i eksportowanie nazw

- Pakiety są paczkami kodu źródłowego
- Służą do grupowania i udostępniania funkcji, typów i zmiennych w celu ich wielokrotnego użycia
- Każdy plik z kodem źródłowym musi przynależeć do jakiegoś pakietu

Chcielibyśmy wydzielić funkcjonalność generowania tokenu do oddzielnej paczki, by nie zawierać całej logiki w pliku `main.go`.



Tworzenie własnych pakietów i eksportowanie nazw

Na początku tworzymy nowy folder `token` w naszym projekcie oraz dodajemy plik `generator.go`

```
pjatk_project/
├── go.mod
├── go.sum
├── main.go
└── token
        └── generator.go
```

Naszą paczkę nazwiemy token i będzie ona odpowiedzialna za operacje na tokenach. W pliku `generator.go` definujemy nową paczkę umieszczając na początku pliku:

```go
package token
```

Tworzenie własnych pakietów i eksportowanie nazw

Przenieśmy logikę generowania tokenu z pliku `main.go` do nowej funkcji w `generator.go`

```go
package token

import "github.com/thanhpk/randstr"

func generate(len int) string {
    return randstr.String(len)
}
```

Tworzenie własnych pakietów i eksportowanie nazw

Następnie w pliku `main.go` dokonajmy modyfikacji by wykorzystać nasz nowy pakiet `token` oraz funkcję `generate`

```
    package main

    import (
        "fmt"
        "pjatk_project/token"
    )

    func main() {
        uniqueToken := token.generate(16)
            fmt.Printf("Twój unikalny token to: %s", uniqueToken)
    }
```

Tworzenie własnych pakietów i eksportowanie nazw

Spróbujmy teraz uruchomić nasz projekt

```
$ go run main.go
    ./main.go:9:23: undefined: token.generate
```

W konsoli pojawił się błąd, ponieważ nasza funkcja `generate` nie została eksportowana i nie jest widoczna nigdzie poza pakietem `token`.

Aby naprawić błąd musimy wykonać eksport. Jedyne co należy zrobić to zmienić nazwę funkcji w `generator.go` tak by zaczynała się z wielkiej litery:

```
func Generate(len int) string
```

Teraz zedytujmy wywołanie funkcji w pliku `main.go`

```
uniqueToken := token.Generate(16)
```

Sukces! Właśnie stworzyliśmy naszą pierwszą paczkę i eksportowaliśmy jej funkcję!

Tworzenie własnych pakietów i eksportowanie nazw

W Go nazwy typów, funkcji, zmiennych, stałych itd. mogą zostać eksportowane poprzez zastosowanie w nazwie wielkiej litery na początku. Takie nazwy są `publiczne` i dostępne również poza pakietem w którym zostały zadeklarowane. Natomiast nazwy zaczynające się małą literą są `prywatne` i dostęp do nich jest ograniczany tylko z poziomu tej samej paczki w której zostały utworzone.

Podsumowanie

- Moduły służą do zarządzania zależnościami w projekcie
- Go udostępnia wbudowane narzędzia (więcej informacji w dokumentacji: https://go.dev/ref/mod)
  - `go mod init <nazwa>`- tworzenie modułu
  - `go get <zależność>` - instalowanie biblioteki
  - `go mod tidy` - aktualizacja zależności
- Pakiety służą do grupowania kodu źródłowego
- By korzystać z zasobów paczki w innych miejscach aplikacji (innych paczkach) musimy zrobić je publiczne poprzez `eksportowanie nazw`