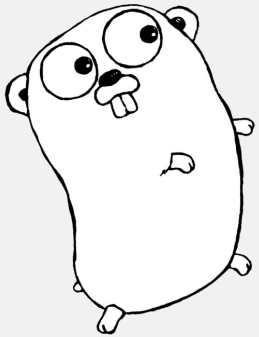

a p p
m o t
i o n



# Go

Boring and Fast

# Was ist Go?

- Entwickelt von Google
- Garbage-Collected
- “Wie könnte C++ heute aussehen?”

# Wer benutzt Go?


docker

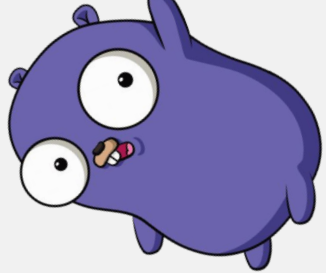
Google

kubernetes

twitch

Uber

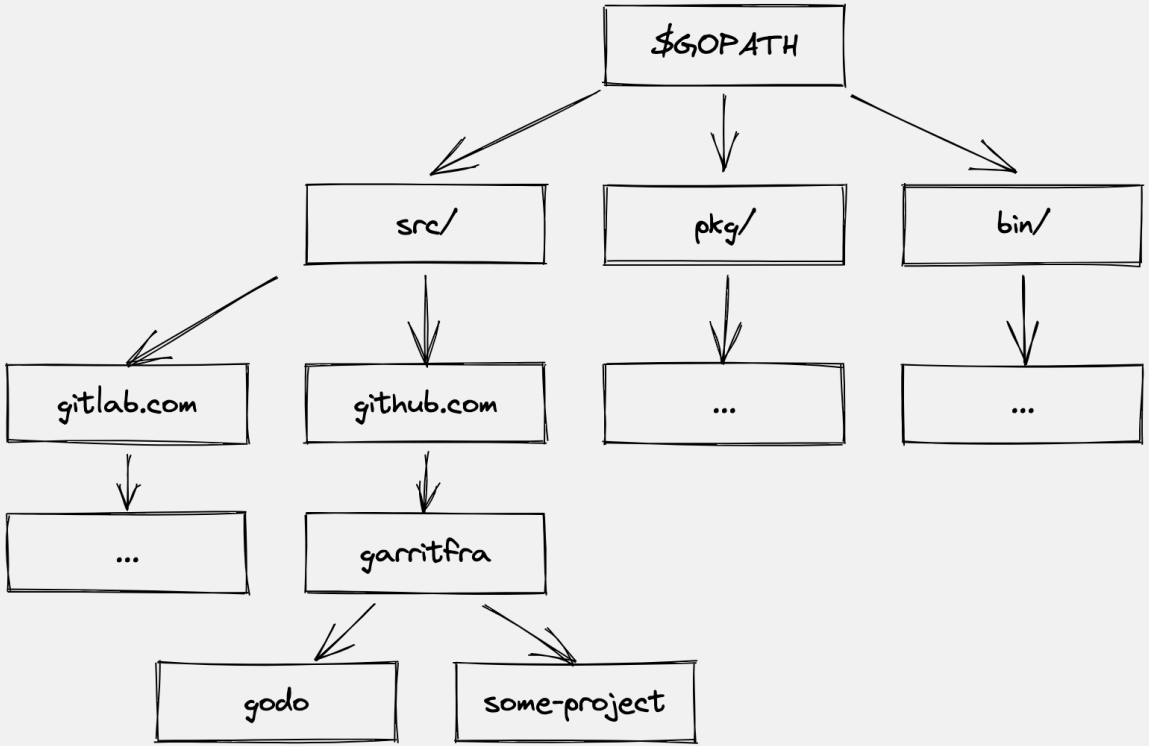
# Todo App - Aspekte



- Projekt Setup
- HTTP-Server
- GraphQL
- Datenbank (SQLite)
- Frontend

# Demo Time!

# Setup - $GOPATH



# Setup - Hello World!

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello World!")
}
```

# HTTP-Server

```go
package main

import "github.com/gin-gonic/gin"

func main() {
    r := gin.Default()
    r.GET("/ping", func(c *gin.Context) {
        c.JSON(200, gin.H{
            "message": "pong",
        })
    })
    r.Run() // listen and serve on 0.0.0.0:8080
}
```

- API vergleichbar mit Express.js
- Nutzt Standard-Bibliothek (net/http)

# GraphQL

```
type Todo {
  id: ID!
  text: String!
  done: Boolean!
}
```

todo.graphql

```
go generate ./...
```

```
type Todo struct {
    ID   string `json:"id"`
    Text string `json:"text"`
    Done bool   `json:"done"`
}
```

todo.go

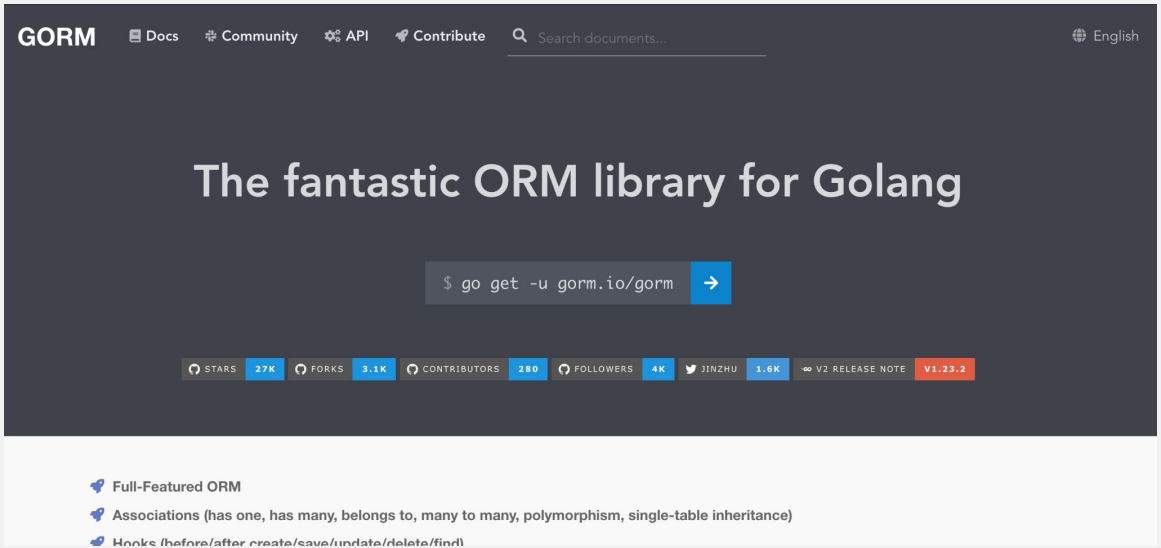
- `gqlgen` Library
- Code Generierung
- Setup sehr straight-forward

# GraphQL - Resolver

```go
func (r *mutationResolver) UpdateTodo(
  ctx context.Context,
  input model.UpdateTodoInput
) (*model.Todo, error) {
    var todo *model.Todo

    result := r.DB.First(&todo).Where("id", input.ID)

    if result.Error != nil {
        return nil, result.Error
    }

    if input.Done != nil {
        todo.Done = *input.Done
    }

    result = r.DB.Save(&todo)

    return todo, result.Error
}
```

- C-ähnliche Patterns
- Error Handling

# Datenbank




- Reiches Set an Features
- Sehr leichtes Setup
- Unterstützt mehrere DBs (SQLite, Postgres, MySQL, …)

# Datenbank - Models

```go
type Todo struct {
    gorm.Model
    ID   string `json:"id" gorm:"column:id"`
    Text string `json:"text" gorm:"column:text"`
    Done bool   `json:"done" gorm:"column:done"`
}
```

- Generierte GraphQL Models können erweitert werden
- Auto-Migrations
- DB-Instanz hängt an Resolver Context

# Todo App - Fazit


- Sehr schnelle Entwicklung
- Keine versteckten Überraschungen
- Syntax erfrischend einfach
- Einheitliches Tooling (Formatting, Docs, …)
- $GOPATH ist gewöhnungsbedürftig

# Go - Erwähnenswert

- Goroutines
- Generics sind noch sehr neu (Type-System sehr flexibel)
- Tooling legt Wert auf Korrektheit (öffentliche Funktionen müssen dokumentiert sein)

# Wann Go?

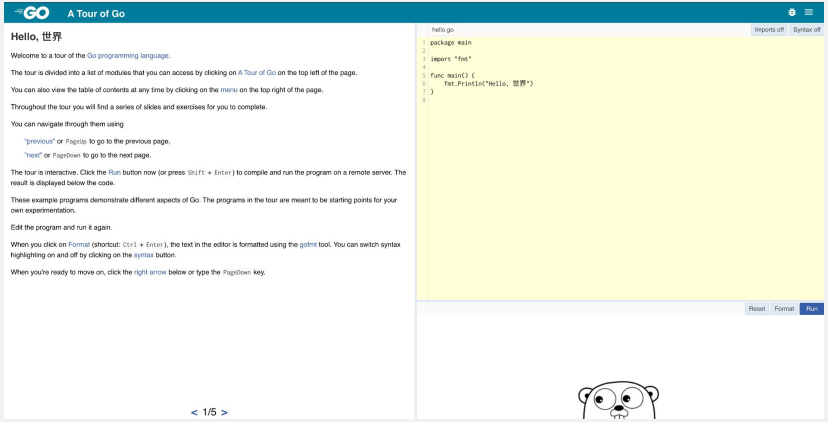
- Performance ist kritisch
- Häufig wechselnde Devs (schnelles Onboarding)
- Langfristige Projekte

# Wann nicht Go?*

*für Web-Applikationen

- Entwicklung von Prototypen
- Content Management Systeme
- Kompilierte Sprachen sind keine Option oder Laufzeit wird nicht unterstützt

# Du willst auch mal Hand anlegen?




https://go.dev/tour

# Fragen?