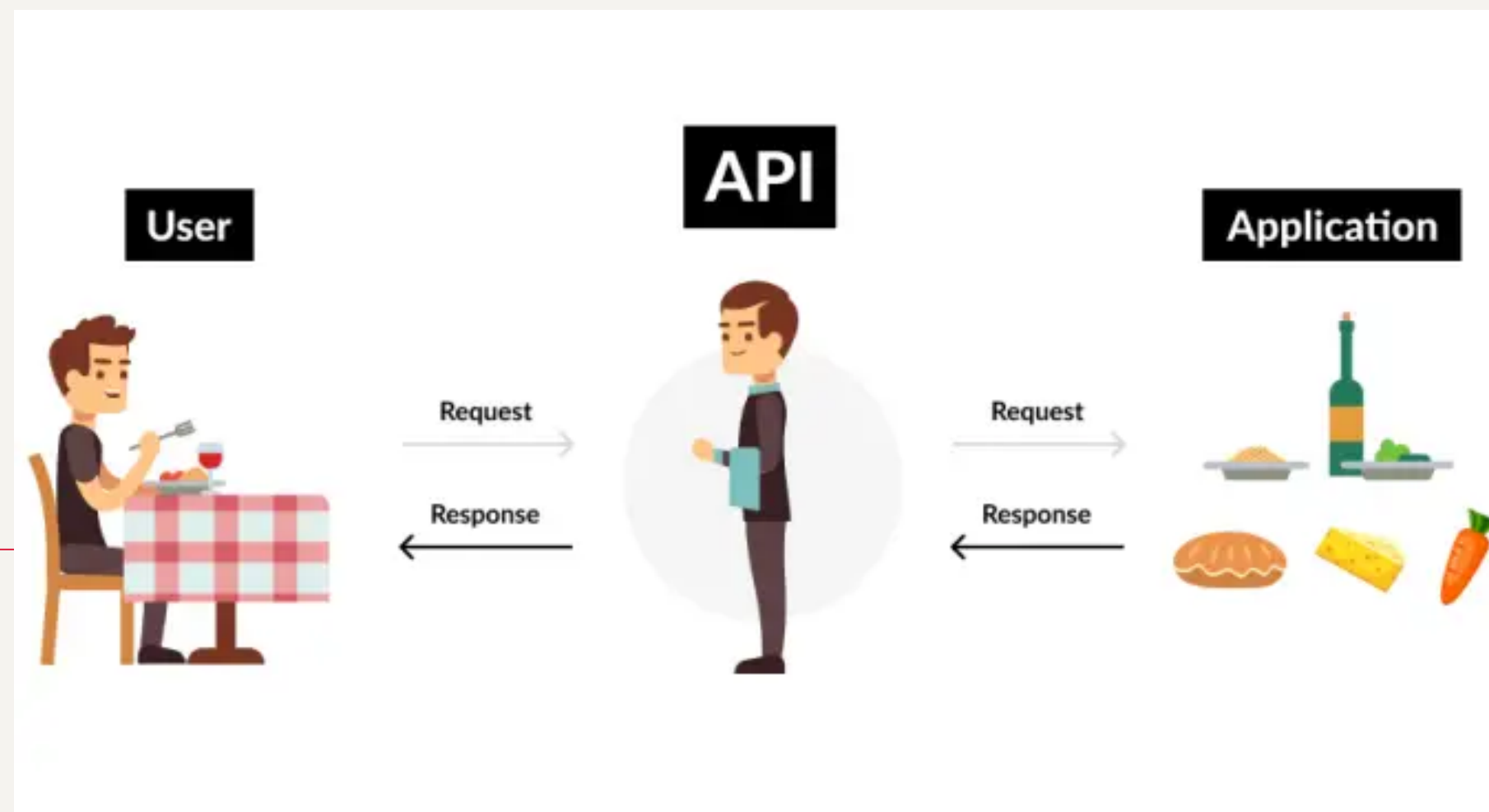


# More with Fiber Framework



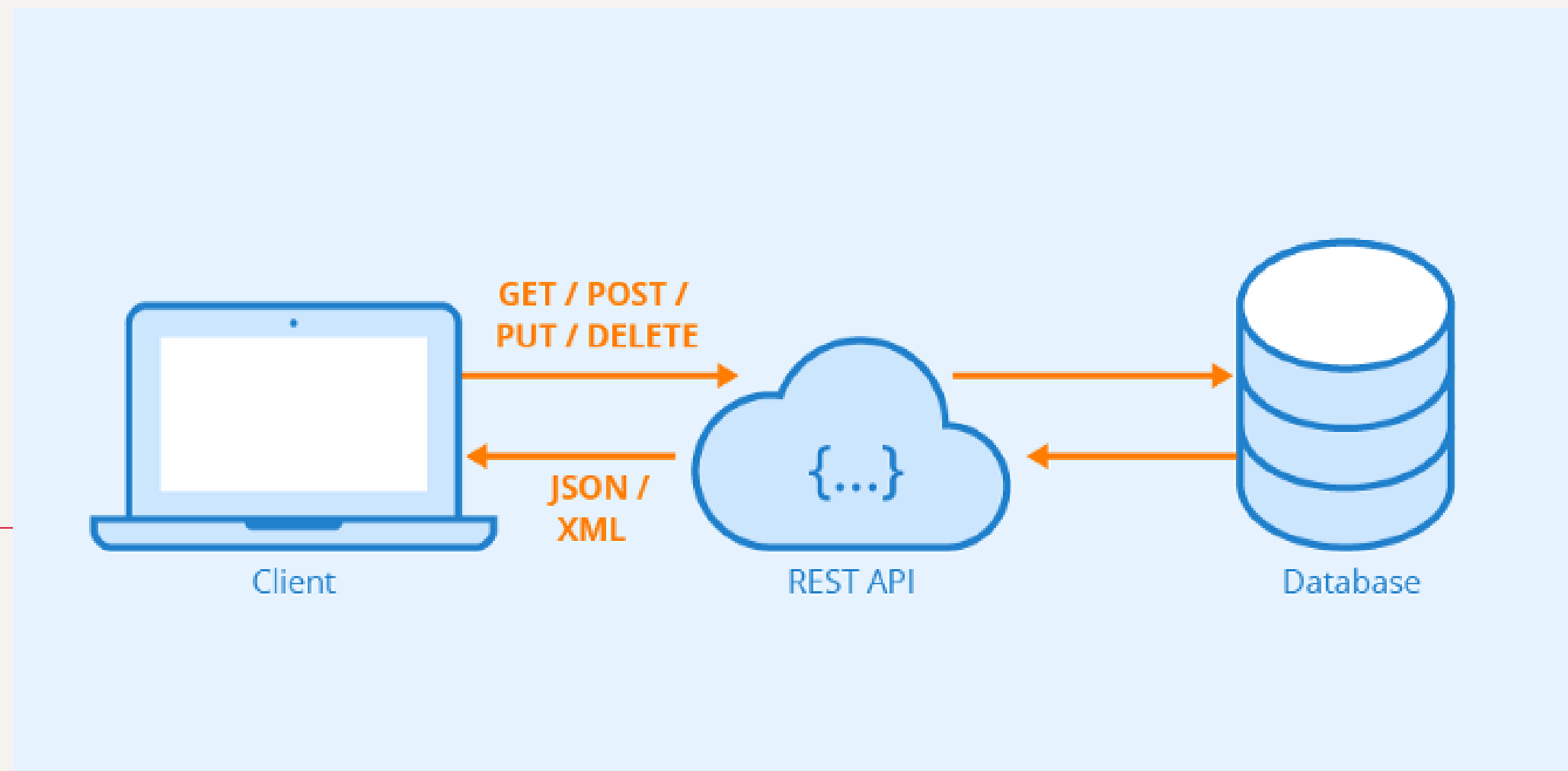
# API

**API** singkatan dari *Application Programming Interface*, merupakan kumpulan aturan / code yang membuat 2 program atau lebih dapat berkomunikasi. Dalam hal ini, API menjadi jembatan antar sistem yang menghubungkan client (browser/aplikasi) dengan server (database).



# Rest API

**REST** atau ***Representational State Transfer*** adalah mekanisme komunikasi antar web yang menggunakan **HTTP/HTTPS** untuk transmisi data dalam berbagai format seperti **JSON**, juga biasa disebut **RESTful API**. **RESTful API** bersifat stateless, artinya tidak ada data client yang disimpan dalam server di antara GET request.



# Method

Method merupakan fungsi yang disediakan server sebagai sarana untuk melakukan request terhadap server

```
carsRoute.Get("/cars/list", getCarLists)
carsRoute.Post("/cars/add", createCar)
carsRoute.Delete("/cars/delete/:id", deleteCarByID)
carsRoute.Put("/cars/update/:id", updateCarByID)
```

# HTTP Status

Merupakan status request yang dikirimkan oleh server ke client

|                           |     |  |
|---------------------------|-----|--|
| StatusOK                  | 200 | Get Data, Request Data, Update Data, dsb |
| StatusCreated             | 201 | Insert Data                              |
| StatusBadRequest          | 400 | Payload Request Salah/Tidak Sesuai       |
| StatusUnauthorized        | 401 | Tidak terautentikasi                     |
| StatusForbidden           | 403 | Akses route tidak diberikan              |
| StatusNotFound            | 404 | Route/data tidak ditemukan               |
| StatusUnprocessableEntity | 422 | Gagal Validasi Payload                   |
| StatusInternalServerError | 503 | Terjadi Function Error                   |

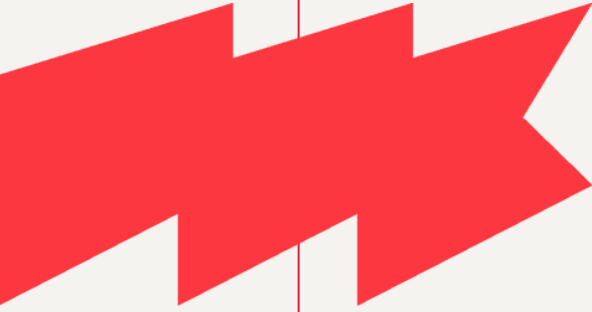
# Routing

Rute atau endpoint yang disediakan oleh server yang dapat diakses oleh client

```
func RegisterCarRoutes(app *fiber.App) {  
    carsRoute := app.Group("/cars")  
    carsRoute.Get("/", getCarLists)  
    carsRoute.Post("/", createCar)  
    carsRoute.Get("/:id", getCarByID)  
    carsRoute.Delete("/:id", deleteCarByID)  
    carsRoute.Put("/:id", updateCarByID)  
}
```

# Parsing Request

Membaca request payload yang dikirim oleh client, baik dari json body, query param, atau parameters



```
func SampleRequest(app *fiber.App) {
    carsRoute := app.Group("/cars")
    carsRoute.Get("/", getCarLists)
    carsRoute.Post("/by-brand", getCarLists)
    carsRoute.Get("/by-id/:id", getCarLists)
}

func getCarLists(c *fiber.Ctx) error {
    type GetCarRequest struct {
        Brand string `json:"brand" valid:"optional"`
        Limit string `query:"limit" valid:"optional"`
    }

    fmt.Println(c.Params("id"))
    fmt.Println(c.Query("search"))

    req := new(GetCarRequest)
    if err := c.BodyParser(req); err != nil {
        // return 422
    }
    if err := c.QueryParser(req); err != nil {
        // return 422
    }

    fmt.Println(req.Brand)
```

# Validation

Memvalidasi request payload yang dikirim oleh client

Disini kita menggunakan bantuan package pihak ke-3 yaitu :

[github.com/asaskevich/govalidator](https://github.com/asaskevich/govalidator)

```
type CreateCarRequest struct {
    Brand      string `json:"brand" valid:"optional"`
    Model      string `json:"model" valid:"optional"`
    Data       int    `json:"data" valid:"required"`
    ContactMail string `json:"contact_mail" valid:"email"`
}

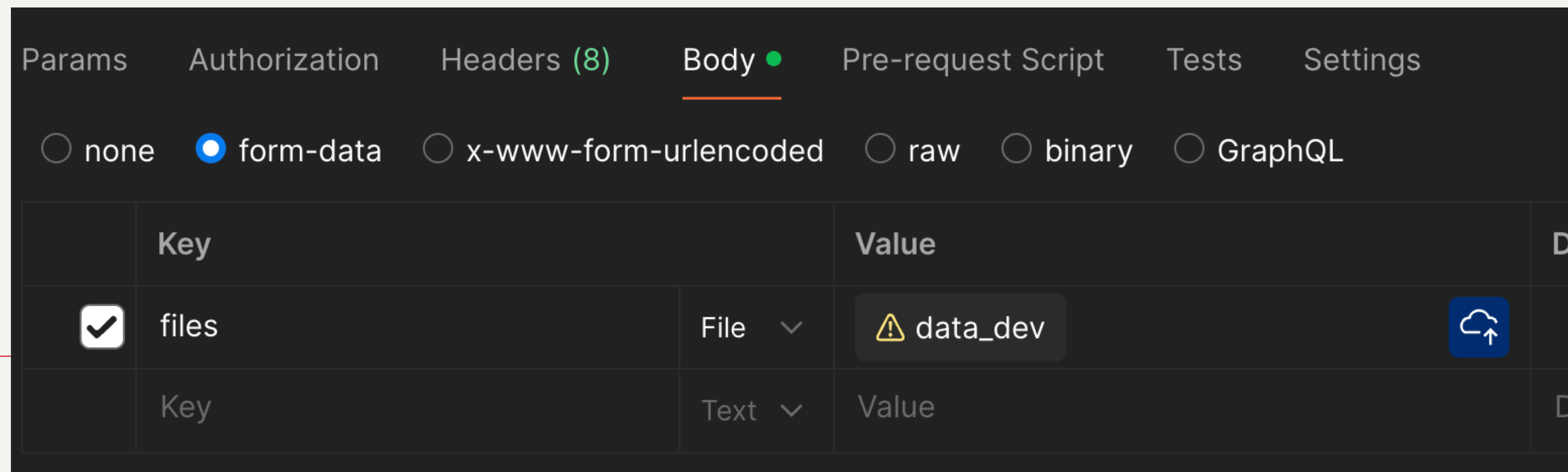
req := new(CreateCarRequest)
if err := c.BodyParser(req); err != nil {
    return c.Status(fiber.StatusBadRequest).JSON(fiber.Map{
        "success": false,
        "message": "Failed to parse body",
        "error":   err,
    })
}

validateResult, err := govalidator.ValidateStruct(req)
if !validateResult {
    return c.Status(fiber.StatusUnprocessableEntity).JSON(fiber.Map{
        "success": false,
        "message": "Failed to parse body",
        "error":   err,
    })
}
```





# Multipart Form

Multipart form adalah salah satu jenis konten yang digunakan dalam permintaan HTTP yang digunakan untuk mengunggah file atau mengirimkan data yang kompleks. Biasanya hanya bisa menerima bentuk file dan string



Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

|                                     | Key   |        | Value  | De |
|-------------------------------------|-------|--------|--|----|
| <input checked="" type="checkbox"/> | files | File ▾ |  data_dev  |    |
|                                     | Key   | Text ▾ | Value  | De |

# Form File Upload

Melakukan upload file melalui form yang biasa disebut Multipart form

```
func SampleUploadRoute(app *fiber.App) {
    apiRoute := app.Group("/upload")
    apiRoute.Post("/", UploadFile)
}

⌘L to chat, ⌘K to generate
func UploadFile(c *fiber.Ctx) error {
    form, err := c.MultipartForm()
    if err != nil {
        return c.SendStatus(fiber.StatusBadRequest)
    }

    files := form.File["file"]
    if len(files) == 0 {
        return c.SendStatus(fiber.StatusBadRequest)
    }

    for _, file := range files {
        fileContents, err := file.Open()
        if err != nil {
            return c.SendStatus(fiber.StatusBadRequest)
        }
        defer fileContents.Close()

        var fileBuffer bytes.Buffer
        _, err = io.Copy(&fileBuffer, fileContents)
        if err != nil {
            return c.SendStatus(fiber.StatusBadRequest)
        }
        os.WriteFile(file.Filename, fileBuffer.Bytes(), 0644)
    }
    return c.SendStatus(fiber.StatusOK)
}
```

# Middleware

Merupakan fungsi perantara/jembatan yang harus dilalui sebelum fungsi yang seharusnya

```
func SampleMiddleware(app *fiber.App) {
    carsRoute := app.Group("/cars", CheckAuth)
    carsRoute.Get("/", IsAdmin, getCarLists)
}

func CheckAuth(c *fiber.Ctx) error {
    authHeaders := string(c.Request().Header.Peek("Authorization"))
    if authHeaders == "" {
        return c.SendStatus(fiber.StatusUnauthorized)
    }
    return c.Next()
}

func IsAdmin(c *fiber.Ctx) error {
    roleHeaders := string(c.Request().Header.Peek("Role"))
    if roleHeaders != "admin" {
        return c.SendStatus(fiber.StatusUnauthorized)
    }
    return c.Next()
}
```