EXAMEN Keuze deel, Inspelen op Innovaties.

Wolf olthuis,

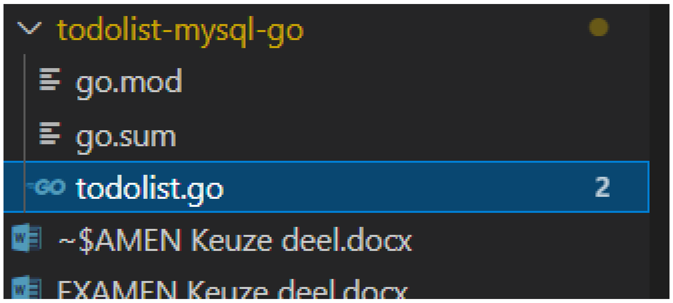
Opdracht 1.4

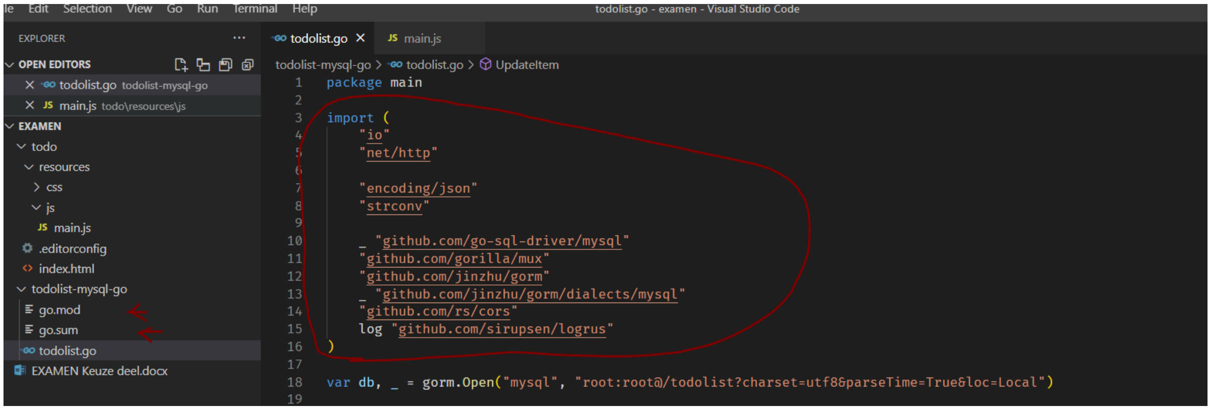
Inhouds Opgave

Opdracht 1.4

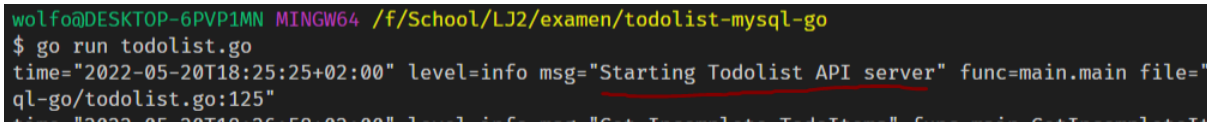
Tutorial Volgen

Stap 1. Project aanmaken, dependencies importeren en API live kunnen zetten.

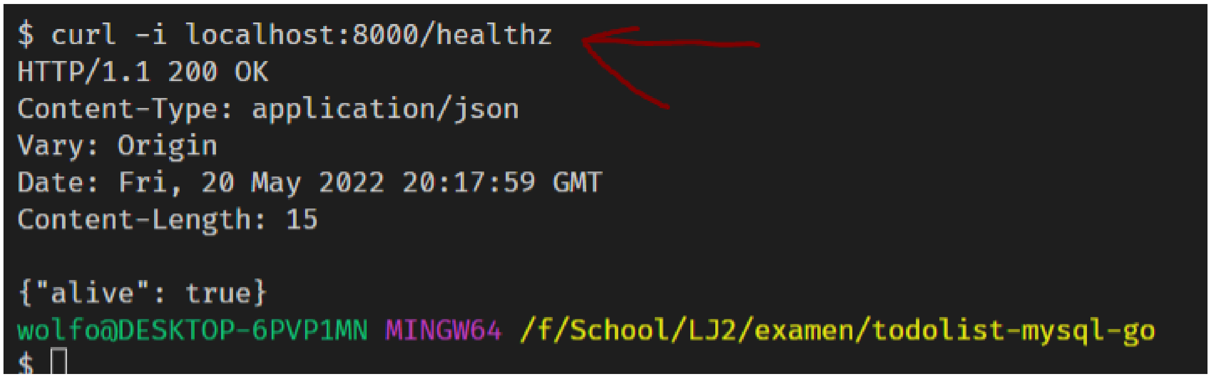



le Edit Selection View Go Run Terminal Help
todolist.go - examen - Visual Studio Code
EXPLORER
OPEN EDITORS
todolist.go todolist-mysql-go
main.js todo\resources\js
EXAMEN
todo
resources
css
js
main.js
.editorconfig
index.html
todolist-mysql-go
go.mod
go.sum
todolist.go
EXAMEN Keuze deel.docx
todolist.go
main.js
todolist-mysql-go > todolist.go > UpdateItem

```
package main

import (
	"io"
	"net/http"

	"encoding/json"
	"strconv"

	_ "github.com/go-sql-driver/mysql"
	"github.com/gorilla/mux"
	"github.com/jinzhu/gorm"
	_ "github.com/jinzhu/gorm/dialects/mysql"
	"github.com/rs/cors"
	log "github.com/sirupsen/logrus"
)

var db, _ = gorm.Open("mysql", "root:root@/todolist?charset=utf8&parseTime=True&loc=Local")
```

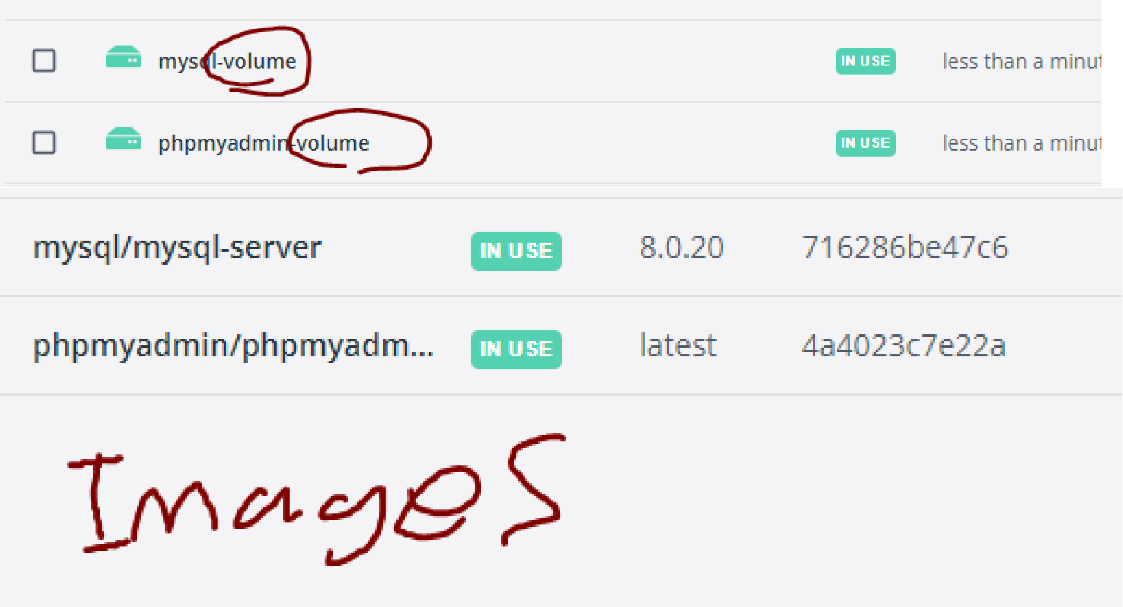


```
wolfo@DESKTOP-6PVP1MN MINGW64 /f/School/LJ2/examen/todolist-mysql-go
$ go run todolist.go
time="2022-05-20T18:25:25+02:00" level=info msg="Starting Todolist API server" func=main.main file="
ql-go/todolist.go:125"
```



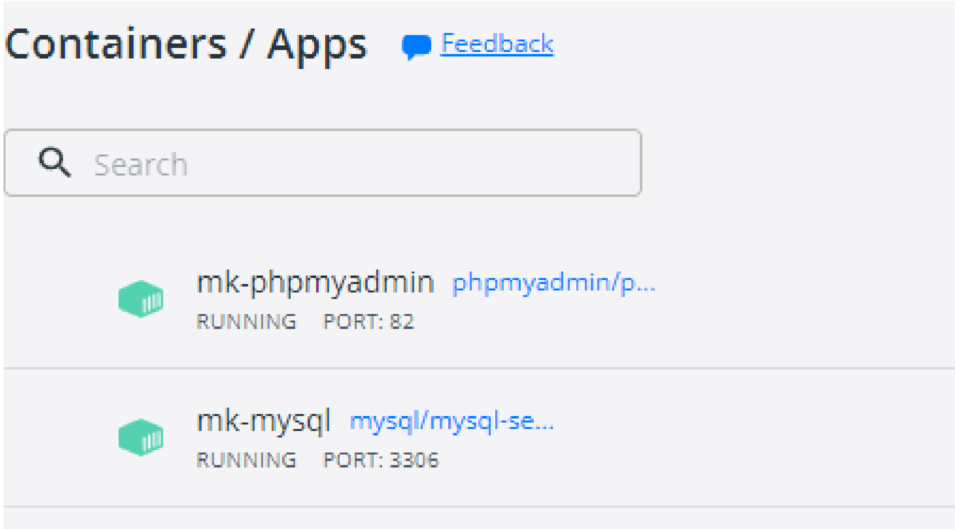
```
$ curl -i localhost:8000/healthz
HTTP/1.1 200 OK
Content-Type: application/json
Vary: Origin
Date: Fri, 20 May 2022 20:17:59 GMT
Content-Length: 15

{"alive": true}
wolfo@DESKTOP-6PVP1MN MINGW64 /f/School/LJ2/examen/todolist-mysql-go
$
```

Stap 2. MySQL instaleren op Docker, PhPMyAdmin instaleren op Docker.



□	mysql-volume	IN USE	less than a minut
□	phpmyadmin-volume	IN USE	less than a minut

mysql/mysql-server	IN USE	8.0.20	716286be47c6
phpmyadmin/phpmyadm...	IN USE	latest	4a4023c7e22a

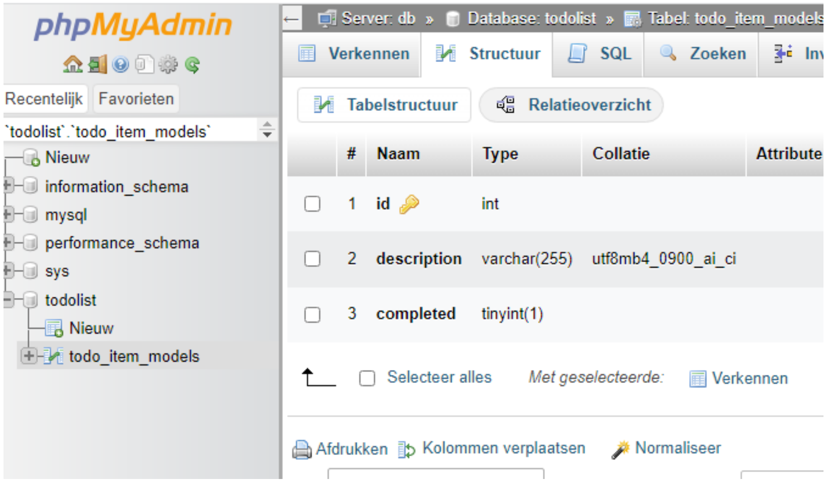


Containers / Apps

mk-phpmyadmin phpmyadmin/p...
RUNNING PORT: 82

mk-mysql mysql/mysql-se...
RUNNING PORT: 3306

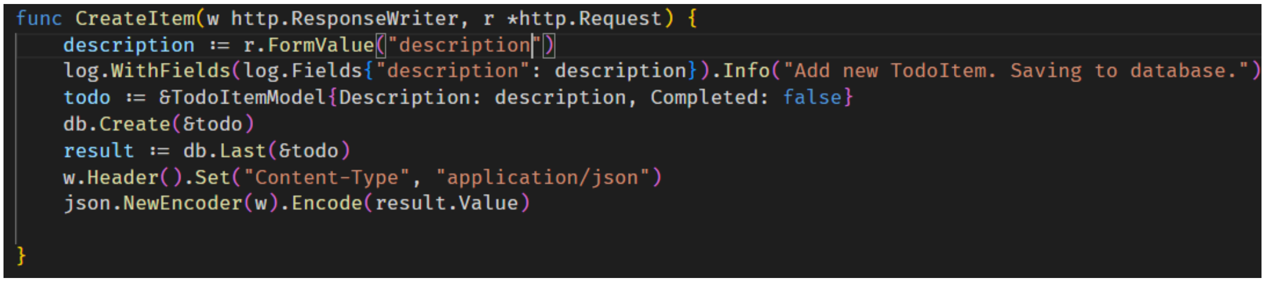
Stap 3. Tabel op Op MySQL aanmaken.



	#	Naam	Type	Collatie	Attribute
□	1	id 🔑	int		
□	2	description	varchar(255)	utf8mb4_0900_ai_ci	
□	3	completed	tinyint(1)		

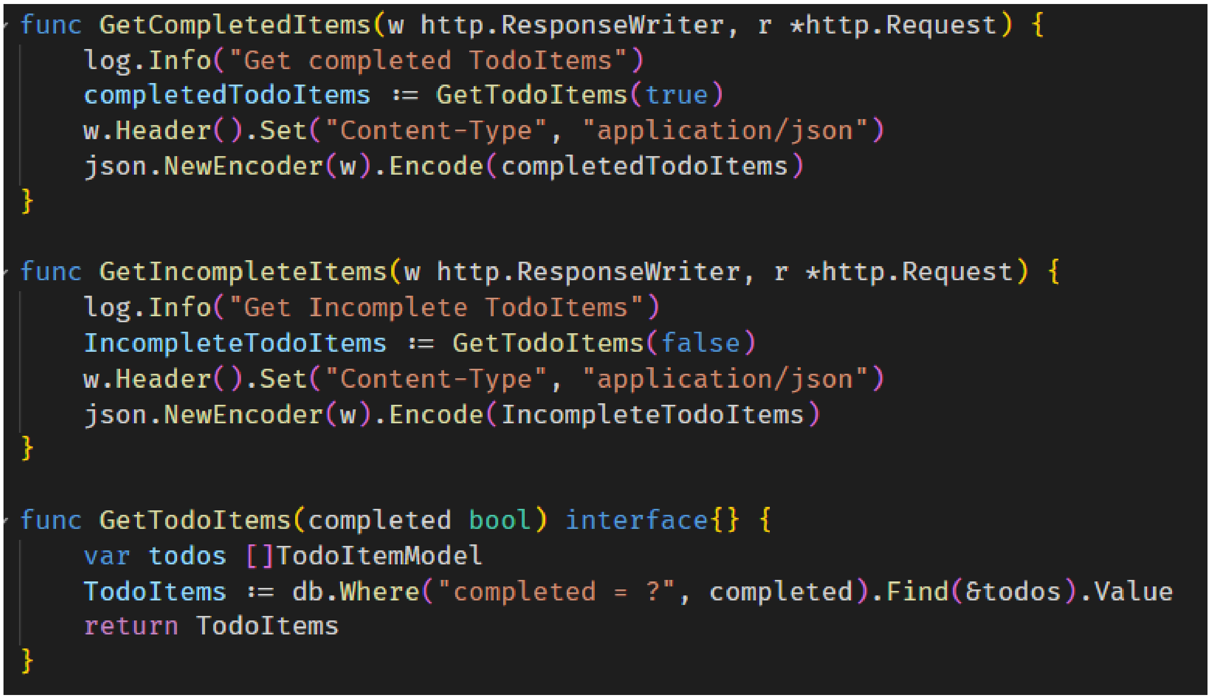
Stap 4. Alle C.R.U.D functies toevoegen.

Create :



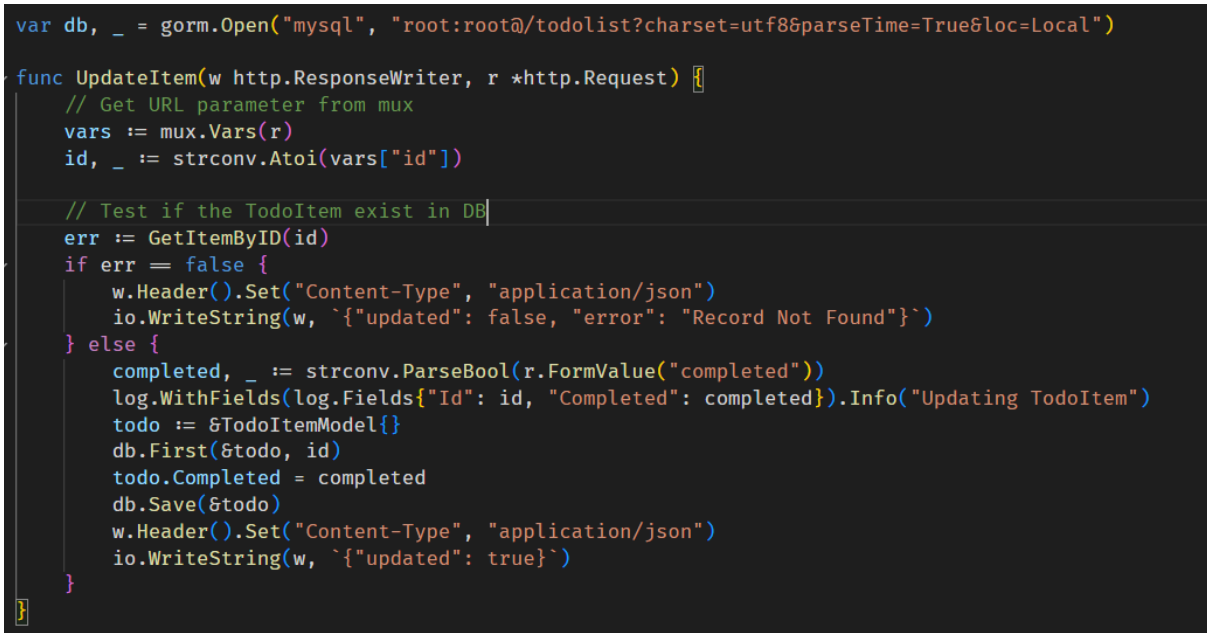
```
func CreateItem(w http.ResponseWriter, r *http.Request) {
    description := r.FormValue("description")
    log.WithFields(log.Fields{"description": description}).Info("Add new TodoItem. Saving to database.")
    todo := &TodoItemModel{Description: description, Completed: false}
    db.Create(&todo)
    result := db.Last(&todo)
    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(result.Value)

}
```

Read :



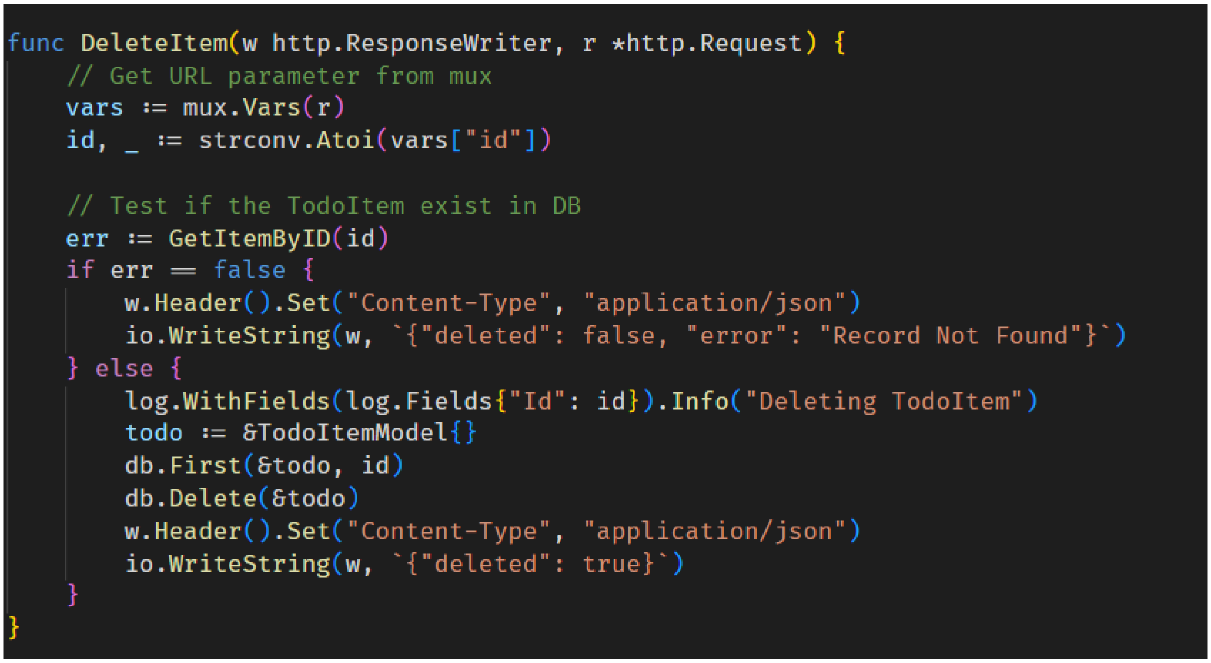
```go
func GetCompletedItems(w http.ResponseWriter, r *http.Request) {
	log.Info("Get completed TodoItems")
	completedTodoItems := GetTodoItems(true)
	w.Header().Set("Content-Type", "application/json")
	json.NewEncoder(w).Encode(completedTodoItems)
}

func GetIncompleteItems(w http.ResponseWriter, r *http.Request) {
	log.Info("Get Incomplete TodoItems")
	IncompleteTodoItems := GetTodoItems(false)
	w.Header().Set("Content-Type", "application/json")
	json.NewEncoder(w).Encode(IncompleteTodoItems)
}

func GetTodoItems(completed bool) interface{} {
	var todos []TodoItemModel
	TodoItems := db.Where("completed = ?", completed).Find(&todos).Value
	return TodoItems
}
```

Update :



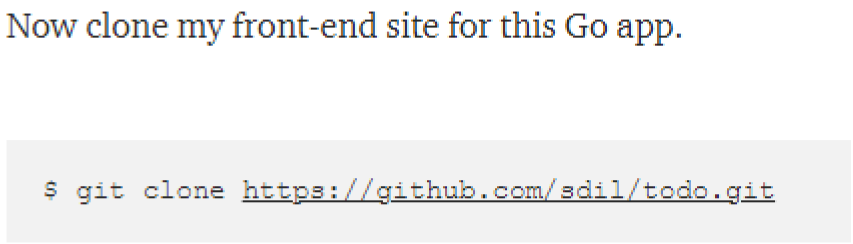
```go
var db, _ = gorm.Open("mysql", "root:root@/todolist?charset=utf8&parseTime=True&loc=Local")

func UpdateItem(w http.ResponseWriter, r *http.Request) {
	// Get URL parameter from mux
	vars := mux.Vars(r)
	id, _ := strconv.Atoi(vars["id"])

	// Test if the TodoItem exist in DB
	err := GetItemByID(id)
	if err == false {
		w.Header().Set("Content-Type", "application/json")
		io.WriteString(w, `{"updated": false, "error": "Record Not Found"}`)
	} else {
		completed, _ := strconv.ParseBool(r.FormValue("completed"))
		log.WithFields(log.Fields{"Id": id, "Completed": completed}).Info("Updating TodoItem")
		todo := &TodoItemModel{}
		db.First(&todo, id)
		todo.Completed = completed
		db.Save(&todo)
		w.Header().Set("Content-Type", "application/json")
		io.WriteString(w, `{"updated": true}`)
	}
}
```

Delete :



```go
func DeleteItem(w http.ResponseWriter, r *http.Request) {
    // Get URL parameter from mux
    vars := mux.Vars(r)
    id, _ := strconv.Atoi(vars["id"])

    // Test if the TodoItem exist in DB
    err := GetItemByID(id)
    if err == false {
        w.Header().Set("Content-Type", "application/json")
        io.WriteString(w, `{"deleted": false, "error": "Record Not Found"}`)
    } else {
        log.WithFields(log.Fields{"Id": id}).Info("Deleting TodoItem")
        todo := &TodoItemModel{}
        db.First(&todo, id)
        db.Delete(&todo)
        w.Header().Set("Content-Type", "application/json")
        io.WriteString(w, `{"deleted": true}`)
    }
}
```

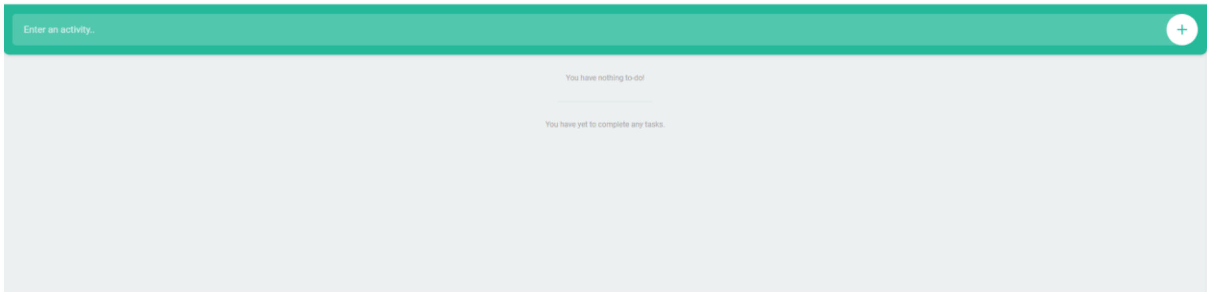
Stap 5. Connecten met FrontEnd



Now clone my front-end site for this Go app.

```
$ git clone https://github.com/sdil/todo.git
```

In de FrontEnd Js word er gepost naar de port van de GoLang Backend



You have nothing to-do!

You have yet to complete any tasks.