build unknown

Introduction

This sample aim to demonstrate a basic RESTful API - based off famcost. It uses mostly built-in Go's packages such as `net/http`, `database/sql`, `strconv`, `html/template` and use a third party packages `gorilla/mux` for teh router and `jackc/pgx` for the PostegreSQL driver

This partial application was created to demonstrate the following:

- Go HTML templates with W3.CSS stylesheet
- RESTful API with gorilla/mux
- Datastorage using PostgreSQL - tested with 15.4.1
- View a record details and add/edit records
- No filtering or sorting has been implemented

The application is a standalone demo requiring no additional WAMP/LAMP dependencies. It has been built and tested on Windows, standalone Ubuntu Linux and WSL:Ubuntu. If the application requires rebuilding, there are two batch files provided. The rebuild assumes there is an existing Go installation on the host computer. The provided batch files will download the required 3rd party packages for the build process.

Building

This application uses the Go programming language - where the latest was Go 1.21 as of writing this application. If you do not have Go installed on your system, you can acquire a copy from Go.dev. The go1.21.0.windows-amd64.msi was used to build this application.

Database configuration

The app assumes a database exists - ESD. Edit the *app.go* to change the default database name. Database defaults in the *app.go* are shown below.

```go
const (
    host     = "localhost"
    port     = 5432
    user     = "postgres"
    password = "postgres"
    dbname   = "ESD"
)
```

To run the server on your Windows system:

1. Run `buildpkg.cmd` in the root of the repo to build the binary (`pwrcost.exe`) using non vendored packages

2. Run `buildvendor.cmd` in the root of the repo to build the binary (`pwrcost.exe`) with the vendor

3. Run the binary `pwrcost.exe` or used the run.cmd (has env variable set)

4. Browse to http://localhost:8080 to test the application out. If port 80 does not work, you can start the app as follows.

   ```
       > pwrcost 8080
   ```

Non Windows

Testing has been performed on WSL & Linux but not MacOS. However, the commands in buildpkg.cmd and buildvendor.cmd can be run manually to build and run this demo.

Build by pkg

```
export GO111MODULE="on"
export GOFLAGS="-mod=mod"
go mod download
:: strip debug info during build
go build -ldflags="-s -w" .
```

Build by vendor

```
export GO111MODULE="on"
export GOFLAGS="-mod=vendor"
go mod vendor
:: strip debug info during build
go build -ldflags="-s -w"
```

Dependencies

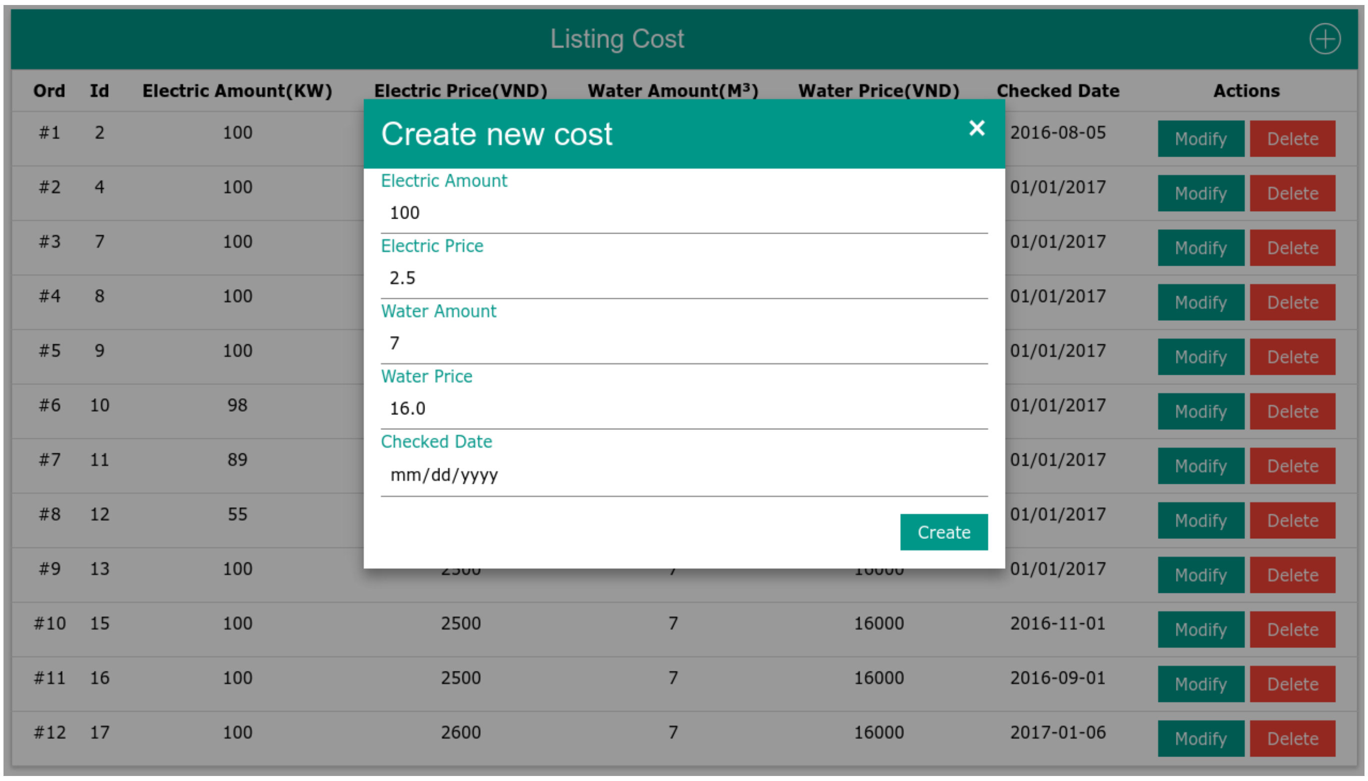
The application uses the following Go packages to build;

- Datastore: PostgreSQL driver
- HTTP router: Gorilla mux

Datastore

This version application requires a separate database to function - PostgreSQL. A few CSV files are imported from the local data folder. This will be imported when the application is run for the first time. Thereafter the application will use the database each time it is executed.

Sample screens



Listing Cost

Ord	Id	Electric Amount(KW)	Electric Price(VND)	Water Amount(M³)	Water Price(VND)	Checked Date	Actions
#1	2	100				2016-08-05	Modify Delete
#2	4	100				01/01/2017	Modify Delete
#3	7	100				01/01/2017	Modify Delete
#4	8	100				01/01/2017	Modify Delete
#5	9	100				01/01/2017	Modify Delete
#6	10	98				01/01/2017	Modify Delete
#7	11	89				01/01/2017	Modify Delete
#8	12	55				01/01/2017	Modify Delete
#9	13	100				01/01/2017	Modify Delete
#10	15	100	2500	7	16000	2016-11-01	Modify Delete
#11	16	100	2500	7	16000	2016-09-01	Modify Delete
#12	17	100	2600	7	16000	2017-01-06	Modify Delete

Create new cost

Electric Amount: 100

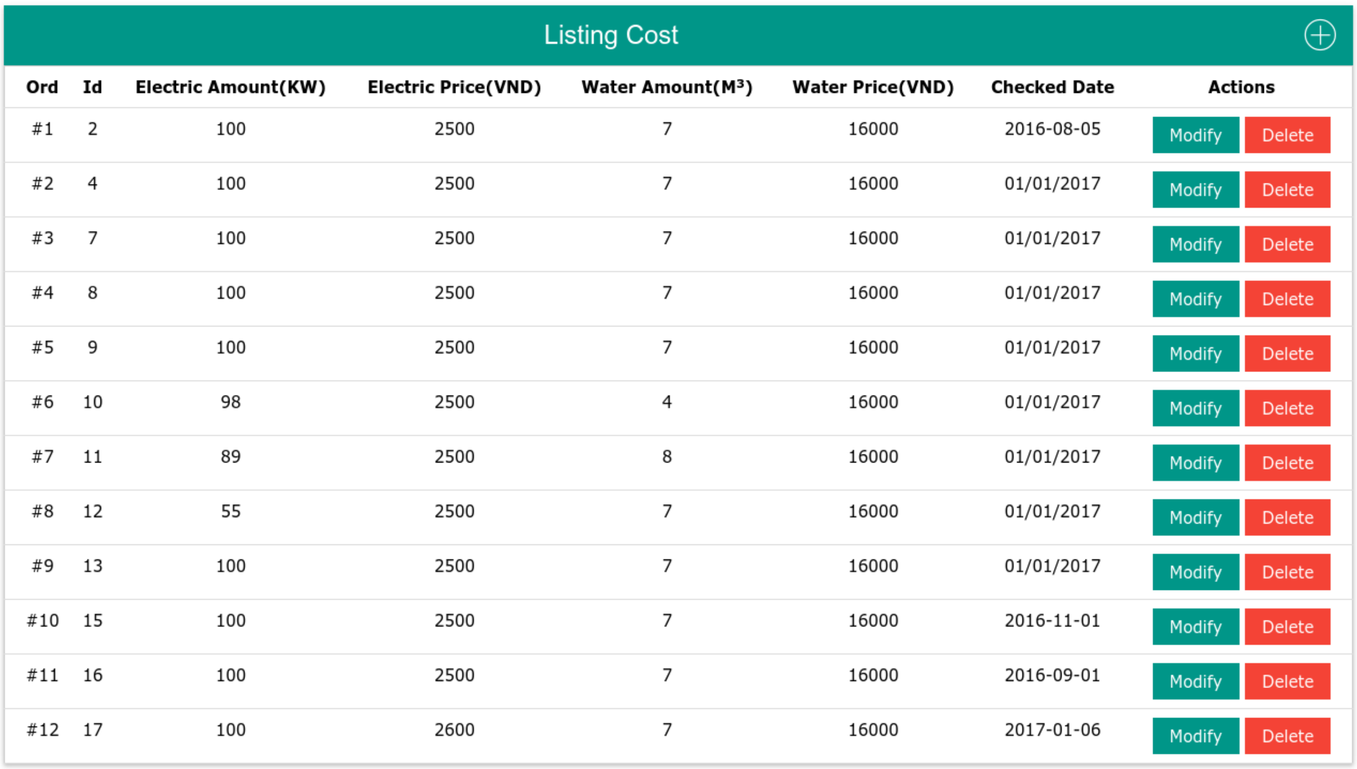
Electric Price: 2.5

Water Amount: 7

Water Price: 16.0

Checked Date: mm/dd/yyyy

Create



Listing Cost

Ord	Id	Electric Amount(KW)	Electric Price(VND)	Water Amount(M³)	Water Price(VND)	Checked Date	Actions
#1	2	100	2500	7	16000	2016-08-05	Modify Delete
#2	4	100	2500	7	16000	01/01/2017	Modify Delete
#3	7	100	2500	7	16000	01/01/2017	Modify Delete
#4	8	100	2500	7	16000	01/01/2017	Modify Delete
#5	9	100	2500	7	16000	01/01/2017	Modify Delete
#6	10	98	2500	4	16000	01/01/2017	Modify Delete
#7	11	89	2500	8	16000	01/01/2017	Modify Delete
#8	12	55	2500	7	16000	01/01/2017	Modify Delete
#9	13	100	2500	7	16000	01/01/2017	Modify Delete
#10	15	100	2500	7	16000	2016-11-01	Modify Delete
#11	16	100	2500	7	16000	2016-09-01	Modify Delete
#12	17	100	2600	7	16000	2017-01-06	Modify Delete

Session management

The application uses the icza/session module to handle some basic sessions for the authentication.

Alternate session management be found at the following sources:

- swithek/sessionup Simple, yet effective HTTP session management and identification package
- gorilla/sessions - provides cookie and filesystem sessions and infrastructure for custom session backends along with the gorilla mux
- alexedwards/scs - session management with over 19 different datastores

- - A dead simple, highly performant, highly customizable sessions service for go http servers using redis as a datastore.