# TSA Innovation Lab – Practical Code Challenge – Documentation - Chung Ho

## Initial thoughts

Looks like I need create an API to accept data from an external source and need to store the data into a database. I assume that is what the word “persist” means. Along with data validation but assume that security is provided by an outer layer in the architecture.

I am to use the “gin\_gonic” framework for this challenge.

To do serve HTTP, I know there is package to handle this, never seen it, hope it work like I think it should.

Phone number must be in E.164 format. I never heard of this, learning something new already, will need to look this up. Got a gut feeling and regular expression will be needed here.

### Assumptions of the data and the database

Full\_name: I have some doubts that the database will store this as full name. More likely there is surname/family name and first name. May even have “other names”. The sample data provide a perfect two-name approach. But the real world is not that simple. For the purposes of this challenge, I will store the data as full name as one field. BUT if I have time, I will split the string by space and assume “<First name> <Surname>” approach.

Email: hopefully I can find a regular express for GO that I can use to verify the email

Phone\_numbers: E.164 I need to look this up as this is new to me. Looks like the one-to-many structure. Looking at the data types coming in, we need to deal with strings.

### Assumptions on the request object to the API

I am assuming the data must all be filled? So full name, email and phone number can not be empty.

## Approach

* Need to get the gin\_gonic package and look it up. “go get github.com/gin-gonic/gin”
* Need examine if the net/http package as I believe this is how to serve HTTP and accept data.
* Should I accept the payload as JSON? Still thinking about it
* Document the method.

### Coding approach

* Get the data structure.
* Code a simple HTTP server to test I got HTTP server right.
* Code something simple with gin to make sure I got the framework right.
* Code something to access a database
* Try and server some data from the database
* Start code the challenge and hit all the point requested.

## Code Design

I imagine that there is a main piece of code in which is the web server. This is ideally should be as thin as possible.

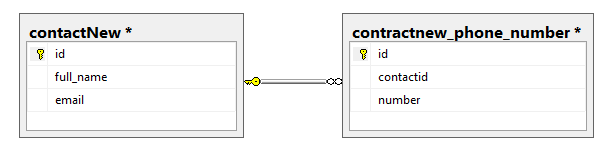
There should be modules/layers where it returns the data from data. A service layer or handler to validate the data coming in. Or could all be in one depending how complex the program is going to be.

First design of the file structures

/tsa – The main serving code go here  
 /controller  
 /models

**Database**

The data should be the main contact with phone numbers in a separate table. The code would likely use the name as the primary key. BUT in production there should a better implementation such auto incremental number or a GUID. The foreign key for the phone numbers should also follow suit. So, in production it should be closer to:



Not sure about TSA DB design conventions the “id” could be replaced with <table>Id. I am not fused.

## Endpoints

The end point will be: <http://localhost:8080/api/v1> as the base point.

**Insert**

This is the main point of the API as it is requested that the data is put at an end point and the data should persists.

End point: <http://localhost:8080/api/v1/newcontact>  
Method: Post  
Header context type: application/json; charset=utf-8

Payload

The follow are the fields, and they are all mandatory.

Full name : string: Expects <firstname> <surname>  
email: string  
phone\_numbers: array of string

Sample

{  
 "full\_name": "fredrik smite",  
 "email": "fred@somedom.com.au",  
 "phone\_numbers": ["+6139888998", "0867354300"]  
}

**List**

End point: <http://localhost:8080/api/v1/contacts>  
Method: Get

Sample return

{  
 "full\_name": "Chung Ho",  
 "email": "chung@domain.com",  
 "phone\_numbers": "0860001234"  
 }, {  
 "full\_name": "fredrik IDESTAM",  
 "email": "fred@somedom.com.au",  
 "phone\_numbers": "+6139888998"  
 }, {  
 "full\_name": "Tsa Test",  
 "email": "test@tsa.com.au",  
 "phone\_numbers": "1800123456"  
 }]

## Begin implementation

The following is like a journal of my time trying to implement the above.

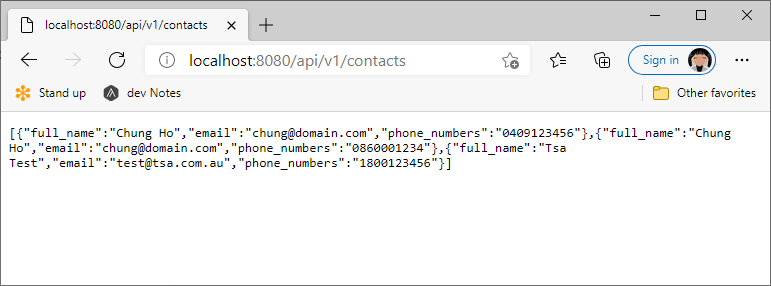
### Starting with gin

Started importing gin and got that working.

### Trying to do a database

Started with sqllite but could not get it to compile on my windows environment. It might be the version of gcc I have but wasted almost two hours on this, so I gave up and try SQL Server. Go I used the github.com/denisenkom/go-mssqldb package.

I manage to find a sql package for go to work and started using that. First attempt to get data from database to return JSON object and server as a GET method. Of course, the phone number data structure is not correct as “Chung Ho” should only appear once. I am just happy to get something working after about fourish hours.



Created POST end point to accept input to accept the data from external source. Took me a while to work out how to do parameterised SQL as I initially had string substitution which mean it is not secure from injection. I think I got it so it can not be injected. Although the validation check before it hit the SQL would not allow the code to continue.

Implemented some base validation to check for empty fields and check that full name field is in the correct format.

Implement a check to see if the person exists in the database so we do not have primary key issues.

End day approx. 7 to 8 hours.

Created a Git repository store everything and also for version control.

## Post day 1 thoughts

The program I wrote was on monolithic program which is not what I had in mind. I would really like to refactor those out those SQL calls out to a different package in a different folder like “controller” or a “handler”? The struct should probable be in its own file/package too?

There is a lot of “if err != nil” type code, just wondering if there is a better way to handle all of those. Or I could do better refactoring to other function or as such. I kind of hack those database calls as the day dragged on.

My design is not great. The POST method could or should handle new and updates at the same time really. This really depends on the requirements. I could had used SQL code such as: (where there is quotes I would use parameters

IF Exists(select full\_name from tsa.dbo.contact where full\_name = 'Chung Ho'  
 update tsa.dbo.contact set email = 'cho@chung.com.au' where full\_name = 'Chung Ho'  
else  
 insert into tsa.dbo.contact (full\_name, email) values ('Chung Ho', 'chung@domain.com')

However, with a “proper” primary key for the contact table the insert method will take it a true(?) and simple insert it as many “John Smith” into the database as you like as this will be closer to the real world. This means that there should be a separate end point for inserts and updates. Updates I can only imagine that an addition field is needed for the ID. This will open another can of worms such as, do we allow name changes?

For updates there is a concern how to do that though. Do we take the input to be true? If that is the case, we need to delete all the phone numbers first and the insert what was provided from the input. The question here what the phone numbers mean for an update with existing phone numbers?

Those connection strings. I hard coded them in EVERY function. There should be a central way to retrieve those, is there a config or something like that? As it stands, not good.

Talking about phone numbers, I was not able to validate the E.164 format. I was going to try out github.com/dongri/phonenumber first as the description sounded it is what I need. I would write a small program to test like the sql folder, before bringing it into the solution.

The email validation regular expression is something I found on The Internet, do not hurt me.

## Testing

I used Insomnia software as the testing tool.

The following was done to test the code for the Insert:

Full\_name:

* <first Name> <surname> to make sure it works.
* Add addition space should fail anyway in this field.
* Empty field or totally omitted.
* Unicode should work… But did not design the database in nvarchar but varchar.

Email

* Correct email address should work.
* Not correct email address should fail.
* Empty or omitted email should fail.

Phone\_numbers

* One phone number should work.
* Two or more phone numbers should work.
* Empty and omitted field should fail.
* NOT E.164 format, this unfortunately was **not implemented**.

## More Time?

First look into the phonenumber package to see if it can do what I need for the validation of the phone number.

Do as much possible in the “Post day 1 thoughts” as possible. The restructuring the database along with the code could take a bit.

Look into the “gorm” package as there are some interesting features for the database mapping/model.

There should a be a new End Point which can take on multiple contacts instead only one. I imagine a loop through the list of contact and insert one at a time. Should the API rollback if one contact fails or should it continue? The design of the code will be different.

I also like to fix the get method to output the correct JSON format.

## Resources

https://github.com/gin-gonic/gin

https://github.com/denisenkom/go-mssqldb

https://golang.org/doc/effective\_go.html

http://go-database-sql.org

https://www.golangprograms.com/regular-expression-to-validate-email-address.html