

A picture containing icon

Description automatically generated

**TUTORIAL IMPLEMENTASI GITHUB API MENGGUNAKAN GOLANG**

Memahami cara menggunakan Github API untuk mengunggah file ke repository melalui Golang

**Rofi Nafiis Zain**

**Haris Riyoni**



**TUTORIAL IMPLEMENTASI GITHUB API MENGGUNAKAN GOLANG**

Memahami cara menggunakan Github API untuk mengunggah file ke repository melalui Golang

***Penulis :***

Rofi Nafiis Zain

Haris Riyoni

ISBN : -

***Editor :***

Rolly Awangga

***Penyunting :***

Rolly Awangga

***Desain sampul dan Tata letak :***

Haris Riyoni

***Penerbit :***

Penerbit Buku Pedia

***Redaksi :***

Athena Residence Blok. E No. 1, Desa Ciwaruga,

Kec. Parongpong, Kab. Bandung Barat 40559

Tel. 628-775-2000-300

Email : penerbit@bukupedia.co.id

***Distributor :***

Informatics Research Center

Jl. Sariasih No. 54

Bandung 40151

Email : irc@poltekpos.ac.id

Cetakan Pertama, 2022

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan

dengan cara apa pun tanpa ijin tertulis dari penerbit

# KATA PENGANTAR

P

uji syukur kita panjatkan ke hadirat Allah SWT, karena atas berkat dan rahmat-Nya, buku ini dapat hadir sebagai panduan yang membahas Tutorial Implementasi Github API menggunakan Golang. Dengan judul "Memahami cara menggunakan Github API untuk mengunggah file ke repository melalui Golang," buku ini membawa kita dalam sebuah perjalanan mendalam untuk memahami integrasi antara Golang dan Github API dalam pengembangan aplikasi web.

Dalam era teknologi yang terus berkembang pesat, penggunaan Github API telah menjadi landasan penting dalam kolaborasi pengembangan perangkat lunak. Melalui buku ini, pembaca akan diajak untuk memahami langkah-langkah konkret dalam menggunakan Github API, khususnya dengan penerapan Golang. Pembaca akan diberikan wawasan mendalam tentang konsep dan teknik yang diperlukan untuk mengunggah file ke repository Github dengan efisien dan efektif menggunakan bahasa pemrograman Golang.

Kami, penulis, merasa sangat bahagia dapat berbagi pengetahuan dan pengalaman kami dalam dunia pengembangan aplikasi web, terutama dalam konteks integrasi antara Golang dan Github API. Buku ini bukan hanya sekadar panduan praktis, tetapi juga merupakan upaya kami untuk memberikan pemahaman menyeluruh dan mendalam tentang topik ini.

Kami berharap bahwa buku ini akan menjadi sumber ilmu yang bermanfaat bagi pembaca, terutama bagi para pengembang web yang ingin menguasai implementasi Github API dengan menggunakan Golang. Semoga buku ini dapat memberikan kontribusi positif bagi perkembangan dunia teknologi, dan semoga Allah SWT senantiasa memberikan keberkahan pada setiap langkah kita.

Akhir kata, terima kasih kepada semua pihak yang telah mendukung dan mendorong terwujudnya buku ini. Semoga bermanfaat dan selamat membaca!

Bandung, November 2023

Penulis

# PRAKATA

B

uku ini, berjudul "Tutorial Implementasi Github API menggunakan Golang: Memahami cara menggunakan Github API untuk mengunggah file ke repository melalui Golang," hadir sebagai upaya kami untuk memberikan panduan yang komprehensif dalam integrasi antara bahasa pemrograman Golang dan Github API. Dalam dunia yang terus berubah dan berkembang, pengetahuan akan integrasi ini menjadi semakin penting untuk pengembang aplikasi web.

Pentingnya kerja sama antara Golang dan Github API semakin terasa dalam pengembangan layanan web yang handal dan efisien. Namun, literatur yang membahas implementasi spesifik ini masih terbilang terbatas. Oleh karena itu, melalui buku ini, kami berupaya untuk mengisi celah tersebut dan menyajikan panduan praktis untuk para pengembang yang ingin memahami cara menggabungkan Golang dengan Github API untuk keperluan pengunggahan file ke repository.

Buku ini dibagi menjadi beberapa bab yang secara sistematis membahas langkah-langkah implementasi, konsep dasar, dan praktik coding yang diperlukan. Harapannya, pembaca akan mendapatkan pemahaman yang mendalam tentang cara memanfaatkan Github API menggunakan Golang, serta bagaimana mengintegrasikan keduanya untuk menciptakan layanan web yang efektif.

Kami berharap buku ini dapat menjadi sumber rujukan yang bermanfaat bagi pembaca, khususnya bagi para pengembang yang tengah menjelajahi kemungkinan integrasi Golang dan Github API. Terima kasih kepada semua pihak yang telah memberikan dukungan dan inspirasi dalam proses penulisan buku ini.

Semoga buku ini memberikan manfaat, inspirasi, dan pengetahuan yang berharga. Selamat menikmati pembelajaran dan eksplorasi dalam implementasi Github API menggunakan Golang.

# DAFTAR ISI

[KATA PENGANTAR i](#_Toc140677487)

[PRAKATA ii](#_Toc140677488)

[DAFTAR ISI iii](#_Toc140677489)

[DAFTAR GAMBAR v](#_Toc140677490)

[PENDAHULUAN vii](#_Toc140677491)

[BAB I 1](#_Toc140677492)

[PENGENALAN DENGAN GO DAN JS 1](#_Toc140677493)

[1.1 PENGENALAN 1](#_Toc140677495)

[1.1.1 APA ITU GO 1](#_Toc140677496)

[1.1.2 MANFAAT MEMPELAJARI GO 2](#_Toc140677498)

[1.1.2 KEUNGGULAN GO DENGAN BAHASA YANG LAIN 2](#_Toc140677499)

[1.1.3 KEKURANGAN DARI GOLANG 3](#_Toc140677500)

[1.1.4 DENGAN BAHASA APA SAJA GO BISA DIINTEGRASIKAN 3](#_Toc140677501)

[1.2 APA ITU JS 4](#_Toc140677502)

[1.2.1 MANFAAT MEMPELAJARI JS 5](#_Toc140677504)

[1.2.2 KEUNGGULAN JS DENGAN BAHASA YANG LAIN 5](#_Toc140677505)

[1.2.3 KEKURANGAN JAVASCRIPT 5](#_Toc140677506)

[1.2.4 DENGAN BAHASA APA SAJA JS BISA DIINTEGRASIKAN 6](#_Toc140677507)

[1.3 PENGERTIAN HEROKU 7](#_Toc140677508)

[1.4 PENGERTIAN MONGODB 8](#_Toc140677509)

[1.5 APA ITU GITHUB 9](#_Toc140677510)

[BAB II 11](#_Toc140677511)

[KEBUTUHAN DAN LOGIKA DALAM INTEGRASI GO DAN JS 11](#_Toc140677512)

[2.1 KEBUTUHAN-KEBUTUHAN YANG HARUS DI PENUHI 11](#_Toc140677513)

[2.1.1 INSTALASI GO 11](#_Toc140677514)

[2.1.2 JAVASCRIPT TIDAK MEMERLUKAN INSTALASI 11](#_Toc140677515)

[2.1.3 AKUN GITHUB 12](#_Toc140677516)

[2.1.4 AKUN HEROKU 12](#_Toc140677517)

[2.1.5 AKUN DAN MONGODB INSTALASI 12](#_Toc140677518)

[2.2 LOGIKA INTEGRASI 13](#_Toc140677519)

[2.2.1 GOLANG SEBAGAI BACKEND 13](#_Toc140677520)

[2.2.2 JAVASCRIPT SEBAGAI FRONTEND 13](#_Toc140677521)

[BAB III 14](#_Toc140677522)

[TAHAP PRAKTEK CODING UNTUK BACK-END DAN FRONT-END 14](#_Toc140677523)

[3.1 PEMBUATAN FOLDER DAN FILE UNTUK PACKAGE BACKEND 14](#_Toc140677524)

[3.1.1 STRUKTUR FILE BACKEND GOLANG 14](#_Toc140677525)

[3.1.2 MEMBUAT STRUCT 15](#_Toc140677526)

[3.1.3 MEMBUAT FUNCTION 17](#_Toc140677527)

[3.1.4 MEMBUAT TEST CODE FUNCTION 20](#_Toc140677528)

[3.1.5 PUSH GITHUB BACK-END DAN MEMBUAT TAG UNTUK PACKAGE BACK-END 22](#_Toc140677529)

[3.1.6 VERSI 2 MENGGUNAKAN GOCROOT BACK-END SEBAGAI FUNCTION DAN STRUCT 22](#_Toc140677530)

[3.2 PENGGUNAAN LIBRARY ITEUNG 24](#_Toc140677531)

[3.2.1 STRUKTUR FOLDER ITEUNG 24](#_Toc140677532)

[3.2.2 SETTING DB 26](#_Toc140677533)

[3.2.3 PEMBUATAN CONTROLLER 28](#_Toc140677534)

[3.2.4 PEMBUATAN URL 30](#_Toc140677535)

[3.2.5 SETTING CORS 30](#_Toc140677536)

[3.2.6 PUSH GITHUB DAN DEPLOY HEROKU ITEUNG 31](#_Toc140677537)

[3.3 PEMBUATAN FOLDER, FILE DAN INTEGRASI FRONTEND 39](#_Toc140677538)

[3.3.1 STRUKTUR FOLDER FRONT-END JAVASCRIPT 39](#_Toc140677539)

[3.3.2 MEMBUAT GET DATA API DAN HTML 40](#_Toc140677540)

[3.3.3 MEMBUAT POST DATA API DAN HTML 43](#_Toc140677541)

[3.3.4 DEPLOY FRONT-END DENGAN GITHUB PAGES 45](#_Toc140677542)

[BAB IV 48](#_Toc140677543)

[Hasil Akhir Tujuan Tampilan dan Debug 48](#_Toc140677544)

[4.1 HASIL TAMPILAN FRONTEND 48](#_Toc140677545)

[4.2 DEBUG HASIL FRONTEND 49](#_Toc140677547)

[DAFTAR PUSTAKA 52](#_Toc140677548)

[GLOSARIUM 53](#_Toc140677549)

[INDEKS 58](#_Toc140677550)

# DAFTAR GAMBAR

**BAB I**

[Gambar 1. 1 Logo Golang 1](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677371)

[Gambar 1. 2 Logo JavaScript 4](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677372)

[Gambar 1. 3 Heroku Logo 7](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677373)

[Gambar 1. 4 MongoDB Logo 8](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677374)

[Gambar 1. 5 Github Logo 9](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677375)

**BAB II**

**BAB III**

[Gambar 3. 1 Struktur File Backend 14](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677376)

[Gambar 3. 2 File Struct 16](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677377)

[Gambar 3. 3 Function Backend 1 18](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677378)

[Gambar 3. 4 Function Backend 2 19](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677379)

[Gambar 3. 5 Function GetNamaUser 20](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677380)

[Gambar 3. 6 File Test Function Code 21](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677381)

[Gambar 3. 7 Test Get Nama dan Surat 21](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677382)

[Gambar 3. 8 Function Module 23](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677383)

[Gambar 3. 9 Struct Package Model 24](#_Toc140677384)

[Gambar 3. 10 Struktur Folder Iteung 24](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677385)

[Gambar 3. 11 Import di Controller 26](#_Toc140677386)

[Gambar 3. 12 Setting Database di db.go 27](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677387)

[Gambar 3. 13 Tampilan Database 27](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677388)

[Gambar 3. 14 Iteung Controller 29](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677389)

[Gambar 3. 15 File Url.go 30](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677390)

[Gambar 3. 16 Setting Cors di Cors.go 31](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677391)

[Gambar 3. 17 Deploy App Heroku 1 32](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677392)

[Gambar 3. 18 Deploy App Heroku 2 32](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677393)

[Gambar 3. 19 Add JawsDB Heroku 1 33](#_Toc140677394)

[Gambar 3. 20 Add JawsDB Heroku 2 34](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677395)

[Gambar 3. 21 Tampilan Sudah TerInstall JawsDB 34](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677396)

[Gambar 3. 22 Tampilan Jika Membuka JawsDB 35](#_Toc140677397)

[Gambar 3. 23 Tampilan Setting App Heroku 35](#_Toc140677398)

[Gambar 3. 24 Setting Config Vars 36](#_Toc140677399)

[Gambar 3. 25 Add IP Heroku Akses Database MongoDB 37](#_Toc140677400)

[Gambar 3. 26 Allow Akses MongoDB 38](#_Toc140677401)

[Gambar 3. 27 Tampilan Aplikasi dengan OpenApp 38](#_Toc140677402)

[Gambar 3. 28 Tampilan Aplikasi 38](#_Toc140677403)

[Gambar 3. 29 Struktur File Javascript 39](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677404)

[Gambar 3. 30 Get Data dari /nama 41](#_Toc140677405)

[Gambar 3. 31 Tampilan Url.js 41](#_Toc140677406)

[Gambar 3. 32 Template dan Isi untuk Data API 42](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677407)

[Gambar 3. 33 Tampilan Code Croot.js 42](#_Toc140677408)

[Gambar 3. 34 Table HTML 43](#_Toc140677409)

[Gambar 3. 35 Tampilan /Surat/Insert 44](#_Toc140677410)

[Gambar 3. 36 Tampilan Url2.js 44](#_Toc140677411)

[Gambar 3. 37 Tampilan Croot2.js 44](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677412)

[Gambar 3. 38 Form HTML 45](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677413)

[Gambar 3. 39 Deploy Frontend Github Pages 46](#_Toc140677414)

[Gambar 3. 40 Tampilan Aplikasi Front-End 46](#_Toc140677415)

[Gambar 3. 41 Struktur File Repo ke Url 47](#_Toc140677416)

[Gambar 3. 42 Tampilan Index.html Github Pages 47](#_Toc140677417)

**BAB IV**

[Gambar 4. 1 Tampilan Index Get API GitPages 48](#_Toc140677418)

[Gambar 4. 2 Tampilan Form Chat 49](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677419)

[Gambar 4. 3 Console Get API 49](#_Toc140677420)

[Gambar 4. 4 Masukan Data Form Chat 50](#_Toc140677421)

[Gambar 4. 5 Form Chat Terkirim 51](#_Toc140677422)

[Gambar 4. 6 Tampilan Hasil Send Post Form Chat 51](file:///C:\Users\imyha\Documents\penting%20buat%20buku\integrasi%20go%20dan%20js%20dalam%20pembuatan%20web%20service%20surat%20menyurat.docx#_Toc140677423)

# PENDAHULUAN

Dalam era revolusi teknologi yang terus berkembang, integrasi perangkat lunak menjadi kunci utama untuk menciptakan solusi aplikasi yang efektif dan inovatif. Salah satu aspek penting dalam pengembangan aplikasi web adalah kemampuan untuk berinteraksi dengan repositori kode, dan Github API telah menjadi fondasi yang tak tergantikan dalam hal tersebut.

Buku ini, berjudul "Tutorial Implementasi Github API menggunakan Golang: Memahami cara menggunakan Github API untuk mengunggah file ke repository melalui Golang," memasuki ranah penting ini dengan fokus pada integrasi antara bahasa pemrograman Golang dan Github API. Tujuan kami adalah memberikan panduan komprehensif bagi pembaca, terutama para pengembang web, untuk memahami dan mengimplementasikan dengan efektif Github API menggunakan Golang.

Dalam dunia pengembangan perangkat lunak, Github telah menjadi pusat kolaborasi dan manajemen kode yang sangat diperlukan. Github API, sebagai antarmuka pemrograman aplikasi dari platform tersebut, memberikan kemampuan untuk mengotomatiskan sejumlah tugas, termasuk pengunggahan file ke repository. Golang, dengan kecepatan eksekusi dan kemudahan penggunaannya, menjadi pilihan yang ideal untuk mengintegrasikan aplikasi dengan Github API.

Buku ini terdiri dari beberapa bab yang secara progresif membimbing pembaca melalui konsep-konsep dasar, langkah-langkah implementasi, dan praktek coding. Bab pertama akan mengenalkan konsep dasar Golang dan Github API, sementara bab-bab berikutnya akan membahas secara mendalam langkah-langkah praktis untuk mengunggah file ke repository melalui Golang.

Pembaca akan mendapatkan pemahaman yang kuat tentang cara menggabungkan kekuatan Golang dan Github API untuk menciptakan layanan web yang efisien dan andal. Melalui contoh kode yang jelas dan penjelasan yang terperinci, buku ini dirancang untuk memberikan wawasan praktis dan kemampuan langsung yang dapat diterapkan dalam proyek pengembangan web.

Mari bersiap untuk memasuki perjalanan menarik dalam mengimplementasikan Github API dengan menggunakan Golang, Selamat Membaca!

# BAB I

# PENGANTAR GO DAN GITHUB

### PENGENALAN

Oke kita masuk ke bagian paling awal dari buku ini yang dimana kita akan berkenalan dengan Bahasa yang akan kita gunakan pada kesempatan kali ini, yaitu Go Language.

### 1.1.1 TENTANG GO

Go, juga dikenal sebagai Golang, adalah bahasa pemrograman open source yang dikembangkan oleh Google pada tahun 2007. Go dirancang dengan tujuan menyederhanakan proses pengembangan perangkat lunak dan meningkatkan kinerja. Keunggulan Go terletak pada sintak yang mudah dipahami, kompilasi yang cepat, serta kemampuannya dalam mengelola konkurensi dan pemrograman paralel. Dengan perpustakaan standar yang kuat, Go memberikan kemudahan dalam membangun aplikasi yang efisien dan andal.



Gambar 1. Logo Golang

### 1.1.2 MANFAAT MEMPELAJARI GO

Mempelajari Go memiliki beberapa manfaat yang signifikan adalah sebagai berikut:

* Pertama, Go menawarkan sintak yang sederhana, mempercepat proses pembelajaran dan membantu pengembang menghasilkan kode yang lebih bersih dan efisien.
* Kedua, Go menyediakan dukungan bawaan untuk konkurensi, memudahkan pengembang dalam menangani proses paralel. Fitur ini sangat berguna dalam pengembangan aplikasi berskala besar. Selain itu, Go juga memiliki performa yang tinggi dan penggunaan memori yang efisien, menjadikannya pilihan yang baik untuk aplikasi yang responsif dan scalable.

### 1.1.2 KEUNGGULAN GO DENGAN BAHASA YANG LAIN

Go memiliki beberapa keunggulan dibandingkan dengan bahasa pemrograman lainnya. Pertama, Go memiliki kinerja yang sangat baik, dengan waktu eksekusi yang cepat dan penggunaan memori yang efisien. Go juga memiliki dukungan bawaan untuk konkurensi, sehingga mudah untuk mengembangkan aplikasi yang berjalan secara paralel.

Selain itu, Go memiliki sistem manajemen memori otomatis (garbage collector) yang canggih, yang mengurangi beban pengembang dalam mengelola alokasi memori. Go juga dikenal dengan stabilitasnya, dengan versi bahasa yang kompatibel secara mundur dan dukungan yang kuat dari komunitas pengembang.

### 1.1.3 KEKURANGAN DARI GOLANG

Golang atau Go adalah bahasa pemrograman yang telah mendapatkan popularitas dalam beberapa tahun terakhir. Namun, seperti halnya bahasa pemrograman lainnya, Golang juga memiliki beberapa kekurangan yang harus dipertimbangkan sebelum memilihnya sebagai bahasa utama dalam pengembangan perangkat lunak. Salah satu kekurangan utama Golang adalah kurangnya dukungan untuk tipe generic. Tipe generic memungkinkan pengembang menulis kode yang bekerja secara generik dengan berbagai jenis tipe data tanpa perlu mengulangi kode yang serupa. Ketidakhadiran tipe generic di Golang dapat mengakibatkan perulangan kode yang dapat mengurangi efisiensi dan kejelasan kode.

Selain itu, beberapa pengembang juga menganggap ekosistem pihak ketiga untuk Golang masih kurang berkembang dibandingkan bahasa pemrograman populer lainnya seperti Python atau JavaScript. Meskipun ekosistem Golang terus berkembang dan semakin beragam, namun terkadang sulit untuk menemukan atau memperoleh pustaka pihak ketiga yang dibutuhkan dalam proyek tertentu. Keterbatasan ini dapat menjadi tantangan bagi pengembang yang bergantung pada pustaka pihak ketiga untuk mempercepat proses pengembangan dan menghadapi kebutuhan khusus.

### 1.1.4 DENGAN BAHASA APA SAJA GO BISA DIINTEGRASIKAN

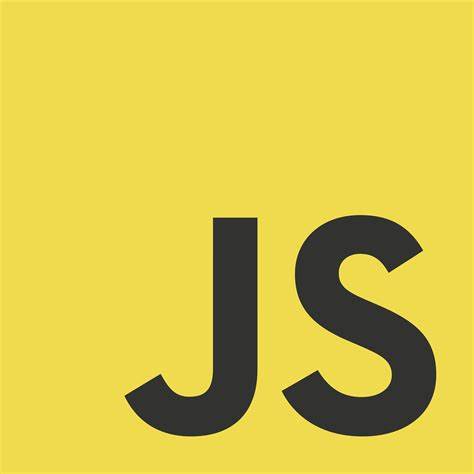
Salah satu keunggulan Golang adalah kemampuannya untuk diintegrasikan dengan bahasa pemrograman lain. Dengan antarmuka C, Golang dapat berkomunikasi dengan bahasa pemrograman lain yang mendukung antarmuka C, seperti C++ dan Rust. Integrasi ini memungkinkan pengembang menggunakan kode dari bahasa-bahasa tersebut sambil tetap memanfaatkan keunggulan kinerja tinggi dari Golang.

Selain itu, Golang juga dapat diintegrasikan dengan bahasa pemrograman Python menggunakan paket Cgo. Dengan dukungan ini, pengembang dapat dengan mudah berkolaborasi antara Golang dan Python, dua bahasa pemrograman yang sering digunakan dalam pengembangan perangkat lunak.

Lebih lanjut, Golang juga berfungsi sebagai antarmuka untuk memanggil kode JavaScript melalui WebAssembly. Fitur ini membuka peluang integrasi yang lebih erat dengan teknologi web modern, sehingga memungkinkan pengembang menggabungkan kekuatan Golang dengan ekosistem JavaScript yang luas dalam pengembangan aplikasi web kompleks dan efisien.

### APA ITU JS

JavaScript, atau biasa disingkat JS, adalah bahasa pemrograman tingkat tinggi yang berfungsi sebagai "otak" di balik aplikasi web. Pertama kali dikembangkan pada tahun 1995, JavaScript telah menjadi bahasa yang sangat populer dan digunakan secara luas dalam pengembangan web. JavaScript memungkinkan interaksi dan manipulasi elemen halaman web secara dinamis, serta mengontrol perilaku dan tampilan halaman tersebut.



### 

Gambar 1. Logo JavaScript

### 1.2.1 MANFAAT MEMPELAJARI JS

Mempelajari JavaScript memberikan manfaat yang sangat penting dalam pengembangan aplikasi web. Pertama, JavaScript didukung oleh hampir semua browser web modern, sehingga mempelajarinya memberikan akses ke berbagai platform dan pengguna. Kedua, JavaScript memungkinkan pengembang untuk menciptakan tampilan yang dinamis dan interaktif, meningkatkan pengalaman pengguna secara keseluruhan. Dengan JavaScript, pengembang dapat mengatur interaksi pengguna, validasi data input, dan memodifikasi tampilan halaman secara real-time. Selain itu, JavaScript juga dapat digunakan untuk pengembangan server-side menggunakan platform seperti Node.js, memungkinkan pengembang untuk membuat aplikasi yang berjalan di sisi server.

### 1.2.2 KEUNGGULAN JS DENGAN BAHASA YANG LAIN

JavaScript memiliki beberapa keunggulan yang membedakannya dari bahasa pemrograman lainnya. Pertama, JavaScript dapat dijalankan langsung di browser web tanpa perlu instalasi tambahan, sehingga memungkinkan pengembang untuk membuat aplikasi web yang mudah diakses oleh pengguna. Kedua, JavaScript memiliki dukungan yang kuat untuk integrasi dengan HTML dan CSS, yang memungkinkan pengembang untuk menciptakan tampilan yang menarik dan dinamis. Selain itu, JavaScript memiliki lingkungan pengembangan yang kaya dengan banyak perpustakaan dan kerangka kerja (framework) yang dapat digunakan untuk mempercepat pengembangan aplikasi.

### 1.2.3 KEKURANGAN JAVASCRIPT

Sebagai bahasa pemrograman yang paling banyak digunakan dalam pengembangan web, JavaScript telah menjadi tulang punggung dalam menciptakan tampilan interaktif dan dinamis pada berbagai situs dan aplikasi. Namun, seperti bahasa pemrograman lainnya, JavaScript juga memiliki beberapa kekurangan yang perlu diperhatikan.

Salah satu kekurangan utama JavaScript adalah dukungan untuk tipe data yang lemah. Dalam JavaScript, tipe data variabel tidak perlu dideklarasikan secara eksplisit, yang dapat menyebabkan kesalahan sulit diidentifikasi saat berinteraksi dengan tipe data yang berbeda. Kekurangan ini dapat mengakibatkan bug yang sulit dilacak dan diperbaiki, terutama dalam proyek-proyek besar yang melibatkan banyak pengembang.

Selain itu, karena JavaScript dijalankan di sisi klien (client-side), kinerja aplikasi dapat dipengaruhi oleh spesifikasi perangkat yang digunakan oleh pengguna akhir. Jika perangkat yang digunakan kurang kuat atau koneksi internetnya lambat, kinerja aplikasi JavaScript dapat mengalami penurunan, menyebabkan pengalaman pengguna yang kurang memuaskan.

### 1.2.4 DENGAN BAHASA APA SAJA JS BISA DIINTEGRASIKAN

JavaScript memiliki kemampuan integrasi yang kuat dengan bahasa pemrograman lain, terutama melalui teknologi WebAssembly. Dengan WebAssembly, kode yang ditulis dalam bahasa lain, seperti C++ atau Rust, dapat dikompilasi menjadi format biner yang dapat dijalankan di lingkungan JavaScript. Integrasi ini memberikan fleksibilitas bagi pengembang untuk menggunakan kode dari bahasa-bahasa tersebut sambil tetap memanfaatkan keunggulan dan kemampuan JavaScript dalam pengembangan aplikasi web yang canggih.

Selain itu, JavaScript juga dapat diintegrasikan dengan bahasa pemrograman server-side seperti Node.js. Node.js memungkinkan penggunaan JavaScript di lingkungan server, membuka peluang untuk mengembangkan aplikasi yang konsisten di kedua sisi, baik sisi klien maupun sisi server.

Integrasi JavaScript dengan bahasa-bahasa pemrograman lain ini memberikan kesempatan bagi para pengembang untuk menciptakan solusi yang lebih kuat dan kompleks dalam dunia pengembangan perangkat lunak. Dengan cara ini, pengembang dapat memanfaatkan kemampuan unik dari masing-masing bahasa untuk mencapai tujuan tertentu dalam proyek pengembangan yang beragam.

### PENGERTIAN HEROKU

Heroku adalah platform cloud yang menyediakan layanan hosting aplikasi web dan layanan pengembangan berbasis cloud. Dengan Heroku, para pengembang dapat dengan mudah mengunggah, mengelola, dan menjalankan aplikasi web mereka tanpa harus khawatir tentang konfigurasi infrastruktur server atau skalabilitas. Heroku berfokus pada menyederhanakan proses pengembangan dan penyebaran aplikasi sehingga pengembang dapat lebih fokus pada kode dan fungsionalitas aplikasi mereka.



Gambar 1. Heroku Logo

Kelebihan Heroku:

- Kemudahan Penggunaan: Heroku menyediakan antarmuka pengguna yang ramah pengembang yang memungkinkan pengunggahan dan penyebaran aplikasi dalam beberapa langkah sederhana.

- Skalabilitas Otomatis: Heroku secara otomatis mengelola dan menyesuaikan sumber daya server sesuai dengan permintaan aplikasi, sehingga memastikan aplikasi dapat diakses dengan lancar tanpa gangguan, bahkan saat menghadapi lonjakan lalu lintas.

- Dukungan Banyak Bahasa: Heroku mendukung banyak bahasa pemrograman termasuk Ruby, Node.js, Python, PHP, Java, dan lainnya, memungkinkan pengembang untuk menggunakan bahasa favorit mereka dalam mengembangkan aplikasi.

- Integrasi dengan Git: Heroku terintegrasi dengan sistem kontrol versi Git, memungkinkan pengembang untuk dengan mudah mengunggah dan mengelola kode aplikasi mereka melalui Git.

### PENGERTIAN MONGODB

MongoDB adalah sistem basis data NoSQL (non-relational) yang sangat populer dalam pengembangan perangkat lunak modern. MongoDB menggunakan struktur dokumen JSON untuk menyimpan data, sehingga memungkinkan fleksibilitas yang lebih besar dibandingkan basis data relasional tradisional. MongoDB dirancang untuk mengelola data yang sangat besar, dan cocok untuk aplikasi yang membutuhkan skalabilitas dan kinerja tinggi.

Gambar 1. MongoDB Logo

Kelebihan MongoDB:

Fleksibilitas Skema: MongoDB tidak memiliki skema yang kaku seperti basis data relasional, memungkinkan pengembang untuk dengan mudah mengubah struktur data tanpa harus memodifikasi seluruh basis data. Ini memudahkan pengembangan dan pengujian aplikasi yang berubah dengan cepat.

Kinerja Tinggi: MongoDB dioptimalkan untuk kinerja yang cepat dan mampu menangani data dengan ukuran besar dan lalu lintas yang tinggi. Indeks yang efisien dan dukungan untuk replikasi dan sharding memastikan kinerja yang optimal dalam skala besar.

Skalabilitas Horizontal: MongoDB dapat dengan mudah diskalakan secara horizontal dengan menambahkan lebih banyak server, sehingga dapat menangani pertumbuhan data dan permintaan pengguna tanpa mengorbankan kinerja.

Dukungan untuk Data Semi-Struktur: Format dokumen JSON pada MongoDB memungkinkan penyimpanan data yang semi-struktur, seperti data yang berubah atau tidak tetap dari waktu ke waktu.

### 1.5 APA ITU GITHUB

GitHub adalah platform hosting yang populer untuk pengembangan perangkat lunak berbasis Git. Ini memungkinkan para pengembang untuk menyimpan, mengelola, dan berbagi kode sumber proyek secara terpusat. GitHub memainkan peran penting dalam kolaborasi dan kerjasama tim dalam pengembangan perangkat lunak dengan menyediakan alat kolaborasi, sistem kontrol versi yang kuat, dan integrasi dengan berbagai layanan pengembangan lainnya.

Gambar 1. Github Logo

Kelebihan GitHub:

Kolaborasi dan Sharing: GitHub memungkinkan pengembang untuk berkolaborasi dalam proyek secara terpusat. Dengan adanya sistem kontrol versi Git, tim dapat bekerja bersama-sama dalam mengembangkan kode, melakukan perubahan, dan menggabungkan kode secara mudah.

Sistem Kontrol Versi yang Kuat: Git adalah sistem kontrol versi yang populer dan kuat, dan GitHub menyediakan antarmuka pengguna yang ramah untuk mengelola repositori Git. Pengembang dapat dengan mudah melakukan perubahan, membuat cabang, melakukan penggabungan, dan mengatasi konflik dengan cepat dan efisien.

Integrasi dengan Layanan Pengembangan Lainnya: GitHub memiliki integrasi yang luas dengan berbagai layanan pengembangan seperti CI/CD, layanan penerapan otomatis, alat manajemen proyek, dan banyak lagi. Hal ini memungkinkan pengembang untuk membangun alur kerja pengembangan yang otomatis dan efisien.

Masyarakat Open Source: GitHub memiliki komunitas yang besar dan aktif dari pengembang dan proyek open source. Ini memudahkan pengembang untuk menemukan proyek open source yang menarik, berkontribusi ke proyek tersebut, atau menggunakan perangkat lunak open source dalam proyek mereka.

Top of Form

# BAB II

# KEBUTUHAN DAN LOGIKA DALAM INTEGRASI GO DAN JS

### 2.1 KEBUTUHAN-KEBUTUHAN YANG HARUS DI PENUHI

Oke kita masuk pada bagian yang pertama dari Buku ini yaitu Kebutuhan dan Logika dalam integrasi Bahasa Golang dan Javascript, yang dimana kita masuk ke kebutuhan atau Persyaratan apa saja untuk mengikuti Pembuatan Web-Service ini, Oke simak ya kebutuhan-kebutuhan yang diperlukan karna ini kebutuhan ini adalah syarat untuk bisa membuat Web-Service ini.

Tujuan dari buku ini yaitu membuat pertama, dapat mengambil data ke dalam index.html dari API yang kita buat dan kedua, membuat Post chat ke dalam database.

### 2.1.1 INSTALASI GO

Tentu selanjutnya yang kita install adalah Golang, untuk instalasi kalian bisa melalui link golang resmi bisa melalui link ini [The Go Programming Language](https://go.dev/). Untuk tambahan informasi Bahasa go ini dibuat Oleh Google dan Go adalah Bahasa yang digadangi akan populer untuk kedepannya. Dan jika kamu sudah menguasai dasar dari GO ini akan sangat mempermudah kamu dalam mengikuti buku ini.

### 2.1.2 JAVASCRIPT TIDAK MEMERLUKAN INSTALASI

Selanjutnya Javascript, Javascript ini tidak memerlukan instalasi seperti Bahasa yang lain, jadi kamu bisa langsung membuat file ber-extensi .js, beda cerita jika kamu ingin menggunakan nodeJS atau ReactJs tentunya ada beberapa yang harus di install jika ingin menggunakan fitur-fitur yang ada.

### 2.1.3 AKUN GITHUB

Membuat akun Github, Github memiliki peran yang penting karena pada kasus apapun dalam membuat project, kamu dapat menyimpan project kamu di github dan juga github banyak platform deploy untuk memudahkan developer atau programmer untuk deploy Back-End atau Front-End dengan Project kita yang tersimpan di Github kita.

Contohnya Github sendiri memiliki deploy Frontend yaitu bernama Github-Pages fitur ini akan digunakan kita kedepannya karena github memberikan fitur ini secara gratis. Platform seperti 000Webhost, Netlify, Heroku dst. Adalah contoh platform yang terintegrasi dengan Github, Netlify untuk deploy atau mempublikasi Web Statis dan Heroku untuk Publikasi Website Dinamis.

Untuk Pendaftaran kamu bisa mengunjungi langsung Website mereka di official resminya atau melalui link ini [GitHub](https://github.com/). Dan diharapkan kamu sudah menguasai git.

### 2.1.4 AKUN HEROKU

Selanjutnya yang dibutuhkan ini adalah akun Heroku yang dimana kita akan mempublikasikan atau deploy Backend Golang kita ke Heroku, tentunya ini ada biayanya, tapi kamu bisa menggunakan akun akademik atau akun yang telah diberikan oleh kampus, untuk mendapatkan uang atau seperti saldo diheroku yang digunakan untuk mendeploy Back-End kita.

Untuk mendaftar kamu dapat mengunjungi website resmi mereka atau bisa melalui link disini [Cloud Application Platform | Heroku](https://www.heroku.com/).

### 2.1.5 AKUN DAN MONGODB INSTALASI

Tahap ini kamu diharuskan untuk membuat akun MongoDB Atlas dan juga Menginstall MongoDB sebagai tempat database kita untuk studi kasus kita tenang ini gratis kok.

Untuk Membuat akun kamu bisa melalui link ini [MongoDB Atlas: Cloud Document Database | MongoDB](https://www.mongodb.com/cloud/atlas/lp/try4?utm_source=bing&utm_campaign=search_bs_pl_evergreen_atlas_core_prosp-brand_gic-null_apac-id_ps-all_desktop_eng_lead&utm_term=atlas%20mongo&utm_medium=cpc_paid_search&utm_ad=e&utm_ad_campaign_id=415204525&adgroup=1211662283600100&msclkid=7b40468ef3af1cb9c732220284086f0c) atau bisa di website resmi mereka ya, kamu juga dapat mendownload dan menginstall MongoDB melalui link ini [MongoDB Compass Download (GUI) | MongoDB](https://www.mongodb.com/try/download/compass) atau bias kunjungi website resmi mereka ya.

### LOGIKA INTEGRASI

Oke Kita sudah masuk kedalam Logika dari Aplikasi yang kita akan buat disini aku mau memberi sedikit pemahaman tentang ke-2 bahasa yaitu Golang dan Javascript mereka digunakan untuk apa aja sih?

### 2.2.1 GOLANG SEBAGAI BACKEND

Pertama Golang, Bahasa Pemrograman Golang ini kita akan gunakan untuk sebagai Backend untuk Aplikasi kita yang dimana tentunya Golang adalah Bahasa dalam Backend terkuat dan tercepat dibanding dengan Bahasa yang lain karena golang meng-convert code program kita menjadi biner yang dimana ini adalah keunggulan Go dari pada Bahasa yang lain.

Nanti kita akan membuat code program package Back-End yang pertama nanti itu akan kita dipanggil atau digunakan pada Library Iteung yang akan kita gunakan nanti, kamu bisa cek Library ini di link ini ya [aiteung/iteung: boilerplate for using aiteung package (github.com)](https://github.com/aiteung/iteung).

### 2.2.2 JAVASCRIPT SEBAGAI FRONTEND

Selanjutnnya Javascript, kita akan menggunakan Javascript untuk Frontend pada Aplikasi kita yang dimana Javascript digunakan untuk sebagai fungsi dari API yang sudah di deploy dari Backend, Tujuan kita ini Mulai dari Get Data dari API juga akan melakukan Post Data API.

Kita akan menggunakan Library dari JSCroot yang dimana ini tersedia di github yang bisa kamu lihat disini [jscroot/jscroot.github.io: VanillaJS Micro Framework](https://github.com/jscroot/jscroot.github.io).

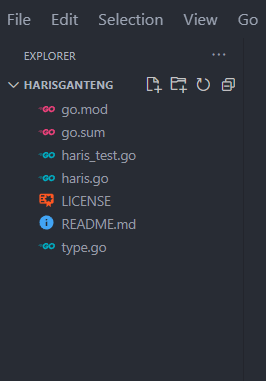
# BAB III

# TAHAP PRAKTEK CODING UNTUK BACK-END DAN FRONT-END

### 3.1 PEMBUATAN FOLDER DAN FILE UNTUK PACKAGE BACKEND

Oke kita masuk ke tahap inti yaitu tahap praktek code untuk aplikasi kita yang dimana disini kita akan mengintegrasikan Golang dengan Javascript.Pertama kita disini akan melakukan tahap pembuatan Back-End di Golang untuk aplikasi kita.

### 3.1.1 STRUKTUR FILE BACKEND GOLANG

Pada tahap ini kita akan membuat dan mengetahui struktur code yang akan kita buat yaitu package Back-End yang dimana ini tahap yang cukup penting untuk kita lakukan, Oke kita lanjut ke struktur file nya.

Gambar 3. Struktur File Backend

Oke, kita masuk ke bagian filenya kita bisa lihat pada gambar diatas terdapat file type.go.

* Type.go

Ini adalah file yang akan kita gunakan sebagai struct kita atau Bahasa gampangnya table dan attribute, Struct ini kita akan jadikan sebagai table-table dan attribute-attribute yang akan kita jadikan database dari Aplikasi kita.

* Haris.go

haris.go ini akan berisi function-function yang dimana kita akan membuat function yang akan membuat data-data dari database kita bisa diakses untuk API atau deploy ke Online Private. Kita mulai akan membuat Function Get dan Post disini ke database kita yaitu dalam Studi kasus ini kita menggunakan MongoDB sebagai database kita.

* Haris\_test.go

Haris\_test.go ini kita akan gunakan sebagai tester untuk function-function yang telah kita buat dalam file haris.go yang dimana untuk mengecek apakah function itu berjalan atau tidak. Peran dari tester ini sangat penting untuk dilakukan karna jika tidak ada tester code maka kita tidak tahu apakah function atau code kita benar atau tidak.

Apalagi dalam sebuah Company, tentunya sekelas company pasti akan mengetest code dari aplikasi-aplikasi yang mereka buat. Maka dari itu kita disini membuat dari testing code dari aplikasi kita.

### 3.1.2 MEMBUAT STRUCT

Oke kita masuk pada ke tahap pembuatan struct kita bisa ambil contoh pada gambar dibawah ini, ini adalah contoh struct pada studi kasus kita kali ini yaitu surat-menyurat.

Sebelum itu kita lalukan go mod init untuk dependensi kita dengan mengetikan pada terminal vscode kalian yaitu **go mod init** **github.com/harisriyoni3/harisganteng**, nama github juga nama repo yang kalian udah buat untuk menaruh file Backend pada Github. Lalu kalian kompilasi dependensi kalian dengan mengetikan diterminal vscode yaitu **go mod tidy.** Lalu kalian boleh membuat struct dibawah ini.



Gambar 3. File Struct

Jadi code diatas ini kamu bisa mengganti package dengan nama kamu dan juga untuk struct kamu bisa mengikuti code diatas atau kamu juga mengganti sesuai keinginan kamu.

* bson : mendefinisikan nama field pada collection di database mongoDB
* json : mendefinisikan nama atribut pada pertukaran json melalui API
* omitempty : diperbolehkan tidak diisi / nullable

Import:"go.mongodb.org/mongo-driver/bson/primitive": Ini adalah paket yang digunakan untuk bekerja dengan BSON (Binary JSON), format dokumen yang digunakan dalam MongoDB. Pada paket ini, kita mengimport tipe data "primitive.ObjectID" yang digunakan sebagai tipe data unik untuk mengidentifikasi dokumen dalam MongoDB.

### 3.1.3 MEMBUAT FUNCTION

Kita masuk ke bagian ke-2 yaitu Membuat Function, Seperti yang kita ketahui function ini yang akan menjadi sebagai pengakses database kita mulai dari membuat function Get, Post, Update dan Juga Delete tetapi kita disini akan membuat 2 function yaitu Insert dan Juga Post untuk akses dari database kita.

Disini kami akan menampilkan keseluruhan dari file haris.go yang dimana disini akan berisi semua function-functionnya, code berada di bawah ini:

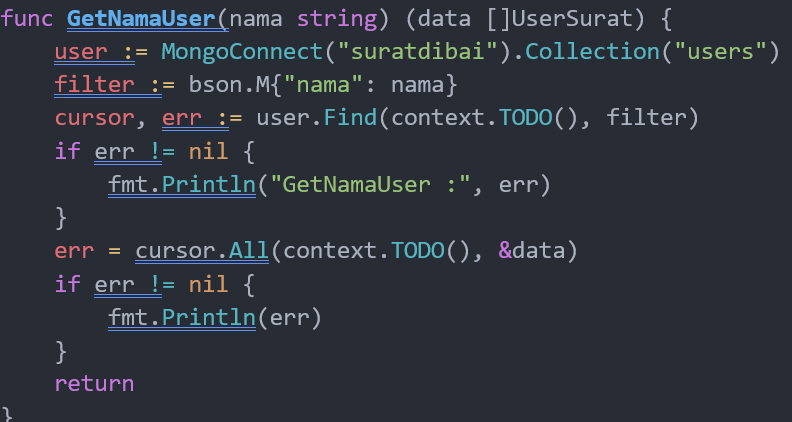


Gambar 3. Function Backend 1



Gambar 3. Function Backend 2

Disini kita akan menampilkan code function yang akan kita gunakan saja yaitu yang pertama adalah GetNamaUser yang dimana disini kita akan mengambil data user untuk mengambil data user kita bisa mengambil 3 opsi yaitu mengambil nama berdasarkan id, nama ataupun nomor, bisa dilihat pada gambar dibawah ini untuk function GetNamaUser:



Gambar 3. Function GetNamaUser

Pada Code diatas berfungsi untuk mengambil data dari database dengan menggunakan nama dari table user.

### 3.1.4 MEMBUAT TEST CODE FUNCTION

Setelah membuat Function-nya untuk kita diwajibkan untuk membuat unit test untuk mengetest code dalam file haris\_test.go codenya sebagai berikut:



Gambar 3. File Test Function Code



Gambar 3. Test Get Nama dan Surat

Pada code diatas ini akan mengetest code kita, dimana disini kita akan get berdasarkan nama user dan juga get surat dengan parameter Kamu.

### 3.1.5 PUSH GITHUB BACK-END DAN MEMBUAT TAG UNTUK PACKAGE BACK-END

Jika kamu telah membuat struct, function dan test code maka tahap selanjutnya adalah kamu membuat tag untuk dipanggil tag tersebut atau digunakan untuk di library iteung yang akan kita gunakan nanti.

Command di terminal vscode kalian:

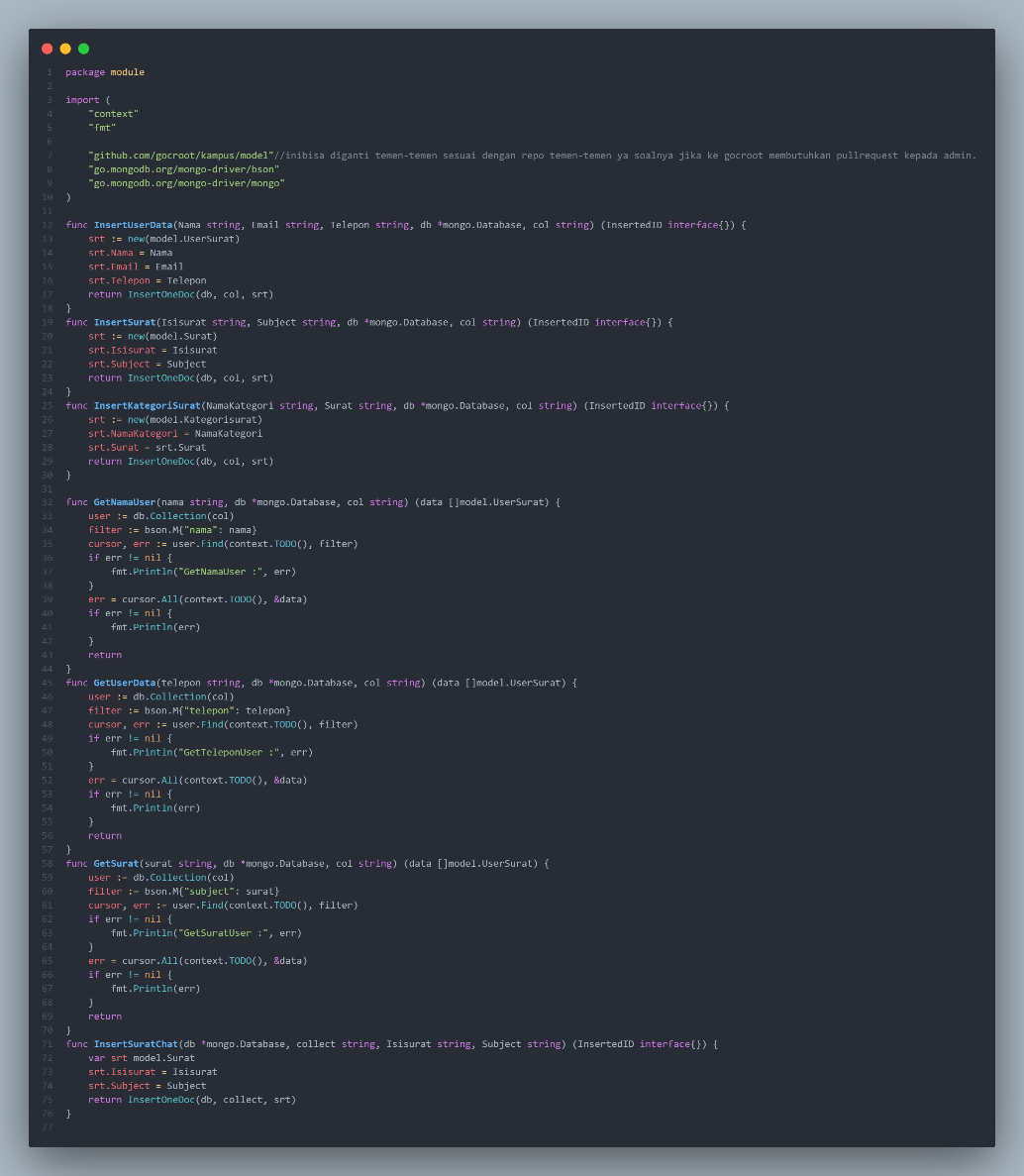
* **git tag**
* **git tag v0.0.1**
* **git push origin --tags**
* **go list -m** [**github.com/usernamegithub/namarepo@v0.0.1**](mailto:github.com/usernamegithub/namarepo@v0.0.1) **(versi tag temen temen)**
* **lalu masuk ke repo library iteung lalu go get** [**github.com/usernamegithub/namarepo@v0.0.1**](mailto:github.com/usernamegithub/namarepo@v0.0.1) **(versi tag temen temen)**

Setelah membuat tag teman-teman tinggal push seperti biasa saja.

### 3.1.6 VERSI 2 MENGGUNAKAN GOCROOT BACK-END SEBAGAI FUNCTION DAN STRUCT

Untuk mempermudah teman-teman bisa menggunakan versi yang ke dua, yaitu teman-teman bisa meng fork pada github ini dan teman-teman jadikan repo ke repo masing-masing [gocroot/kampus: modul kampus (github.com)](https://github.com/gocroot/kampus). lalu masukan struct didalam folder **model** dan function di dalam **module**.

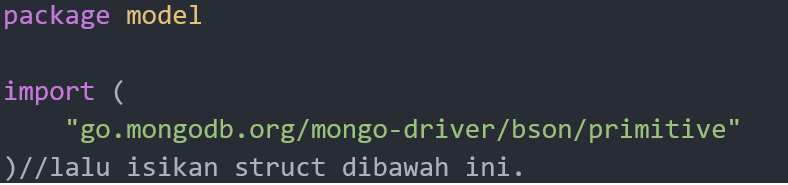
Lalu teman-teman samakan fungctionnya dengan code dibawah ini atau contoh dalam folder module contohnya :



Gambar 3. Function Module

Nah, Untuk package juga harus diubah sesuai dengan nama **Folder** di folder module tempat function dinamakan **package Module dan sedangkan** untuk **struct** dinamakan Folder model dengan begitu menggunakan **package Model,** Selanjutnya kamu push saja ke repo kamu.

Contoh codenya :



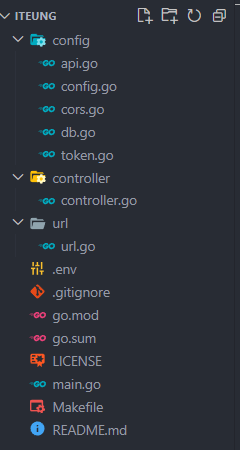
Gambar 3. Struct Package Model

### 3.2 PENGGUNAAN LIBRARY ITEUNG

Oke kita masuk ke penggunaan library iteung, selamat sudah mencapai tahap Backend ini kepada teman-teman, selanjutnya memahami structure folder dari Iteung:

### 3.2.1 STRUKTUR FOLDER ITEUNG

Oke kita masuk ke tahap struktur folder, disini kita bisa lihat pada gambar dibawah ini untuk struktur foldernya :



Gambar 3. Struktur Folder Iteung

Untuk mendapatkan Library ini kalian bisa mengecek ke repo saya atau pun langsung ke repo dari aiteung bisa di cek link dibawah ini:

* repo aiteung : [aiteung/iteung: boilerplate for using aiteung package (github.com)](https://github.com/aiteung/iteung)
* repo iteung : [harisriyoni3/iteung: boilerplate for using aiteung package (github.com)](https://github.com/harisriyoni3/iteung)

Penjelasan Folder :

1. Config

Folder Config berfungsi untuk mengatur config dalam aplikasi backend dan juga mengatur Cors, database, token, api.

1. Controller

Folder Controller berfungsi untuk membuat Controller atau disini kita akan memanggil function yang telah kita buat dalam module gocroot untuk dijadikan atau dipanggil dalam file url.

1. Url

Folder Url berisikan file Url disini akan membuat dan memanggil controller untuk dijadikan API untuk Backend kita.

Setelah itu, Teman-teman boleh tinggal copas dengan repo yang telah diberikan pada link diatas lalu temen-temen ubah secara manual dan uabh dependensinya lagi dengan go mod init github/usernamegithub/repoiteung **(repo iteung yang temen-temen fork atau yang sudah teman-teman ambil dari link diatas)** dan akhiri dengan **go mod init**.

Setelah teman-teman merapikan dependensi, teman-teman jangan khawatir dengan error yang ada, misalnya temen-temen error dengan pengimportan.



Gambar 3. Import di Controller

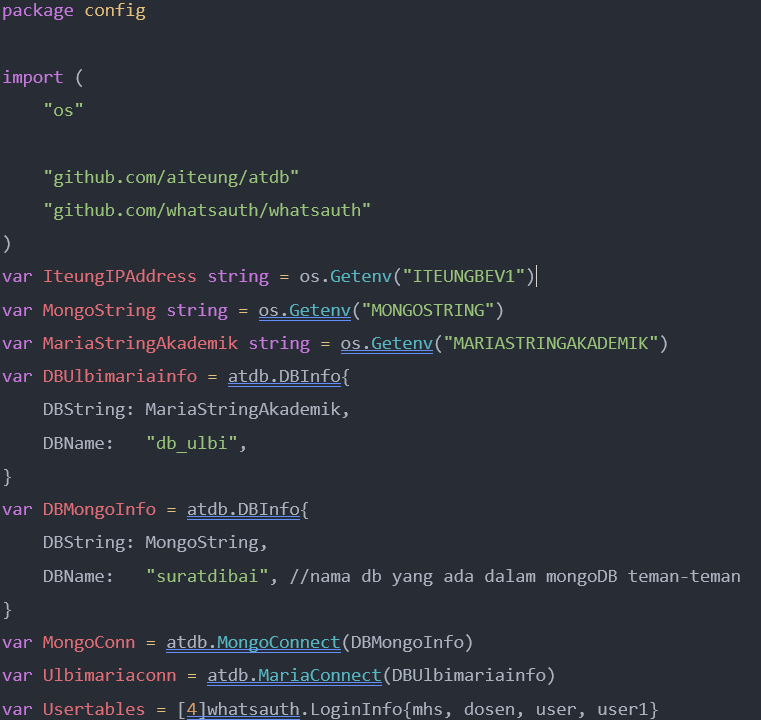
Misalnya, Teman-teman erorr dalam mengimport package model atau module dengan repo teman-teman, kalian bisa menggunakan **go get github.com/gocroot/kampus/model** atau **github.com/gocroot/kampus/module**

Lalu, teman-teman juga rapihkan dependensinya dengan melakukan **go mod init**

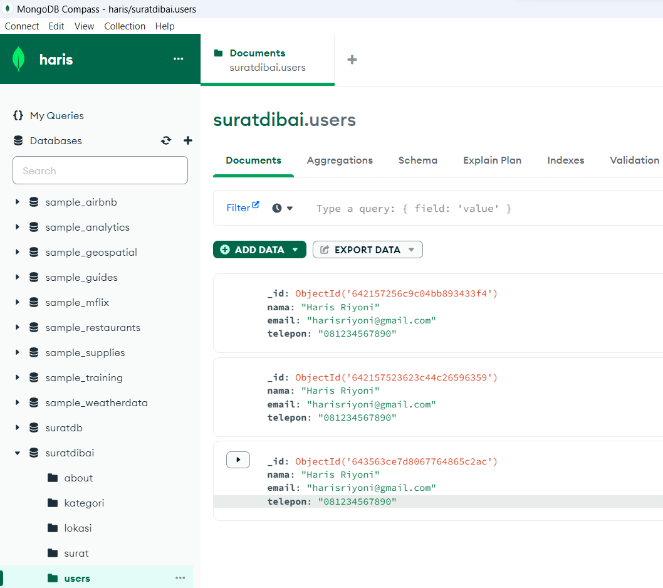
Lalu teman-teman bisa alias untuk memanggilnya saya menmanggil alias dengan **gege** untuk **model** untuk memanggil **struct** dan untuk **module** dengan **gaga** untukmemanggil **function.**

### 3.2.2 SETTING DB

Selanjutnya tahap mensetting DB dengan db teman-teman atau mongoDB, contohnya sebagai berikut:



Gambar 3. Setting Database di db.go

Contoh DB di MongoDB seperti gambar dibawah ini:

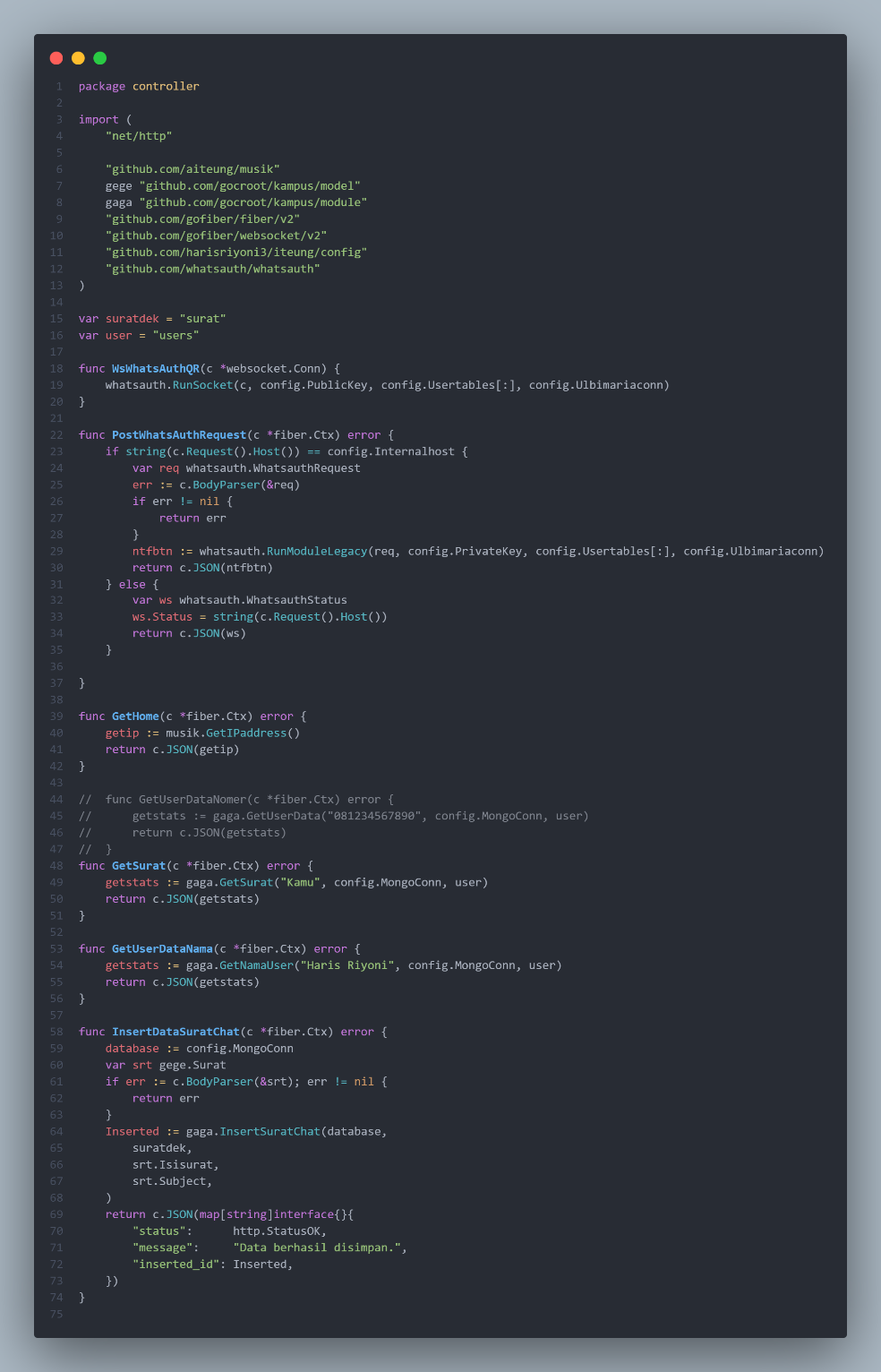
Gambar 3. Tampilan Database

Pada sidebar di kiri, terdapat database-database teman-teman, jika tidak muncul berarti teman-teman belum melakukan running terhadap test code atau testing code yang telah kita buat diatas. Teman-teman running kan dulu testing dan jika mendapatkan status Oke berarti teman-teman telah berhasil menginsert dan juga meng get data dari database jika data tersedia ataupun acuan dari database ada maka akan muncul data yang ingin teman-teman munculkan.

### 3.2.3 PEMBUATAN CONTROLLER

Oke kita masuk ke tahap, Pembuatan Controller yang dimana kita akan membuat jembatan untuk mengambil data atau melakukan CRUD API untuk Database kita, disini adalah tempatnya.

Untuk Contoh Codenya bisa kalian lihat pada gambar di bawah ini:



Gambar 3. Iteung Controller

Nah Code diatas itu cara untuk memanggil function dan struct itu melalui link github yang di repo **usernamekamu/kampus/model** untuk struct dan untuk function dengan memanggil **usernamekamu/kampus/model**.

### 3.2.4 PEMBUATAN URL

Setelah kita membuat Controller kita akan membuat URL untuk dapat di akses oleh kita yaitu dengan membuat URL yang akan memanggil Controller kedalam file ini yang akan dijadikan Sebagai URL nanti.

Contoh Codenya bisa kalian lihat dibawah ini:

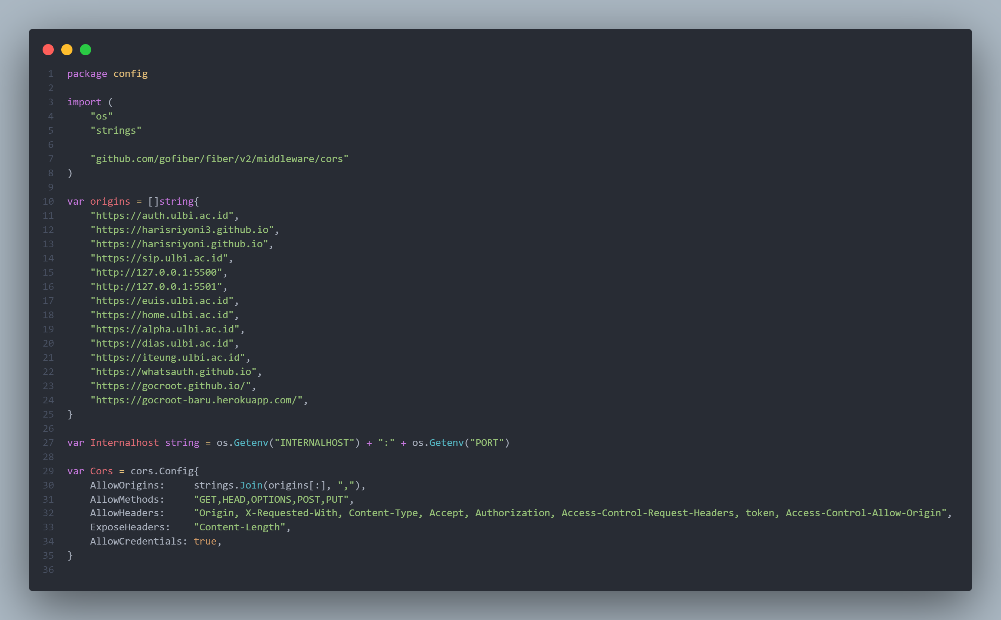


Gambar 3. File Url.go

### 3.2.5 SETTING CORS

Selanjutnya tahap Setting Cors kepanjangan dari cors ini yaitu Cross Origin Resource Sharing, Cors ini merupakan kebijakan untuk keamanan pada Web Browser untuk melindungi jalur lintasan data.

Kita disini kita akan membuat Cors dalam folder config dengan file cors.go, disini kita akan membuat header,key dan web atau aplikasi yang kita ingin ijinkan untuk siapa saja yang dapat melintasi jalur data kita pada intinya seperti itu. Contoh code nya teman-teman bisa mengikuti pada code dibawah ini:



Gambar 3. Setting Cors di Cors.go

Oke pada code diatas terdapat Origins, dan untuk **var Cors** teman-teman mengikuti saja seperti yang terdapat pada code diatas untuk **var origin** teman-teman pertama buat dulu aplikasi front-end dan jangan lupa untuk front-end nya di deploy menggunakan **github pages.**  Contoh atau pembuatan Front-End terdapat pada sub-bab **3.3 Pembuatan Folder, File dan Integrasi Front-End.**

Setelah membuat pages lalu teman-teman cantumkan saja link github.io teman-teman contoh dari saya atau github saya yaitu “[**https://harisriyoni3.github.io**](https://harisriyoni3.github.io)**”.** yang dimana ini untuk perizinan untuk mendapatkan akses jalur lintasan. Jika dirasa sudah teman-teman tinggal mendeploy aplikasi Back-End di Heroku.

### 3.2.6 PUSH GITHUB DAN DEPLOY HEROKU ITEUNG

Sebelum Deploy tentunya kita akan Push terlebih dahulu dan baru kita akan Deploy ke Heroku dengan langkah-langkah sebagai berikut :

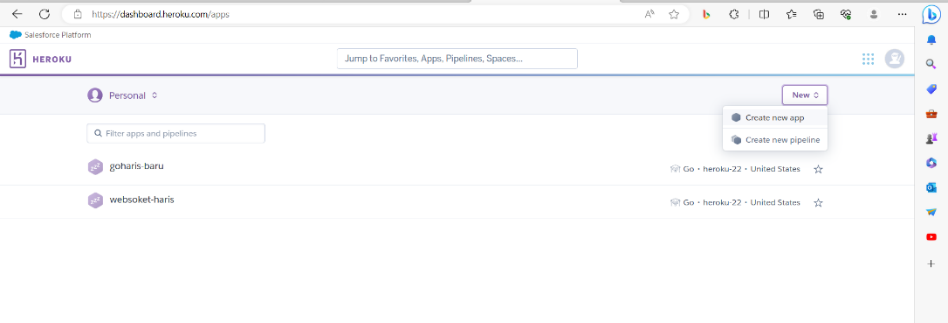
1. Push Github

Push dengan command

* Git add .
* Git status
* Git commit -m “status push/nama push”
* Git push main/master

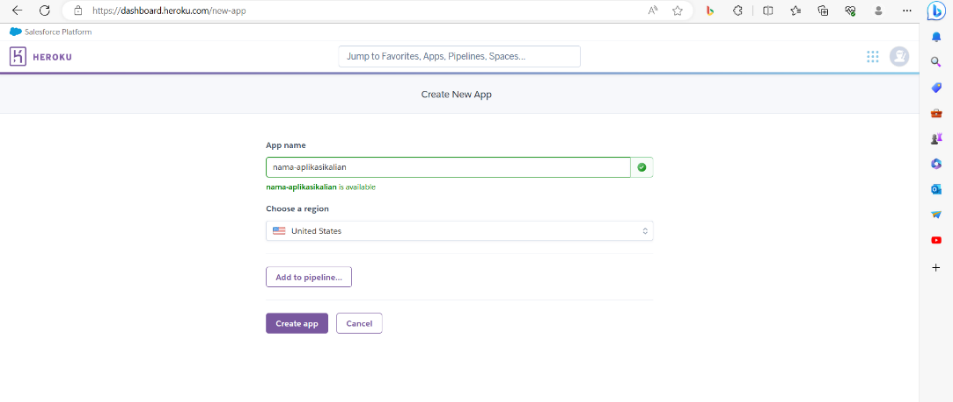
1. Buat App baru Heroku

Pertama, tentu kalian harus mempunyai akunnya dan sudah di satukan dengan debit teman-teman selanjutnya teman-teman pilih create new app seperti pada contoh gambar dibawah ini:



Gambar 3. Deploy App Heroku 1

Lalu create saja.



Gambar 3. Deploy App Heroku 2

1. Add remote Heroku di terminal

Add remote Heroku dengan mengikuti comman disini heroku

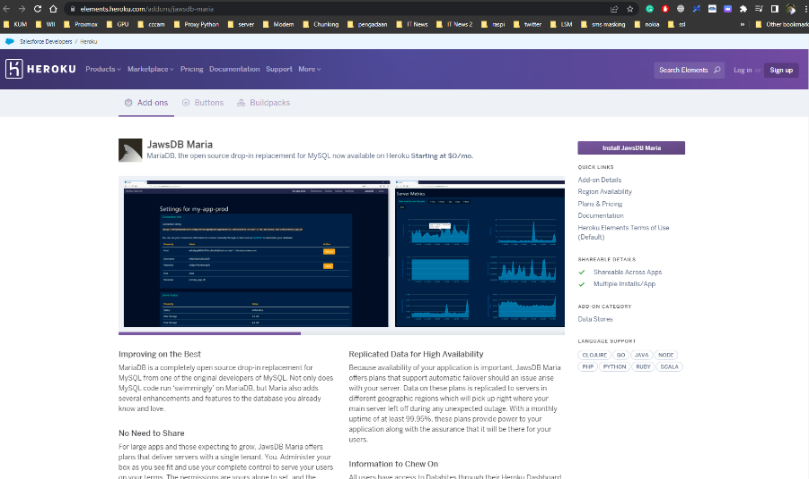
**git:remote -a nama-aplikasikalian**

1. Push Back-End ke Heroku

Jika sudah di add remote herokunya, selanjutnya kalian ketikan command untuk deploy aplikasi herokunya dengan command ini :**git push heroku master/main.**

1. Add on dan Setting ENV heroku Mariadb Jawsdb

Untuk menambahkan adds kalian bisa mengunjungi link ini **https://elements.heroku.com/addons/jawsdb-maria,** contohnya:

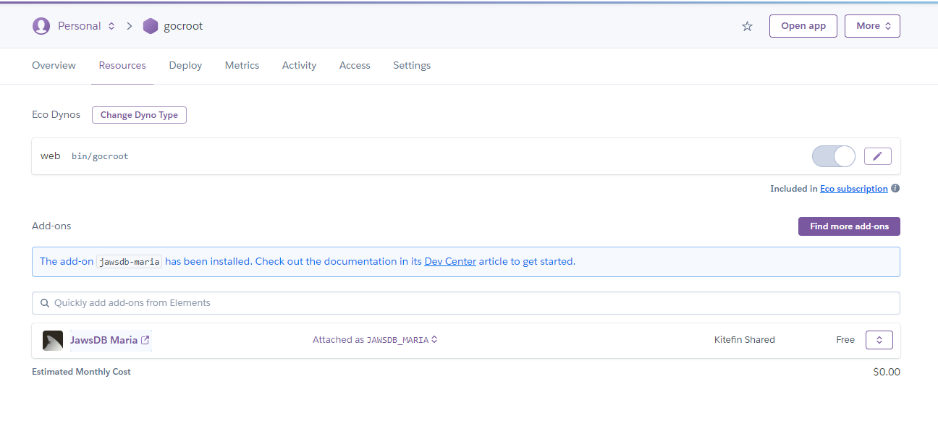


Gambar 3. Add JawsDB Heroku 1

Lalu kalian klik install pada seperti pada gambar diatas, lalu kalian pilih yang free seperti gambar dibawah ini:

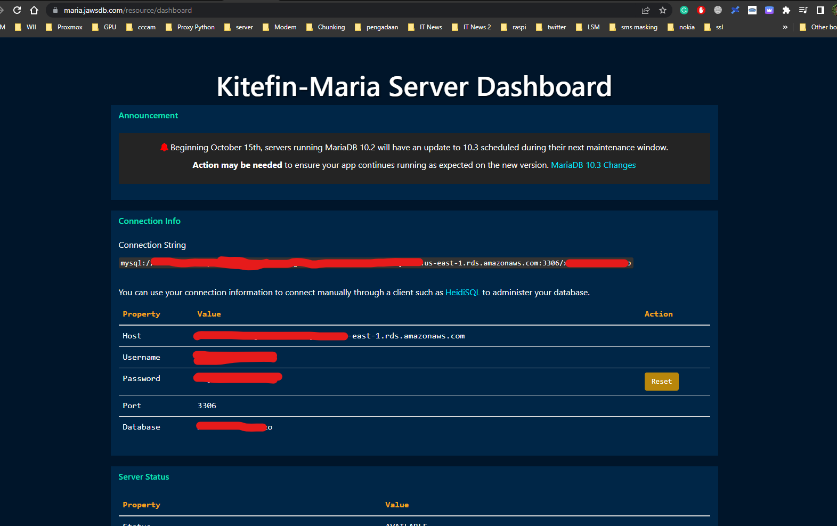


Gambar 3. Add JawsDB Heroku 2

dalam memakainya kita hanya akan mengambil atau hanya butuh Connection String yang bisa di dapatkan dengan klik JawsDB Maria di bagian Add-ons:

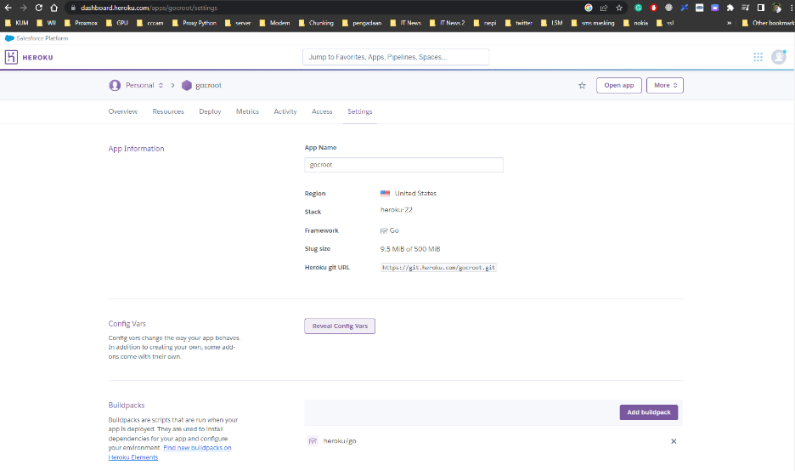
Gambar 3. Tampilan Sudah TerInstall JawsDB

Setelah mengklik akan muncul direct ke halaman seperti dibawah ini:



Gambar 3. Tampilan Jika Membuka JawsDB

Nah, ini adalah key-key yang akan kita gunakan seperti host, username, password dll. Selanjutnya ke tahap Edit atau Setting Env di aplikasi kita di Heroku.



Gambar 3. Tampilan Setting App Heroku

Di menu Setting kalian akan menemukan nama aplikasi kalian dan juga kalian akan menemukan Kata Convig vars, nah dibagian ini yang akan kita ubah, kalian cukup klik Reveal Convig Vars seperti yang terlihat dibawah ini:

Untuk saat ini kita tambahkan MARIASTRINGAKADEMIK dan MONGOSTRING terlebih dahulu contoh:

* **“MARIASTRINGAKADEMIK=**username**:**password**@tcp(ao9moanwus0rjiex.cbetxkdyhwsb.us-east-1.rds.amazonaws.com:3306)/”**
* **“MONGOSTRING=mongodb+srv://**username**:**password**@cluster0.wghp85v.mongodb.net/test”**

Seperti yang terlihat pada gambar dibawah ini:



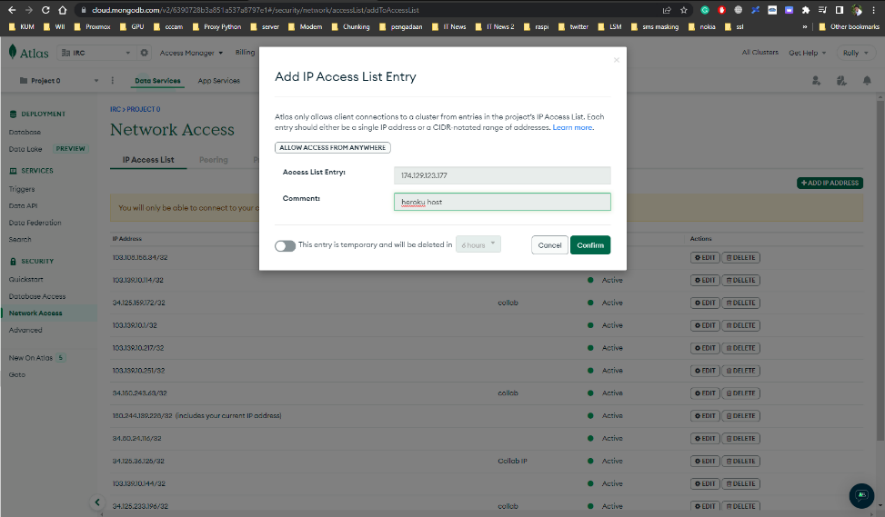
Gambar 3. Setting Config Vars

Jika terdapat error seperti diatas maka modif Connection string menjadi seperti berikut:

**“MARIASTRINGAKADEMIK=**username**:**password**@tcp(ao9moanwus0rjiex.cbetxkdyhwsb.us-east-1.rds.amazonaws.com:3306)/”**

1. Setting Network Acces MongoDB Atlas Website

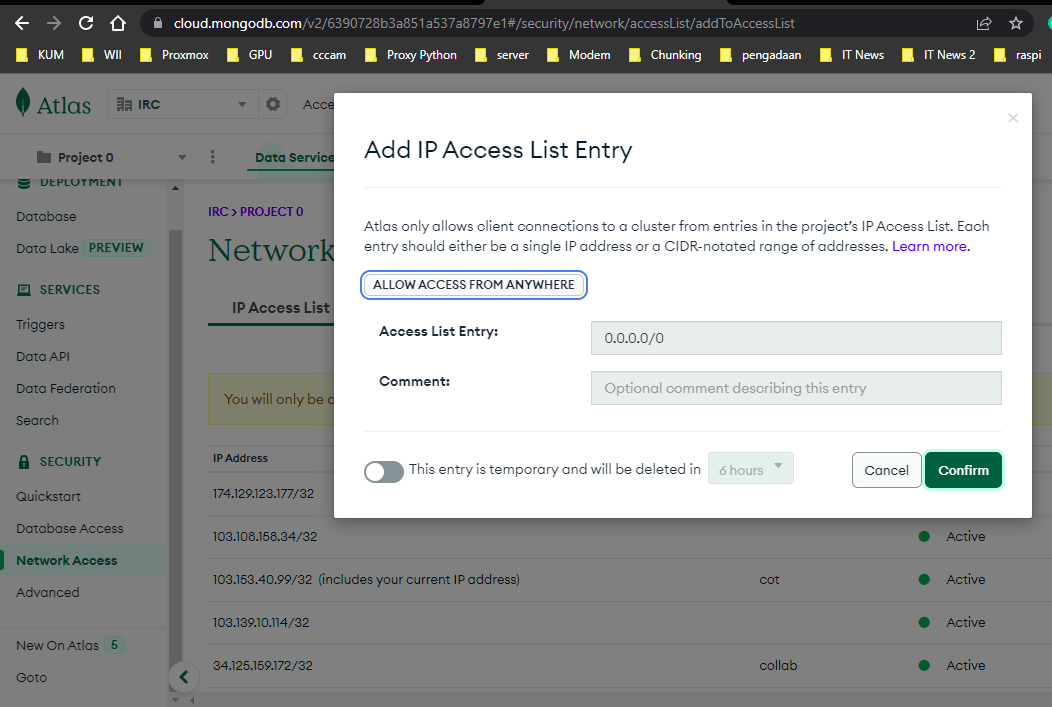
Untuk memungkinkan aplikasi di Heroku mengakses MongoDB yang sudah dibuat di MongoDB.com, Kamu perlu menyediakan alamat IP host Heroku yang diperoleh saat membuat controller homepage ke MongoDB.com. dapat terlihat pada gamabr dibawah ini:



Gambar 3. Add IP Heroku Akses Database MongoDB

Setelah itu, silahkan masukkan data dummy pada collection dengan mongo compass yang diakses oleh package yang dikembangkan sesuai yang diinginkan.

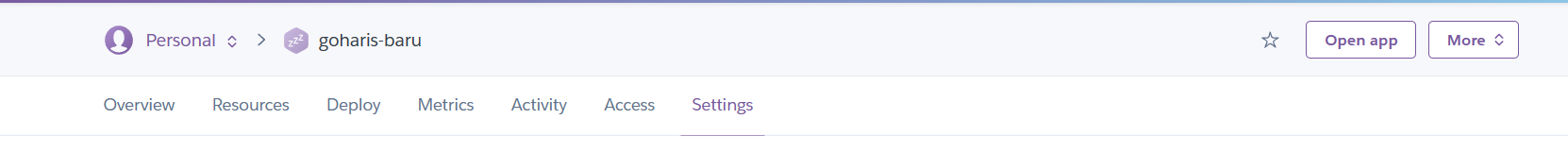
Jika masih belum, kamu klik Allow Acces From Anywhere dan isikan 0.0.0.0/0 lalu confirm seperti yang terlihat pada gambar dibawah ini:



Gambar 3. Allow Akses MongoDB

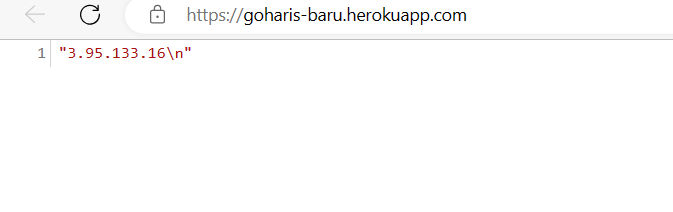
Untuk Hasil Untuk Setiap URL kalian bisa cek pada BAB IV atau Terakhir untuk Hasil Implementasi yang telah kita buat.

Setelah itu kalian bisa lihat aplikasi kalian dengan Open App pada Heroku kalian bisa lihat pada gambar dibawah ini:



Gambar 3. Tampilan Aplikasi dengan OpenApp

Jika sudah mendapatkan linknya kalian bisa melihat aplikasi yang kalian telah buat, contoh:



Gambar 3. Tampilan Aplikasi

Nah kalian tinggal kasih “/” setelah .com missal dikasus saya .com/surat ini tergantung kalian dalam pembuatan di file URL. Untuk melihat hasil kalian bisa lihat di BAB terakhir Ya.

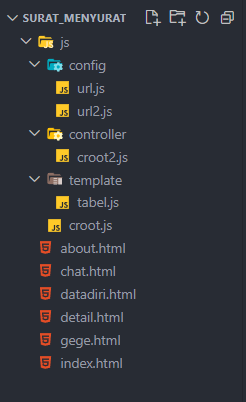
### 3.3 PEMBUATAN FOLDER, FILE DAN INTEGRASI FRONTEND

Oke kita, Masuk kedalam Tahap Frontend, Kalian diharuskan mencari atau membuat Frontend yang cantik atau yang sederhana juga tidak apa-apa, untuk mempermudah kalian dapat mencari digoogle template Frontend yang cocok dengan studi kasus kita.

Kalian dapat mengetikan di Google dengan Keyword Template Free Front-End Tailwind CSS Free. Setelah mendapatkan kalian dapat memodifikasinya. Ataupun kalian dapat mengikuti Front-End dalam github Saya atau Buku kalian bisa dapatkan dalam link ini [app/administrasi/Surat\_Menyurat at main · harisriyoni3/app · GitHub](https://github.com/harisriyoni3/app/tree/main/administrasi/Surat_Menyurat).

### 3.3.1 STRUKTUR FOLDER FRONT-END JAVASCRIPT

Oke kita masuk kedalam Struktur Folder dari Js dan file Html kita, Seperti pada gambar dibawah ini:



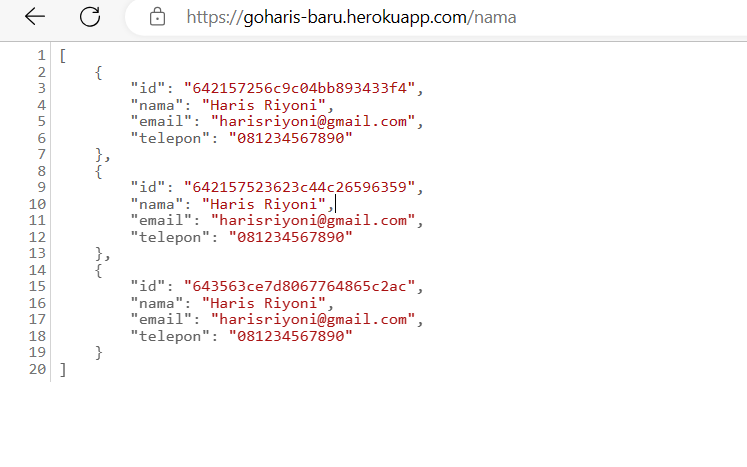
Gambar 3. Struktur File Javascript

Bisa kita lihat pada gambar diatas berisi mulai dari File HTMl juga berisi Folder JS yang akan kita gunakan untuk mengambil data dari API juga digunakan untuk POST API.

* Mulai Dari Folder Config ini berisi URL-URL yang telah kita buat mulai dari API Untuk GET data dari database yang sudah menjadi API untuk Aplikasi Back-End kita contohnya **urlapp.com/nama** dan telah membuat ALL Method dalam aplikasi kita pada URL **/surat/insert**.
* Setelah itu ada Controller yang akan dipanggil untuk sebagai module di html untuk media post kita ke database melalui aplikasi front-end kita.
* Folder Template ini berisi data API yang akan kita simpan untuk html table kita yang dimana akan di impot dalam croot.js
* File Croot.js ini yang akan dipanggil dalam HTMl kita yang dimana berisi method untuk pengambilan Get data API Back-End kita.

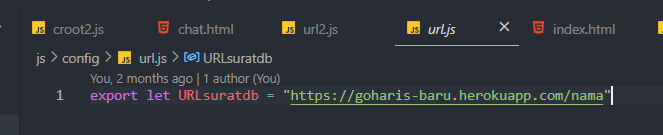
### 3.3.2 MEMBUAT GET DATA API DAN HTML

Pertama, Lakukan Setting pada File Url.js untuk menaruh link Get data kita, Contohnya:



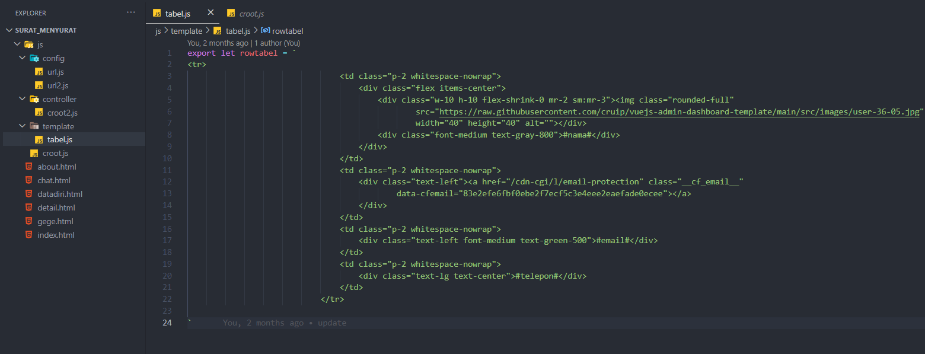
Gambar 3. Get Data dari /nama

Nah URL [goharis-baru.herokuapp.com/nama](https://goharis-baru.herokuapp.com/nama) kita taruh di url.js seperti gambar dibawah ini:



Gambar 3. Tampilan Url.js

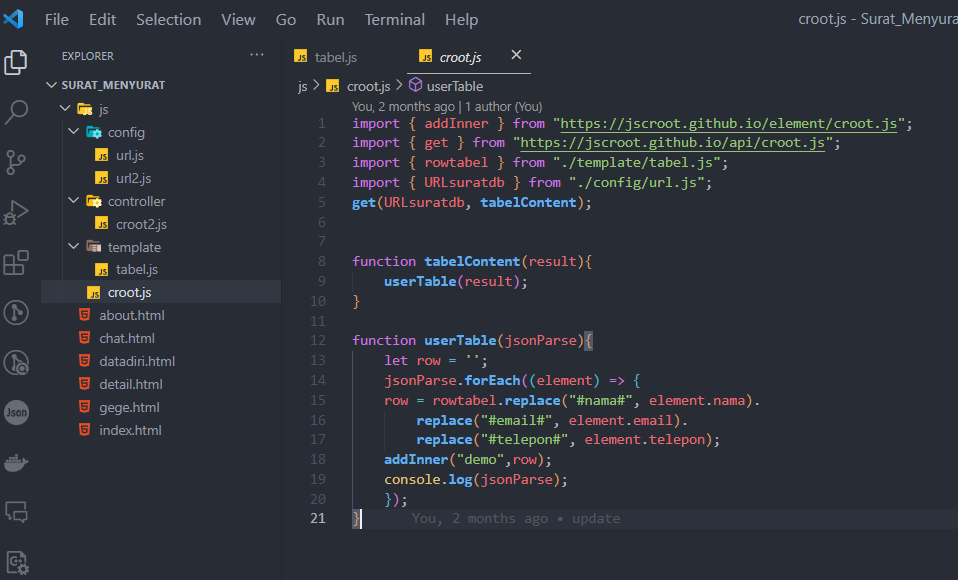
* Kedua, kita siapkan Template untuk tempat menyimpan atau menampilkan data API kita di table dengan sebagai berikut:



Gambar 3. Template dan Isi untuk Data API

Seperti yang terlihat pada code diatas yaitu terdapat #nama# untuk penampungan data nama dari user, #email# untuk penampungan email dari user dan #telepon# untuk sebagai menampilkan data telefon user dari database. Lalu file ini akan di import di croot.js.

* Ketiga, Membuat Croot.js



Gambar 3. Tampilan Code Croot.js

Seperti terlihat pada code diatas sama kan saja kalau teman-teman telah mengikuti dan sama untuk struct dan URL nya. Jika teman-teman ada yang diubah sebelumnya teman-teman cukup ubah sesuai dengan struct teman-teman buat, dengan mengubah element.namafield yang ada di API URL teman-teman. Dengan Addinner Demo dimana teman-teman tinggal tambahkan id=”demo” pada table html.

* Keempat, dalam html kalian panggil jsnya dengan cara masukan dalam head atau setelah </body> dengan command **<script type="module" src="js/croot.js"></script>** dengan begitu js teman-teman sudah terpanggil dan sudah disiap digunakan, untuk id demo teman teman tambahkan pada tbody dari table teman-teman contoh seperti pada gambar di bawah ini:

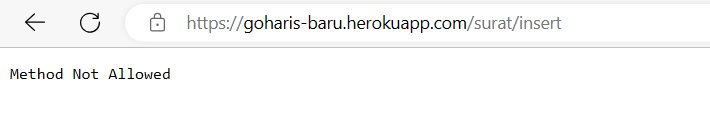


Gambar 3. Table HTML

Jika terdapat Erorr jangan khawatir teman-teman coba solve dan buka StackOverflow dan meriview buku ini lagi jika terdapat erorr juga mengecek erorr dalam javascript menggunakan console dalam web browser pada saat html dibuka untuk hasil bisa dilihat pada BAB terakhir.

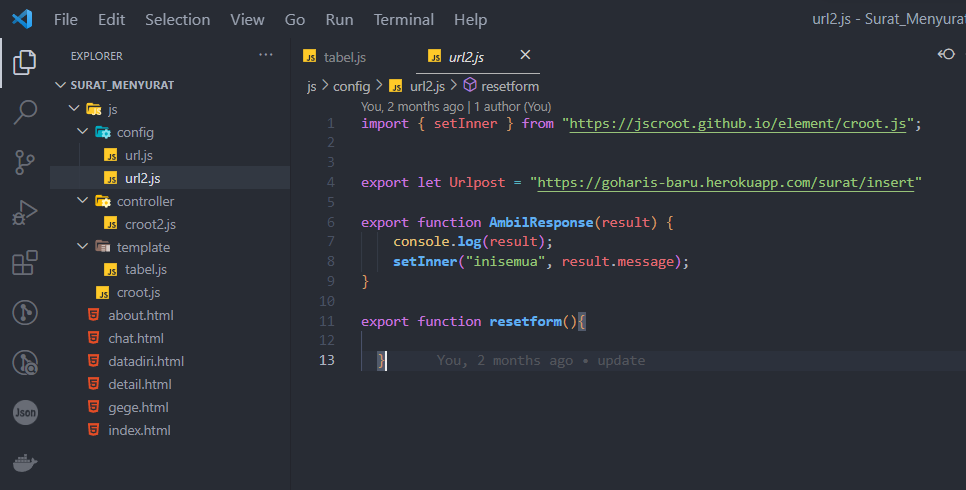
### 3.3.3 MEMBUAT POST DATA API DAN HTML

* Pertama, Setting Url2.js dan ubah sesuai dengan link aplikasi teman-teman seperti contoh link [goharis-baru.herokuapp.com/surat/insert](https://goharis-baru.herokuapp.com/surat/insert) adalah contoh aplikasi saya, tampilannya seperti ini:



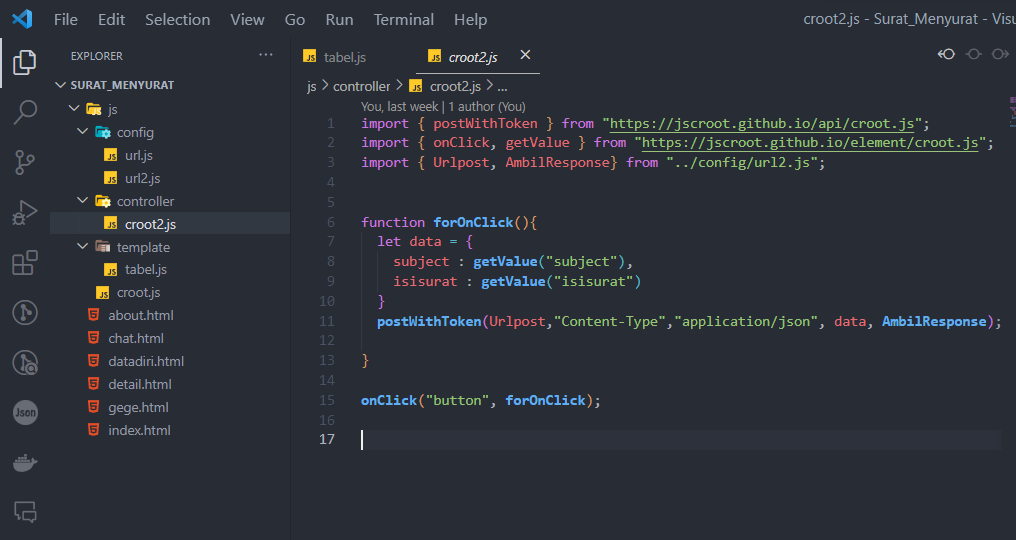
Gambar 3. Tampilan /Surat/Insert

* Dan Setting pada Code nya



Gambar 3. Tampilan Url2.js

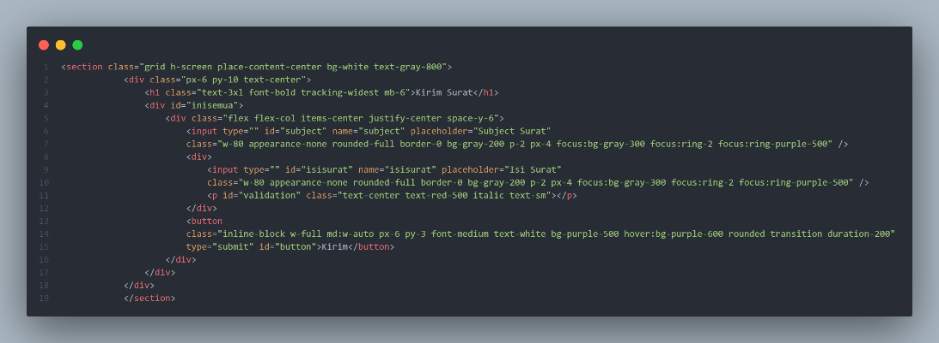
Pada Setinner ini akan menginisialisasi form yang ber id=”inisemua”

* Selanjutnya, Croot2.js

Gambar 3. Tampilan Croot2.js

Pada Function ForOnclick ini akan mendapatkan value dari form yang telah kita isi misalkan disini subject dan isisurat yang dimana form tersebut dengan id subect dan id isisurat.

Dan jangan lupa memberikan token yang kita sudah setting dan beri akses pada cors yang telah kita ubah ataupun tambahkan di Bac-End.

* Selanjutnya, jangan lupa untuk Mengambil Js nya dengan command:**<script type="module" src="js/controller/croot2.js"></script>**
* Dan juga setting untuk form kita seperti contoh pada gambar form dibawah ini:

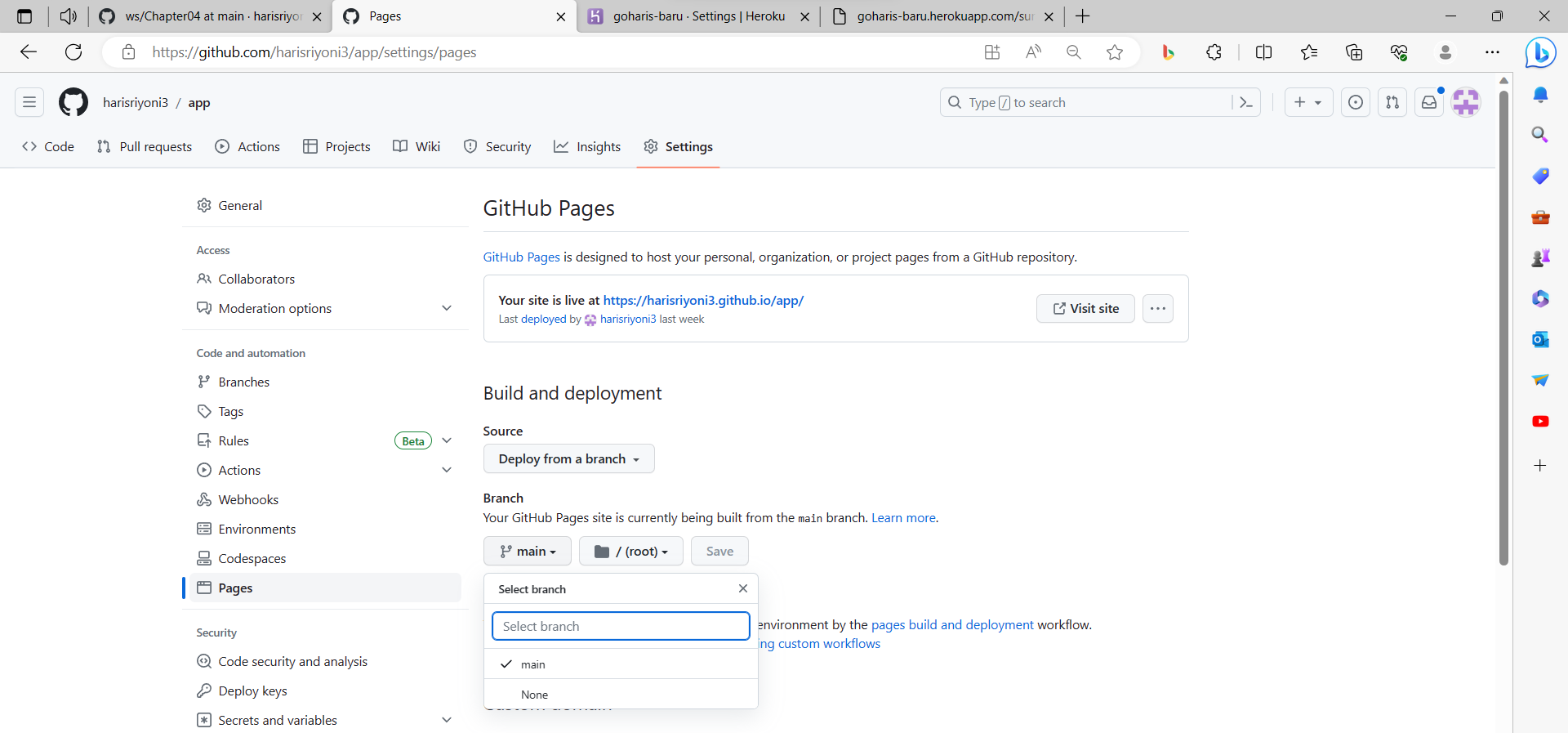
Gambar 3. Form HTML

Jangan lupa juga untuk memberikan id= button pada button di form kita, karena dalam js kita terdapat Onclick pada ber id = button.

### 3.3.4 DEPLOY FRONT-END DENGAN GITHUB PAGES

Oke kita masuk ke tahap paling terakhir yaitu deployment untuk front-end kita dalam pembuatan aplikasi surat menyurat, Oke langsung saja kalian menuju github ke repo kalian pastikan semua dari repo kalian sudah ter-update atau sudah diperbarui.

Kita langsung saja menuju repo teman-teman, bisa terlihat pada gambar dibawah ini:

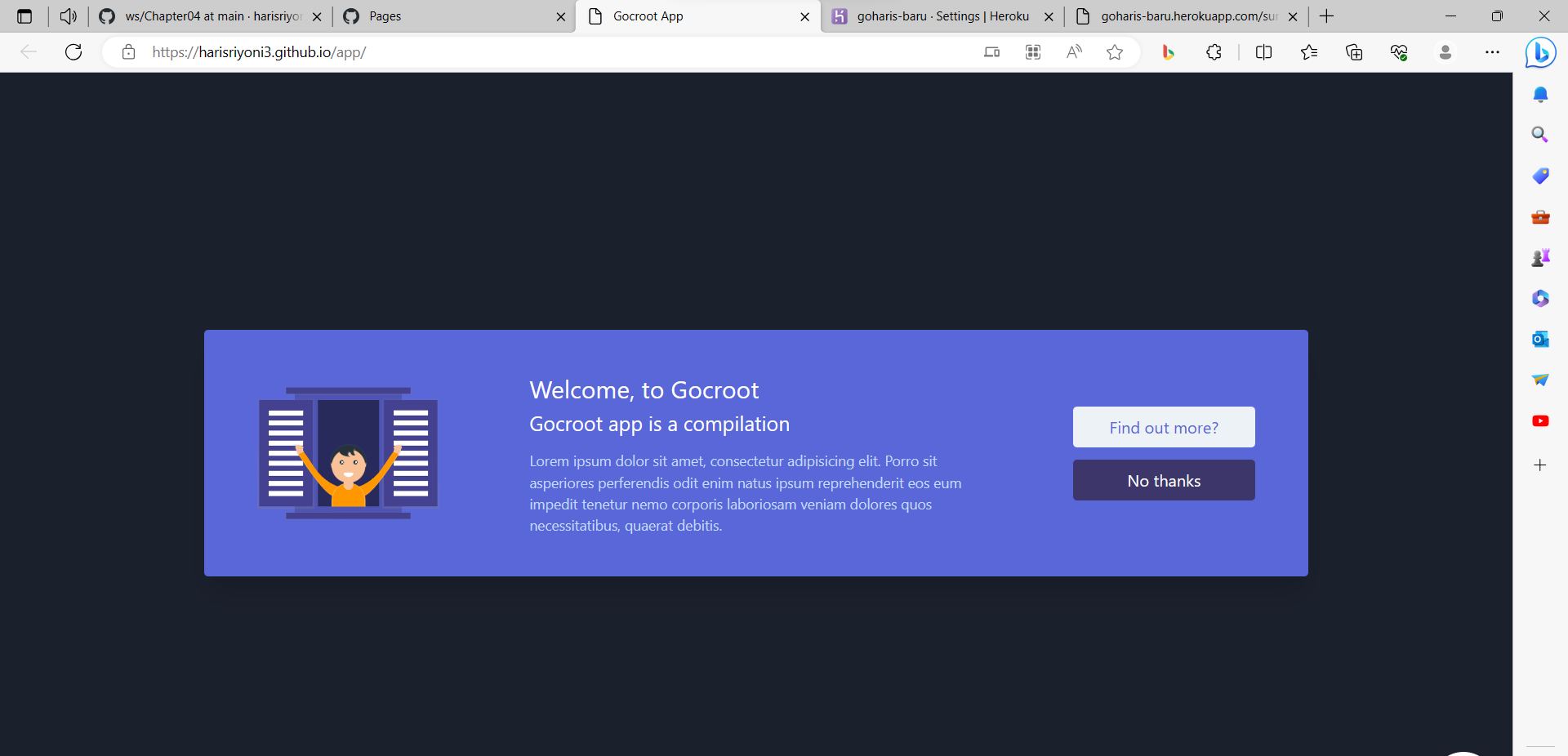


Gambar 3. Deploy Frontend Github Pages

Kalian, tinggal menuju ke repo terus jika sudah berada pada repo teman-teman, Selanjutnya pada bagian navbar terdapat Setting dan pilih pada side bar kiri yaitu Pages dan akan tertampil seperti gambar diatas.

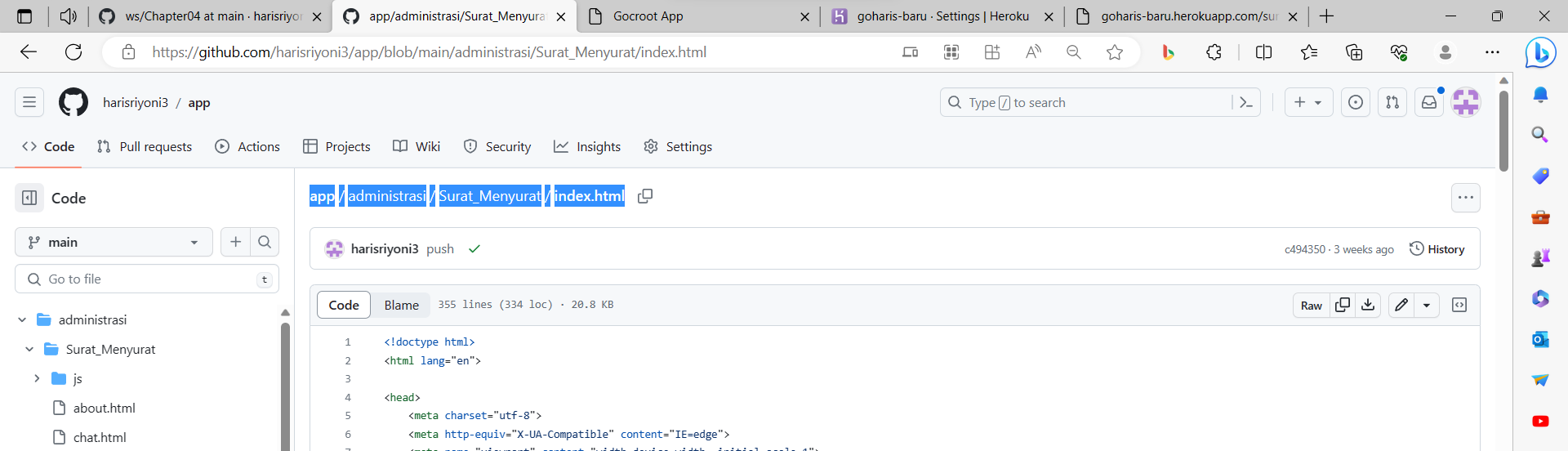
Selanjutnya dapat dilihat pada gambar, Build and Deployment untuk Source kita pilih Deploy from a branch, untuk branch ubah menjadi main atau master lalu pilih save. Lalu jika sudah memilih save, tunggu beberapa saat untuk Deployment Frontend mungkin membutuhkan 1-5 menit untuk muncul Visit Site.

Jika dirasa sudah akan muncul seperti ini:



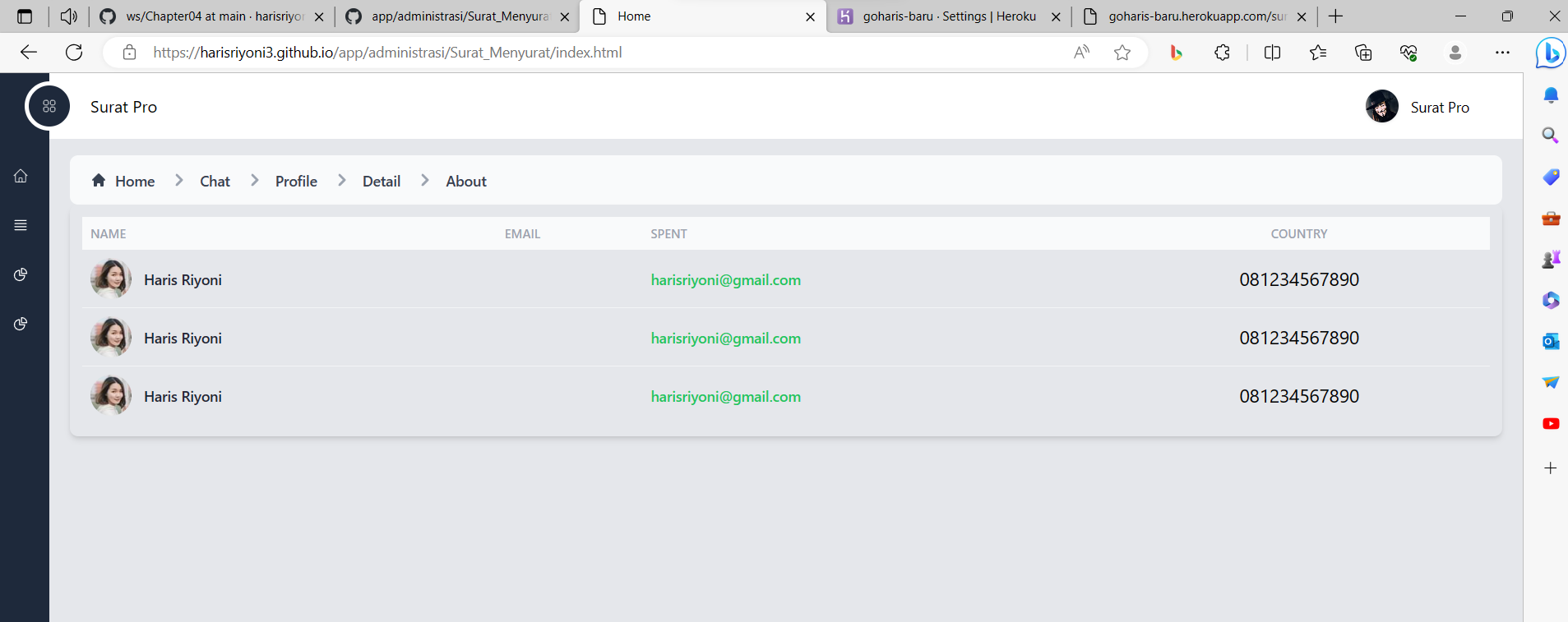
Gambar 3. Tampilan Aplikasi Front-End

Lalu tinggal direksikan ke file .html teman-teman contohnya:



Gambar 3. Struktur File Repo ke Url

Seperti yang terlihat, teman-teman cukup ganti linknya menjadi link yang terdireksi ke file index.html teman-teman. Dalam kasus saya terdapat saya akan menggunakan app administrasi dan surat\_menyurat dan hasil link nya sebagai berikut: [Home (harisriyoni3.github.io)](https://harisriyoni3.github.io/app/administrasi/Surat_Menyurat/index.html).



Gambar 3. Tampilan Index.html Github Pages

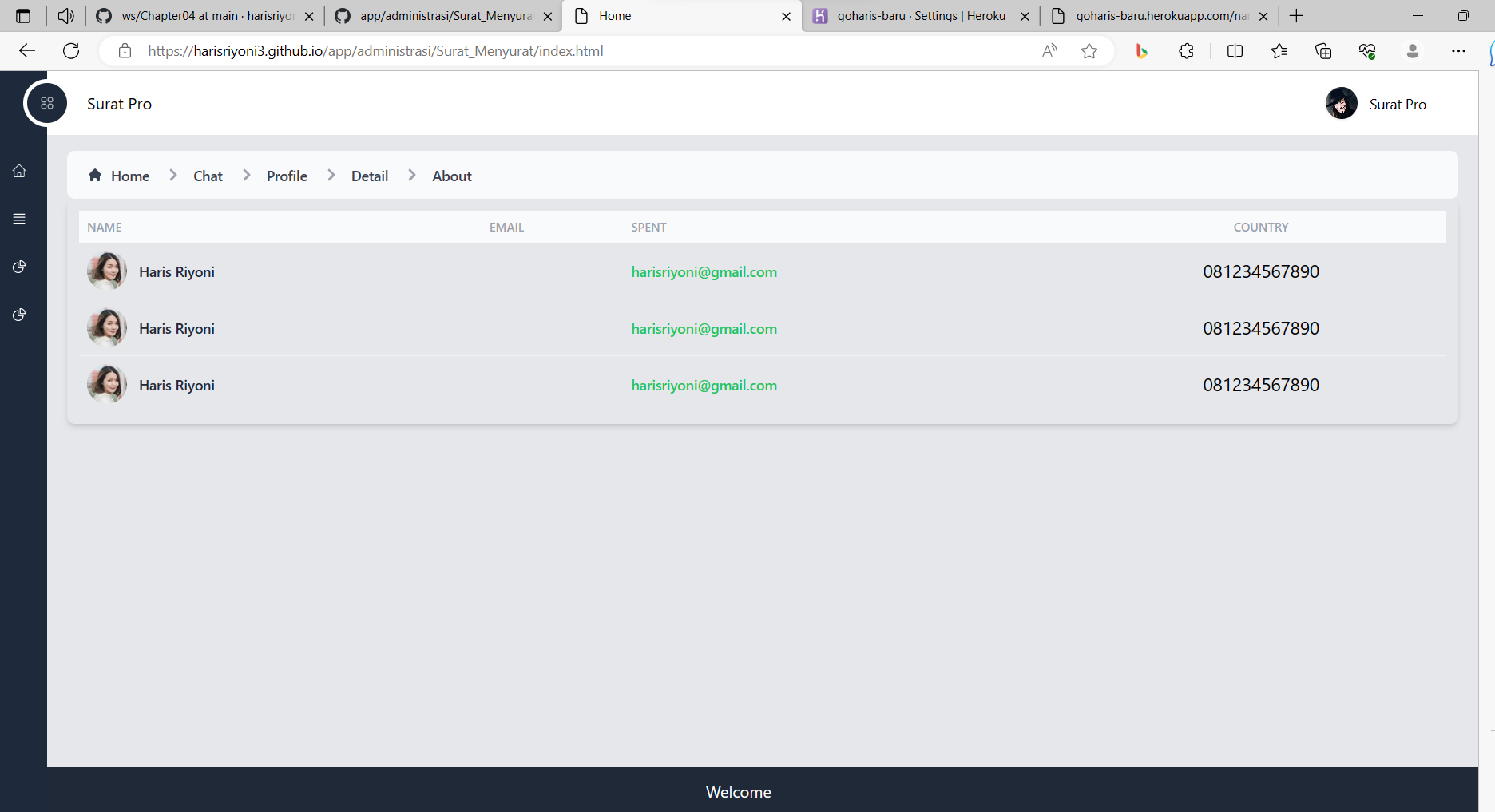
# BAB IV

# Hasil Akhir Tujuan Tampilan dan Debug

### 4.1 HASIL TAMPILAN FRONTEND

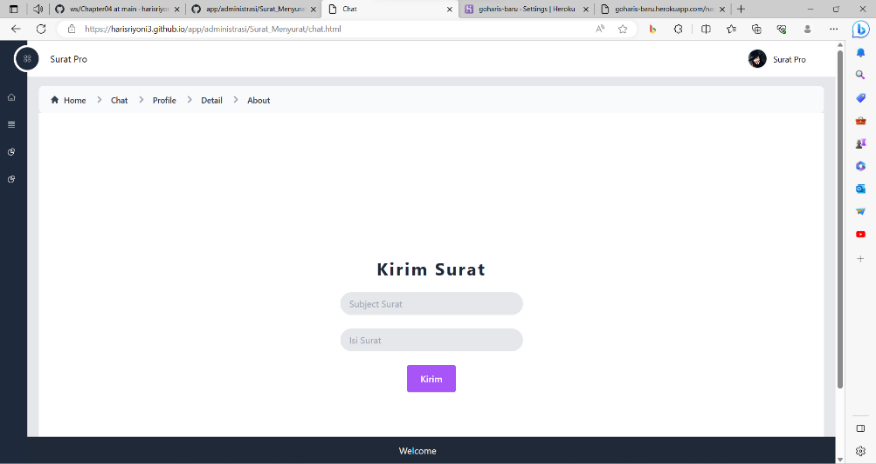
Pada Bab ini kita akan melihat hasil dari Frontend kita mulai dari Index.html dan Chat.html

* Untuk index sebagai berikut:



Gambar 4. Tampilan Index Get API GitPages

* Untuk Chat.html sebagai berikut:

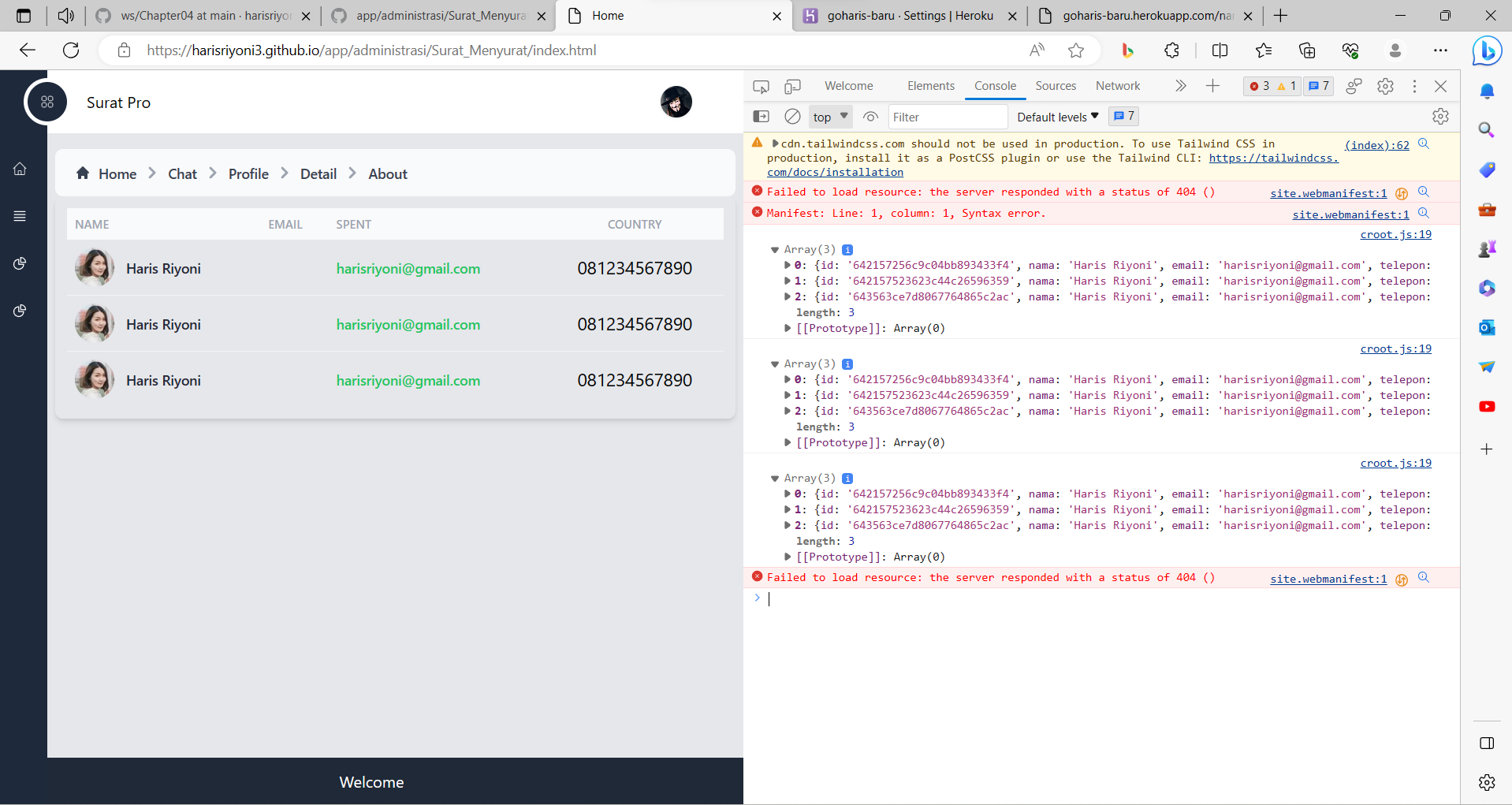


Gambar 4. Tampilan Form Chat

### 4.2 DEBUG HASIL FRONTEND

Kita masuk ke bagian debug console untuk index.html Muncul dalam Console data-data yang telah di get oleh Js kita sehingga muncul di Console juga muncul di index.html.

* Untuk Index.html:



Gambar 4. Console Get API

* Untuk Chat.html

Kita masuk ke bagian debug console untuk chat.html Muncul dalam Console data-data yang telah di Post oleh Js kita sehingga muncul di Database juga muncul dalam MongoDB teman-teman juga dapat memunculkan data-data real yang baru dipost kedalam API dan ini masuk kedalam Get data API ini untuk latihan teman-teman untuk memunculkan data yang telah dipost.

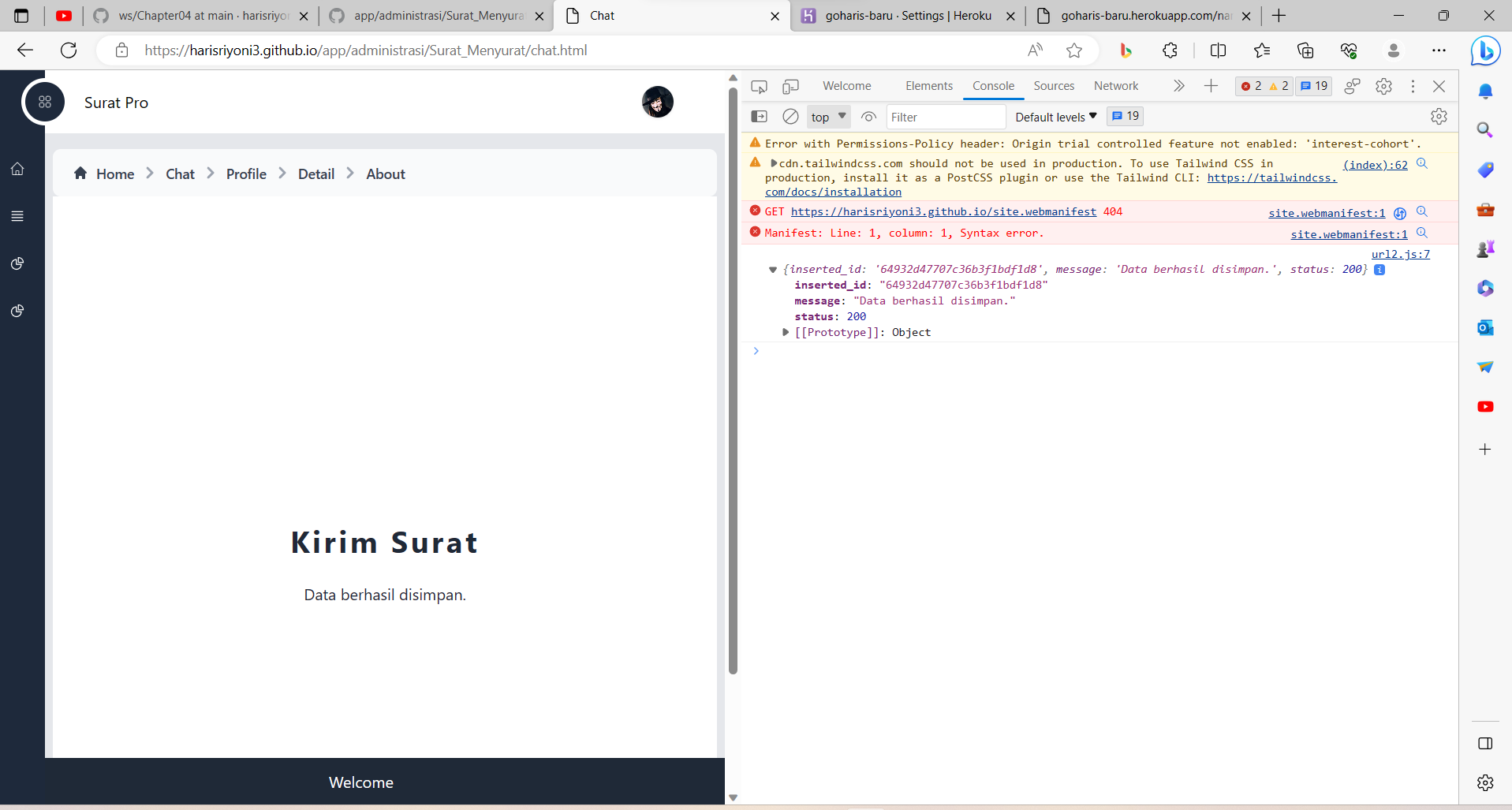
* Memasukan Field:



Gambar 4. Masukan Data Form Chat

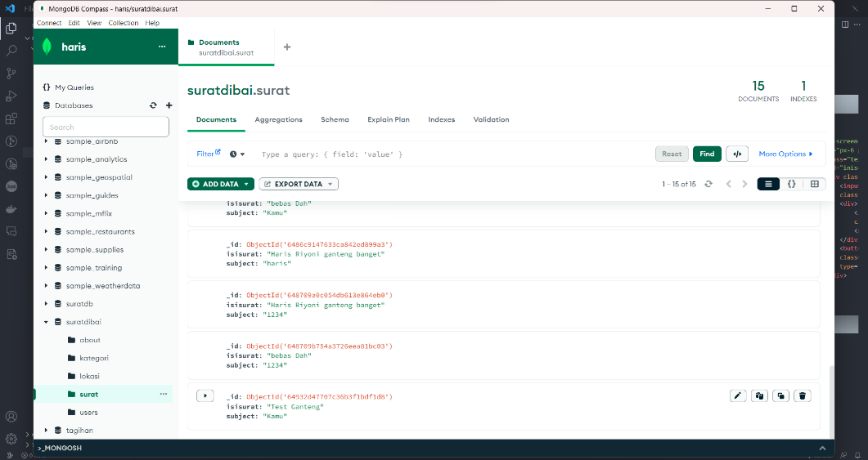
* Send Post

Terlihat di Console data Inserted ini berarti data berhasil di Post atau dikirim ke Database, kita akan cek didatabase:



Gambar 4. Form Chat Terkirim

Cek Database, dan terlihat pada gamabr dibawah ini datanya muncul “Test Ganteng” dan “Kamu”.



Gambar 4. Tampilan Hasil Send Post Form Chat

Selamat untuk teman-teman yang telah mengikuti buku ini sampai akhir dan saya ucapkan terimakasih banyak, tetap semangat dalam mempelajari dunia programmer dan sedikit Quote “Coding terasa sulit karna tak pernah dicicil karena jika dicicil pasti akan selesai sesulit apapun sulitnya Erorr”.

# DAFTAR PUSTAKA

Chabbi & Ramanathan, (2022) A study of real-world data races in Golang.

Kristanto et al., (2020) Implementasi Golang dan New Simple Queue pada Sistem Sandbox Pihak Ketiga Berbasis REST API.

Susanto, (2021) Riset Pengembangan API Generator Menggunakan Golang.

Kristanto, Harjoseputro, Samodra, et al., (2020) Golang and New Simple Queue Implementation on Third Party Sandbox System Based on REST API.

Ramadhanu & Priandika, (2021) Rancang Bangun Web Service Api Aplikasi Sentralisasi Produk Umkm Pada Uptd Plut Kumkm Provinsi Lampung.

Ardiansyah & Fatwanto, (2022) Comparison of Memory usage between REST API in Javascript and Golang.

Nascimento et al., (2021) JavaScript API Deprecation Landscape: A Survey and Mining Study.

Guthals, (2023) GitHub For Dummies.

Irawan, (2021) Implementasi Arsitektur Microservice Untuk Input Nilai Praktikum Mahasiswa Stmik Akakom Yogyakarta Menggunakan Restful Api.

Cahyono & others, (2022) Pengembangan Backend Pada Startup Sajiloka.

Sari & Hidayat, (2022) Designing website vaccine booking system using golang programming language and framework react JS.

Tsoukalos, (2019) Mastering Go: Create Golang production applications using network libraries, concurrency, machine learning, and advanced data structures.

# GLOSARIUM

A

**Table**: Bisa dari struct entity atau bisa dari database.

**Attribute**: Kolom yang berada dalam suatu table atau entity dari struct atau database.

B

**Button**: Tombol.

**Backend**: Bagian belankang dari aplikasi seperti code interaksi dengan database dll.

C

**Cloud**: Suatu Penyampaian, penyimpanan dan lain lain melalui layanan melalui internet.

**Commit**: Yang mendekteksi perubahan yg terjadi.

**Cache**: Proses penyimpanan sementara data.

**Command**: Sebuah perintah.

**Code editor**: Tempat menyusun, mengedit, menghapus code.

D

**Deploy:** Sama halnya seperti hosting.

**Drag**: Seret.

**Drop:** Lepas.

**Database**: Kumpulan data yang dikelola berdasarkan ketentuan tertentu, saling berhubungan sehingga mudah dalam pengelolaannya.

**Debug**: Mengetest atau melihat hasil dari aplikasi berhasil atau ada kendala dalam aplikasi.

**Dependensi**: Merapikan Struktur code project file dan folder dalam kasus ini kita untuk merapikan file dan folder dalam Bahasa Golang.

E

**Error**: Code atau Dependensi Bermasalah.

**ENV**: Environment.

F

**Frontend**: Bagian depan dari aplikasi seperti tampilan aplikasi.

**Function**: Fungsi-fungsi code yang telah dibuat.

**Field**: Attribute atau kolom.

G

**Get**: Mendapatkan.

H

**Hosting**: Mempublikasikan web atau aplikasi ke server internet atau jaringan online.

I

**Integrasi**: Menyatukan.

**Import**: Menggunakan, bisa seperti menggunakan library yang telah disediakan oleh bahasa tersebut atau dapat dari orang lain.

J

K

 L

**Library**: Sebuah kerangka code yang bisa kita manfaatkan untuk aplikasi kita.

M

**Merge**: Menggabungkan.

**Master:** Sebuah cabang di Git.

**Main:** Sebuah cabang di Git.

N

O

**Open source:** Suatu software atau kode pemrograman komputer yang dipublikasikan secara umum pada orang-orang.

P

**Push:** Mendorong atau mengirim sebuah project ke suatu repository.

**Programmer**: Sebuah profesi.

**Pull request:** Suatu permintaan untuk menggabungkan (merge) kode yang kita modifikasi dengan repositori utama atau repositori lain.

**Post**: Mengunggah.

Q

R

**Repository**: Tempat penyimpanan di Github.

S

**Source code**: Merupakan deretan kode.

**Skill:** Kemampuan.

**Sintak**: penulisan codenya.

**Solve**: Memperbaiki Code yang bermasalah.

T

U

**Upload**: Mengunggah.

V

**Verifikasi**: Menyetujui.

W

X

Y

Z

# INDEKS

**A**

Apa Itu Go? ,1

Apa Itu JS? ,2

Akun Gihtub, 4

Akun Heroku, 5

Akun dan MongoDB Instalasi, 5

**B**

**C**

**D**

Deploy Front-End dengan Github Pages, 33

Debug Hasil Front-End, 37

**E**

**F**

Fitur Pada GitHub ,13

**G**

Golang sebagai Back-End, 5

**H**

Hasil Akhir Tujuan Tampilan dan Debug, 36

Hasil Tampilan Front-End, 36

**I**

Instalasi Go, 4

**J**

Javascript tidak Memerlukan Instalasi, 4

Javascript sebagai Front-End, 6

**K**

keunggulan JS dengan Bahasa yang lain, 3

Kebutuhan Dan Logika Dalam Integrasi Go Dan Js, 4

Kebutuhan-Kebutuhan Yang Harus Di Penuhi & Tujuan, 4

**L**

**M**

Manfaat Mempelajari GO, 2

Manfaat Mempelajari JS, 3

Membuat Struct, 8

Membuat Function, 9

Membuat Test Code Function, 12

Membuat Get Data API dan Html, 29

Membuat Post Data API dan Html, 32

**N**

**O**

**P**

Pengenalan Dengan Go Dan Js, 1

Pengenalan, 1

Pembuatan Folder dan File untuk Package Backend, 7

Push Github Back-End dan Membuat Tag untuk Package Back-End, 13

Penggunaan Library Iteung, 16

Pembuatan Controller, 19

Pembuatan URL , 20

Push Github dan Deploy Heroku Iteung, 21

Pembuatan Folder, File dan Integrasi Front-End, 28

**Q**

**R**

**S**

Struktur File Backend Golang, 7

Struktur Folder Iteung, 16

Setting DB, 17

Setting CORS, 20

Setting Network Acces MongoDB Atlas Website, 26

Struktur Folder Front-End Javascript, 28

**T**

Tahap Praktek Coding Untuk Back-End Dan Front-End, 7

