

Astara

Realizado por: Juan David Ramírez Bernal

Curso académico: 2020/21
Grado Superior de Desarrollo de Aplicaciones Webs.

Índice

Introducción	pag 3.
Back-End	pag 4.
Front-End	pag 7.
Diseño	pag 9.
Despliegue	pag 16.
Bibliografía	pag 22.

Introducción

Astara pretende ser una aplicación web en la que los usuarios puedan controlar sus tareas y objetivos de una forma sencilla y clara, de esa forma incrementar el rendimiento de dichas personas.

Mi objetivo con Astara ha sido crear una web a la que posteriormente le siga dando uso y siga desarrollando, es por esto que decidí no hacer una tienda web, o similares, ya que pensaba que no le iba dar uso más allá de la entrega de este proyecto.

Astara está compuesta ,en términos generales, por dos partes, una API, que sería el back-end del proyecto, la cual está desarrollando en Golang (Go), elegido este lenguaje por su eficiencia y rapidez, así como su fácil aprendizaje. La segunda parte, es un front-end desarrollado en Vue.js, una librería de JavaScript que nos permite hacer un front-end ligero y versátil.

Back-End

Como ya he mencionado anteriormente, la parte Back-End del proyecto consiste en una API desarrollada en Golang, el cual se apoya en el cliente http "Fiber" para poder controlar las peticiones que le lleguen.

Podemos encontrar numerosas webs que comparan a Fiber con Express, una librería que nos permite controlar peticiones http cuando el Back-End está desarrollado en Node.js, y es que Fiber intenta imitar a Express, con una sintaxis y metodología similar.

Con Fiber podemos aprovechar la concurrencia que nos Brinda Golang, pudiendo así responder a varias peticiones http al mismo tiempo.

Gracias a su rapidez y sencillez Golang ha ido adquiriendo mucha popularidad estos últimos años, creando así una gran comunidad de desarrollo, la cual fomenta buenas prácticas y metodologías como veremos más adelante.

Una vez instalado y configurado Golang, empecé creando el proyecto y dividiéndolo en subpartes, de esta forma, en el proyecto se encuentran carpetas dedicadas para los controladores y módulos, así como funcionalidades comunes para todos estos.

La carpeta que contiene ficheros comunes, está dividida a su vez por otras carpetas, y es que Golang nos obliga a realizar esta práctica, a la hora de declarar nuevos paquetes, y de esta forma mantener el código más organizado.

Es importante denotar que dentro de cada fichero, las variables, funciones y tipos que empiezan por letra mayúscula indican que son "exportadas" fuera de su paquete, es decir, son visibles para el resto de archivos fuera de su paquete, y las que empiezan por minúscula, solo son visibles a nivel del propio paquete.

Para la implementación de las sesiones he utilizado JWT (JSON Web Tokens). Paso ahora a hacer una breve descripción de cómo lo hago.

Cuando un usuario se intenta autenticar ante la aplicación , lo primero que hará será escribir su nombre en el login, ya que nombre y correo electrónico son únicos y con ellos se puede identificar al usuario.

La API realizará una comprobación para ver si el usuario existe o no. Si el usuario no existe mandará un error y si sí existe pasará al siguiente paso, la introducción de la contraseña por parte del usuario. Una vez introducida, se recoge la contraseña de la base de datos y se comprueba que ambas sean iguales. Si son distintas se manda un error y si son iguales, se crea un JWT con información del usuario, exactamente se guarda el Id del usuario y el rol que tiene, además de una fecha de caducidad, que será igual a la actual más diez minutos.

Con esta información se genera una Cookie y se manda al navegador, esta cookie es HttpOnly, lo que quiere decir que el usuario no tendrá acceso a ella, además de establecerse que es de SameSite, por lo que solo será “útil” en el mismo dominio.

Con cada petición realizada a la API, se le envía la cookie en la cabecera, así que antes de trabajar la petición se realiza una validación de los valores de la cookie. a través de un middleware. En caso de que la cookie haya sido corrompida por el usuario, se le devuelve a este un estatus 401, y se le redirige al login. Esto es gracias, en parte, al interceptor que hay en el Front-End, que pasaré a explicar más adelante.

Si la cookie es correcta, se le añaden diez minutos más a su fecha de expiración, así el usuario podrá seguir navegando por el sitio web.

De esta forma, logro no guardar absolutamente nada en la base de datos que relacione al usuario con su estado, siendo así la API, complemente Stateless.

Si un usuario malintencionado captura una cookie con el token de un usuario, tendrá escasos diez minutos para poder descifrar su contenido, para lo cual deberá de encontrar la “private key” que guarda el servidor, lo cual no es una tarea fácil.

Por otro lado, la conexión de la base de datos está dividida en tres sub conexiones, cada una de ellas es un usuario distinto de la base de datos, cada uno con privilegios distintos, de esa forma añado una capa más de seguridad al sitio web.

Estas tres sub conexiones son , la primera la “Non user”, la cual se utiliza para comprobar las credenciales de los usuarios al registrarse, haría las veces de lo que comúnmente se llama “un logger”. La segunda la de “User” la cual tendrá cada usuario que se loguea, y que podrá crear, editar, borrar las tablas de targets, task, goals y user, para la actualización de la información de usuario. La tercera y última, la de “Admin”, la cual tienen más permisos, pero ahora mismo no se puede obtener dicha conexión, ya que mi sitio web no dispone de una parte de administración, por lo que la conexión no es dada a ningún usuario que entre.

Son con estas tres y solo estas tres instancias, que la web funciona. Cuando un usuario se intenta loguear o intenta realizar una operación dentro de la web, pide la instancia de la base de datos correspondiente y si no existe aún crea una, que podrá ser usada posteriormente por el resto de usuarios que la necesiten, en esencia, la base de datos utiliza el patrón de diseño Singleton.

Debido al carácter concurrente de Golang, se podría dar el caso de que dos usuarios hayan pedido una instancia a la vez, y que esta no estuviese definida aún, por lo que ambos generarían una nueva instancia, pudiéndose dar el caso de que uno sea más rápido que otro y settee la instancia, produciendo un error, ya que el otro también intentará settee la instancia, lo cual es un problema. Para solucionar esta situación, he implementado un sistema de bloqueo, por el cual si un usuario requiere de la creación de una instancia, esta es bloqueada para el resto de usuarios y una vez esté creada, será desbloqueada y accesible para el resto de usuarios.

Me gustaría destacar también el uso de punteros y referencias en la obtención de objetivos y tareas, ya que fue un problema bastante tedioso.

En la web, se pueden crear objetivos los cuales tenga a su vez dentro de ellos otras tareas, es decir, un objetivo puede tener tareas anidadas dentro de él. En este caso solo permito una anidación al primer nivel.

Para explicar mejor esto, me gustaría decir que ambos, tareas y objetivos, vienen de un “elemento” superior, un padre, llamado “target”. De aquí tareas y objetivos reciben “contenido” común a ambos, es decir, “target” es el padre de “tareas” y “objetivos” que son sus hijos.

Dicho esto, al realizar una petición para recibir los objetivos de cierto usuario, le pido a la base de datos que me devuelva, no solo los objetivos, sino todos los targets, paginados, de ese usuario, para un área en específico.

Posteriormente paso a procesar todo el contenido devuelto. Para ello creo un mapa clave valor de los “targets”, el motivo de usar un mapa (hashmap) es que la complejidad de la inserción y búsqueda es $O(1)$. Si el “target” pertenece a otro, es decir, corresponde a la anidación de otro target, entonces, se le añade al array de anidados de ese “target”. Pero esto tiene un problema y es que Golang le da un valor nulo a ese array, y en el procesado del hashmap no se puede hacer un “push” al array, así que la solución viene del uso de punteros. De ese modo, si el array es nulo, en el caso de Golang, lo llama “nil”, creo un nuevo array y asocio el array de anidación del “target” con valor nulo, al nuevo creado, (a través de punteros y referencia) que contendrá el “target” hijo anidado.

Debido a este proceso, el mapa se desordena, dando lugar a cambios de sitio en la posición de las tareas y objetivos en el Front-end. Para solucionar este problema, luego de anidar los “targets” se ordenan por su identificador y finalmente se convierte en array y luego en string, para poder ser enviado en forma de JSON, al cliente.

Front-End

En el Front-End he usado Vue-cli y Vue-router, además de Axios, para realizar las peticiones al servidor.

Paso ahora a comentar un poco la estructura de las carpetas y su contenido.

El proyecto consta de tres “views” que podemos encontrar en su correspondiente carpeta, estas tres “views” son la “Landing” page, la “Home” page y la página de “Login”. Dentro cada componente se encuentran anidados el resto de componentes que se encuentran dentro de la carpeta “components” y que están distribuidos por su “temática”.

También podremos encontrar la carpeta “assets/” donde se almacenan estilos comunes a los distintos componentes, la carpeta “auth/” que hace de auxiliar para la autenticación del usuario, la carpeta “js/” que contiene código JavaScript auxiliar, y la carpeta “router/” donde se encuentra el router de la aplicación.

Dentro de la carpeta “auth/” se definen cuatro cosas, una instancia de Axios, la cual es importada por el resto de componentes que la necesiten. El uso de esta instancia es importante, ya que de otro modo, cada vez que crease una instancia de Axios, debería darle una configuración, gracias a la instancia, puedo darle una configuración a ella y repartir la instancia a los componentes.

También se define un interceptor, el cual se ejecutará antes de procesar la respuesta mandada por el servidor, de este modo, si el servidor manda una respuesta con estatus 401 el interceptor la recibirá y mandará al usuario fuera de la aplicación, a la pantalla de logueo.

Junto a estas instancias se definen dos funciones, la primera “Validate” comprueba que las respuestas del servidor entran dentro del rango 200 y la segunda “AreaCorrespond” comprueba que el área al que está intentando entrar el usuario existe para ese usuario, de otro modo será llevado al Dashboard.

También realizo una validación de los inputs de los formularios, tanto en el front-end como en el back-end. En caso de que haya algún error, uso o bien un componente que me ayuda a mostrarlo o un mensaje personalizado.

Dentro de los componentes podemos encontrar los típicos datos, métodos, vistas condicionadas, listas, etc, además de propiedades computadas, enlazado de estilos condicional, manejo de eventos de padrea a hijo y de hijo a padre, con emits, y refs, etc.

También he usado transiciones, no solo a través de estilos sino a través de las Vue-transitions. He intentado implementar las animaciones de JQuery, aunque estas no hacen buena complementación con vue. Por ejemplo, al usar el “fadeOut” de JQuery, debido a la indexación de Vue, en lugar de eliminar una tarea, eliminaba dos, por esa razón opté por no utilizar JQuery para las animaciones.

Para practicamente todo el desarrollo del front-end me he apoyado en el uso de JQuery.

Desgraciadamente no he podido implementar JQueryUI ni animaciones. También he usado SplittingJS, que es una librería de JS que permite dividir texto en diferentes secciones, líneas, palabras, caracteres, etc, y que me ha sido de gran utilidad a la hora de dar estilos a la web.

También he creado un shortcode, a través del cual se puede acceder al perfil de manera más rápida.

Además de hecho uso del two ways databinding que nos ofrece Vue, para hacer más interacciones con el usuario. Por ejemplo al modificar el nombre de usuario en el perfil, este cambia automáticamente en el sidebar donde se muestra el nombre de usuario.

En la sección de tareas y objetivos, existe al final de cada listado un elemento observable, el cual me ayuda a realizar la paginación de estos elementos. Gracias al uso de varios intersection observers, (uno en la sección de objetivos y otro en la sección de tareas) cuando el elemento es visto, se realiza una petición al servidor que devuelve la siguiente tanda de resultados.

Esto es muy útil en la búsqueda de los elementos, ya que en la parte superior podemos encontrar una barra de búsqueda, esta está asociada a una propiedad computada del tareas y objetivos, de modo que solo muestra los elementos cuyo nombre contiene la cadena que se busca, como consecuencia de esto, el elemento observable se vuelve visible ya que viene hacia arriba, de este modo, si no se ha encontrado el resultado, se harán sucesivas peticiones al servidor hasta que este devuelva todos los elementos, encontrando el objetivo si es que se encuentra en ese área o si es que existe.

Diseño

Para los estilos he usado el plugin de Vue que permite el uso de Sass. En la carpeta “assets/” podemos encontrar estilos comunes a todos los componentes, así como las media queries. Aunque hubiera sido mejor no usar el plugin, ya que no controlo demasiado bien el orden en el que Vue, añade los estilos, por lo que los elementos de las media queries llevan todos el “!important”, lo cual no me gusta nada. Quizás hubiera sido mejor no usar este plugin que nos ofrece Vue, y añadir los estilos de una forma más tradicional.

Para la elección de la paleta de colores he elegido el amarillo como color principal he elegido el amarillo, color simbólico de las estrella, como referencia al nombre de la web. También he elegido una escala de grises, que me permitiese generar dos modos, uno claro y otro oscuro.

Siendo mi paleta una variante de esta otra, que usé como inspiración. He usado el mismo color amarillo claro (#F0A202) y el mismo gris (#E032DB). El resto de colores aunque no los he usado han sido de inspiración para crear la gama de colores.



Estos son algunos de los colores añadidos, con sus correspondientes nombres usado en la aplicación.

\$black (primary) = #242423k

\$lightBlack (secondary) = #353535

\$tertiary = #808080

\$white= #eff2f1

Con esta gama de grises, he creado tema claro y otro oscuro. Al más frecuente, lo llamo “primary”, además uso el color “secondary” y “tertiary” para añadir detalles. Estos colores son cambiados por su versión clara, al cambiar de tema.

Dejo a continuación una foto del tema claro y oscuro.

PERSONAL

Buscar:

Editar

X

Tareas

Objetivos

+ Goal

COMPRAR GUITARRA ELÉCTRICA

Editar

X

Fecha límite: 2021-06-24

Siempre he querido aprender a tocar la guitarra y me gustaría poder comprarme una

Crear tarea

Ocultar tareas

☐

AHORRAR 500\$

Fecha límite: 2021-06-22

Planeado para: 2021-07-01

Editar

X

☐

BUSCAR UNA GUITARRA

Fecha límite: 2021-06-14

Planeado para: 2021-06-28

Editar

X

Tareas

Objetivos

+ Goal

APRENDER A TOCAR LA GUITARRA

Editar

X

Fecha límite: 2021-09-10

Me gustaría aprender a tocar la guitarra eléctrica por que es mi instrumento musical favorito y siempre he querido tocarla desde que era pequeño

Crear tarea

Ocultar tareas

☐

IR A LA ACADEMÍA

Fecha límite: 2021-06-24

Planeado para: 2021-06-20

Editar

X

☐

COMPRAR GUITARRA

Fecha límite: 2021-06-14

Planeado para: 2021-06-03

Editar

X

COMPETIR EN NATACIÓN

Editar

X

Fecha límite: 2021-06-24

Competir este año en el torneo de natación

Crear tarea

Dejo a continuación una imagen que sirvió como inspiración para el diseño de la página web.

Para las fuentes he usado Lora, una fuente serif y la clásica Roboto, que es sans-serif, creo que ambas proporcionan un buen contraste a la vez de ser legibles. Para el título de las áreas, las descripciones de los objetivos y en los textos de la landing page, he usado Lora, mientras que en el resto de títulos he usado Roboto, con más espaciado entre las líneas y a veces aumentando su grosor para darle más peso.



He exagerado los títulos de las áreas y sobretodo del dashboard, haciéndolos muy grandes, muy visibles, también he incrementado el tamaño de los nombres de las tareas y objetivos dándoles mayor peso visual.

Las imágenes de la landing page tienen un efecto parallax, además de un diseño “roto”.

El logo es un SVG hecho con Inkspace. Dejo una foto del logo, aunque también se puede encontrar el svg dentro de la carpeta “astara/src/assets/”.



El diseño es responsive, lo que no me termina de cuadrar es el sidebar, en versión movil, ya que no se adapta del todo bien, obviando esto, el resto de elementos se adapta bien al tamaño movil.

Dejo a continuación imágenes de ello.



La transparencia de la barra de navegación es adrede.



Astara Entrar

Sé más productivo con los short-cuts. Usa atajos de teclado para agilizar tus tareas, se más rápido y productivo. Abre ventanas, edita el perfil, y mucho más.



WHITE MODE

Para los amantes del white mode. Ahora tienes la opción de usar la aplicación en white mode. Disfruta de la aplicación de



Astara Inicio

Order is key to control chaos

Entra y empieza organiza tus tareas y objetivos para ser más productivo.

Entrar

pruebados

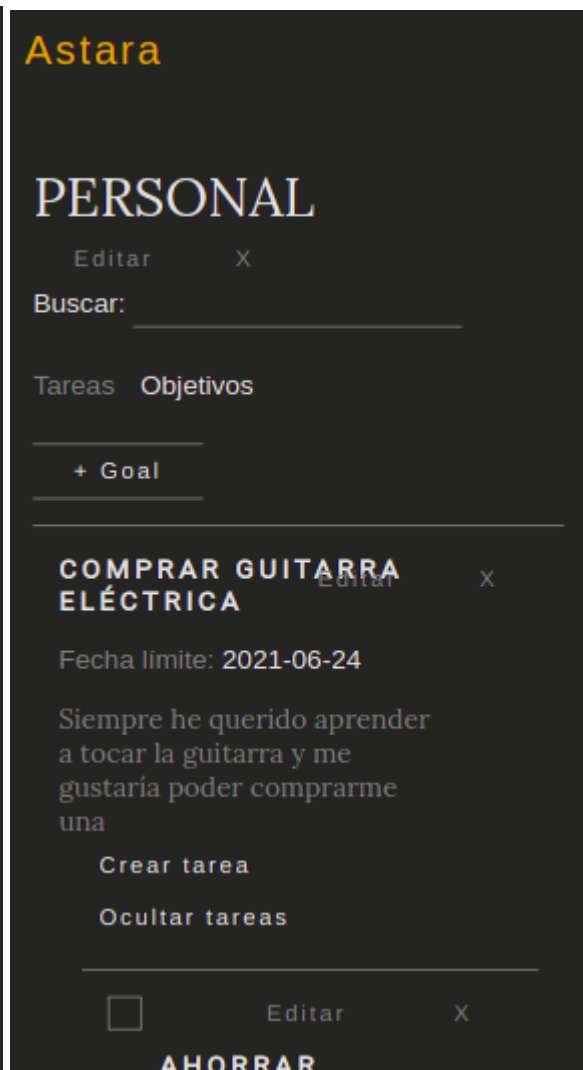
.....|



Mostar contraseña

Aceptar

¿Nuevo en Astara? Hazte una cuenta



Dejo a continuación a tamaño pc.



Astara

AREAS
MAIN
NOVEDOSO
PERSONAL

+ Area

pruebados

OBJETIVOS PRINCIPALES

Perfil

Salir

Nombre de usuario
El nombre de usuario es único y te identifica en la aplicación.
pruebados

Email
Recuerda que también puedes usar tu correo electrónico para acceder a la aplicación.

Cambiar contraseña
Introduce tu contraseña antigua para cambiarla.

Mostrar

Comprobar contraseña

Theme
Oscuro

Aceptar

Cancelar

Editar

X

Crear tarea

Editar

X

Editar

X

Astara

AREAS
MAIN
NOVEDOSO
PERSONAL

+ Area

pruebados

OBJETIVOS PRINCIPALES

COMPRAR GUITARRA ELÉCTRICA

Editar

X

Fecha límite: 2021-06-24

Siempre he querido aprender a tocar la guitarra y me gustaría poder comprarme una

Crear tarea

TAREAS

☐

NUEVA TAREA

Fecha límite: 2021-07-06

Planeado para: 2021-06-25

Editar

X

☐

PRUEBA

Fecha límite: 2021-06-30

Planeado para: 2021-06-30

Editar

X

Despliegue

Para el despliegue de la aplicación web compré el dominio `astarapp.site` en Hostinger y un hosting en DigitalOcean, configuré ambos para que se apuntasen y monté un servidor de aplicaciones Nginx, en una máquina ubuntu 20.02 LTS dentro del servidor.

Para instalar Nginx, simplemente ejecuté el comando “sudo apt install nginx”.

```
v4: 138.68.135.
root@astara: ~
systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:
   Active: active (running) since Tue 2021-06-15 18:19:34 UTC; 1 day 11h ago
     Docs: man:nginx(8)
    Main PID: 309575 (nginx)
      Tasks: 2 (limit: 1136)
     Memory: 3.7M
    CGroup: /system.slice/nginx.service
           └─309575 nginx: master process /usr/sbin/nginx -g daemon on; maste
              └─309576 nginx: worker process

Jun 15 18:19:34 astara systemd[1]: Starting A high performance web server and a
Jun 15 18:19:34 astara systemd[1]: Started A high performance web server and a
root@astara: #
```

Dentro del servidor instalé mysql-server, cloné el frontend y traspasé la API por ftp.

Para la instalación de mysql-server, ejecutando el comando “sudo apt install mysql-server” se instala.

Luego con el comando “mysql_secure_installation” creé otro usuario para poder conectarme a la base de datos.

Una vez tuve la base de datos montada en el servidor, intenté importar la base de datos que tenía en local, pero finalmente, copie y pegué las instrucciones de la creación de las tablas que general la importación directamente.


```
root@astara: ~  
mysql> SHOW TABLES  
+-----+  
| Tables_in_astara |  
+-----+  
| Areas            |  
| Goals            |  
| Status           |  
| Targets          |  
| Task             |  
| Users            |  
+-----+  
6 rows in set (0.00 sec)  
  
mysql>
```

Los servidores de DigitalOcean ya tiene instalado git, así que no tuve que instalarlo para clonar el repositorio.

Pero no tienen “npm” instalado, pero simplemente ejecutando “sudo apt install npm” se instala.

Solo creé una nueva carpeta en “/var/www” con el nombre de la aplicación y clone allí el contenido del repositorio, eliminé lo que no formaba parte del fronted y luego, generé un directorio “dist/” que sería la versión de producción de nuestro front-end que nos ofrece Vue. Pero antes de ejecutar este comando, cambié la dirección a la que Axios, hacía las peticiones para que apuntase al servidor. Luego ejecuté el comando “npm run build” para generar el directorio “dist/”.

```
root@astara: ~  
root@astara:~# ls /var/www/astara/  
README.md babel.config.js dist node_modules package-lock.json package.json public src  
root@astara:~#
```

Para conectarme al servidor he usado SSH he usado la contraseña que me proporcionó DigitalOcean, que más tarde cambié y los archivos los he pasado de dos formas, a través con Git ejecutando el comando “git clone” a mi repositorio, y la siguiente forma es a través de FTP, para lo cual monté el servicio svftpd en el servidor e instalé Filezilla en mi ordenador personal.

Q Search by resource name or public IP (Ctrl+B)

Create ▾



USAGE
\$1.43




Astara
Web Application

→ Move Resources

Resources Activity Settings

DROPLETS (1)

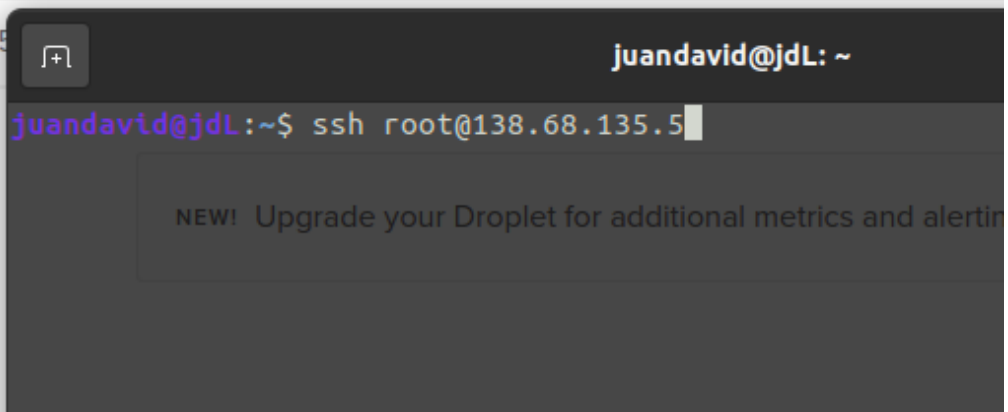
  astara	138.68.135.5	go +1	 ...
---	--------------	-------	---

DOMAINS (1)

astarapp.site	2 A / 3 NS / 1 SOA	...
----------------------	--------------------	-----

IPv4: 138.68.135.5

raphs
ccess
ower
olumes
esize



Dejo a continuación una imagen con la configuración del servidor de aplicaciones nginx.

```

server {
    listen 80;
    listen [::]:80;
    server_name astarapp.site;

    root /var/www/astara/dist;

    index index.html index.htm index.nginx-debian.html;

    error_page 404 /;

    location / {
        add_header Access-Control-Allow-Origin *;
        add_header Access-Control-Allow-Credentials true always;
    }

    location /api/v1 {
        proxy_pass http://138.68.135.5:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Cookie $http_cookie;
        proxy_set_header language es;
        proxy_ignore_headers Set-Cookie;

        add_header Access-Control-Allow-Origin *;
        add_header Access-Control-Allow-Credentials true always;
    }

    location ~ /\.ht {
        deny all;
    }
}

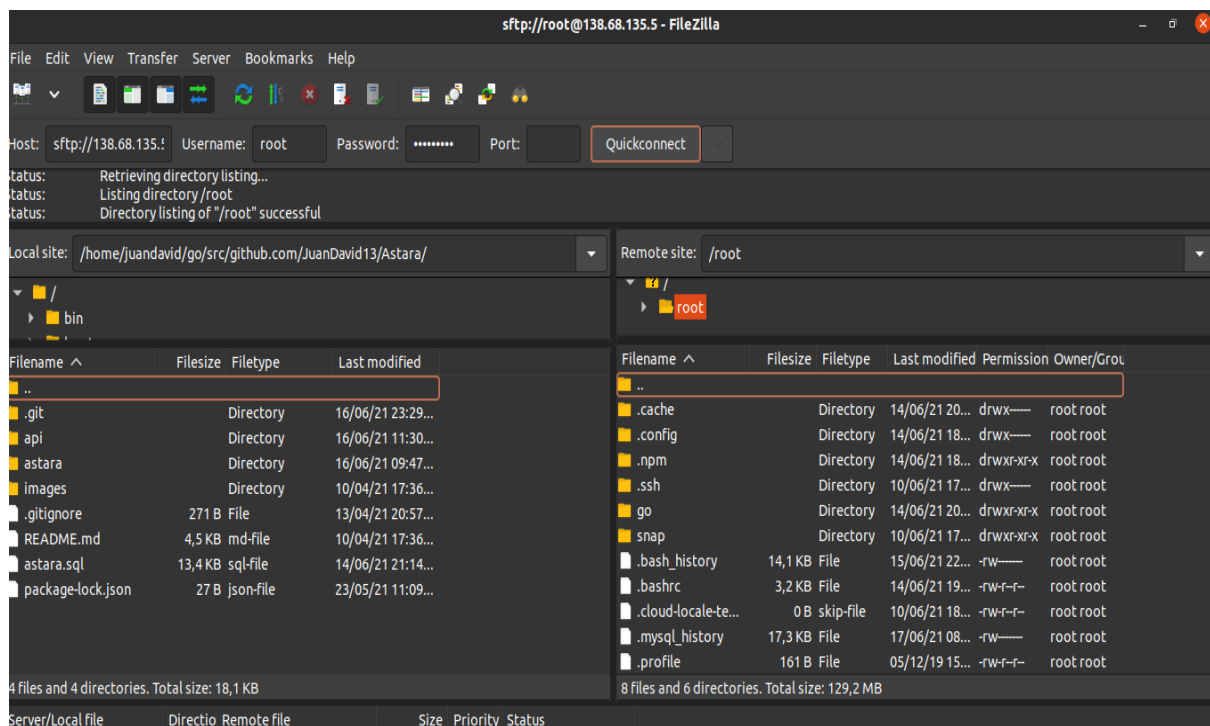
```

```

root@astara:~# ls ./astara/
astara  commons  controllers  go.mod  go.sum  main.go  models
root@astara:~#

```

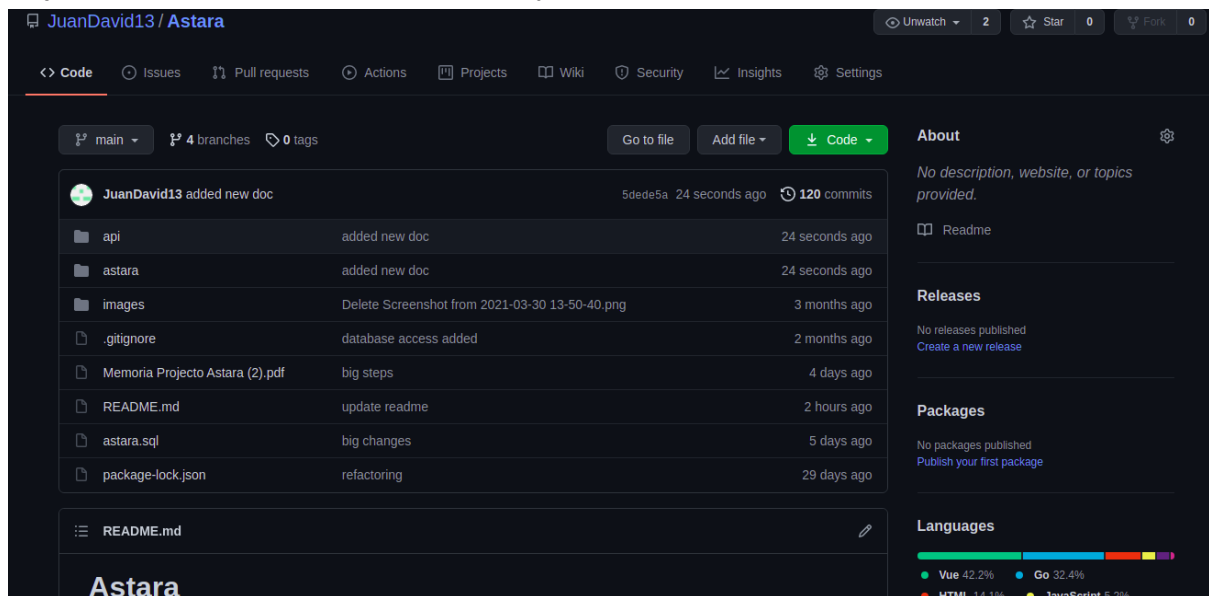
```
root@astara: ~  
listen=NO  
listen_ipv6=YES  
anonymous_enable=NO  
local_enable=YES  
dirmessage_enable=YES  
use_localtime=YES  
xferlog_enable=YES  
connect_from_port_20=YES  
secure_chroot_dir=/var/run/ssh/empty  
pam_service_name=vsftpd  
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem  
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key  
ssl_enable=NO  
~  
~
```



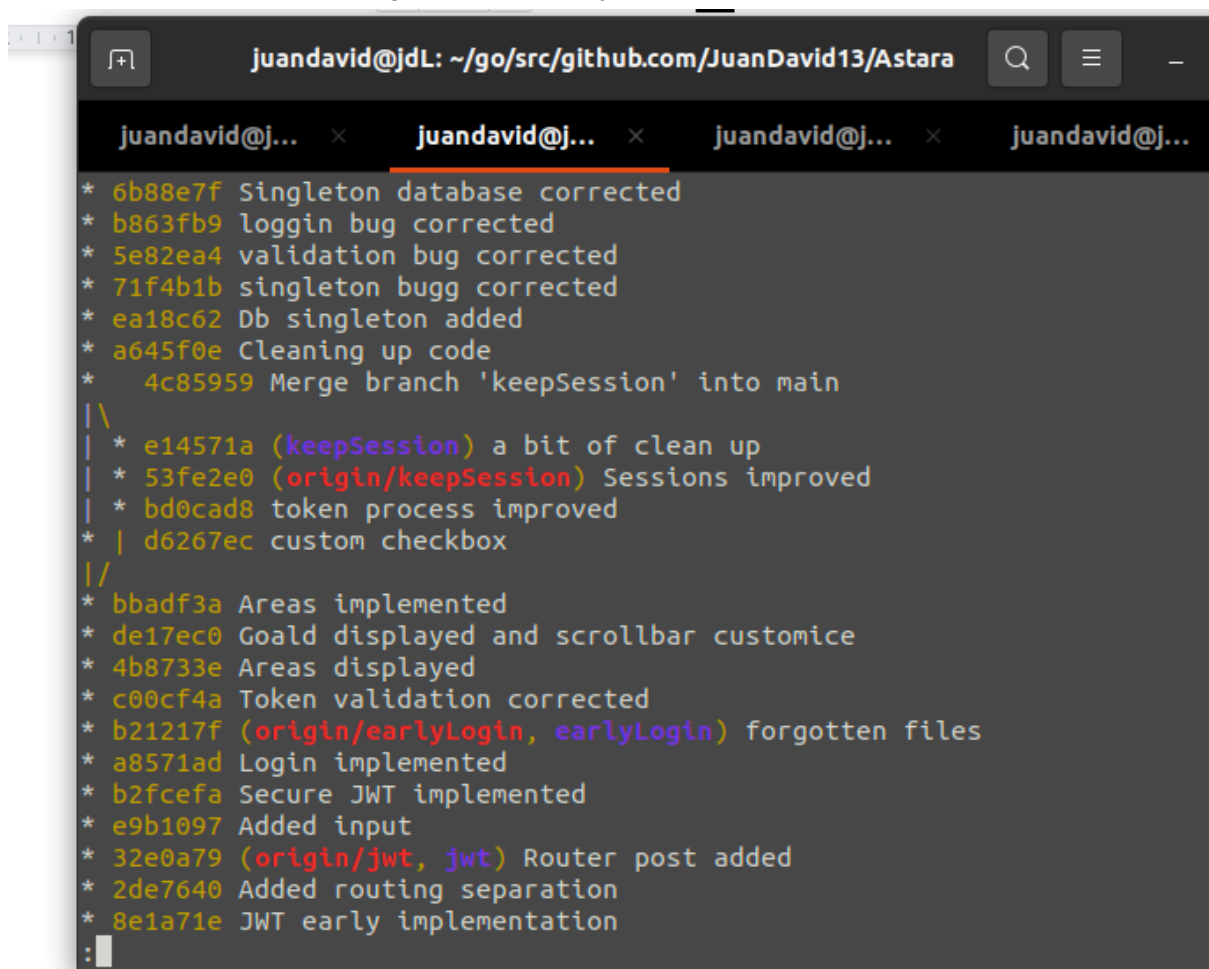
La configuración del servicio ftp, es sumamente insegura, el motivo de que sea así, es que al principio, para no tener problemas estuve haciéndolo todo como root, para tener los menos problemas posibles, pero finalmente tuve un problema en su momento yo pensaba que era con nginx, ya que las peticiones que hacía el navegador ninguna traía consigo la cabecera “Set-Cookie” que debería de mandar el servidor, sin embargo cuando realizaba la misma petición desde Postman, si la traía. A día de hoy aún no he solucionado el problema, pero seguiré trabajando para encontrar una solución.

Después de pasar horas y horas sin poder solucionar el problema, no continué con la configuración del servidor, ya que aún me faltaban cosas por arreglar del resto de la aplicación.

Con respecto al uso de Git, he usado la plataforma de Github para alojar el repositorio del proyecto, el cual tiene unos 120 commits y 4 ramas.



Muestro a continuación un segmento de la trayectoria del repositorio.



Respecto a la documentación automática, he usado JsDoc para la creación de documentación.

Golang no dispone, o al menos yo no he encontrado, ningún paquete o programa que genere documentación de forma automática.

Bibliografía

<https://jwt.io/>

<https://pkg.go.dev/github.com/dgrijalva/jwt-go>

<https://pkg.go.dev/github.com/joho/godotenv>

<https://axios-http.com/docs/example>

<https://splitting.js.org/guide.html#splitting>

<https://vuejs.org/>

<https://golang.org/ref/spec>

<https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>

<https://dev.to/koddr/how-to-deploy-golang-application-on-any-gnu-linux-system-but-without-docker-59m1>

<https://qvault.io/cryptography/jwts-in-golang/>

<https://medium.com/@vo9312/consider-go-fiber-as-an-upgrade-for-express-js-1569176f65d>

<https://flaviocopes.com/golang-sql-database/>

<https://medium.com/golang-issue/how-singleton-pattern-works-with-golang-2fdd61cd5a7f>

<https://medium.com/@haxzie/dark-and-light-theme-switcher-using-css-variables-and-pure-javascript-zocada-dd0059d72fa2>

<https://v3.vuejs.org/guide/transitions-list.html#staggering-list-transitions>

<https://docs.digitalocean.com/products/droplets/how-to/connect-with-ssh/openssh/>