Сервер приложений

Содержание

[Введение 5](#_Toc41893307)

[1 Технические характеристики 6](#_Toc41893308)

[1.1 Среда разработки 6](#_Toc41893309)

[2 Описание программы 7](#_Toc41893310)

[2.1 Сторонние библиотеки 7](#_Toc41893311)

[2.2 Структура RoleAccess.json 8](#_Toc41893312)

[2.3 Настроечный файл Config.toml 8](#_Toc41893313)

[2.4 Структуры 9](#_Toc41893314)

[2.4.1 Account 9](#_Toc41893315)

[2.4.2 AreaInfo 10](#_Toc41893316)

[2.4.3 BoxPoint 10](#_Toc41893317)

[2.4.4 BusyArm 10](#_Toc41893318)

[2.4.5 CacheData 11](#_Toc41893319)

[2.4.6 CheckData 11](#_Toc41893320)

[2.4.7 Config 11](#_Toc41893321)

[2.4.8 DBConfig 11](#_Toc41893322)

[2.4.9 DeviceLog 12](#_Toc41893323)

[2.4.10 DeviceLogInfo 12](#_Toc41893324)

[2.4.11 EditCrossInfo 12](#_Toc41893325)

[2.4.12 FillingInfo 12](#_Toc41893326)

[2.4.13 Locations 12](#_Toc41893327)

[2.4.14 PassChange 13](#_Toc41893328)

[2.4.15 Permission 13](#_Toc41893329)

[2.4.16 PngSettings 13](#_Toc41893330)

[2.4.17 Point 13](#_Toc41893331)

[2.4.18 Privilege 14](#_Toc41893332)

[2.4.19 RegionInfo 14](#_Toc41893333)

[2.4.20 Role 15](#_Toc41893334)

[2.4.21 RoleAccess 15](#_Toc41893335)

[2.4.22 RoleData 15](#_Toc41893336)

[2.4.23 RouteInfo 15](#_Toc41893337)

[2.4.24 SelectedData 15](#_Toc41893338)

[2.4.25 ServerLogInfo 16](#_Toc41893339)

[2.4.26 ShortAccount 16](#_Toc41893340)

[2.4.27 TLSostInfo 17](#_Toc41893341)

[2.4.28 Token 17](#_Toc41893342)

[2.4.29 TrafficLights 17](#_Toc41893343)

[2.4.30 LogFile 17](#_Toc41893344)

[2.4.31 StateResult 18](#_Toc41893345)

[2.4.32 ArmCommandMessage 18](#_Toc41893346)

[2.4.33 StateMessage 19](#_Toc41893347)

[2.4.34 TCPConfig 19](#_Toc41893348)

[2.5 Утилиты 19](#_Toc41893349)

[2.6 Middleware route 20](#_Toc41893350)

[2.7 Обработчики 20](#_Toc41893351)

[2.7.1 ActAddAccount 20](#_Toc41893352)

[2.7.2 ActChangePw 20](#_Toc41893353)

[2.7.3 ActDeleteAccount 20](#_Toc41893354)

[2.7.4 ActUpdateAccount 21](#_Toc41893355)

[2.7.5 BuildCross 21](#_Toc41893356)

[2.7.6 BuildMapPage 21](#_Toc41893357)

[2.7.7 CheckAllCross 21](#_Toc41893358)

[2.7.8 CheckSelectedDirCross 21](#_Toc41893359)

[2.7.9 ControlCheckButton 21](#_Toc41893360)

[2.7.10 ControlCloseCross 22](#_Toc41893361)

[2.7.11 ControlCreateButton 22](#_Toc41893362)

[2.7.12 ControlCross 22](#_Toc41893363)

[2.7.13 ControlDeleteButton 22](#_Toc41893364)

[2.7.14 ControlEditableCross 22](#_Toc41893365)

[2.7.15 ControlSendButton 22](#_Toc41893366)

[2.7.16 ControlTestState 23](#_Toc41893367)

[2.7.17 CrossEditFree 23](#_Toc41893368)

[2.7.18 CrossEditInfo 23](#_Toc41893369)

[2.7.19 DevCrossInfo 23](#_Toc41893370)

[2.7.20 DispatchControlButtons 23](#_Toc41893371)

[2.7.21 DisplayAccInfo 24](#_Toc41893372)

[2.7.22 DisplayDeviceLogFile 24](#_Toc41893373)

[2.7.23 DisplayServerLogFile 24](#_Toc41893374)

[2.7.24 DisplayServerLogInfo 24](#_Toc41893375)

[2.7.25 LocationButtonMapPage 24](#_Toc41893376)

[2.7.26 LogDeviceInfo 25](#_Toc41893377)

[2.7.27 LoginAcc 25](#_Toc41893378)

[2.7.28 LoginAccOut 25](#_Toc41893379)

[2.7.29 MainCrossCreator 25](#_Toc41893380)

[2.7.30 MakeSelectedDirCross 25](#_Toc41893381)

[2.7.31 UpdateMapPage 26](#_Toc41893382)

[2.7.32 LicenseInfo 26](#_Toc41893383)

[2.7.32 LicenseInfo 26](#_Toc41893384)

[2.7.33 LicenseNewKey 26](#_Toc41893385)

[2.7.34 TechSupp 26](#_Toc41893386)

[2.7.34 ChatEngine 26](#_Toc41893387)

# Введение

Сервер приложений (TLServer) осуществляет обработку запросов от пользователей.

Данные передаются посредством протокола HTTP (HyperText Transfer Protocol - протокол передачи гипертекста) и WebSocket — протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

Хранение информации осуществляется посредством  свободной объектно-реляционная системы управления базами данных PostgreSQL, базируется на языке SQL и поддерживает многие из возможностей стандарта SQL.

Сервер приложений осуществляет связь посредством запросов с сервером устройств Ag-Server.

1 Технические характеристики

* 1. Среда разработки

Языком программирования выбран GOlang - компилируемый многопоточный язык программирования, разработанный внутри компании Google.

Версия языка на 04.2020 является 1.14.3.

# 2 Описание программы

## 2.1 Сторонние библиотеки

1) github.com/BurntSushi/toml v0.3.1

Библиотека для работы с TOML (формат конфигурационных файлов), позволяется кодировать и раскодировать файлы данного формата.

2) github.com/dgrijalva/jwt-go v3.2.0+incompatible

Библиотека поддерживает синтаксический анализ и проверку, а также генерацию и подписание JWT (json web token). В настоящее время поддерживаются следующие алгоритмы подписи: HMAC SHA, RSA, RSA-PSS и ECDSA.

3) github.com/gorilla/websocket v1.4.2

Библиотека обработки протокола связи websocket – содержит в себе набор инструментов, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

4) github.com/gin-gonic/gin v1.6.3

Библиотека реализует маршрутизатор и диспетчер запросов для сопоставления входящих запросов с их соответствующим обработчиком.

5) github.com/jmoiron/sqlx v1.2.0

Библиотека реализует интерфейсы для работы с БД.

6) github.com/lib/pq v1.3.0 // indirect

Драйвер для работы с postgresql.

7) github.com/pkg/errors v0.9.1

Библиотека, обеспечивающая простую обработку ошибок.

8) golang.org/x/crypto v0.0.0-20200323165209-0ec3e9974c59

Стандартная библиотека, предоставляющая возможности по шифрованию информации.

9) github.com/ruraomsk/ag-server v0.0.0-20200325075148-7bcc97b69ad3

Сервер управления АСУД.

10) github.com/jordan-wright/email v0.0.0-20200521030443-c069f37d901d

Библиотека, предоставляющая инструменты для работы с почтовыми клиентами.

## 2.2 Структура RoleAccess.json

RoleAccess.json содержит информацию для осуществления контроля доступа пользователей к ресурсам.

Файл содержит 3 ключа:

1. roles – информация о роли

ключи:

name – имя,

permissions - массив разрешений;

1. permissions – информация о разрешении

ключи:

id – уникальный номер,

description – описание,

visible - флаг видимости, необходим для работы с гибкими ролями;

1. routes – информация о маршруте

ключи:

id – уникальный номер,

description – описание,

permission – к какому разрешению относится маршрут,

path – маршрут ресурса.

## 2.3 Настроечный файл Config.toml

Config.toml содержит всю настроечную информацию для корректной работы сервера.

Структура toml файла:

logger\_path = "" //путь до каталога с логами сервера

static\_path = "" //путь до каталога static (содержит все ресурсы для отображения перекрестков)

web\_path = "" //путь до каталога web

server\_ip = " //ip сервера / порт

ya\_key = "" //ключ авторизации для яндекса

token\_password = "" //ключ для шифрования токенов доступа

Настройка подключения к тсп серверу обемена командами  
[tcpServer]  
tcpServerAddress = "" //адресс сервера  
portState = " //порт для обмена Стате  
portArmCommand = " //порт для обмена арм командами

portMessageInfo = " //порт для обмена Message командами  
  
Настройки для постгреса  
[database]  
db\_name = "" //имя бд  
db\_password = "" //пароль доступа к бд  
db\_user = "" //пользователь для обращения к бд  
db\_type = "" //тип бд  
db\_host = "" //ip сервера бд  
db\_port = "" //порт обращения к бд  
db\_SetMaxOpenConst = //максимальное количество пустых соединений с бд  
db\_SetMaxIdleConst = //максимальное количество соединений с бд

## 2.4 Структуры

### 2.4.1 Account

Account структура аккаунта пользователя

type Account struct {

ID int `json:"id",sql:"id"` //уникальный номер пользователя

Login string `json:"login",sql:"login"` //Имя пользователя

Password string `json:"password"` //Пароль

BoxPoint locations.BoxPoint `json:"boxPoint",sql:"-"` //Точки области отображения

WorkTime time.Duration `json:"workTime",sql:"workTime"` //Время работы пользователя в часах

YaMapKey string `json:"ya\_key",sql:"-"` //Ключ доступа к яндекс карте

Token string `json:"token",sql:"-"` //Токен пользователя

}

Методы:

1. ChangePW изменение пароля пользователя

func (account \*Account) ChangePW() u.Response

1. Create создание аккаунта для пользователей

func (account \*Account) Create(privilege Privilege) u.Response

1. Delete удаление аккаунта из БД

func (account \*Account) Delete() u.Response

1. GetInfoForUser собор информацию для пользователя, который авторизировался

func (account \*Account) GetInfoForUser() u.Response

1. ParserBoxPointsUser заполняет BoxPoint

func (account \*Account) ParserBoxPointsUser() (err error)

1. Update обновление данных аккаунта

func (account \*Account) Update(privilege Privilege) u.Response

1. Validate проверка аккаунда в бд

func (account \*Account) Validate() (u.Response, bool)

### 2.4.2 AreaInfo

AreaInfo информация о районе

type AreaInfo struct {

Num string `json:"num"` //уникальный номер района

NameArea string `json:"nameArea"` //расшифровка номера

}

Методы:

1. SetAreaInfo установить в структуру номер и имя района по номеру района и региона

func (area \*AreaInfo) SetAreaInfo(numReg, numArea string)

2.4.3 BoxPoint

BoxPoint координаты для отрисовки зоны работы пользователя

type BoxPoint struct {

Point0 Point `json:"point0"` //левая нижняя точка на карте

Point1 Point `json:"point1"` //правая верхняя точка на карте

}

2.4.4 BusyArm

BusyArm информация о занятом перекрестке

type BusyArm struct {

Region string `json:"region"` //регион

Area string `json:"area"` //район

ID int `json:"ID"` //ID

Description string `json:"description"` //описание

}

2.4.5 CacheData

CacheData Данные для обновления в определенный период

type CacheData struct {

mapRegion map[string]string //регионы  
mapArea map[string]map[string]string //районы  
mapTLSost map[int]string //светофоры

}

2.4.6 CheckData

CheckData структура проверки для перекрестков

type CheckData struct {

ID string `json:"ID"` //ID устройства

PngStatus bool `json:"pngStatus"` //флаг наличия map.png

SvgStatus bool `json:"svgStatus"` //флаг наличия cross.svg

}

2.4.7 Config

Config структура с объявлением всех переменных config.toml файла

type Config struct {

YaKey [string](http://localhost:6060/pkg/builtin/#string) `toml:"ya\_key"` //ключ авторизации для яндекса

TokenPassword [string](http://localhost:6060/pkg/builtin/#string) `toml:"token\_password"` //ключ для шифрования токенов доступа

TCPConfig [tcpConnect](http://localhost:6060/pkg/github.com/JanFant/TLServer/internal/app/tcpConnect/).[TCPConfig](http://localhost:6060/pkg/github.com/JanFant/TLServer/internal/app/tcpConnect/#TCPConfig) `toml:"tcpServer"` //информация о tcp соединении с сервером устройств

DBConfig [DBConfig](http://localhost:6060/pkg/github.com/JanFant/TLServer/int1/model/config/#DBConfig) `toml:"database"` //информация о соединении с базой данных

}

2.4.8 DBConfig

DBConfig настройки для работы с базой данных

type DBConfig struct {

Name [string](http://localhost:6060/pkg/builtin/#string) `toml:"db\_name"` //имя БД

Password [string](http://localhost:6060/pkg/builtin/#string) `toml:"db\_password"` //пароль доступа к БД

User [string](http://localhost:6060/pkg/builtin/#string) `toml:"db\_user"` //пользователя для обращения к бд

Type [string](http://localhost:6060/pkg/builtin/#string) `toml:"db\_type"` //тип бд

Host [string](http://localhost:6060/pkg/builtin/#string) `toml:"db\_host"` //ip сервера бд

Port [string](http://localhost:6060/pkg/builtin/#string) `toml:"db\_port"` //порт для обращения к бд

SetMaxOpenConst [int](http://localhost:6060/pkg/builtin/#int) `toml:"db\_SetMaxOpenConst"` //максимальное количество пустых соединений с бд

SetMaxIdleConst [int](http://localhost:6060/pkg/builtin/#int) `toml:"db\_SetMaxIdleConst"` //максимальное количество соединенияй с бд

}

2.4.9 DeviceLog

DeviceLog описание таблицы, хранящей лог от устройств

type DeviceLog struct {

Time time.Time `json:"time"` //время записи

ID int `json:"id"` //id устройства которое прислало информацию

Text string `json:"text"` //информация о событие

Devices BusyArm `json:"devices"` //информация о девайсе

}

2.4.10 DeviceLogInfo

DeviceLogInfo структура запроса пользователя за данными в бд

type DeviceLogInfo struct {

Devices []BusyArm `json:"devices"` //информация о девайсах

TimeStart time.Time `json:"timeStart"` //время начала отсчета

TimeEnd time.Time `json:"timeEnd"` //время конца отсчета

}

2.4.11 EditCrossInfo

EditCrossInfo информация о пользователе, занявшем перекресток на изменение

type EditCrossInfo struct {

Login string `json:"login"` //логин пользователя

EditFlag bool `json:"editFlag"` //флаг разрешения на редактирование перекрестка

Kick bool `json:"kick"` //флаг закрытия арма у данного пользователя

// contains filtered or unexported fields

}

2.4.12 FillingInfo

FillingInfo информация о запросе

type FillingInfo struct {

User string //пользователь запросивший заполнение таблицы

Status bool //статус выполнения запроса

}

2.4.13 Locations

Locations информация о запрашиваемом регионе и районе карты

type Locations struct {

Region string `json:"region"` //регион

Area []string `json:"area"` //районы

}

Методы:

1. MakeBoxPoint расчет координат для перемещения по карте

func (location \*Locations) MakeBoxPoint() (box BoxPoint, err error)

2.4.14 PassChange

PassChange структура для изменения пароля

type PassChange struct {

OldPW string `json:"oldPW"` //старый пароль

NewPW string `json:"newPW"` //новый пароль

}

Методы:

1. ValidOldNewPW проверка данных полученных от пользователя для изменения своего пароля

func (passChange \*PassChange) ValidOldNewPW(login string) (account \*Account, err error)

### 2.4.15 Permission

Permission структура полномочий содержит ID, команду и описание команды

type Permission struct {

ID int `json:"id"` //ID порядковый номер

Visible bool `json:"visible"` //флаг отображения пользователю

Description string `json:"description"` //описание команды

}

### 2.4.16 PngSettings

PngSettings настройки размеров создаваемой map.png

type PngSettings struct {

SizeX int `json:"sizeX",toml:"png\_sizeX"` //размер по координате X

SizeY int `json:"sizeY",toml:"png\_sizeY"` //размер по координате Y

Z int `json:"z",toml:"png\_Z"` //величина отдаление

}

### 2.4.17 Point

Point координаты точки

type Point struct {

Y, X float64 //Координата Х и Y

}

Методы:

1. GetPoint возврашает значение координаты

func (points \*Point) GetPoint() (y, x float64)

1. SetPoint задать значение координаты

func (points \*Point) SetPoint(y, x float64)

1. StrToFloat преобразует строку, полученную из БД в структуру Point

func (points \*Point) StrToFloat(str string)

### 2.4.18 Privilege

Privilege структура для запросов к БД

type Privilege struct {

Role Role `json:"role"` //информация о роли пользователя

Region string `json:"region"` //регион пользователя

Area []string `json:"area"` //массив районов пользователя

PrivilegeStr string `json:"-"` //строка для декодирования

}

Методы:

1. ConvertToJson из строки в структуру

func (privilege \*Privilege) ConvertToJson() (err error)

1. DisplayInfoForAdmin отображение информации о пользователях для администраторов

func (privilege \*Privilege) DisplayInfoForAdmin(mapContx map[string]string) u.Response

1. ReadFromBD прочитать данные из бд и разобрать

func (privilege \*Privilege) ReadFromBD(login string) error

1. ToSqlStrUpdate запись привилегий в базу

func (privilege \*Privilege) WriteRoleInBD(login string) (err error)

### 2.4.19 RegionInfo

RegionInfo расшифровка региона

type RegionInfo struct {

Num string `json:"num"` //уникальный номер региона

NameRegion string `json:"nameRegion"` //расшифровка номера

}

Методы:

1. SetRegionInfo установить в структуру номер и имя региона по номеру

func (region \*RegionInfo) SetRegionInfo(num string)

### 2.4.20 Role

Role информация о роли

type Role struct {

Name string `json:"name"` //название роли

Perm []int `json:"permissions"` //массив полномочий

}

### 2.4.21 RoleAccess

RoleAccess информация наборах ролей и полномочий

type RoleAccess struct {

Roles []Role `json:"roles"` //массив ролей

Permission []Permission `json:"permissions"` //массив разрешений

Routes []RouteInfo `json:"routes"` //массив маршрутов

}

Методы:

1. ReadRoleAccessFile чтение RoleAccess файла

func (roleAccess \*RoleAccess) ReadRoleAccessFile() (err error)

### 2.4.22 RoleData

RoleData структура, включающая всю информацию о ролях, привелегиях, и маршрутах

type RoleData struct {

MapRoles map[string][]int //роли

MapPermisson map[int]Permission //привилегии

MapRoutes map[string]RouteInfo //маршруты

}

### 2.4.23 RouteInfo

RouteInfo информация о всех расписанных маршрутах

type RouteInfo struct {

ID int `json:"id"` //уникальный номер маршрута

Permission int `json:"permission"` //номер разрешения к которому относится этот маршрут

Path string `json:"path"` //путь (url) обращения к ресурсу

Description string `json:"description"` //описание маршрута

}

### 2.4.24 SelectedData

SelectedData общая структура обмена

type SelectedData struct {

SelectedData map[string]map[string][]CheckData `json:"selected"` //хранилище перекрестков которые были выбраны

PngSettings PngSettings `json:"pngSettings"` //настройки для создания map.png

}

### 2.4.25 ServerLogInfo

ServerLogInfo данные для хранения информации лог файлов

type ServerLogInfo struct {

Type string `json:"type"` //тип лог сообщения

Time string `json:"time"` //врямя когда произошло событие

IP string `json:"IP"` //IP с которого делали запрос

Login string `json:"login"` //логин пользователя который делал запрос

Resource string `json:"resource"` //путь к ресурсу на котором произошло событие

Message string `json:"message"` //расшифровка действия пользователя

}

### 2.4.26 ShortAccount

ShortAccount удобная структура аккаунта для обмена с пользователем

type ShortAccount struct {

Login string `json:"login"` //логин пользователя

WorkTime int `json:"workTime"` //время сеанса пользователя

Password string `json:"password"` //пароль пользователя

Role Role `json:"role"` //роль пользователя

Privilege string `json:"-"` //привелегии (не уходят на верх)

Region RegionInfo `json:"region"` //регион работы пользователя

Area []AreaInfo `json:"area"` //районы работы пользователя

}

Методы:

1. ConvertShortToAcc преобразование информации об аккаунте

func (shortAcc \*ShortAccount) ConvertShortToAcc() (account Account, privilege Privilege)

1. DecodeRequest расшифровываем json данные полученные от пользователя

func (shortAcc \*ShortAccount) DecodeRequest(w http.ResponseWriter, r \*http.Request) error

1. ValidChangePW проверка данных полученных от админа для смены паролей пользователя

func (shortAcc \*ShortAccount) ValidChangePW(role string, region string) (account \*Account, err error)

1. ValidCreate проверка данных полученных от пользователя на создание нового пользователя

func (shortAcc \*ShortAccount) ValidCreate(role string, region string) (err error)

1. ValidDelete проверка данных полученных от пользователя на удаление аккаунта

func (shortAcc \*ShortAccount) ValidDelete(role string, region string) (account \*Account, err error)

### 2.4.27 TLSostInfo

TLSostInfo информация о состояния светофоров

type TLSostInfo struct {

Num int `json:"num"` //номер состояния

Description string `json:"description"` //описание состояния

}

### 2.4.28 Token

Token (JWT) структура токена доступа

type Token struct {

UserID uint //Уникальный ID пользователя

Login string //Уникальный логин пользователя

IP string //IP пользователя

Role string //Роль

Permission []int //Привелегии

Region string //Регион пользователя

jwt.StandardClaims

}

### 2.4.29 TrafficLights

TrafficLights информация о светофоре

type TrafficLights struct {

ID int `json:"ID"` //Уникальный ID светофора

Region RegionInfo `json:"region"` //Регион

Area AreaInfo `json:"area"` //Район

Subarea int `json:"subarea"` //ПодРайон

Idevice int `json:"idevice"` //Реальный номер устройства

Sost TLSostInfo `json:"tlsost"` //Состояние светофора

Description string `json:"description"` //Описание светофора

Points Point `json:"points"` //Координата где находится светофор

}

### 2.4.30 LogFile

LogFile информация о лог файле

type LogFile struct {

flog \*os.File //дескриптор открытого файла

path string //путь да текущего файла лога

date string //текущая дата

}

Методы:

1. Close закрытие лог файла

func (l \*LogFile) Close() error

1. Read чтение лог файла

func (l \*LogFile) Read(p []byte) (n int, err error)

1. Write записать лог файла

func (l \*LogFile) Write(p []byte) (n int, err error)

### 2.4.31 StateResult

StateResult информация о прорке данных

type StateResult struct {

SumResult []string //накоплении информации о ходе проверки

Err error //признак ошибки при проверке

}

Методы:

1. CtrlVerified проверка недельных карт

func CtrlVerified(cross \*agS\_pudge.Cross) (result StateResult)

1. DaySetsVerified проверка суточных карт

func DaySetsVerified(cross \*agS\_pudge.Cross) (result StateResult)

1. MouthSetsVerified проверка месячных(годовых) карт

func MouthSetsVerified(cross \*agS\_pudge.Cross, empty IsEmpty) (result StateResult)

1. TimeUseVerified проверка недельных карт

func TimeUseVerified(cross \*agS\_pudge.Cross) (result StateResult)

### 2.4.32 ArmCommandMessage

ArmCommandMessage ARM информация для отправки на сервер

type ArmCommandMessage struct {

User string //пользователь отправляющий данные (логин)

CommandStr string //данные подготовленные к отправке

Message string //информация о результате передачи данных

}

### 2.4.33 StateMessage

StateMessage state информация для отправки на сервер

type StateMessage struct {

User string //пользователь отправляющий данные (логин)

Info string //короткая информация о state

StateStr string //данные подготовленные к отправке

Message string //информация о результате передачи данных

}

### 2.4.34 TCPConfig

TCPConfig настройки для тсп соединения

type TCPConfig struct {

ServerAddr string `toml:"tcpServerAddress"` //адресс сервера

PortState string `toml:"portState"` //порт для обмена Стате

PortArmComm string `toml:"portArmCommand"` //порт для обмена арм командами

}

## 2.5 Утилиты

1) Message создает map для ответа пользователю

func Message(code int, message string) Response

1. ParserInterface разбирает рефлексией интерфейс в map[string]string

func ParserInterface(in interface{}) (contx map[string]string)

3) Respond формирует ответ пользователю

func SendRespond(c \*gin.Context, resp Response)

4) WriteLogMessage обработчик u.message преобразует сообщение для записи в лог файл

func WriteLogMessage(ip string, url string, data u.Response, info interface{})

## 2.6 Middleware route

Middleware route – промежуточные фильтры, которые работают перед обработчиком запроса.

1) AccessControl проверка разрешен ли пользователя доступ к запрашиваемому ресурсу

var AccessControl = func() gin.HandlerFunc

2) JwtAuth контроль токена для всех прошедших регистрацию и обрашающихся к ресурсу

var JwtAuth = func() gin.HandlerFunc

1. JwtFile упрошенных контроль токена для получения данных из файлового хранилища

var JwtFile = func() gin.HandlerFunc

2.7 Обработчики

### 2.7.1 ActAddAccount

ActAddAccount обработчик запроса добавления (работа с пользователями).

var ActAddAccount = func(c \*gin.Context) {}

Маршрут: /manage/add

### 2.7.2 ActChangePw

ActChangePw обработчик запроса смены пароля для пользователя

var ActChangePw = func(c \*gin.Context) {}

Маршрут: /manage/changepw

### 2.7.3 ActDeleteAccount

ActDeleteAccount обработчик запроса удаления (работа с пользователями)

var ActDeleteAccount = func(c \*gin.Context) {}

Маршрут: /manage/delete

### 2.7.4 ActUpdateAccount

ActUpdateAccount обработчик запроса обновления (работа с пользователями)

var ActUpdateAccount = func(c \*gin.Context) {}

Маршрут: /manage/update

### 2.7.5 BuildCross

BuildCross обработчик собора данных для отображения перекрёстка

var BuildCross = func(c \*gin.Context) {}

Маршрут: /cross

### 

### 2.7.6 BuildMapPage

BuildMapPage собираем данные для авторизованного пользователя

var BuildMapPage = func(c \*gin.Context) {}

Маршрут: /map

### 2.7.7 CheckAllCross

CheckAllCross обработчик проверки всех перекрестков из БД

var CheckAllCross = func(c \*gin.Context) {}

Маршрут: /manage/crossCreator/checkAllCross

### 2.7.8 CheckSelectedDirCross

CheckSelectedDirCross обработчик проверки регионов, районов и перекрестков, выбранных пользователем

var CheckSelectedDirCross = func(c \*gin.Context){}

Маршрут: /manage/crossCreator/checkSelected

### 2.7.9 ControlCheckButton

ControlCheckButton обработчик данных для их проверки

var ControlCheckButton = func(c \*gin.Context) {}

Маршрут: /cross/control/checkButton

### 2.7.10 ControlCloseCross

ControlCloseCross обработчик закрытия перекрестка

var ControlCloseCross = func(c \*gin.Context) {}

Маршрут: /cross/control/close

### 2.7.11 ControlCreateButton

ControlCreateButton обработчик данных для создания перекрестка и отправка на устройство(сервер)

var ControlCreateButton = func(c \*gin.Context) {}

Маршрут: /cross/control/createButton

### 2.7.12 ControlCross

ControlCross обработчик данных для заполнения таблиц управления

var ControlCross = func(c \*gin.Context) {}

Маршрут: /cross/control

### 2.7.13 ControlDeleteButton

ControlDeleteButton обработчик данных для удаления перекрестка

var ControlDeleteButton = func(c \*gin.Context) {}

Маршрут: /cross/control/deleteButton

### 2.7.14 ControlEditableCross

ControlEditableCross обработчик проверки редактирования перекрестка

var ControlEditableCross = func(c \*gin.Context) {}

Маршрут: /cross/control/editable

### 2.7.15 ControlSendButton

ControlSendButton обработчик данных для отправки на устройство(сервер)

var ControlSendButton = func(c \*gin.Context) {}

Маршрут: /cross/control/sendButton

### 2.7.16 ControlTestState

ControlTestState обработчик проверки State

var ControlTestState = func(c \*gin.Context) {}

Маршрут: /manage/stateTest

### 2.7.17 CrossEditFree

CrossEditFree освобождение перекрестков

var CrossEditFree = func(c \*gin.Context) {}

Маршрут: /manage/crossEditControl/free

### 2.7.18 CrossEditInfo

CrossEditInfo сбор информации о занятых перекрестках

var CrossEditInfo = func(c \*gin.Context) {}

Маршрут: /manage/crossEditControl

### 2.7.19 DevCrossInfo

DevCrossInfo обработчик собора данных для отображения перекрёстка (idevice информация)

var DevCrossInfo = func(c \*gin.Context) {}

Маршрут: /cross/dev

### 2.7.20 DispatchControlButtons

DispatchControlButtons обработчик кнопок диспетчерского управления

var DispatchControlButtons = func(c \*gin.Context){}

Маршрут: /cross/DispatchControlButtons

### 2.7.21 DisplayAccInfo

DisplayAccInfo отображение информации об аккаунтах для администрирования

var DisplayAccInfo = func(c \*gin.Context) {}

Маршрут: /manage

### 2.7.22 DisplayDeviceLogFile

DisplayDeviceLogFile обработчик отображения файлов лога устройства

var DisplayDeviceLogFile = func(c \*gin.Context){}

Маршрут: /map/deviceLog

### 2.7.23 DisplayServerLogFile

DisplayServerLogFile обработчик отображения файлов лога сервера

var DisplayServerLogFile = func(c \*gin.Context){}

Маршрут: /manage/serverLog

### 2.7.24 DisplayServerLogInfo

DisplayServerLogInfo обработчик выгрузки содержимого лог файла сервера

var DisplayServerLogInfo = func(c \*gin.Context){}

Маршрут: /manage/serverLog/info

### 2.7.25 LocationButtonMapPage

LocationButtonMapPage обработка запроса на получение новых координат рабочей области

var LocationButtonMapPage = func(c \*gin.Context) {}

Маршрут: /map/locationButton

### 2.7.26 LogDeviceInfo

LogDeviceInfo обработчик запроса на выгрузку информации логов устройства за определенный период

var LogDeviceInfo = func(c \*gin.Context) {}

Маршрут: /map/deviceLog/info

### 2.7.27 LoginAcc

LoginAcc обработчик входа в систему

var LoginAcc = func(c \*gin.Context) {}

Маршрут: /login

### 2.7.28 LoginAccOut

LoginAccOut обработчик выхода из системы

var LoginAccOut = func(c \*gin.Context) {}

Маршрут: /map/logOut

### 2.7.29 MainCrossCreator

MainCrossCreator сборка информации для странички создания каталогов перекрестков

var MainCrossCreator = func(c \*gin.Context) {}

Маршрут: /manage/crossCreator

### 2.7.30 MakeSelectedDirCross

MakeSelectedDirCross обработчик проверки регионов, районов и перекрестков, выбранных пользователем

var MakeSelectedDirCross = func(c \*gin.Context){}

Маршрут: /manage/crossCreator/makeSelected

### 2.7.31 UpdateMapPage

UpdateMapPage обновление информации о попавших в область светофорах

var UpdateMapPage = func(c \*gin.Context) {}

Маршрут: /map/update

### 2.7.32 LicenseInfo

LicenseInfo обработчик сборки начальной информации

var LicenseInfo = func(c \*gin.Context)

Маршрут: /license

### 2.7.32 LicenseInfo

LicenseInfo обработчик сборки начальной информации

var LicenseInfo = func(c \*gin.Context)

Маршрут: /license

### 2.7.33 LicenseNewKey

LicenseNewKey обработчик сохранения нового токена

var LicenseNewKey  = func(c \*gin.Context)

Маршрут: /license/newToken

### 2.7.34 TechSupp

TechSupp обработчик отправления сообщения в тех поддержку

var TechSupp = func(c \*gin.Context)

Маршрут: /techSupp/send

### 2.7.34 ChatEngine

ChatEngine обработчик вебсокета для работы с чатом

var ChatEngine = func(c \*gin.Context)

Маршрут: /chatW