

6.RPC服务开发

1.安装rpc测试工具

1.1 安装

1.2 使用

2.user模块

2.1 user.proto

2.2 生成RPC代码

2.3 把默认etcd注册中心更换为consul

2.4 model代码生成

2.5 数据库配置

2.6 日志配置

2.7 新增用户接口

2.8 删除用户接口

2.9 用户登录接口

2.10 更新用户信息接口

2.11 获取用户列表

3.system模块

3.1 system.proto

3.1 获取系统配置详情接口

3.3 修改系统配置接口

3.4 获取密钥配置接口

3.5 修改密钥配置接口

4.group模块

4.1 group.proto

4.2 增加群组

4.3 删除群组

4.4 群组详情

4.5 群组列表

5.task模块

5.1 task.proto

5.2 增加task接口

1.安装rpc测试工具

官网链接: <https://github.com/fullstorydev/grpcui>

1.1安装

```
1 go install github.com/fullstorydev/grpcui/cmd/grpcui@latest
```

1.2 使用

```
1 grpcui -plaintext localhost:8080
```

2.user模块

2.1 user.proto

```
1  syntax = "proto3";
2
3  package user;
4
5  option go_package = "./user";
6
7  // Login
8  message loginRequest {
9      string username = 1;
10     string password = 2;
11 }
12 message loginResponse {
13     string message = 1;
14     int64 userID = 2;
15     string username = 3;
16 }
17 // Login
18
19 // UpdateUserInfo
20 message updateUserInfoRequest {
21     string password = 1;
22     string realName = 2;
23     string phoneNumber = 3;
24     int64 departmentId = 4;
25     string email = 5;
26     string jenkinsAccount = 6;
27     string jenkinsPassword = 7;
28     string avatar = 8;
29     int64 id = 9;
30 }
31 message updateUserInfoResponse {
32     string message = 1;
33 }
34 // UserList
35 message userInfo {
36     int64 userID = 1;
37     string username = 2;
38 }
39 message userListRequest {
40 }
41 message userListResponse {
42     repeated userInfo userList = 1;
43 }
44 // UserList
45
```

```

46 // UserDetails
47 message userDetailsRequest {
48     string username = 1;
49 }
50 message userDetailsResponse {
51     int64 userID = 1;
52     string username = 2;
53     string realName = 3;
54     string phoneNumber = 4;
55     string departmentName = 5;
56     string email = 6;
57     string jenkinsAccount = 7;
58     string isAdmin = 8;
59 }
60 // UserDetails
61 // AddUser
62 message addUserRequest {
63     string username = 1;
64     string password = 2;
65     string realName = 4;
66     string phoneNumber = 5;
67     int64 departmentId = 6;
68     string email = 7;
69     string jenkinsAccount = 8;
70     string jenkinsPassword = 9;
71     string avatar = 10;
72     string isAdmin = 11;
73 }
74 message addUserResponse {
75     string message = 1;
76     int64 userID = 2;
77     string username = 3;
78 }
79
80 // AddUser
81
82 // DelUser
83 message delUserRequest {
84     string username = 1;
85 }
86 message delUserResponse {
87     string message = 1;
88     int64 userID = 2;
89     string username = 3;
90 }
91 // DelUser
92
93 // Services

```

```

94  service User {
95      rpc login(loginRequest) returns (loginResponse);
96      rpc addUser(addUserRequest) returns (addUserResponse);
97      rpc updateUserInfo(updateUserInfoRequest) returns (updateUserInfoResp
98  onse);
99      rpc userList(userListRequest) returns (userListResponse);
100     rpc userDetail(userDetailRequest) returns (userDetailResponse);
101     rpc delUser(delUserRequest) returns (delUserResponse);
102 }
    // Services

```

2.2 生成RPC代码

```

1  goctl rpc protoc user.proto --go_out=./ --go-grpc_out=./ --zrpc_out=.

```

安装依赖

```

1  go mod tidy

```

2.3 把默认etcd注册中心更换为consul

需要安装docker和docker-compose，并用docker启动一个consul服务 安装步骤见下面博客链接
<https://www.liwenzhou.com/posts/Go/consul/>

修改user.yaml配置文件

```
1 Name: user.rpc
2 Mode: dev
3 ListenOn: 0.0.0.0:8080
4 # Etcd:
5 #   Hosts:
6 #     - 127.0.0.1:2379
7 #   Key: user.rpc
8 # 使用consul来作为注册中心
9 Consul:
10   Host: 127.0.0.1:8500
11   Key: consul-user.rpc
```

修改 config.go文件

```
1 package config
2
3 import ("github.com/zeromicro/go-zero/zrpc"
4         "github.com/zeromicro/zero-contrib/zrpc/registry/consul"
5        )
6
7 type Config struct {
8     zrpc.RpcServerConf
9
10    // 引入consul
11    Consul consul.Conf
12 }
13
```

user.go文件中匿名导入consul，并进行服务注册

```
1  package main
2
3  import (
4      "flag"
5      "fmt"
6
7      "chatops-server/internal/user/rpc/internal/config"
8      "chatops-server/internal/user/rpc/internal/server"
9      "chatops-server/internal/user/rpc/internal/svc"
10     "chatops-server/internal/user/rpc/user"
11
12     "github.com/zeromicro/go-zero/core/conf"
13     "github.com/zeromicro/go-zero/core/service"
14     "github.com/zeromicro/go-zero/zrpc"
15     "github.com/zeromicro/zero-contrib/zrpc/registry/consul"
16     _ "github.com/zeromicro/zero-contrib/zrpc/registry/consul"
17     "google.golang.org/grpc"
18     "google.golang.org/grpc/reflection"
19 )
20
21 var configFile = flag.String("f", "etc/user.yaml", "the config file")
22
23 func main() {
24     flag.Parse()
25
26     var c config.Config
27     conf.MustLoad(*configFile, &c)
28     ctx := svc.NewServiceContext(c)
29
30     s := zrpc.MustNewServer(c.RpcServerConf, func(grpcServer *grpc.Server) {
31         user.RegisterUserServer(grpcServer, server.NewUserServer(ctx))
32
33         if c.Mode == service.DevMode || c.Mode == service.TestMode {
34             reflection.Register(grpcServer)
35         }
36     })
37
38     // 将服务注册到consul
39     consul.RegisterService(c.ListenOn, c.Consul)
40     defer s.Stop()
41
42     fmt.Printf("Starting rpc server at %s...\n", c.ListenOn)
43     s.Start()
44 }
```

2.4 model代码生成

生成model层代码（不带缓存）

```
1 goctl model mysql datasource -url="chatops:Chatops_123@tcp(116.204.88.174:3306)/chatops_server" -table="user" -dir=.
2 goctl model mysql datasource -url="chatops:Chatops_123@tcp(116.204.88.174:3306)/chatops_server" -table="department" -dir=.
```

2.5 数据库配置

user.yaml添加mysql配置（）

```
1 Name: user.rpc
2 Mode: dev
3 ListenOn: 0.0.0.0:8080
4 # Etcd:
5 #   Hosts:
6 #     - 127.0.0.1:2379
7 #   Key: user.rpc
8 # 使用consul来作为注册中心
9 Consul:
10   Host: 127.0.0.1:8500
11   Key: consul-user.rpc
12 Mysql:
13   DataSource: chatops:Chatops_123@tcp(116.204.88.174:3306)/chatops_server?
    charset=utf8mb4&parseTime=true&loc=Asia%2FShanghai
```

config.go 添加配置


```
1  package config
2
3  import (
4      "github.com/zeromicro/go-zero/zrpc"
5      "github.com/zeromicro/zero-contrib/zrpc/registry/consul"
6  )
7
8  type Config struct {
9      zrpc.RpcServerConf
10
11      // 引入consul
12      Consul consul.Conf
13
14      // mysql配置
15      Mysql struct {
16          DataSource string
17      }
18  }
19
```

servicecontext.go 添加配置

```
1 package svc
2
3 import (
4     "chatops-server/internal/user/model"
5     "chatops-server/internal/user/rpc/internal/config"
6
7     "github.com/zeromicro/go-zero/core/stores/sqlx"
8 )
9
10 type ServiceContext struct {
11     Config      config.Config
12     UserModel   model.UserModel
13     Conn        sqlx.SqlConn
14 }
15
16 func NewServiceContext(c config.Config) *ServiceContext {
17     conn := sqlx.NewMysql(c.Mysql.DataSource)
18     return &ServiceContext{
19         Config:      c,
20         UserModel:   model.NewUserModel(conn),
21         Conn:        conn,
22     }
23 }
24
```

2.6 日志配置

user.yaml 增加配置

```
1 Name: user.rpc
2 Mode: dev
3 ListenOn: 0.0.0.0:10001
4 # Etcd:
5 #   Hosts:
6 #     - 127.0.0.1:2379
7 #   Key: user.rpc
8 # 使用consul来作为注册中心
9 Consul:
10   Host: 127.0.0.1:8500
11   Key: consul-user.rpc
12 Mysql:
13   DataSource: chatops:Chatops_123@tcp(116.204.88.174:3306)/chatops_server?
    charset=utf8mb4&parseTime=true&loc=Asia%2FShanghai
14 # 日志配置
15 Log:
16   ServiceName: "user-rpc"
17   Mode: file
18   Encoding: plain
19   Path: /var/log/chatops-server
20   Level: debug
21   Stat: true
```

2.7 新增用户接口

```
1  package logic
2
3  import (
4      "context"
5
6      "chatops-server/internal/user/model"
7      "chatops-server/internal/user/rpc/internal/svc"
8      "chatops-server/internal/user/rpc/user"
9      md5secret "chatops-server/pkg/md5Secret"
10
11      "github.com/zeromicro/go-zero/core/logx"
12 )
13
14 type AddUserLogic struct {
15     ctx      context.Context
16     svcCtx *svc.ServiceContext
17     logx.Logger
18 }
19
20 func NewAddUserLogic(ctx context.Context, svcCtx *svc.ServiceContext) *AddUserLogic {
21     return &AddUserLogic{
22         ctx:      ctx,
23         svcCtx:   svcCtx,
24         Logger:   logx.WithContext(ctx),
25     }
26 }
27
28 func (l *AddUserLogic) AddUser(in *user.AddUserRequest) (*user.AddUserResponse, error) {
29     // 参数校验在apis层实现
30     // 加密密码
31     secret := l.svcCtx.Config.Secret
32     password := md5secret.MD5Secret(in.Password, secret)
33     // 将新增用户插入user表
34     userInfo := &model.User{
35         Username:      in.Username,
36         Password:      password,
37         RealName:      in.RealName,
38         PhoneNumber:   in.PhoneNumber,
39         DepartmentId:  in.DepartmentId,
40         Email:         in.Email,
41         JenkinsAccount: in.JenkinsAccount,
42         JenkinsPassword: in.JenkinsPassword, // md5加密不可逆, 该密码还要用于
        登录Jenkins, 所以不能加密
    }
```

```

43         Avatar:         in.Avatar,
44         IsAdmin:        in.IsAdmin,
45     }
46     result, err := l.svcCtx.UserModel.Insert(l.ctx, userInfo)
47     if err != nil {
48         logx.Errorw("user.rpc.UserModel.Insert failed", logx.LogField{Key
49 : "err", Value: err})
50         return nil, err
51     }
52     logx.Infow("user.rpc.UserModel.Insert success", logx.LogField{Key: "us
53 ername", Value: in.Username})
54     // 返回响应
55     userID, _ := result.LastInsertId()
56     return &user.AddUserResponse{
57         Message: "ok",
58         UserID:   userID,
59         Username: in.Username,
60     }, nil

```

2.8 删除用户接口

```
1  package logic
2
3  import (
4      "context"
5      "errors"
6
7      "chatops-server/internal/user/rpc/internal/svc"
8      "chatops-server/internal/user/rpc/user"
9
10     "github.com/zeromicro/go-zero/core/logx"
11     "github.com/zeromicro/go-zero/core/stores/sqlx"
12 )
13
14 type DelUserLogic struct {
15     ctx      context.Context
16     svcCtx   *svc.ServiceContext
17     logx.Logger
18 }
19
20 func NewDelUserLogic(ctx context.Context, svcCtx *svc.ServiceContext) *DelUserLogic {
21     return &DelUserLogic{
22         ctx:      ctx,
23         svcCtx:   svcCtx,
24         Logger:   logx.WithContext(ctx),
25     }
26 }
27
28 func (l *DelUserLogic) DelUser(in *user.DelUserRequest) (*user.DelUserResponse, error) {
29     // 先根据username来获取userID
30     userInfo, err := l.svcCtx.UserModel.FindOneByUsername(l.ctx, in.Username)
31     if errors.Is(err, sqlx.ErrNotFound) {
32         return nil, errors.New("用户不存在")
33     }
34     if err != nil {
35         logx.Errorw("user.rpc.UserModel.FindOneByUsername failed", logx.LogField{Key: "err", Value: err})
36         return nil, err
37     }
38     // 再根据用户userID删除用户
39     err = l.svcCtx.UserModel.Delete(l.ctx, userInfo.Id)
40     if err != nil {
41
```

```
42         logx.Errorw("user.rpc.UserModel.Delete failed", logx.LogField{Key
43 : "err", Value: err})
44         return nil, err
45     }
46     logx.Infoln("user.rpc.UserModel.Delete success")
47     // 返回响应
48     return &user.DelUserResponse{
49         Message: "ok",
50         UserID:  userInfo.Id,
51         Username: in.Username,
52     }, nil
}
```

2.9 用户登录接口

```
1  package logic
2
3  import (
4      "context"
5      "errors"
6
7      "chatops-server/internal/user/rpc/internal/svc"
8      "chatops-server/internal/user/rpc/user"
9      md5secret "chatops-server/pkg/md5Secret"
10
11      "github.com/zeromicro/go-zero/core/logx"
12      "github.com/zeromicro/go-zero/core/stores/sqlx"
13 )
14
15 type LoginLogic struct {
16     ctx      context.Context
17     svcCtx *svc.ServiceContext
18     logx.Logger
19 }
20
21 func NewLoginLogic(ctx context.Context, svcCtx *svc.ServiceContext) *LoginLogic {
22     return &LoginLogic{
23         ctx:      ctx,
24         svcCtx: svcCtx,
25         Logger: logx.WithContext(ctx),
26     }
27 }
28
29 func (l *LoginLogic) Login(in *user.LoginRequest) (*user.LoginResponse, error) {
30     // 根据username查询password 和 userID
31     userInfo, err := l.svcCtx.UserModel.FindOneByUsername(l.ctx, in.Username)
32     if errors.Is(err, sqlx.ErrNotFound) {
33         return nil, errors.New("用户不存在")
34     }
35     if err != nil {
36         logx.Errorw("user.rpc.UserModel.FindOneByUsername failed", logx.LogField{Key: "err", Value: err})
37         return nil, err
38     }
39     // 判断密码是否正确
40     // 用户传入的password加密后和数据库查出来的做比较
41     secret := l.svcCtx.Config.Secret
```



```
42     password := md5secret.MD5Secret(in.Password, secret)
43     if password != userInfo.Password {
44         return nil, errors.New("用户名或密码错误")
45     }
46     // 返回响应
47     return &user.LoginResponse{
48         Message: "ok",
49         UserID:  userInfo.Id,
50         Username: userInfo.Username,
51     }, nil
52 }
53
```

2.10 更新用户信息接口

```
1  package logic
2
3  import (
4      "context"
5
6      "chatops-server/internal/user/model"
7      "chatops-server/internal/user/rpc/internal/svc"
8      "chatops-server/internal/user/rpc/user"
9      md5secret "chatops-server/pkg/md5Secret"
10
11      "github.com/zeromicro/go-zero/core/logx"
12 )
13
14 type UpdateUserInfoLogic struct {
15     ctx      context.Context
16     svcCtx *svc.ServiceContext
17     logx.Logger
18 }
19
20 func NewUpdateUserInfoLogic(ctx context.Context, svcCtx *svc.ServiceContext) *UpdateUserInfoLogic {
21     return &UpdateUserInfoLogic{
22         ctx:      ctx,
23         svcCtx:   svcCtx,
24         Logger:   logx.WithContext(ctx),
25     }
26 }
27
28 func (l *UpdateUserInfoLogic) UpdateUserInfo(in *user.UpdateUserInfoRequest) (*user.UpdateUserInfoResponse, error) {
29     // 更新用户信息
30     // 新密码加密
31     newPassword := md5secret.MD5Secret(in.Password, l.svcCtx.Config.Secret)
32
33     userInfoNew := &model.User{
34         Id:           in.Id,
35         Password:     newPassword,
36         RealName:     in.RealName,
37         PhoneNumber:  in.PhoneNumber,
38         DepartmentId: in.DepartmentId,
39         Email:        in.Email,
40         JenkinsAccount: in.JenkinsAccount,
41         JenkinsPassword: in.JenkinsPassword,
42     }
43     err := l.svcCtx.UserModel.Update(l.ctx, userInfoNew)
```

```
43 // 返回响应
44 if err != nil {
45     logx.Errorw("user.rpc UserModel.Update failed", logx.LogField{Key
46 : "err", Value: err})
47     return nil, err
48 }
49 return &user.UpdateUserInfoResponse{
50     Message: "ok",
51 }, nil
52 }
```

2.11 获取用户列表

由于自动生成的model代码没有对应的查询语句，需要执行原生sql

修改servicecontext.go 配置

```
1  package svc
2
3  import (
4      "chatops-server/internal/user/model"
5      "chatops-server/internal/user/rpc/internal/config"
6
7      "github.com/zeromicro/go-zero/core/stores/sqlx"
8  )
9
10 type ServiceContext struct {
11     Config      config.Config
12     UserModel   model.UserModel
13     DepartmentModel model.DepartmentModel
14     Conn        sqlx.SqlConn
15 }
16
17 func NewServiceContext(c config.Config) *ServiceContext {
18     conn := sqlx.NewMysql(c.Mysql.DataSource)
19
20     return &ServiceContext{
21         Config:      c,
22         UserModel:   model.NewUserModel(conn),
23         DepartmentModel: model.NewDepartmentModel(conn),
24         Conn:        conn,
25     }
26 }
27
```

```
1  package logic
2
3  import (
4      "context"
5      "errors"
6
7      "chatops-server/internal/user/rpc/internal/svc"
8      "chatops-server/internal/user/rpc/user"
9
10     "github.com/zeromicro/go-zero/core/logx"
11     "github.com/zeromicro/go-zero/core/stores/sqlx"
12 )
13
14 type UserListLogic struct {
15     ctx      context.Context
16     svcCtx   *svc.ServiceContext
17     logx.Logger
18 }
19
20 func NewUserListLogic(ctx context.Context, svcCtx *svc.ServiceContext) *UserListLogic {
21     return &UserListLogic{
22         ctx:      ctx,
23         svcCtx:   svcCtx,
24         Logger:   logx.WithContext(ctx),
25     }
26 }
27
28 type userDbInfo struct {
29     ID      int64 `db:"id"`
30     Username string `db:"username"`
31 }
32
33 func (l *UserListLogic) UserList(in *user.UserListRequest) (*user.UserListResponse, error) {
34     // 查询用户信息
35     // 由于自动生成的model代码没有对应的查询语句，需要执行原生sql
36     var userListFromDb []userDbInfo
37     querySQL := `select id, username from user;`
38     err := l.svcCtx.Conn.QueryRowsCtx(l.ctx, &userListFromDb, querySQL)
39     if errors.Is(err, sqlx.ErrNotFound) {
40         logx.Errorw("user.rpc.Conn.Exec failed", logx.LogField{Key: "err", Value: sqlx.ErrNotFound.Error()})
41         return nil, errors.New("查询用户列表失败")
42     }
43 }
```

```

43     if err != nil {
44         logx.Errorw("user.rpc.Conn.Exec failed", logx.LogField{Key: "err"
, Value: err})
45         return nil, err
46     }
47     logx.Infoln("user.rpc.Conn.Exec success")
48     var userList []*user.UserInfo
49     for _, userFromDb := range userListFromDb {
50         userResponse := user.UserInfo{
51             UserID:    userFromDb.ID,
52             Username:  userFromDb.Username,
53         }
54         userList = append(userList, &userResponse)
55     }
56     return &user.UserListResponse{
57         UserList: userList,
58     }, nil
59 }
60

```

3.system模块

3.1 system.proto

```
1  syntax = "proto3";
2
3  package system;
4
5  option go_package = "./system";
6
7  // SystemDetailInfo
8  message systemDetailInfoRequest {
9  }
10 message systemDetailInfoResponse {
11     int64 detectDuration = 1; // 探测钉钉群组时间间隔
12     int64 historyDuration = 2; // task历史保留时间
13 }
14 // SystemDetailInfo
15
16 // UpdateSystemInfo
17 message updateSystemInfoRequest {
18     int64 detectDuration = 1; // 探测钉钉群组时间间隔
19     int64 historyDuration = 2; // task历史保留时间
20 }
21 message updateSystemInfoResponse {
22     string message = 1;
23 }
24 // UpdateSystemInfo
25
26 // SecretDetailInfo
27 message secretDetailInfoRequest {
28 }
29 message secretDetailInfoResponse {
30     string message = 1;
31     string agentID = 2;
32     string appKey = 3;
33     string appSecret = 4;
34 }
35 // SecretDetailInfo
36
37 // UpdateSecretInfo
38 message updateSecretInfoRequest {
39     string agentID = 1;
40     string appKey = 2;
41     string appSecret = 3;
42     string name = 4;
43 }
44 message updateSecretInfoResponse {
45     string message = 1;
```

```

46  }
47  // UpdateSecretInfo
48
49  // Services
50  service System {
51      rpc systemDetailInfo(systemDetailInfoRequest) returns(systemDetailInfo
52      Response);
53      rpc updateSystemInfo(updateSystemInfoRequest) returns(updateSystemInfo
54      Response);
55      rpc secretDetailInfo(secretDetailInfoRequest) returns(secretDetailInfo
56      Response);
57      rpc updateSecretInfo(updateSecretInfoRequest) returns(updateSecretInfo
58      Response);
59  }
60  // Services

```

rpc代码生成和model代码生成和user模块一致，数据库/日志/注册中心等也一致，参考user层进行配置即可，下面的其他模块不再进行赘述

3.1 获取系统配置详情接口


```
1 package logic
2
3 import (
4     "context"
5     "errors"
6
7     "chatops-server/internal/system/rpc/internal/svc"
8     "chatops-server/internal/system/rpc/system"
9
10    "github.com/zeromicro/go-zero/core/logx"
11    "github.com/zeromicro/go-zero/core/stores/sqlx"
12 )
13
14 type SystemDetailInfoLogic struct {
15     ctx      context.Context
16     svcCtx *svc.ServiceContext
17     logx.Logger
18 }
19
20 func NewSystemDetailInfoLogic(ctx context.Context, svcCtx *svc.ServiceContext) *SystemDetailInfoLogic {
21     return &SystemDetailInfoLogic{
22         ctx:      ctx,
23         svcCtx: svcCtx,
24         Logger: logx.WithContext(ctx),
25     }
26 }
27
28 func (l *SystemDetailInfoLogic) SystemDetailInfo(in *system.SystemDetailInfoRequest) (*system.SystemDetailInfoResponse, error) {
29     SystemInfo, err := l.svcCtx.SystemModel.FindOne(l.ctx, 1)
30     if errors.Is(err, sqlx.ErrNotFound) {
31         logx.Errorw("system.rpc.SystemModel.FindOne failed", logx.LogField{Key: "err", Value: err})
32         return nil, errors.New("未查询到系统配置")
33     }
34     logx.Infow("system.rpc.Conn.SystemModel.FindOne success")
35     // 返回响应
36     return &system.SystemDetailInfoResponse{
37         DetectDuration: SystemInfo.DetectDuration,
38         HistoryDurantion: SystemInfo.HistoryDurantion,
39     }, nil
40 }
41
```

3.3 修改系统配置接口

```
1 package logic
2
3 import (
4     "context"
5
6     "chatops-server/internal/system/model"
7     "chatops-server/internal/system/rpc/internal/svc"
8     "chatops-server/internal/system/rpc/system"
9
10    "github.com/zeromicro/go-zero/core/logx"
11 )
12
13 type UpdateSystemInfoLogic struct {
14     ctx      context.Context
15     svcCtx *svc.ServiceContext
16     logx.Logger
17 }
18
19 func NewUpdateSystemInfoLogic(ctx context.Context, svcCtx *svc.ServiceContext) *UpdateSystemInfoLogic {
20     return &UpdateSystemInfoLogic{
21         ctx:      ctx,
22         svcCtx: svcCtx,
23         Logger: logx.WithContext(ctx),
24     }
25 }
26
27 func (l *UpdateSystemInfoLogic) UpdateSystemInfo(in *system.UpdateSystemInfoRequest) (*system.UpdateSystemInfoResponse, error) {
28     // 更新系统设置
29     updateSystemInfo := model.System{
30         Id: 1,
31         DetectDuration: in.DetectDuration,
32         HistoryDuration: in.HistoryDuration,
33     }
34     err := l.svcCtx.SystemModel.Update(l.ctx, &updateSystemInfo)
35     if err != nil {
36         logx.Errorw("system.rpc.SystemModel.Update failed", logx.LogField{
37             Key: "err", Value: err})
38         return nil, err
39     }
40     logx.Infow("system.rpc.Conn.SystemModel.Update success")
41     // 返回响应
42     return &system.UpdateSystemInfoResponse{
43         Message: "ok",
44     }, nil
45 }
```

```
43     }, nil  
44 }  
45 }
```

3.4 获取密钥配置接口

```
1 package logic
2
3 import (
4     "context"
5     "errors"
6
7     "chatops-server/internal/system/rpc/internal/svc"
8     "chatops-server/internal/system/rpc/system"
9
10    "github.com/zeromicro/go-zero/core/logx"
11    "github.com/zeromicro/go-zero/core/stores/sqlx"
12 )
13
14 type SecretDetailInfoLogic struct {
15     ctx      context.Context
16     svcCtx *svc.ServiceContext
17     logx.Logger
18 }
19
20 func NewSecretDetailInfoLogic(ctx context.Context, svcCtx *svc.ServiceContext) *SecretDetailInfoLogic {
21     return &SecretDetailInfoLogic{
22         ctx:      ctx,
23         svcCtx: svcCtx,
24         Logger: logx.WithContext(ctx),
25     }
26 }
27
28 func (l *SecretDetailInfoLogic) SecretDetailInfo(in *system.SecretDetailInfoRequest) (*system.SecretDetailInfoResponse, error) {
29     // 从数据库获取机器人的密钥信息
30     robotInfo, err := l.svcCtx.RobotModel.FindOne(l.ctx, 1)
31     if errors.Is(err, sqlx.ErrNotFound) {
32         logx.Errorw("system.rpc.RobotModel.FindOne failed", logx.LogField{
33             Key: "err", Value: err})
34         return nil, errors.New("未查找到机器人信息")
35     }
36     logx.Info("system.rpc.RobotModel.FindOne success")
37     // 返回响应
38     return &system.SecretDetailInfoResponse{
39         Message:    "ok",
40         AgentID:    robotInfo.AgentId,
41         AppKey:     robotInfo.Appkey,
42         AppSecret:  robotInfo.Appsecret,
43     }, nil
44 }
```

```
43 }  
44
```

3.5 修改密钥配置接口

```
1  package logic
2
3  import (
4      "context"
5
6      "chatops-server/internal/system/model"
7      "chatops-server/internal/system/rpc/internal/svc"
8      "chatops-server/internal/system/rpc/system"
9
10     "github.com/zeromicro/go-zero/core/logx"
11 )
12
13 type UpdateSecretInfoLogic struct {
14     ctx      context.Context
15     svcCtx *svc.ServiceContext
16     logx.Logger
17 }
18
19 func NewUpdateSecretInfoLogic(ctx context.Context, svcCtx *svc.ServiceContext) *UpdateSecretInfoLogic {
20     return &UpdateSecretInfoLogic{
21         ctx:      ctx,
22         svcCtx: svcCtx,
23         Logger: logx.WithContext(ctx),
24     }
25 }
26
27 func (l *UpdateSecretInfoLogic) UpdateSecretInfo(in *system.UpdateSecretInfoRequest) (*system.UpdateSecretInfoResponse, error) {
28     // 更新密钥设置
29     updateSecretInfo := model.Robot{
30         Id:      1,
31         Name:     in.Name,
32         Appkey:   in.AppKey,
33         Appsecret: in.AppSecret,
34         AgentId:  in.AgentID,
35     }
36     err := l.svcCtx.RobotModel.Update(l.ctx, &updateSecretInfo)
37     if err != nil {
38         logx.Errorw("system.rpc.RobotModel.Update failed", logx.LogField{Key: "err", Value: err})
39         return nil, err
40     }
41     logx.Infow("system.rpc.Conn.RobotModel.Update success")
42     return &system.UpdateSecretInfoResponse{
```

```
43         Message: "ok",  
44     }, nil  
45 }  
46
```

4.group模块

4.1 group.proto


```
1  syntax = "proto3";
2
3  package group;
4
5  option go_package = "./group";
6
7
8  // AddGroup
9  message addGroupRequest {
10     string name = 1;
11     string conversionID = 2;
12 }
13 message addGroupResponse {
14     string message = 1;
15 }
16 // AddGroup
17
18 // DeleteGroup
19 message deleteGroupRequest {
20     string name = 1;
21 }
22 message deleteGroupResponse {
23     string message = 1;
24 }
25 // DeleteGroup
26
27 // GroupList
28 message groupInfo {
29     string name = 1;
30     string conversionID = 2;
31 }
32 message groupListRequest {
33 }
34 message groupListResponse {
35     string message = 1;
36     repeated groupInfo groupList = 2;
37 }
38 // GroupList
39
40 // GroupDetail
41 message groupDetailRequest {
42     string name = 1;
43 }
44 message groupDetailResponse {
45     string message = 1;
```

```
46     string name = 2;
47     string conversionID = 3;
48 }
49 // GroupDetail
50
51 // Services
52 service Group {
53     rpc AddGroup(addGroupRequest) returns(addGroupResponse);
54     rpc DeleteGroup(deleteGroupRequest) returns(deleteGroupResponse);
55     rpc GroupList(groupListRequest) returns(groupListResponse);
56     rpc GroupDetail(groupDetailRequest) returns(groupDetailResponse);
57 }
58 // Services
59
```

4.2 增加群组

```
1 package logic
2
3 import (
4     "context"
5     "errors"
6
7     "chatops-server/internal/group/model"
8     "chatops-server/internal/group/rpc/group"
9     "chatops-server/internal/group/rpc/internal/svc"
10
11     "github.com/zeromicro/go-zero/core/logx"
12 )
13
14 type AddGroupLogic struct {
15     ctx      context.Context
16     svcCtx   *svc.ServiceContext
17     logx.Logger
18 }
19
20 func NewAddGroupLogic(ctx context.Context, svcCtx *svc.ServiceContext) *AddGroupLogic {
21     return &AddGroupLogic{
22         ctx:      ctx,
23         svcCtx:   svcCtx,
24         Logger:   logx.WithContext(ctx),
25     }
26 }
27
28 func (l *AddGroupLogic) AddGroup(in *group.AddGroupRequest) (*group.AddGroupResponse, error) {
29     // 往数据库中插入新数据
30     newGroup := model.Groups{
31         Name:      in.Name,
32         ConversionId: in.ConversionID,
33     }
34     _, err := l.svcCtx.GroupModel.Insert(l.ctx, &newGroup)
35     if err != nil {
36         logx.Errorw("group.rpc.GroupModel.Insert failed", logx.LogField{Key: "err", Value: err})
37         return nil, errors.New("新增group失败")
38     }
39     // 返回响应
40
41     return &group.AddGroupResponse{
42         Message: "ok",
43     }, nil
44 }
```

```
43     }, nil
44 }
45 }
```

4.3 删除群组

```
1 package logic
2
3 import (
4     "context"
5     "errors"
6
7     "chatops-server/internal/group/rpc/group"
8     "chatops-server/internal/group/rpc/internal/svc"
9
10    "github.com/zeromicro/go-zero/core/logx"
11    "github.com/zeromicro/go-zero/core/stores/sqlx"
12 )
13
14 type DeleteGroupLogic struct {
15     ctx      context.Context
16     svcCtx   *svc.ServiceContext
17     logx.Logger
18 }
19
20 func NewDeleteGroupLogic(ctx context.Context, svcCtx *svc.ServiceContext)
    *DeleteGroupLogic {
21     return &DeleteGroupLogic{
22         ctx:      ctx,
23         svcCtx:   svcCtx,
24         Logger:   logx.WithContext(ctx),
25     }
26 }
27
28 func (l *DeleteGroupLogic) DeleteGroup(in *group.DeleteGroupRequest) (*gro
    up.DeleteGroupResponse, error) {
29     // 根据name查出要删除的id
30     delGroupInfo, err := l.svcCtx.GroupModel.FindOneByName(l.ctx, in.Name)
31     if errors.Is(err, sqlx.ErrNotFound) {
32         logx.Errorw("group.rpc.GroupModel.FindOneByName failed", logx.LogF
            ield{Key: "err", Value: err})
33         return nil, errors.New("未查询到指定组")
34     }
35     // 根据id删除对应的组信息
36     err = l.svcCtx.GroupModel.Delete(l.ctx, delGroupInfo.Id)
37     if err != nil {
38         logx.Errorw("group.rpc.GroupModel.Delete failed", logx.LogField{Ke
            y: "err", Value: err})
39         return nil, errors.New("删除组失败")
40     }
41     logx.Info("group.rpc.GroupModel.Delete success")

```

```
42 // 返回响应
43
44 return &group.DeleteGroupResponse{
45     Message: "ok",
46 }, nil
47 }
48
```

4.4 群组详情

```
1 package logic
2
3 import (
4     "context"
5     "errors"
6
7     "chatops-server/internal/group/rpc/group"
8     "chatops-server/internal/group/rpc/internal/svc"
9
10    "github.com/zeromicro/go-zero/core/logx"
11    "github.com/zeromicro/go-zero/core/stores/sqlx"
12 )
13
14 type GroupDetailLogic struct {
15     ctx    context.Context
16     svcCtx *svc.ServiceContext
17     logx.Logger
18 }
19
20 func NewGroupDetailLogic(ctx context.Context, svcCtx *svc.ServiceContext)
    *GroupDetailLogic {
21     return &GroupDetailLogic{
22         ctx:    ctx,
23         svcCtx: svcCtx,
24         Logger: logx.WithContext(ctx),
25     }
26 }
27
28 func (l *GroupDetailLogic) GroupDetail(in *group.GroupDetailRequest) (*gro
    up.GroupDetailResponse, error) {
29     // 根据name来查询group详情
30     groupInfo, err := l.svcCtx.GroupModel.FindOneByName(l.ctx, in.Name)
31     if errors.Is(err, sqlx.ErrNotFound) {
32         logx.Errorw("group.rpc.GroupModel.FindOneByName failed", logx.LogF
            ield{Key: "err", Value: err})
33         return nil, errors.New("未查询到指定组")
34     }
35     // 返回响应
36     return &group.GroupDetailResponse{
37         Message:    "ok",
38         Name:       groupInfo.Name,
39         ConversionID: groupInfo.ConversionId,
40     }, nil
41 }
42
```

4.5 群组列表


```
1 package logic
2
3 import (
4     "context"
5     "errors"
6
7     "chatops-server/internal/group/rpc/group"
8     "chatops-server/internal/group/rpc/internal/svc"
9
10    "github.com/zeromicro/go-zero/core/logx"
11    "github.com/zeromicro/go-zero/core/stores/sqlx"
12 )
13
14 type GroupListLogic struct {
15     ctx    context.Context
16     svcCtx *svc.ServiceContext
17     logx.Logger
18 }
19
20 func NewGroupListLogic(ctx context.Context, svcCtx *svc.ServiceContext) *GroupListLogic {
21     return &GroupListLogic{
22         ctx:    ctx,
23         svcCtx: svcCtx,
24         Logger: logx.WithContext(ctx),
25     }
26 }
27
28 type groupDbInfo struct {
29     Name          string `db:"name"`
30     ConversionID  string `db:"conversion_id"`
31 }
32
33 func (l *GroupListLogic) GroupList(in *group.GroupListRequest) (*group.GroupListResponse, error) {
34     // 查询群组信息
35     var groupListFromDb []groupDbInfo
36     querySQL := "select name, conversion_id from `groups`"
37     err := l.svcCtx.Conn.QueryRowsCtx(l.ctx, &groupListFromDb, querySQL)
38     if errors.Is(err, sqlx.ErrNotFound) {
39         logx.Errorw("group.rpc.Conn.QueryRowsCtx failed", logx.LogField{Key: "err", Value: sqlx.ErrNotFound.Error()})
40         return nil, errors.New("查询群组列表失败")
41     }
42     if err != nil {
```

```

43         logx.Errorw("group.rpc.Conn.QueryRowsCtx failed", logx.LogField{Key: "err", Value: err})
44         return nil, err
45     }
46     logx.Infow("group.rpc.Conn.QueryRowsCtx success")
47     var groupList []*group.GroupInfo
48     for _, groupFromDb := range groupListFromDb {
49         groupResponse := group.GroupInfo{
50             Name:          groupFromDb.Name,
51             ConversionID: groupFromDb.ConversionID,
52         }
53         groupList = append(groupList, &groupResponse)
54     }
55     return &group.GroupListResponse{
56         Message: "ok",
57         GroupList: groupList,
58     }, nil
59 }
60

```

5.task模块

5.1 task.proto

```
1  syntax = "proto3";
2
3  package task;
4
5  option go_package = "./task";
6
7  // AddTask
8  message addTaskRequest {
9      string name = 1;
10     string type = 2;
11     int64 remoteID = 3;
12     string script = 4;
13     string jenkinsJobName = 5;
14     string execParams = 6;
15     int64 groupID = 7;
16     int64 userID = 8;
17 }
18 message addTaskResponse {
19     string message = 1;
20 }
21 // AddTask
22
23 // DeleteTask
24 message deleteTaskRequest {
25     string name = 1;
26 }
27 message deleteTaskResponse {
28     string message = 1;
29 }
30 // DeleteTask
31
32 // UpdateTask
33 message updateTaskRequest {
34     string name = 1;
35     string type = 2;
36     int64 remoteID = 3;
37     string script = 4;
38     string jenkinsJobName = 5;
39     string execParams = 6;
40     int64 groupID = 7;
41     int64 userID = 8;
42 }
43 message updateTaskResponse {
44     string message = 1;
45 }
```

```

46 // UpdateTask
47
48 // TaskDetail
49 message taskDetailRequest {
50     string name = 1;
51 }
52 message taskDetailResponse {
53     string message = 1;
54     string name = 2;
55     string type = 3;
56     int64 remoteID = 4;
57     string script = 5;
58     string jenkinsJobName = 6;
59     string execParams = 7;
60     string groupName = 8;
61     string username = 9;
62 }
63 // TaskDetail
64
65 // TaskList
66 message taskInfo {
67     string name = 1;
68     string type = 2;
69     string groupName = 3;
70     string username = 4;
71 }
72 message taskListRequest {
73 }
74 message taskListResponse {
75     string message = 1;
76     repeated taskInfo taskList = 2;
77 }
78 // TaskList
79
80 // Services
81 service Task {
82     rpc addTask(addTaskRequest) returns(addTaskResponse);
83     rpc deleteTask(deleteTaskRequest) returns(deleteTaskResponse);
84     rpc updateTask(updateTaskRequest) returns(updateTaskResponse);
85     rpc taskDetail(taskDetailRequest) returns(taskDetailResponse);
86     rpc taskList(taskListRequest) returns(taskListResponse);
87 }
88 // Services

```

5.2 增加task接口

