Сервис

«Менеджер паролей GophKeeper»

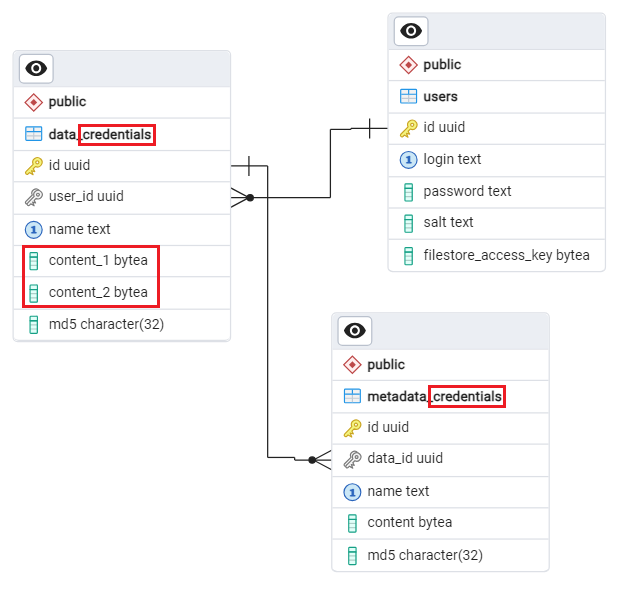
Функциональность сервиса

– Регистрация/логин пользователя

– Запись/удаление данных/метаданных

– Чтение данных с фильтрацией по тегу и/или метаданным

– Загрузка/скачивание файлов

База данных

Таблицы:

***\_credentials***

login, password

***\_credit\_card***

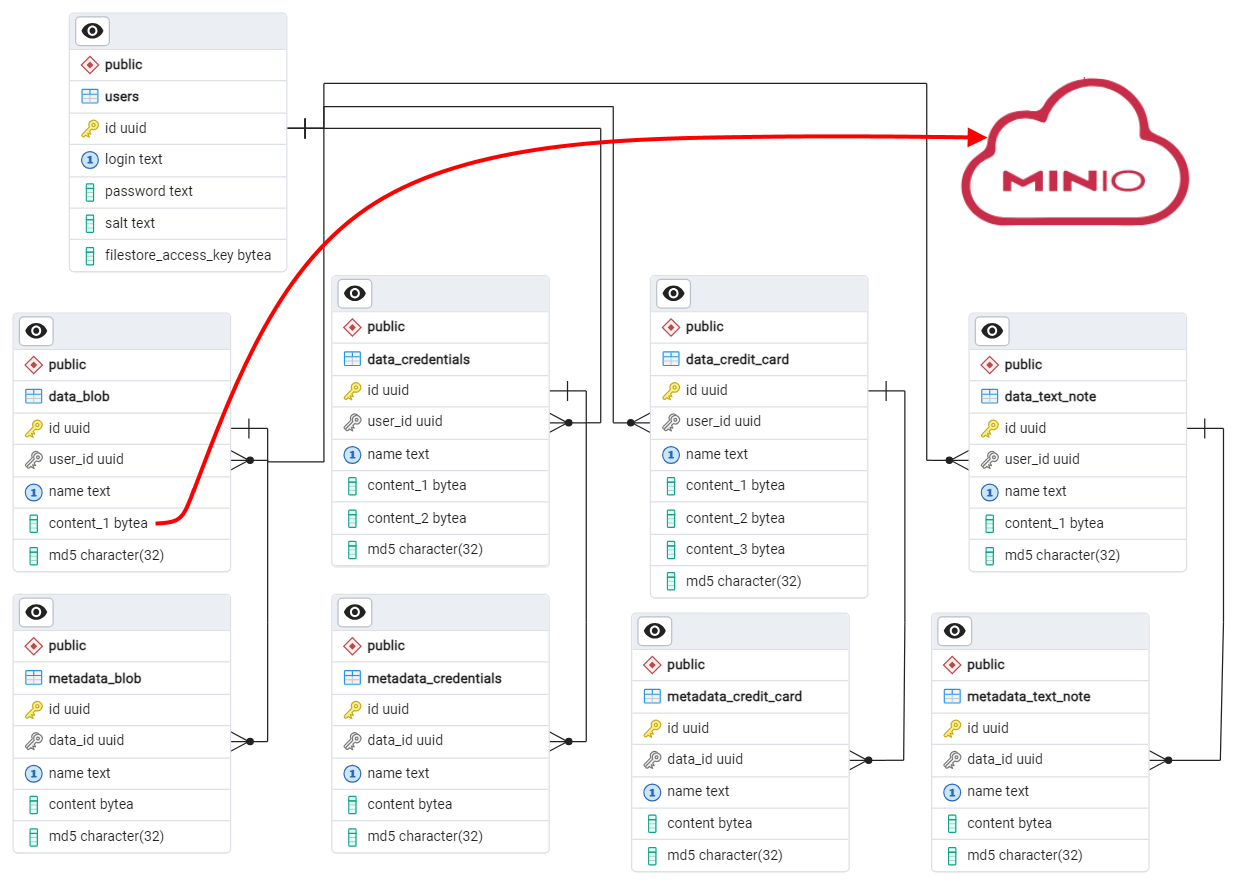
number, until, holder

***\_text\_note***

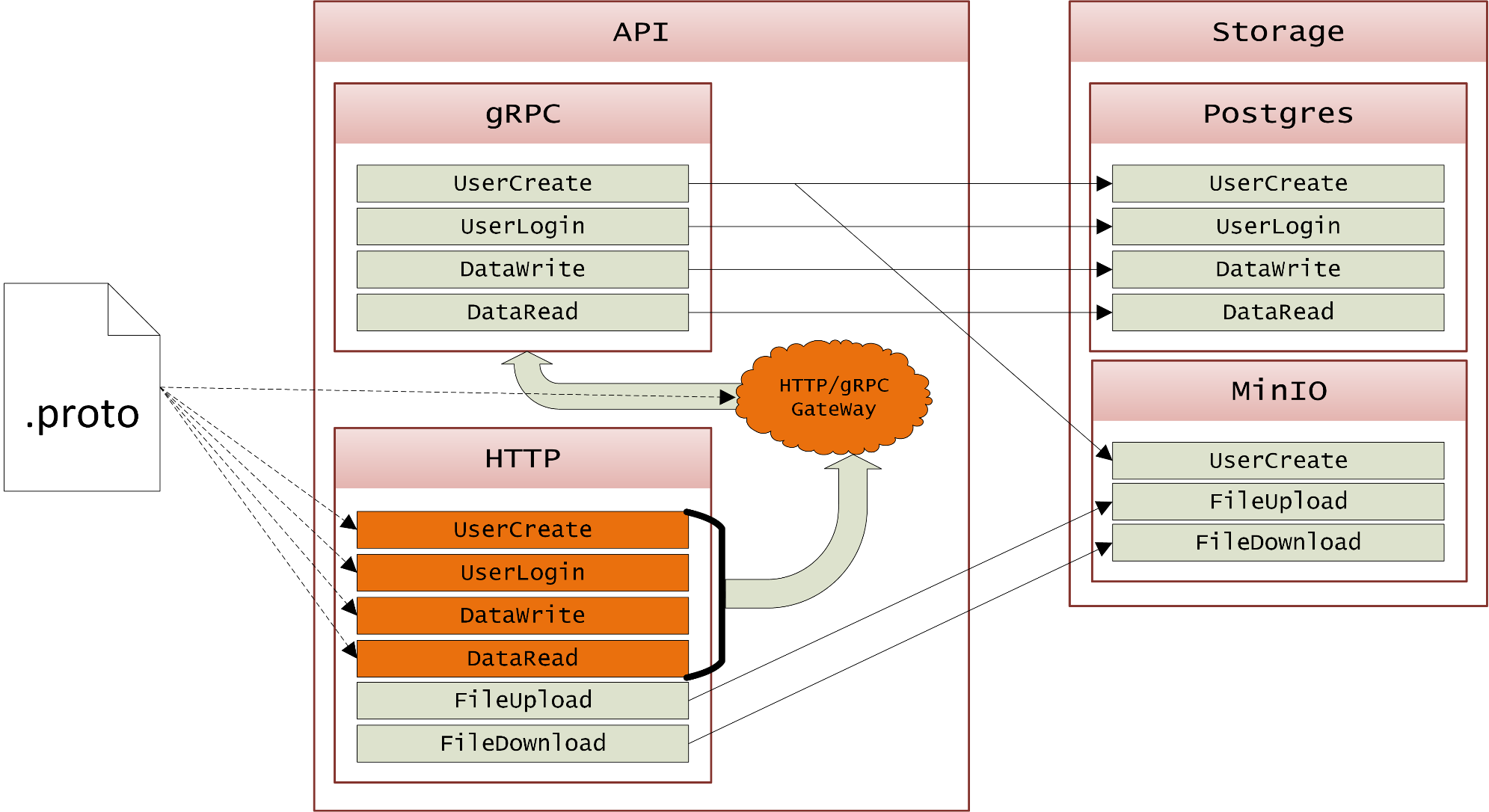
text

***\_blob***

file\_name

База данных

Структура сервиса



Описание API сервиса

**protoc** **-I ./internal/proto/src \**

**-I ../googleapis \**

**--go\_out ./internal/proto/gen \**

**--go\_opt paths=source\_relative \**

**--go-grpc\_out ./internal/proto/gen \**

**--go-grpc\_opt paths=source\_relative \**

**--grpc-gateway\_out ./internal/proto/gen \**

**--grpc-gateway\_opt paths=source\_relative \**

**service.proto**

Клиент

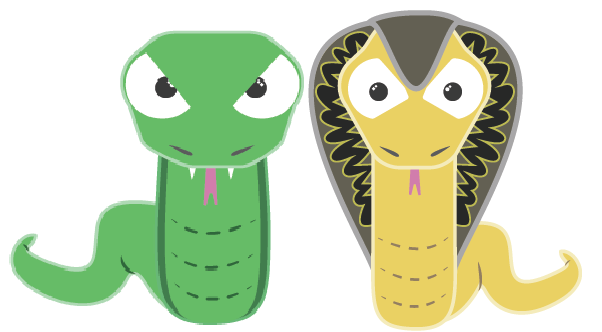
**./client user ( create | login ) -l “Victoria“ -p “Victoria’s secret“**

**./client data print -n “%“ -t ( “any“, “cred“, “card“, “note“, “file“ )**

**-m “site=google.com“ -m “type=main“**

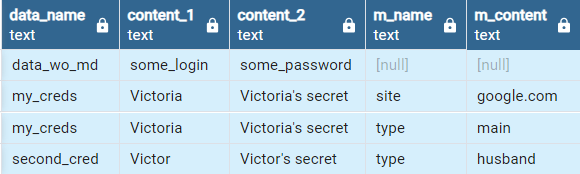
**./client data write ( cred | card | note | metadata ) -n “my\_note“ --dtext=“Text“**

**./client data delete ( cred | card | note | file | metadata ) -n “my\_file“**

**./client file upload -n “my\_file“ --fname=“D:\movie.avi“**

**./client file download -n “my\_file“**

Запрос данных с фильтрацией

**./client data print -n % -t cred -m “site=google.com“ -m “type=main“**

SELECT

    data.name AS data\_name,

    pgp\_sym\_decrypt(data.content\_1, 'secret') AS content\_1,

    pgp\_sym\_decrypt(data.content\_2, 'secret') AS content\_2,

    metadata.name AS m\_name,

    pgp\_sym\_decrypt(metadata.content, 'secret') AS m\_content

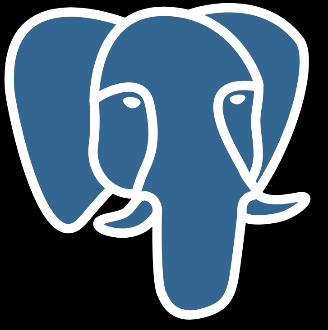
FROM

    (SELECT \* FROM data\_credentials WHERE user\_id = '898130ef-0785-4467-a75f-ee4ef9e40bf3') AS data

    LEFT JOIN metadata\_credentials AS metadata ON

        data.id = metadata.data\_id

WHERE

    data.name LIKE '%' AND

    EXISTS ( SELECT id FROM metadata\_credentials AS metadata WHERE

        data.id = metadata.data\_id AND (

        (metadata.name = 'site' AND

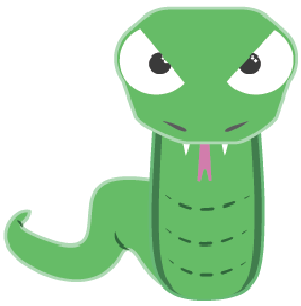
            pgp\_sym\_decrypt(metadata.content, 'secret') LIKE 'google.com') OR

        (metadata.name = 'type' AND

            pgp\_sym\_decrypt(metadata.content, 'secret') LIKE 'main')

    ))

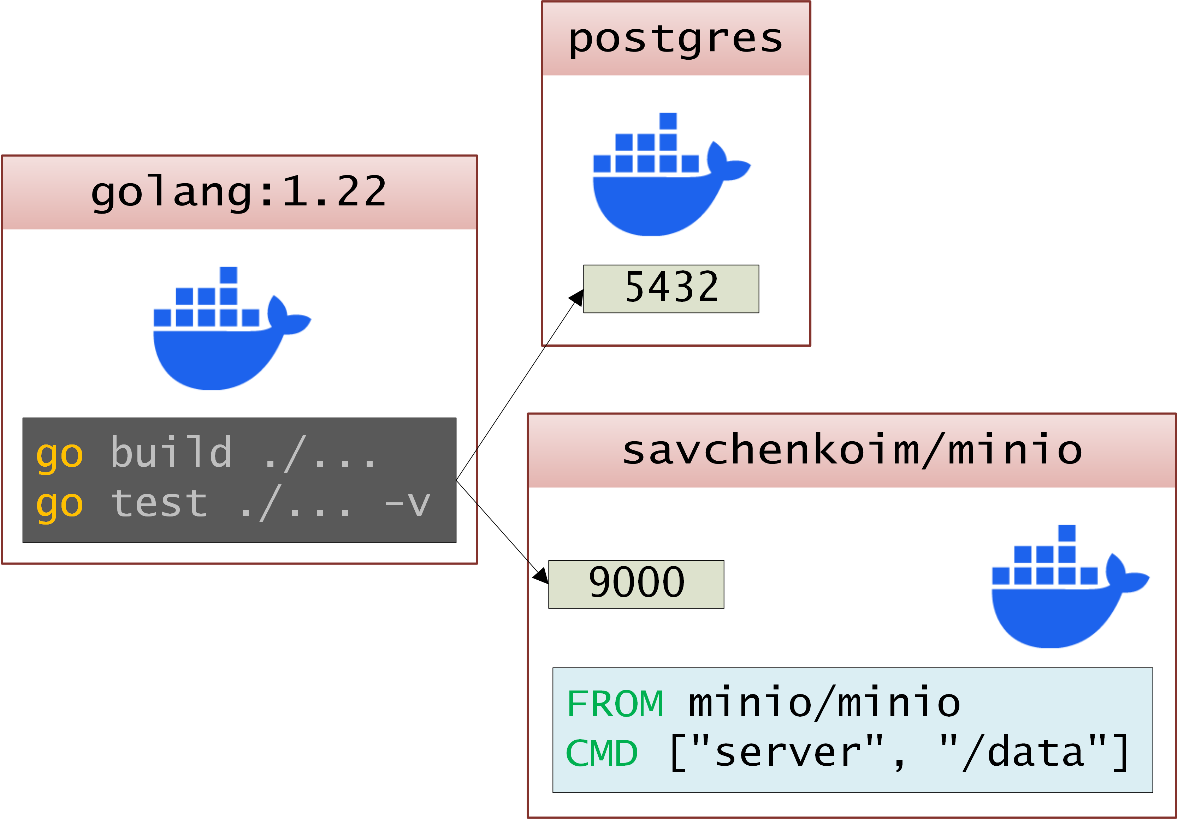
Пакет Viper для парсинга конфигурации

 flags := pflag.NewFlagSet("server\_cfg", pflag.*ExitOnError*)  
 flags.String("par1", "val1", "Parameter 1")  
 flags.String("par2", "val2", "Parameter 2")  
 flags.String("cfg", "", "Config file")  
 err = append(err, flags.Parse(os.Args))  
  
 err = append(err, viper.BindPFlags(flags))  
  
 err = append(err, viper.BindEnv("par1", "PARAMETER\_1"))  
 err = append(err, viper.BindEnv("par2", "PARAMETER\_2"))  
 err = append(err, viper.BindEnv("cfg", "CONFIG\_FILE"))  
  
 cfgSet := viper.GetString("cfg") != ""  
 if cfgSet {  
 viper.SetConfigFile(viper.GetString("cfg"))  
 err = append(err, viper.ReadInConfig())  
 }  
  
 // Do something  
 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // End doing something  
  
 if cfgSet {  
 err = append(err, viper.WriteConfig())  
 }

GitHub actions

t.Run("Setting\_Up\_Test\_Environment", func(t \*testing.T) {  
 if \_, ok := os.LookupEnv("GITHUB\_TEST\_RUN"); ok {  
 minioEndpoint = "minio:9000"  
 dbConnectionString = "postgresql://postgres:postgres@postgres/postgres?sslmode=disable"  
 } else {  
 cMinio, err = testcontainers.NewMinioContainer()  
 require.NoError(t, err)  
 minioEndpoint, err = cMinio.EndPoint()  
 require.NoError(t, err)  
 cPostgres, err = testcontainers.NewPostgresContainer()  
 require.NoError(t, err)  
 dbConnectionString, err = cPostgres.ConnectionString()  
 require.NoError(t, err)  
 }  
})

GitHub actions



Спасибо!