Московский государственный технический университет имени Н. Э. Баумана.

Факультет “Информатика и системы управления”
Кафедра ИУ5 ”Системы обработки информации и управления”

Курс “Парадигмы и конструкции языков программирования”
Отчет по лабораторной работе №1.

Выполнила:
Студент группы ИУ5-31Б
Савельева Д.А

Подпись и дата:

Проверил:
Преподаватель кафедры ИУ5
Нардид А.Н.

Подпись и дата:

Москва,2024 г.

Постановка задачи.

Создать программу с использованием технологий grpc, protobuf. Подключить к этой программе базу данных, используя postgres. Запустить программу используя Docker.

Текст программы:

Client.go

```
package cl

import (
	"context"
	"first_try/weather"
	"fmt"
	"google.golang.org/grpc"
)

type WeatherClient struct {
	client weather.WeatherServiceClient
}

func NewWeatherClient(address string) (*WeatherClient, error) {
	conn, err := grpc.Dial(address, grpc.WithInsecure())
	if err != nil {
		return nil, fmt.Errorf("failed to dial: %v", err)
	}
	return &WeatherClient{
		client: weather.NewWeatherServiceClient(conn),
	}, nil
}

func (c *WeatherClient) GetWeather(city string) (*weather.WeatherData, error) {
	req := &weather.CityRequest{
		City: city,
	}
	res, err := c.client.GetWeather(context.Background(), req)
	if err != nil {
		return nil, fmt.Errorf("failed to get weather data: %v", err)
	}

	return res, nil
}
```

Start_cl.go

```
package main

import (
	"bufio"
```

```
	"first_try/client/cl"
	"fmt"
	_ "github.com/lib/pq"
	"os"
)

func main() {
	reader := bufio.NewReader(os.Stdin)
	fmt.Print("Enter the name of the city: ")
	city, _ := reader.ReadString('\n')
	city = city[:len(city)-1] // убирает символ перевода строки

	weatherClient, err := cl.NewWeatherClient(":5000")
	if err != nil {
		fmt.Println(err)
		return
	}

	weatherData, err := weatherClient.GetWeather(city)

	if err != nil {
		fmt.Println(err)
		return
	}

	fmt.Printf("The weather in %s: Temperature is %.1f°C, feels like %.1f°C, humidity is %d%%, wind
speed is %.1f m/s\n",
		city, weatherData.Main.Temp, weatherData.Main.FeelsLike,
weatherData.Main.Humidity, weatherData.Wind.Speed)

}
```

Докерфайл для клиента:

```
FROM golang

WORKDIR /app

COPY . .
COPY ../cl ./

EXPOSE 3000
CMD ["go", "run", "start_cl.go"]
```

Сервер.

Server.go

```
package sv

import (
	"context"
	"database/sql"
```

```
	"encoding/json"
	"first_try/weather"
	"fmt"
	_ "github.com/lib/pq"
	"google.golang.org/grpc"
	"google.golang.org/grpc/codes"
	"google.golang.org/grpc/status"
	"io/ioutil"
	_ "os"

	"net"
	"net/http"
	"time"
)

const apiKey = "cae0bfaacc6a42deb153581a503b95f7" // API-ключ

type weatherServer struct {
	weather.UnimplementedWeatherServiceServer
}

func (s *weatherServer) GetWeather(ctx context.Context, req *weather.CityRequest)
(*weather.WeatherData, error) {
	url :=
fmt.Sprintf("http://api.openweathermap.org/data/2.5/weather?q=%s&appid=%s&units=metric",
req.City, apiKey)
	resp, err := http.Get(url)
	if err != nil {
		return nil, status.Errorf(codes.Internal, "failed to get: %v", err)
	}
	defer resp.Body.Close()

	body, err := ioutil.ReadAll(resp.Body)
	if err != nil {
		return nil, status.Errorf(codes.Internal, "failed to read: %v", err)
	}

	var weatherData weather.WeatherData

	err = json.Unmarshal(body, &weatherData) //записывает данные в weatherdata
	if err != nil {
		return nil, status.Errorf(codes.Internal, "failed to unmarshal: %v", err)
	}

	connStr := "user=postgres password=11111111 dbname=weatherdb sslmode=disable"
	db, err := sql.Open("postgres", connStr)
	if err != nil {
		return nil, status.Errorf(codes.Internal, "failed to open db: %v", err)
	}
	defer db.Close()

	result, err := db.Exec("insert into Weather (City, Time, Temperature, Feels, Humidity, Wind)
values ($1,$2,$3,$4,$5,$6)", weatherData.Name, time.Now(), weatherData.Main.Temp,
weatherData.Main.FeelsLike, weatherData.Main.Humidity, weatherData.Wind.Speed)
```

```
	if err != nil {
		panic(err)
	}

	rowsAffected, err := result.RowsAffected()
	if err != nil {
		return nil, status.Errorf(codes.Internal, "failed to add: %v", err)
	}
	fmt.Println("Rows affected:", rowsAffected)

	return &weatherData, nil
}

func RunServer(port string) error {
	lis, err := net.Listen("tcp", port)
	if err != nil {
		return status.Errorf(codes.Internal, "failed to listen: %v", err)
	}
	s := grpc.NewServer()
	weather.RegisterWeatherServiceServer(s, &weatherServer{})
	fmt.Printf("gRPC server started on port %s\n", port)
	return s.Serve(lis)
}
```

Start_serv.go

```
package main

import (
	"first_try/server/sv"
	"flag"
	"log"
)

func main() {
	port := flag.String("port", ":5000", "gRPC server port")
	flag.Parse()

	err := sv.RunServer(*port)
	if err != nil {
		log.Fatalf("failed to start server: %v", err)
	}
}
```

Докерфайл для сервера

```
FROM golang

WORKDIR /app

COPY . .
COPY ../sv ./
```

```
EXPOSE 5000
CMD ["go", "run", "start_serv.go"]
```

Docker_compose.yml

```
services:
  client:
    build: ./client
    ports:
      - "3000:3000"
    depends_on:
      - server
  server:
    build: ./server
    ports:
      - "5000:5000"
  postgres:
    image: bitnami/postgresql
    container_name: weather_postgres
    volumes:
      - "./migrations/postgres:/docker-entrypoint-initdb.d"
    environment:
      POSTGRES_PASSWORD: 11111111
      POSTGRES_DB: weatherdb
    ports:
      - "5432:5432"
    restart: unless-stopped
```

Прото файл

Weather.proto

```
syntax = "proto3";

package weather;

option go_package = "first_try/weather";

message WeatherData {
  message Coord {
    float lon = 1;
    float lat = 2;
  }

  message WeatherCondition {
    int32 id = 1;
    string main = 2;
    string description = 3;
    string icon = 4;
  }
```

```
 message MainData {
  float temp = 1;
  float feels_like = 2;
  float temp_min = 3;
  float temp_max = 4;
  int32 pressure = 5;
  int32 humidity = 6;
  int32 sea_level = 7;
  int32 grnd_level = 8;
 }

 message Wind {
  float speed = 1;
  int32 deg = 2;
  float gust = 3;
 }

 message Clouds {
  int32 all = 1;
 }

 message Sys {
  int32 type = 1;
  int32 id = 2;
  string country = 3;
  int64 sunrise = 4;
  int64 sunset = 5;
 }

 Coord coord = 1;
 repeated WeatherCondition weather = 2;
 string base = 3;
 MainData main = 4;
 int32 visibility = 5;
 Wind wind = 6;
 Clouds clouds = 7;
 int64 dt = 8;
 Sys sys = 9;
 int32 timezone = 10;
 int32 id = 11;
 string name = 12;
 int32 cod = 13;
}

service WeatherService {
 rpc GetWeather (CityRequest) returns (WeatherData);
}

message CityRequest {
 string city = 1;
}
```

Файлы, сгенерированные protobuf

Weather.pb.go

```
// Code generated by protoc-gen-go. DO NOT EDIT.
// versions:
//         protoc-gen-go v1.35.1
//         protoc        v5.29.0--rc2
// source: weather.proto

package weather

import (
        protoreflect "google.golang.org/protobuf/reflect/protoreflect"
        protoimpl "google.golang.org/protobuf/runtime/protoimpl"
        reflect "reflect"
        sync "sync"
)

const (
        // Verify that this generated code is sufficiently up-to-date.
        _ = protoimpl.EnforceVersion(20 - protoimpl.MinVersion)
        // Verify that runtime/protoimpl is sufficiently up-to-date.
        _ = protoimpl.EnforceVersion(protoimpl.MaxVersion - 20)
)

type WeatherData struct {
        state        protoimpl.MessageState
        sizeCache    protoimpl.SizeCache
        unknownFields protoimpl.UnknownFields

        Coord      *WeatherData_Coord           `protobuf:"bytes,1,opt,name=coord,proto3"
json:"coord,omitempty"`
        Weather    []*WeatherData_WeatherCondition `protobuf:"bytes,2,rep,name=weather,proto3"
json:"weather,omitempty"`
        Base       string                       `protobuf:"bytes,3,opt,name=base,proto3" json:"base,omitempty"`
        Main       *WeatherData_MainData        `protobuf:"bytes,4,opt,name=main,proto3"
json:"main,omitempty"`
        Visibility int32                        `protobuf:"varint,5,opt,name=visibility,proto3"
json:"visibility,omitempty"`
        Wind       *WeatherData_Wind            `protobuf:"bytes,6,opt,name=wind,proto3"
json:"wind,omitempty"`
        Clouds     *WeatherData_Clouds          `protobuf:"bytes,7,opt,name=clouds,proto3"
json:"clouds,omitempty"`
        Dt         int64                        `protobuf:"varint,8,opt,name=dt,proto3" json:"dt,omitempty"`
        Sys        *WeatherData_Sys             `protobuf:"bytes,9,opt,name=sys,proto3"
json:"sys,omitempty"`
        Timezone   int32                        `protobuf:"varint,10,opt,name=timezone,proto3"
json:"timezone,omitempty"`
        Id         int32                        `protobuf:"varint,11,opt,name=id,proto3" json:"id,omitempty"`
        Name       string                       `protobuf:"bytes,12,opt,name=name,proto3"
json:"name,omitempty"`
        Cod        int32                        `protobuf:"varint,13,opt,name=cod,proto3" json:"cod,omitempty"`
}
```

```
func (x *WeatherData) Reset() {
	*x = WeatherData{}
	mi := &file_weather_proto_msgTypes[0]
	ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
	ms.StoreMessageInfo(mi)
}

func (x *WeatherData) String() string {
	return protoimpl.X.MessageStringOf(x)
}

func (*WeatherData) ProtoMessage() {}

func (x *WeatherData) ProtoReflect() protoreflect.Message {
	mi := &file_weather_proto_msgTypes[0]
	if x != nil {
		ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
		if ms.LoadMessageInfo() == nil {
			ms.StoreMessageInfo(mi)
		}
		return ms
	}
	return mi.MessageOf(x)
}

// Deprecated: Use WeatherData.ProtoReflect.Descriptor instead.
func (*WeatherData) Descriptor() ([]byte, []int) {
	return file_weather_proto_rawDescGZIP(), []int{0}
}

func (x *WeatherData) GetCoord() *WeatherData_Coord {
	if x != nil {
		return x.Coord
	}
	return nil
}

func (x *WeatherData) GetWeather() []*WeatherData_WeatherCondition {
	if x != nil {
		return x.Weather
	}
	return nil
}

func (x *WeatherData) GetBase() string {
	if x != nil {
		return x.Base
	}
	return ""
}

func (x *WeatherData) GetMain() *WeatherData_MainData {
	if x != nil {
		return x.Main
```

```
	}
	return nil
}

func (x *WeatherData) GetVisibility() int32 {
	if x != nil {
		return x.Visibility
	}
	return 0
}

func (x *WeatherData) GetWind() *WeatherData_Wind {
	if x != nil {
		return x.Wind
	}
	return nil
}

func (x *WeatherData) GetClouds() *WeatherData_Clouds {
	if x != nil {
		return x.Clouds
	}
	return nil
}

func (x *WeatherData) GetDt() int64 {
	if x != nil {
		return x.Dt
	}
	return 0
}

func (x *WeatherData) GetSys() *WeatherData_Sys {
	if x != nil {
		return x.Sys
	}
	return nil
}

func (x *WeatherData) GetTimezone() int32 {
	if x != nil {
		return x.Timezone
	}
	return 0
}

func (x *WeatherData) GetId() int32 {
	if x != nil {
		return x.Id
	}
	return 0
}

func (x *WeatherData) GetName() string {
```

```
	if x != nil {
		return x.Name
	}
	return ""
}

func (x *WeatherData) GetCod() int32 {
	if x != nil {
		return x.Cod
	}
	return 0
}

type CityRequest struct {
	state         protoimpl.MessageState
	sizeCache     protoimpl.SizeCache
	unknownFields protoimpl.UnknownFields

	City string `protobuf:"bytes,1,opt,name=city,proto3" json:"city,omitempty"`
}

func (x *CityRequest) Reset() {
	*x = CityRequest{}
	mi := &file_weather_proto_msgTypes[1]
	ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
	ms.StoreMessageInfo(mi)
}

func (x *CityRequest) String() string {
	return protoimpl.X.MessageStringOf(x)
}

func (*CityRequest) ProtoMessage() {}

func (x *CityRequest) ProtoReflect() protoreflect.Message {
	mi := &file_weather_proto_msgTypes[1]
	if x != nil {
		ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
		if ms.LoadMessageInfo() == nil {
			ms.StoreMessageInfo(mi)
		}
		return ms
	}
	return mi.MessageOf(x)
}

// Deprecated: Use CityRequest.ProtoReflect.Descriptor instead.
func (*CityRequest) Descriptor() ([]byte, []int) {
	return file_weather_proto_rawDescGZIP(), []int{1}
}

func (x *CityRequest) GetCity() string {
	if x != nil {
		return x.City
```

```
	}
	return ""
}

type WeatherData_Coord struct {
	state         protoimpl.MessageState
	sizeCache     protoimpl.SizeCache
	unknownFields protoimpl.UnknownFields

	Lon float32 `protobuf:"fixed32,1,opt,name=lon,proto3" json:"lon,omitempty"`
	Lat float32 `protobuf:"fixed32,2,opt,name=lat,proto3" json:"lat,omitempty"`
}

func (x *WeatherData_Coord) Reset() {
	*x = WeatherData_Coord{}
	mi := &file_weather_proto_msgTypes[2]
	ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
	ms.StoreMessageInfo(mi)
}

func (x *WeatherData_Coord) String() string {
	return protoimpl.X.MessageStringOf(x)
}

func (*WeatherData_Coord) ProtoMessage() {}

func (x *WeatherData_Coord) ProtoReflect() protoreflect.Message {
	mi := &file_weather_proto_msgTypes[2]
	if x != nil {
		ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
		if ms.LoadMessageInfo() == nil {
			ms.StoreMessageInfo(mi)
		}
		return ms
	}
	return mi.MessageOf(x)
}

// Deprecated: Use WeatherData_Coord.ProtoReflect.Descriptor instead.
func (*WeatherData_Coord) Descriptor() ([]byte, []int) {
	return file_weather_proto_rawDescGZIP(), []int{0, 0}
}

func (x *WeatherData_Coord) GetLon() float32 {
	if x != nil {
		return x.Lon
	}
	return 0
}

func (x *WeatherData_Coord) GetLat() float32 {
	if x != nil {
		return x.Lat
	}
```

```
	return 0
}

type WeatherData_WeatherCondition struct {
	state         protoimpl.MessageState
	sizeCache     protoimpl.SizeCache
	unknownFields protoimpl.UnknownFields

	Id          int32  `protobuf:"varint,1,opt,name=id,proto3" json:"id,omitempty"`
	Main        string `protobuf:"bytes,2,opt,name=main,proto3" json:"main,omitempty"`
	Description string `protobuf:"bytes,3,opt,name=description,proto3"
json:"description,omitempty"`
	Icon        string `protobuf:"bytes,4,opt,name=icon,proto3" json:"icon,omitempty"`
}

func (x *WeatherData_WeatherCondition) Reset() {
	*x = WeatherData_WeatherCondition{}
	mi := &file_weather_proto_msgTypes[3]
	ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
	ms.StoreMessageInfo(mi)
}

func (x *WeatherData_WeatherCondition) String() string {
	return protoimpl.X.MessageStringOf(x)
}

func (*WeatherData_WeatherCondition) ProtoMessage() {}

func (x *WeatherData_WeatherCondition) ProtoReflect() protoreflect.Message {
	mi := &file_weather_proto_msgTypes[3]
	if x != nil {
		ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
		if ms.LoadMessageInfo() == nil {
			ms.StoreMessageInfo(mi)
		}
		return ms
	}
	return mi.MessageOf(x)
}

// Deprecated: Use WeatherData_WeatherCondition.ProtoReflect.Descriptor instead.
func (*WeatherData_WeatherCondition) Descriptor() ([]byte, []int) {
	return file_weather_proto_rawDescGZIP(), []int{0, 1}
}

func (x *WeatherData_WeatherCondition) GetId() int32 {
	if x != nil {
		return x.Id
	}
	return 0
}

func (x *WeatherData_WeatherCondition) GetMain() string {
	if x != nil {
```

```
		return x.Main
	}
	return ""
}

func (x *WeatherData_WeatherCondition) GetDescription() string {
	if x != nil {
		return x.Description
	}
	return ""
}

func (x *WeatherData_WeatherCondition) GetIcon() string {
	if x != nil {
		return x.Icon
	}
	return ""
}

type WeatherData_MainData struct {
	state         protoimpl.MessageState
	sizeCache     protoimpl.SizeCache
	unknownFields protoimpl.UnknownFields

	Temp      float32 `protobuf:"fixed32,1,opt,name=temp,proto3" json:"temp,omitempty"`
	FeelsLike float32 `protobuf:"fixed32,2,opt,name=feels_like,json=feelsLike,proto3"
json:"feels_like,omitempty"`
	TempMin   float32 `protobuf:"fixed32,3,opt,name=temp_min,json=tempMin,proto3"
json:"temp_min,omitempty"`
	TempMax   float32 `protobuf:"fixed32,4,opt,name=temp_max,json=tempMax,proto3"
json:"temp_max,omitempty"`
	Pressure  int32   `protobuf:"varint,5,opt,name=pressure,proto3" json:"pressure,omitempty"`
	Humidity  int32   `protobuf:"varint,6,opt,name=humidity,proto3" json:"humidity,omitempty"`
	SeaLevel  int32   `protobuf:"varint,7,opt,name=sea_level,json=seaLevel,proto3"
json:"sea_level,omitempty"`
	GrndLevel int32   `protobuf:"varint,8,opt,name=grnd_level,json=grndLevel,proto3"
json:"grnd_level,omitempty"`
}

func (x *WeatherData_MainData) Reset() {
	*x = WeatherData_MainData{}
	mi := &file_weather_proto_msgTypes[4]
	ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
	ms.StoreMessageInfo(mi)
}

func (x *WeatherData_MainData) String() string {
	return protoimpl.X.MessageStringOf(x)
}

func (*WeatherData_MainData) ProtoMessage() {}

func (x *WeatherData_MainData) ProtoReflect() protoreflect.Message {
	mi := &file_weather_proto_msgTypes[4]
```

```
		if x != nil {
			ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
			if ms.LoadMessageInfo() == nil {
				ms.StoreMessageInfo(mi)
			}
			return ms
		}
		return mi.MessageOf(x)
}

// Deprecated: Use WeatherData_MainData.ProtoReflect.Descriptor instead.
func (*WeatherData_MainData) Descriptor() ([]byte, []int) {
		return file_weather_proto_rawDescGZIP(), []int{0, 2}
}

func (x *WeatherData_MainData) GetTemp() float32 {
		if x != nil {
			return x.Temp
		}
		return 0
}

func (x *WeatherData_MainData) GetFeelsLike() float32 {
		if x != nil {
			return x.FeelsLike
		}
		return 0
}

func (x *WeatherData_MainData) GetTempMin() float32 {
		if x != nil {
			return x.TempMin
		}
		return 0
}

func (x *WeatherData_MainData) GetTempMax() float32 {
		if x != nil {
			return x.TempMax
		}
		return 0
}

func (x *WeatherData_MainData) GetPressure() int32 {
		if x != nil {
			return x.Pressure
		}
		return 0
}

func (x *WeatherData_MainData) GetHumidity() int32 {
		if x != nil {
			return x.Humidity
		}
```

```
	return 0
}

func (x *WeatherData_MainData) GetSeaLevel() int32 {
	if x != nil {
		return x.SeaLevel
	}
	return 0
}

func (x *WeatherData_MainData) GetGrndLevel() int32 {
	if x != nil {
		return x.GrndLevel
	}
	return 0
}

type WeatherData_Wind struct {
	state        protoimpl.MessageState
	sizeCache    protoimpl.SizeCache
	unknownFields protoimpl.UnknownFields

	Speed float32 `protobuf:"fixed32,1,opt,name=speed,proto3" json:"speed,omitempty"`
	Deg   int32   `protobuf:"varint,2,opt,name=deg,proto3" json:"deg,omitempty"`
	Gust  float32 `protobuf:"fixed32,3,opt,name=gust,proto3" json:"gust,omitempty"` // Added gust
field
}

func (x *WeatherData_Wind) Reset() {
	*x = WeatherData_Wind{}
	mi := &file_weather_proto_msgTypes[5]
	ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
	ms.StoreMessageInfo(mi)
}

func (x *WeatherData_Wind) String() string {
	return protoimpl.X.MessageStringOf(x)
}

func (*WeatherData_Wind) ProtoMessage() {}

func (x *WeatherData_Wind) ProtoReflect() protoreflect.Message {
	mi := &file_weather_proto_msgTypes[5]
	if x != nil {
		ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
		if ms.LoadMessageInfo() == nil {
			ms.StoreMessageInfo(mi)
		}
		return ms
	}
	return mi.MessageOf(x)
}

// Deprecated: Use WeatherData_Wind.ProtoReflect.Descriptor instead.
```

```
func (*WeatherData_Wind) Descriptor() ([]byte, []int) {
	return file_weather_proto_rawDescGZIP(), []int{0, 3}
}

func (x *WeatherData_Wind) GetSpeed() float32 {
	if x != nil {
		return x.Speed
	}
	return 0
}

func (x *WeatherData_Wind) GetDeg() int32 {
	if x != nil {
		return x.Deg
	}
	return 0
}

func (x *WeatherData_Wind) GetGust() float32 {
	if x != nil {
		return x.Gust
	}
	return 0
}

type WeatherData_Clouds struct {
	state         protoimpl.MessageState
	sizeCache     protoimpl.SizeCache
	unknownFields protoimpl.UnknownFields

	All int32 `protobuf:"varint,1,opt,name=all,proto3" json:"all,omitempty"`
}

func (x *WeatherData_Clouds) Reset() {
	*x = WeatherData_Clouds{}
	mi := &file_weather_proto_msgTypes[6]
	ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
	ms.StoreMessageInfo(mi)
}

func (x *WeatherData_Clouds) String() string {
	return protoimpl.X.MessageStringOf(x)
}

func (*WeatherData_Clouds) ProtoMessage() {}

func (x *WeatherData_Clouds) ProtoReflect() protoreflect.Message {
	mi := &file_weather_proto_msgTypes[6]
	if x != nil {
		ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
		if ms.LoadMessageInfo() == nil {
			ms.StoreMessageInfo(mi)
		}
		return ms
```

```
	}
	return mi.MessageOf(x)
}

// Deprecated: Use WeatherData_Clouds.ProtoReflect.Descriptor instead.
func (*WeatherData_Clouds) Descriptor() ([]byte, []int) {
	return file_weather_proto_rawDescGZIP(), []int{0, 4}
}

func (x *WeatherData_Clouds) GetAll() int32 {
	if x != nil {
		return x.All
	}
	return 0
}

type WeatherData_Sys struct {
	state         protoimpl.MessageState
	sizeCache     protoimpl.SizeCache
	unknownFields protoimpl.UnknownFields

	Type    int32  `protobuf:"varint,1,opt,name=type,proto3" json:"type,omitempty"`
	Id      int32  `protobuf:"varint,2,opt,name=id,proto3" json:"id,omitempty"`
	Country string `protobuf:"bytes,3,opt,name=country,proto3" json:"country,omitempty"`
	Sunrise int64  `protobuf:"varint,4,opt,name=sunrise,proto3" json:"sunrise,omitempty"`
	Sunset  int64  `protobuf:"varint,5,opt,name=sunset,proto3" json:"sunset,omitempty"`
}

func (x *WeatherData_Sys) Reset() {
	*x = WeatherData_Sys{}
	mi := &file_weather_proto_msgTypes[7]
	ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
	ms.StoreMessageInfo(mi)
}

func (x *WeatherData_Sys) String() string {
	return protoimpl.X.MessageStringOf(x)
}

func (*WeatherData_Sys) ProtoMessage() {}

func (x *WeatherData_Sys) ProtoReflect() protoreflect.Message {
	mi := &file_weather_proto_msgTypes[7]
	if x != nil {
		ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
		if ms.LoadMessageInfo() == nil {
			ms.StoreMessageInfo(mi)
		}
		return ms
	}
	return mi.MessageOf(x)
}

// Deprecated: Use WeatherData_Sys.ProtoReflect.Descriptor instead.
```

```
func (*WeatherData_Sys) Descriptor() ([]byte, []int) {
	return file_weather_proto_rawDescGZIP(), []int{0, 5}
}

func (x *WeatherData_Sys) GetType() int32 {
	if x != nil {
		return x.Type
	}
	return 0
}

func (x *WeatherData_Sys) GetId() int32 {
	if x != nil {
		return x.Id
	}
	return 0
}

func (x *WeatherData_Sys) GetCountry() string {
	if x != nil {
		return x.Country
	}
	return ""
}

func (x *WeatherData_Sys) GetSunrise() int64 {
	if x != nil {
		return x.Sunrise
	}
	return 0
}

func (x *WeatherData_Sys) GetSunset() int64 {
	if x != nil {
		return x.Sunset
	}
	return 0
}

var File_weather_proto protoreflect.FileDescriptor

var file_weather_proto_rawDesc = []byte{
	0x0a, 0x0d, 0x77, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x2e, 0x70, 0x72, 0x6f, 0x74, 0x6f, 0x12,
	0x07, 0x77, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x22, 0xb5, 0x08, 0x0a, 0x0b, 0x57, 0x65, 0x61,
	0x74, 0x68, 0x65, 0x72, 0x44, 0x61, 0x74, 0x61, 0x12, 0x30, 0x0a, 0x05, 0x63, 0x6f, 0x6f, 0x72,
	0x64, 0x18, 0x01, 0x20, 0x01, 0x28, 0x0b, 0x32, 0x1a, 0x2e, 0x77, 0x65, 0x61, 0x74, 0x68, 0x65,
	0x72, 0x2e, 0x57, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x44, 0x61, 0x74, 0x61, 0x2e, 0x43, 0x6f,
	0x6f, 0x72, 0x64, 0x52, 0x05, 0x63, 0x6f, 0x6f, 0x72, 0x64, 0x12, 0x3f, 0x0a, 0x07, 0x77, 0x65,
	0x61, 0x74, 0x68, 0x65, 0x72, 0x18, 0x02, 0x20, 0x03, 0x28, 0x0b, 0x32, 0x25, 0x2e, 0x77, 0x65,
	0x61, 0x74, 0x68, 0x65, 0x72, 0x2e, 0x57, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x44, 0x61, 0x74,
	0x61, 0x2e, 0x57, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x43, 0x6f, 0x6e, 0x64, 0x69, 0x74, 0x69,
	0x6f, 0x6e, 0x52, 0x07, 0x77, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x12, 0x12, 0x0a, 0x04, 0x62,
	0x61, 0x73, 0x65, 0x18, 0x03, 0x20, 0x01, 0x28, 0x09, 0x52, 0x04, 0x62, 0x61, 0x73, 0x65, 0x12,
	0x31, 0x0a, 0x04, 0x6d, 0x61, 0x69, 0x6e, 0x18, 0x04, 0x20, 0x01, 0x28, 0x0b, 0x32, 0x1d, 0x2e,
```

0x77, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x2e, 0x57, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x44,
0x61, 0x74, 0x61, 0x2e, 0x4d, 0x61, 0x69, 0x6e, 0x44, 0x61, 0x74, 0x61, 0x52, 0x04, 0x6d, 0x61,
0x69, 0x6e, 0x12, 0x1e, 0x0a, 0x0a, 0x76, 0x69, 0x73, 0x69, 0x62, 0x69, 0x6c, 0x69, 0x74, 0x79,
0x18, 0x05, 0x20, 0x01, 0x28, 0x05, 0x52, 0x0a, 0x76, 0x69, 0x73, 0x69, 0x62, 0x69, 0x6c, 0x69,
0x74, 0x79, 0x12, 0x2d, 0x0a, 0x04, 0x77, 0x69, 0x6e, 0x64, 0x18, 0x06, 0x20, 0x01, 0x28, 0x0b,
0x32, 0x19, 0x2e, 0x77, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x2e, 0x57, 0x65, 0x61, 0x74, 0x68,
0x65, 0x72, 0x44, 0x61, 0x74, 0x61, 0x2e, 0x57, 0x69, 0x6e, 0x64, 0x52, 0x04, 0x77, 0x69, 0x6e,
0x64, 0x12, 0x33, 0x0a, 0x06, 0x63, 0x6c, 0x6f, 0x75, 0x64, 0x73, 0x18, 0x07, 0x20, 0x01, 0x28,
0x0b, 0x32, 0x1b, 0x2e, 0x77, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x2e, 0x57, 0x65, 0x61, 0x74,
0x68, 0x65, 0x72, 0x44, 0x61, 0x74, 0x61, 0x2e, 0x43, 0x6c, 0x6f, 0x75, 0x64, 0x73, 0x52, 0x06,
0x63, 0x6c, 0x6f, 0x75, 0x64, 0x73, 0x12, 0x0e, 0x0a, 0x02, 0x64, 0x74, 0x18, 0x08, 0x20, 0x01,
0x28, 0x03, 0x52, 0x02, 0x64, 0x74, 0x12, 0x2a, 0x0a, 0x03, 0x73, 0x79, 0x73, 0x18, 0x09, 0x20,
0x01, 0x28, 0x0b, 0x32, 0x18, 0x2e, 0x77, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x2e, 0x57, 0x65,
0x61, 0x74, 0x68, 0x65, 0x72, 0x44, 0x61, 0x74, 0x61, 0x2e, 0x53, 0x79, 0x73, 0x52, 0x03, 0x73,
0x79, 0x73, 0x12, 0x1a, 0x0a, 0x08, 0x74, 0x69, 0x6d, 0x65, 0x7a, 0x6f, 0x6e, 0x65, 0x18, 0x0a,
0x20, 0x01, 0x28, 0x05, 0x52, 0x08, 0x74, 0x69, 0x6d, 0x65, 0x7a, 0x6f, 0x6e, 0x65, 0x12, 0x0e,
0x0a, 0x02, 0x69, 0x64, 0x18, 0x0b, 0x20, 0x01, 0x28, 0x05, 0x52, 0x02, 0x69, 0x64, 0x12, 0x12,
0x0a, 0x04, 0x6e, 0x61, 0x6d, 0x65, 0x18, 0x0c, 0x20, 0x01, 0x28, 0x09, 0x52, 0x04, 0x6e, 0x61,
0x6d, 0x65, 0x12, 0x10, 0x0a, 0x03, 0x63, 0x6f, 0x64, 0x18, 0x0d, 0x20, 0x01, 0x28, 0x05, 0x52,
0x03, 0x63, 0x6f, 0x64, 0x1a, 0x2b, 0x0a, 0x05, 0x43, 0x6f, 0x6f, 0x72, 0x64, 0x12, 0x10, 0x0a,
0x03, 0x6c, 0x6f, 0x6e, 0x18, 0x01, 0x20, 0x01, 0x28, 0x02, 0x52, 0x03, 0x6c, 0x6f, 0x6e, 0x12,
0x10, 0x0a, 0x03, 0x6c, 0x61, 0x74, 0x18, 0x02, 0x20, 0x01, 0x28, 0x02, 0x52, 0x03, 0x6c, 0x61,
0x74, 0x1a, 0x6c, 0x0a, 0x10, 0x57, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x43, 0x6f, 0x6e, 0x64,
0x69, 0x74, 0x69, 0x6f, 0x6e, 0x12, 0x0e, 0x0a, 0x02, 0x69, 0x64, 0x18, 0x01, 0x20, 0x01, 0x28,
0x05, 0x52, 0x02, 0x69, 0x64, 0x12, 0x12, 0x0a, 0x04, 0x6d, 0x61, 0x69, 0x6e, 0x18, 0x02, 0x20,
0x01, 0x28, 0x09, 0x52, 0x04, 0x6d, 0x61, 0x69, 0x6e, 0x12, 0x20, 0x0a, 0x0b, 0x64, 0x65, 0x73,
0x63, 0x72, 0x69, 0x70, 0x74, 0x69, 0x6f, 0x6e, 0x18, 0x03, 0x20, 0x01, 0x28, 0x09, 0x52, 0x0b,
0x64, 0x65, 0x73, 0x63, 0x72, 0x69, 0x70, 0x74, 0x69, 0x6f, 0x6e, 0x12, 0x12, 0x0a, 0x04, 0x69,
0x63, 0x6f, 0x6e, 0x18, 0x04, 0x20, 0x01, 0x28, 0x09, 0x52, 0x04, 0x69, 0x63, 0x6f, 0x6e, 0x1a,
0xe7, 0x01, 0x0a, 0x08, 0x4d, 0x61, 0x69, 0x6e, 0x44, 0x61, 0x74, 0x61, 0x12, 0x12, 0x0a, 0x04,
0x74, 0x65, 0x6d, 0x70, 0x18, 0x01, 0x20, 0x01, 0x28, 0x02, 0x52, 0x04, 0x74, 0x65, 0x6d, 0x70,
0x12, 0x1d, 0x0a, 0x0a, 0x66, 0x65, 0x65, 0x6c, 0x73, 0x5f, 0x6c, 0x69, 0x6b, 0x65, 0x18, 0x02,
0x20, 0x01, 0x28, 0x02, 0x52, 0x09, 0x66, 0x65, 0x65, 0x6c, 0x73, 0x4c, 0x69, 0x6b, 0x65, 0x12,
0x19, 0x0a, 0x08, 0x74, 0x65, 0x6d, 0x70, 0x5f, 0x6d, 0x69, 0x6e, 0x18, 0x03, 0x20, 0x01, 0x28,
0x02, 0x52, 0x07, 0x74, 0x65, 0x6d, 0x70, 0x4d, 0x69, 0x6e, 0x12, 0x19, 0x0a, 0x08, 0x74, 0x65,
0x6d, 0x70, 0x5f, 0x6d, 0x61, 0x78, 0x18, 0x04, 0x20, 0x01, 0x28, 0x02, 0x52, 0x07, 0x74, 0x65,
0x6d, 0x70, 0x4d, 0x61, 0x78, 0x12, 0x1a, 0x0a, 0x08, 0x70, 0x72, 0x65, 0x73, 0x73, 0x75, 0x72,
0x65, 0x18, 0x05, 0x20, 0x01, 0x28, 0x05, 0x52, 0x08, 0x70, 0x72, 0x65, 0x73, 0x73, 0x75, 0x72,
0x65, 0x12, 0x1a, 0x0a, 0x08, 0x68, 0x75, 0x6d, 0x69, 0x64, 0x69, 0x74, 0x79, 0x18, 0x06, 0x20,
0x01, 0x28, 0x05, 0x52, 0x08, 0x68, 0x75, 0x6d, 0x69, 0x64, 0x69, 0x74, 0x79, 0x12, 0x1b, 0x0a,
0x09, 0x73, 0x65, 0x61, 0x5f, 0x6c, 0x65, 0x76, 0x65, 0x6c, 0x18, 0x07, 0x20, 0x01, 0x28, 0x05,
0x52, 0x08, 0x73, 0x65, 0x61, 0x4c, 0x65, 0x76, 0x65, 0x6c, 0x12, 0x1d, 0x0a, 0x0a, 0x67, 0x72,
0x6e, 0x64, 0x5f, 0x6c, 0x65, 0x76, 0x65, 0x6c, 0x18, 0x08, 0x20, 0x01, 0x28, 0x05, 0x52, 0x09,
0x67, 0x72, 0x6e, 0x64, 0x4c, 0x65, 0x76, 0x65, 0x6c, 0x1a, 0x42, 0x0a, 0x04, 0x57, 0x69, 0x6e,
0x64, 0x12, 0x14, 0x0a, 0x05, 0x73, 0x70, 0x65, 0x65, 0x64, 0x18, 0x01, 0x20, 0x01, 0x28, 0x02,
0x52, 0x05, 0x73, 0x70, 0x65, 0x65, 0x64, 0x12, 0x10, 0x0a, 0x03, 0x64, 0x65, 0x67, 0x18, 0x02,
0x20, 0x01, 0x28, 0x05, 0x52, 0x03, 0x64, 0x65, 0x67, 0x12, 0x12, 0x0a, 0x04, 0x67, 0x75, 0x73,
0x74, 0x18, 0x03, 0x20, 0x01, 0x28, 0x02, 0x52, 0x04, 0x67, 0x75, 0x73, 0x74, 0x1a, 0x1a, 0x0a,
0x06, 0x43, 0x6c, 0x6f, 0x75, 0x64, 0x73, 0x12, 0x10, 0x0a, 0x03, 0x61, 0x6c, 0x6c, 0x18, 0x01,
0x20, 0x01, 0x28, 0x05, 0x52, 0x03, 0x61, 0x6c, 0x6c, 0x1a, 0x75, 0x0a, 0x03, 0x53, 0x79, 0x73,
0x12, 0x12, 0x0a, 0x04, 0x74, 0x79, 0x70, 0x65, 0x18, 0x01, 0x20, 0x01, 0x28, 0x05, 0x52, 0x04,
0x74, 0x79, 0x70, 0x65, 0x12, 0x0e, 0x0a, 0x02, 0x69, 0x64, 0x18, 0x02, 0x20, 0x01, 0x28, 0x05,
0x52, 0x02, 0x69, 0x64, 0x12, 0x18, 0x0a, 0x07, 0x63, 0x6f, 0x75, 0x6e, 0x74, 0x72, 0x79, 0x18,
0x03, 0x20, 0x01, 0x28, 0x09, 0x52, 0x07, 0x63, 0x6f, 0x75, 0x6e, 0x74, 0x72, 0x79, 0x12, 0x18,

```
	0x0a, 0x07, 0x73, 0x75, 0x6e, 0x72, 0x69, 0x73, 0x65, 0x18, 0x04, 0x20, 0x01, 0x28, 0x03, 0x52,
	0x07, 0x73, 0x75, 0x6e, 0x72, 0x69, 0x73, 0x65, 0x12, 0x16, 0x0a, 0x06, 0x73, 0x75, 0x6e, 0x73,
	0x65, 0x74, 0x18, 0x05, 0x20, 0x01, 0x28, 0x03, 0x52, 0x06, 0x73, 0x75, 0x6e, 0x73, 0x65, 0x74,
	0x22, 0x21, 0x0a, 0x0b, 0x43, 0x69, 0x74, 0x79, 0x52, 0x65, 0x71, 0x75, 0x65, 0x73, 0x74, 0x12,
	0x12, 0x0a, 0x04, 0x63, 0x69, 0x74, 0x79, 0x18, 0x01, 0x20, 0x01, 0x28, 0x09, 0x52, 0x04, 0x63,
	0x69, 0x74, 0x79, 0x32, 0x4a, 0x0a, 0x0e, 0x57, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x53, 0x65,
	0x72, 0x76, 0x69, 0x63, 0x65, 0x12, 0x38, 0x0a, 0x0a, 0x47, 0x65, 0x74, 0x57, 0x65, 0x61, 0x74,
	0x68, 0x65, 0x72, 0x12, 0x14, 0x2e, 0x77, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x2e, 0x43, 0x69,
	0x74, 0x79, 0x52, 0x65, 0x71, 0x75, 0x65, 0x73, 0x74, 0x1a, 0x14, 0x2e, 0x77, 0x65, 0x61, 0x74,
	0x68, 0x65, 0x72, 0x2e, 0x57, 0x65, 0x61, 0x74, 0x68, 0x65, 0x72, 0x44, 0x61, 0x74, 0x61, 0x42,
	0x13, 0x5a, 0x11, 0x66, 0x69, 0x72, 0x73, 0x74, 0x5f, 0x74, 0x72, 0x79, 0x2f, 0x77, 0x65, 0x61,
	0x74, 0x68, 0x65, 0x72, 0x62, 0x06, 0x70, 0x72, 0x6f, 0x74, 0x6f, 0x33,
}

var (
	file_weather_proto_rawDescOnce sync.Once
	file_weather_proto_rawDescData = file_weather_proto_rawDesc
)

func file_weather_proto_rawDescGZIP() []byte {
	file_weather_proto_rawDescOnce.Do(func() {
		file_weather_proto_rawDescData =
protoimpl.X.CompressGZIP(file_weather_proto_rawDescData)
	})
	return file_weather_proto_rawDescData
}

var file_weather_proto_msgTypes = make([]protoimpl.MessageInfo, 8)
var file_weather_proto_goTypes = []any{
	(*WeatherData)(nil),               // 0: weather.WeatherData
	(*CityRequest)(nil),               // 1: weather.CityRequest
	(*WeatherData_Coord)(nil),         // 2: weather.WeatherData.Coord
	(*WeatherData_WeatherCondition)(nil), // 3: weather.WeatherData.WeatherCondition
	(*WeatherData_MainData)(nil),      // 4: weather.WeatherData.MainData
	(*WeatherData_Wind)(nil),          // 5: weather.WeatherData.Wind
	(*WeatherData_Clouds)(nil),        // 6: weather.WeatherData.Clouds
	(*WeatherData_Sys)(nil),           // 7: weather.WeatherData.Sys
}
var file_weather_proto_depIdxs = []int32{
	2, // 0: weather.WeatherData.coord:type_name -> weather.WeatherData.Coord
	3, // 1: weather.WeatherData.weather:type_name -> weather.WeatherData.WeatherCondition
	4, // 2: weather.WeatherData.main:type_name -> weather.WeatherData.MainData
	5, // 3: weather.WeatherData.wind:type_name -> weather.WeatherData.Wind
	6, // 4: weather.WeatherData.clouds:type_name -> weather.WeatherData.Clouds
	7, // 5: weather.WeatherData.sys:type_name -> weather.WeatherData.Sys
	1, // 6: weather.WeatherService.GetWeather:input_type -> weather.CityRequest
	0, // 7: weather.WeatherService.GetWeather:output_type -> weather.WeatherData
	7, // [7:8] is the sub-list for method output_type
	6, // [6:7] is the sub-list for method input_type
	6, // [6:6] is the sub-list for extension type_name
	6, // [6:6] is the sub-list for extension extendee
	0, // [0:6] is the sub-list for field type_name
}
```

```
func init() { file_weather_proto_init() }
func file_weather_proto_init() {
	if File_weather_proto != nil {
		return
	}
	type x struct{}
	out := protoimpl.TypeBuilder{
		File: protoimpl.DescBuilder{
			GoPackagePath: reflect.TypeOf(x{}).PkgPath(),
			RawDescriptor: file_weather_proto_rawDesc,
			NumEnums:     0,
			NumMessages:   8,
			NumExtensions: 0,
			NumServices:   1,
		},
		GoTypes:         file_weather_proto_goTypes,
		DependencyIndexes: file_weather_proto_depIdxs,
		MessageInfos:     file_weather_proto_msgTypes,
	}.Build()
	File_weather_proto = out.File
	file_weather_proto_rawDesc = nil
	file_weather_proto_goTypes = nil
	file_weather_proto_depIdxs = nil
}
```

Weather_grpc.pb.go

```
// Code generated by protoc-gen-go-grpc. DO NOT EDIT.
// versions:
// - protoc-gen-go-grpc v1.5.1
// - protoc           v5.29.0--rc2
// source: weather.proto

package weather

import (
	context "context"
	grpc "google.golang.org/grpc"
	codes "google.golang.org/grpc/codes"
	status "google.golang.org/grpc/status"
)

// This is a compile-time assertion to ensure that this generated file
// is compatible with the grpc package it is being compiled against.
// Requires gRPC-Go v1.64.0 or later.
const _ = grpc.SupportPackageIsVersion9

const (
	WeatherService_GetWeather_FullMethodName = "/weather.WeatherService/GetWeather"
)

// WeatherServiceClient is the client API for WeatherService service.
//
```

```
// For semantics around ctx use and closing/ending streaming RPCs, please refer to
https://pkg.go.dev/google.golang.org/grpc/?tab=doc#ClientConn.NewStream.
type WeatherServiceClient interface {
	GetWeather(ctx context.Context, in *CityRequest, opts ...grpc.CallOption) (*WeatherData, error)
}

type weatherServiceClient struct {
	cc grpc.ClientConnInterface
}

func NewWeatherServiceClient(cc grpc.ClientConnInterface) WeatherServiceClient {
	return &weatherServiceClient{cc}
}

func (c *weatherServiceClient) GetWeather(ctx context.Context, in *CityRequest, opts
...grpc.CallOption) (*WeatherData, error) {
	cOpts := append([]grpc.CallOption{grpc.StaticMethod()}, opts...)
	out := new(WeatherData)
	err := c.cc.Invoke(ctx, WeatherService_GetWeather_FullMethodName, in, out, cOpts...)
	if err != nil {
		return nil, err
	}
	return out, nil
}

// WeatherServiceServer is the server API for WeatherService service.
// All implementations must embed UnimplementedWeatherServiceServer
// for forward compatibility.
type WeatherServiceServer interface {
	GetWeather(context.Context, *CityRequest) (*WeatherData, error)
	mustEmbedUnimplementedWeatherServiceServer()
}

// UnimplementedWeatherServiceServer must be embedded to have
// forward compatible implementations.
//
// NOTE: this should be embedded by value instead of pointer to avoid a nil
// pointer dereference when methods are called.
type UnimplementedWeatherServiceServer struct{}

func (UnimplementedWeatherServiceServer) GetWeather(context.Context, *CityRequest)
(*WeatherData, error) {
	return nil, status.Errorf(codes.Unimplemented, "method GetWeather not implemented")
}
func (UnimplementedWeatherServiceServer) mustEmbedUnimplementedWeatherServiceServer() {}
func (UnimplementedWeatherServiceServer) testEmbeddedByValue()                       {}

// UnsafeWeatherServiceServer may be embedded to opt out of forward compatibility for this service.
// Use of this interface is not recommended, as added methods to WeatherServiceServer will
// result in compilation errors.
type UnsafeWeatherServiceServer interface {
	mustEmbedUnimplementedWeatherServiceServer()
}
```

```
func RegisterWeatherServiceServer(s grpc.ServiceRegistrar, srv WeatherServiceServer) {
	// If the following call pancis, it indicates UnimplementedWeatherServiceServer was
	// embedded by pointer and is nil.  This will cause panics if an
	// unimplemented method is ever invoked, so we test this at initialization
	// time to prevent it from happening at runtime later due to I/O.
	if t, ok := srv.(interface{ testEmbeddedByValue() }); ok {
		t.testEmbeddedByValue()
	}
	s.RegisterService(&WeatherService_ServiceDesc, srv)
}

func _WeatherService_GetWeather_Handler(srv interface{}, ctx context.Context, dec func(interface{})
error, interceptor grpc.UnaryServerInterceptor) (interface{}, error) {
	in := new(CityRequest)
	if err := dec(in); err != nil {
		return nil, err
	}
	if interceptor == nil {
		return srv.(WeatherServiceServer).GetWeather(ctx, in)
	}
	info := &grpc.UnaryServerInfo{
		Server:     srv,
		FullMethod: WeatherService_GetWeather_FullMethodName,
	}
	handler := func(ctx context.Context, req interface{}) (interface{}, error) {
		return srv.(WeatherServiceServer).GetWeather(ctx, req.(*CityRequest))
	}
	return interceptor(ctx, in, info, handler)
}

// WeatherService_ServiceDesc is the grpc.ServiceDesc for WeatherService service.
// It's only intended for direct use with grpc.RegisterService,
// and not to be introspected or modified (even as a copy)
var WeatherService_ServiceDesc = grpc.ServiceDesc{
	ServiceName: "weather.WeatherService",
	HandlerType: (*WeatherServiceServer)(nil),
	Methods: []grpc.MethodDesc{
		{
			MethodName: "GetWeather",
			Handler:    _WeatherService_GetWeather_Handler,
		},
	},
	Streams:  []grpc.StreamDesc{},
	Metadata: "weather.proto",
}
```

Результат:

```
D:\dassa\dassa\grcp\grcp_weather\client>go run start_cl.go
Enter the name of the city: London
The weather in London: Temperature is 9.0°C, feels like 7.6°C, humidity is 94%, wind speed is 2.7 m/s

D:\dassa\dassa\grcp\grcp_weather\client>
```

```
D:\dassa\dassa\grcp\grcp_weather\server>go run start_serv.go
gRPC server started on port :5000
Rows affected: 1
```

Таким образом, программа показывает информацию о погоде в любом городе и добавляет ее в базу данных.