eloy-aws-api-service/src/handlers/addDevice/addDevice.go (92.1%) not tracked not covered covered

```go
package main

import (
	"types"
	"fmt"
	"os"
	"encoding/json"
	"errors"

	"github.com/aws/aws-sdk-go/aws"
	"github.com/aws/aws-lambda-go/lambda"
	"github.com/aws/aws-lambda-go/events"
	"github.com/aws/aws-sdk-go/aws/session"
	"github.com/aws/aws-sdk-go/service/dynamodb"
	"github.com/aws/aws-sdk-go/service/dynamodb/dynamodbattribute"
	"github.com/aws/aws-sdk-go/service/dynamodb/dynamodbiface"
)

type SuccessResponse struct{
	Status		string		`json:"status"`
	Device		types.Device		`json:"data"`
}

type dynamoDBAPI struct{
	DynamoDB dynamodbiface.DynamoDBAPI
}

var databseStruct *types.DatabseStruct
var dynamodbapi *dynamoDBAPI

func init(){
	databseStruct = new(types.DatabseStruct)
	region := os.Getenv("AWS_REGION")
	dynamodbapi = new(dynamoDBAPI) // crate a setter that  can be used for inserting
	sess, err := session.NewSession(&aws.Config{Region: &region},)
	databseStruct.SessionError = err
	svc := dynamodb.New(sess)
	dynamodbapi.DynamoDB = dynamodbiface.DynamoDBAPI(svc)

	if err != nil {
		fmt.Println("There is an error while creating database session: " + err.Error())
	}

	// Get table name from OS's environment
	fetchedTableName :=os.Getenv("DEVICES_TABLE_NAME")
	if len(fetchedTableName)==0 {
		databseStruct.TableName =  nil;
		fmt.Println("It is not possible to fetch device tabel name")
	}else{
		databseStruct.TableName = aws.String(fetchedTableName)
	}
}



// main AWS lambda function starting point.
// It gets some inputs from client as json, parse it and tries to insert it into dynamodb.
// valid input json is like types.Device struct
func AddDevice(request events.APIGatewayProxyRequest) (events.APIGatewayProxyResponse, error) {

	// there is some internal server error
	if databseStruct.SessionError != nil || databseStruct.TableName == nil {
		return events.APIGatewayProxyResponse{
			Body:		createErrorResponseJson(500, "Internal Server's Error occured"),
			StatusCode: 500,
		}, nil
	}

	// validate inputs of client's request (APIGatewayProxyRequest).
	newDevice, err := validateInputs(request)

	// if inputs are not suitable, return HTTP 400 error
	if err != nil {
		return events.APIGatewayProxyResponse{
			Body:		"" + err.Error(),
			StatusCode: 400,
		}, nil
	}

	_, err = dynamodbapi.insertItemToDatabase(newDevice)

	// If an internal error occured in the database  , return HTTP error 500
	if err != nil {
		return events.APIGatewayProxyResponse{
			Body:		createErrorResponseJson(500,"Internal Server's Error occured"),
			StatusCode: 500,
		}, nil
```

```
	}

	// looks fine, item inserted and result will be returned.
	return createSuccessResponseJson(newDevice)
}

func validateInputs(request events.APIGatewayProxyRequest) (types.Device, error) {

	var errorFlag bool = false

	// Initialize device json object(struct)
	device := types.Device{
		ID:                                  "",
		DeviceModel:        "",
		Name:                          "",
		Note:                          "",
		Serial:                          "",
	}

	errorMessage := ""

	if len(request.Body) == 0 {
		errorMessage = "No inputs provided, please provide inputs in json format."
		return types.Device{}, errors.New(createErrorResponseJson(400, errorMessage))
	}

	// Parse request body, gets body of request then parse it to json and finally assigns it to device
	var err = json.Unmarshal([]byte(request.Body), &device)

	if err != nil {
		errorMessage = "Wrong format: Inputs must be a valid json."
		return types.Device{}, errors.New(createErrorResponseJson(400, errorMessage))
	}

	errorMessage = "Following fields are not provided: "

	if len(device.ID) == 0 {
		errorMessage += "id, "
		errorFlag = true
	}

	if len(device.DeviceModel) == 0 {
		errorMessage += "deviceModel, "
		errorFlag = true
	}

	if len(device.Name) == 0 {
		errorMessage += "name, "
		errorFlag = true
	}

	if len(device.Note) == 0 {
		errorMessage += "note, "
		errorFlag = true
	}

	if len(device.Serial) == 0 {
		errorMessage += "serial, "
		errorFlag = true
	}

	// if some fields are missin, report it as an error
	if errorFlag == true {
		return types.Device{}, errors.New(createErrorResponseJson(400, errorMessage))
	}
	// everything looks fine, return created device
	return device, nil
}

func createErrorResponseJson(errorCode int, errorMessage string) (jsonString string) {

	errorResponse := types.ErrorResponse { ErrorMessage: types.ErrorMessage { Code: errorCode, Message: errorMessage,},}
	errorResponseJson, _ := json.MarshalIndent(&errorResponse, "", "\t")
	return string(errorResponseJson)
}

func createSuccessResponseJson(newDevice types.Device) (events.APIGatewayProxyResponse, error){
	successResponse := SuccessResponse {
		"requested item inserted",
		newDevice,
	}

	successResponseJson, _ := json.MarshalIndent(&successResponse, "", "\t")

	return events.APIGatewayProxyResponse {
		Body: string(successResponseJson),
		StatusCode: 201,
	}, nil
}
```

```
// function that just insert requested item to dynamodb's table.
func (ig *dynamoDBAPI) insertItemToDatabase(newDevice types.Device)(*dynamodb.PutItemOutput, error){

        // marshal newDevice struct(object) as a dynamodb item
        item, _ := dynamodbattribute.MarshalMap(newDevice)

        // preparing an input for dynamodb
        input := &dynamodb.PutItemInput{
                Item: item,
                TableName: databseStruct.TableName,
        }

        // put created input to dynamodb
        output, err := ig.DynamoDB.PutItem(input)
        return output, err
}

func main(){
        // aws lambda function calls it
        lambda.Start(AddDevice)
}
```