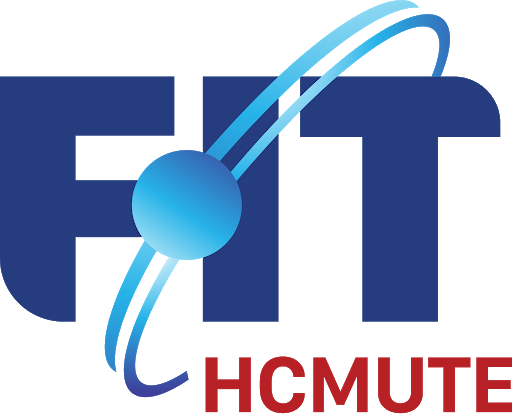
**.**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN ĐIỆN TOÁN ĐÁM MÂY**

-----🙞🙜🕮🙞🙜-----

****

**ĐỀ TÀI:**

**XÂY DỰNG ỨNG DỤNG TRÊN AWS CHO PHÉP TẠO DATABASE VÀ CUNG CẤP API ĐỂ THÊM, XÓA, SỬA TRÊN DATABASE**

Khóa 2019-2022

**GVHD: TS. HUỲNH XUÂN PHỤNG**

**SVTH:**

**VŨ MẠNH CƯỜNG 20110449**

**NGUYỄN NHƯ SÂM 20110557**

**TRẦN ĐỨC LONG 20110058**

Khóa 2019 -2023

**KHÓA 2020-2024**

|  |  |
| --- | --- |
| **ĐH SƯ PHẠM KỸ THUẬT TP. HCM**  **KHOA CNTT**  \*\*\*\*\*\*\* | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh Phúc**  \*\*\*\*\*\*\* |

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên Sinh viên 1: **VŨ MẠNH CƯỜNG** MSSV 1: **20110449**

Họ và tên Sinh viên 2: **NGUYỄN NHƯ SÂM** MSSV 2: **20110557**

Họ và tên Sinh viên 3: **TRẦN ĐỨC LONG** MSSV 3: **20110058**

Ngành: **Kỹ thuật dữ liệu**

Tên đề tài: **Xây dựng ứng dụng trên AWS cho phép tạo database và cung cấp API để thêm, xóa sửa trên database**

Họ và tên Giáo viên hướng dẫn: **TS. Huỳnh Xuân Phụng**

**NHẬN XÉT**

1. Về nội dung đề tài & khối lượng thực hiện:

2. Ưu điểm:

3. Khuyết điểm:

4. Đề nghị cho bảo vệ hay không?

5. Đánh giá loại:

6. Điểm:

|  |  |  |
| --- | --- | --- |
| *Tp. Hồ Chí Minh, ngày tháng năm 2022* | | |
|  | Giáo viên hướng dẫn  *(Ký & ghi rõ họ tên)* |

# 

# **LỜI CẢM ƠN**

*“Lời đầu tiên nhóm xin phép được gửi lời cảm ơn chân thành đến Khoa Công Nghệ Thông Tin – Trường Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh đã tạo điều kiện cho nhóm chúng em được học tập, phát triển nền tảng kiến thức sâu sắc và thực hiện đề tài này.*

*Bên cạnh đó chúng em xin gửi lời cảm ơn đến thầy Huỳnh Xuân Phụng đã chỉ dạy những kiến thức bổ ích của bộ môn Điện Toán Đám Mây và hướng dẫn chúng em thực hiện đề tài Xây Dựng Ứng Dụng Trên AWS Cho Phép Tạo Database Và Cung Cấp API Để Thêm, Xóa, Sửa Trên Database.*

*Chúng em xin cảm ơn thầy vì thời gian qua đã hướng dẫn và chỉ dạy chúng em nhiệt tình giúp chúng em nắm được những kiến thức bổ ích của môn học này.*

*Tuy nhiên lượng kiến thức là vô tận nên chúng em đã cố gắng để có thể hoàn thành đề tài này, nên khó tránh khỏi thiếu sót chúng em mong thầy có thể thông cảm. Chúng em mong nhận được sự góp ý của thầy để có thêm kinh nghiệm để có thể hoàn thiện đề tài hơn.”*

*Chúng em xin chân thành cảm ơn!*

***Nhóm thực hiện***

Vũ Mạnh Cường 20110449

Nguyễn Như Sâm 20110557

Trần Đức Long 20110058

**MỤC LỤC**

**[LỜI CẢM ƠN 3](#_Toc7768)**

**[PHẦN MỞ ĐẦU 5](#_Toc17168)**

[1. Tính cấp thiết của đề tài 5](#_Toc29386)

[Cơ sở dữ liệu (database) nói chung và cơ sở dữ liệu không quan hệ (noSQL database) nói riêng là một thành phần không thể thiếu trong việc tạo nên các phần mềm ứng dụng hoặc website có như cầu lưu trữ thông tin có thể hoặc không do người dùng nhập vào. Nhờ chúng mà dữ liệu với những insight quan trọng có thể được lưu trữ một cách dễ dàng. Vì vậy mà việc tạo nên một hệ thống có khả năng làm cho việc tương tác đến cơ sở dữ liệu của các lập trình viên trở nên dễ dàng hơn là một việc cấp thiết cần phải được thực hiện. 5](#_Toc5929)

[2. Đối tượng nghiên cứu 5](#_Toc10201)

[3. Phạm vi nghiên cứu 5](#_Toc29456)

[4. Kết quả dự kiến đạt được 5](#_Toc30088)

**[PHẦN NỘI DUNG 7](#_Toc28151)**

**[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 7](#_Toc16531)**

[1.1 Tổng quan về Lambda 7](#_Toc8888)

[1.2 Các tính năng của Lambda 7](#_Toc1357)

[1.3 . Các dịch vụ liên quan đến Lambda 7](#_Toc15792)

[1.4 Tổng quan về DynamoDB 8](#_Toc17043)

[1.5 . Các tính năng, dịch vụ liên quan đến DynamoDB 8](#_Toc32370)

**[1.6 . Tại sao nên sữ dụng Dynamodb? 9](#_Toc11568)**

**[CHƯƠNG 2: THỰC HIỆN 11](#_Toc29723)**

[2.1 Cài đặt 11](#_Toc30551)

*[2.1.1 Cài đặt Function Lamda](#_Toc12630)* [11](#_Toc12630)

*[2.1.2 Cài đặt Simple Queue Service:](#_Toc5427)* [15](#_Toc5427)

*[2.1.3 Setup Web (PHP):](#_Toc23875)* [16](#_Toc23875)

[- Source Code: 16](#_Toc28302)

[2.2 Triển khai Project: 19](#_Toc28901)

*[2.2.1. Local:](#_Toc12069)* [19](#_Toc12069)

*[2.2.2. EC2:](#_Toc11466)* [21](#_Toc11466)

[2.3 Thiết kế DynamoDB minh họa cho việc tương tác: 22](#_Toc12832)

**[Danh sách các bảng 22](#_Toc28923)**

**[PHẦN KẾT LUẬN 22](#_Toc10950)**

[1. Kết quả đạt được 22](#_Toc12672)

*[1.1. Kiến thức tìm hiểu được](#_Toc31598)* [22](#_Toc31598)

*[1.2. Chương trình đã làm được](#_Toc21503)* [23](#_Toc21503)

[2. Ưu điểm 23](#_Toc29997)

[3. Nhược điểm 23](#_Toc26420)

[4. Hướng phát triển 23](#_Toc19931)

**PHẦN MỞ ĐẦU**

Ngày nay, các ứng dụng công nghệ thông tin đã được áp dụng trong mọi lĩnh vực nghiên cứu khoa học kĩ thuật cũng như trong đời sống, máy tính trở thành công cụ trợ giúp đắc lực cho con người trong lưu trữ, phân tích và xử lý thông tin. Ở nước ta, cùng với sự phát triển chung của nền kinh tế, tin học đang ngày được ứng dụng rộng rãi trong công tác quản lý và từng bước khẳng định sức mạnh cũng như vị trí quan trọng của mình.

## **1. Tính cấp thiết của đề tài**

## Cơ sở dữ liệu (database) nói chung và cơ sở dữ liệu không quan hệ (noSQL database) nói riêng là một thành phần không thể thiếu trong việc tạo nên các phần mềm ứng dụng hoặc website có như cầu lưu trữ thông tin có thể hoặc không do người dùng nhập vào. Nhờ chúng mà dữ liệu với những insight quan trọng có thể được lưu trữ một cách dễ dàng. Vì vậy mà việc tạo nên một hệ thống có khả năng làm cho việc tương tác đến cơ sở dữ liệu của các lập trình viên trở nên dễ dàng hơn là một việc cấp thiết cần phải được thực hiện.

## **2. Đối tượng nghiên cứu**

Đối với đề tài này, đối tượng nghiên cứu là Lambda, DynamoDB. Đồng thời kết hợp với các kiến thức về Front-end và Back-end để tạo một trang web cụ thể.

## **3. Phạm vi nghiên cứu**

Đề tài này chủ yếu tập trung vào việc kết hợp sử dụng Lambda và DynamoDB nhằm thực hiện các câu truy vấn để thêm, sửa, xóa dữ liệu trên database.

## **4. Kết quả dự kiến đạt được**

* Cài đặt các giao diện
* Cài Đặt phía backend
* Thêm, xóa các table trong DynamoDB
* Thêm, sửa, xóa các item trong DynamoDB

# 

# **PHẦN NỘI DUNG**

# **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

## **Tổng quan về Lambda**

AWS Lambda là một dịch vụ điện toán phi máy chủ, theo định hướng sự kiện, giúp bạn chạy mã cho hầu hết mọi loại ứng dụng hoặc dịch vụ backend mà không cần cung cấp hay quản lý máy chủ. Bạn có thể kích hoạt Lambda từ hơn 200 dịch vụ AWS và các ứng dụng phần mềm dưới dạng dịch vụ (SaaS), đồng thời bạn chỉ phải trả tiền theo mức sử dụng.AWS Lamda được hỗ trợ và quản lý bởi đội ngũ

## **Các tính năng của Lambda**

Các tính năng chính sau đây giúp bạn phát triển các ứng dụng Lambda có khả năng mở rộng, bảo mật và dễ dàng mở rộng:

* AWS Lambda dễ dàng scale cơ sở hạ tầng mà không cần bất kỳ cấu hình bổ sung nào. Dịch vụ này giúp các developer giảm công việc vận hành liên quan tới hạ tầng hệ thống.
* Dịch vụ cung cấp nhiều tùy chọn như dịch vụ Amazon S3, Amazon CloudWatch, Amazon DynamoDB, Amazon API Gateway, Amazon Kinesis, Amazon CodeCommit và nhiều tùy chọn khác để trigger event.
* Không yêu cầu chi phí trả trước (upfront). Bạn chỉ trả tiền cho bộ nhớ được sử dụng bởi lambda function và chi phí tối thiểu cho số lượng request do đó AWS Lambda khá tiết kiệm chi phí cho người dùng.
* AWS Lambda sử dụng AWS IAM để xác định tất cả các vai trò và chính sách bảo mật do đảm bảo tính bảo mật cho ứng dụng của người dùng.
* AWS Lambda cũng cung cấp khả năng chịu lỗi (fault tolerance) cho các services run code và function. Người dùng sẽ không cần lo về downtime của ứng dụng.

## **. Các dịch vụ liên quan đến Lambda**

Lambda tích hợp với các dịch vụ AWS khác để gọi các hàm dựa trên các sự kiện được chỉ định:

* API Gateway cung cấp các đường dẫn có tính bảo mật cao, có thể mở rộng với các API web định tuyến đến các request HHTP và kích hoạt đến các Lamda Function
* Đối với các dịch vụ tạo hàng đợi hoặc các luồng dữ liệu (chẳng hạn như DynamoDB và Kinesis), Lambda sẽ xem xét các hàng đợi hoặc luồng dữ liệu từ dịch vụ và gọi Function để xử lý dữ liệu của các hàng đợi hoặc luồng dữ liệu.
* Xác định các sự kiện Amazon S3 gọi một hàm Lambda để xử lý các đối tượng Amazon S3, chẳng hạn như khi một đối tượng được tạo hoặc xóa.
* Sử dụng chức năng Lambda để xử lý các thông báo Amazon SQS messages hoặc Amazon Simple Notification Service (Amazon SNS).
* Sử dụng Chức năng từng bước AWS để kết nối các chức năng Lambda với nhau thành quy trình làm việc không máy chủ được gọi là máy trạng thái.

## **Tổng quan về DynamoDB**

DynamoDB cho phép người dùng tạo cơ sở dữ liệu có khả năng lưu trữ và truy xuất bất kỳ lượng dữ liệu nào cũng như phục vụ bất kỳ lượng truy cập nào. Nó tự động phân phối dữ liệu và lưu lượng qua các máy chủ để quản lý động các yêu cầu của từng khách hàng và cũng duy trì hiệu suất nhanh chóng.

## **. Các tính năng, dịch vụ liên quan đến DynamoDB**

* Trực tiếp nhập dữ liệu từ Amazon S3 vào bảng DynamoDB mới mà không phải viết mã hoặc quản lý cơ sở hạ tầng bổ sung.
* AWS Glue Elastic Views hỗ trợ DynamoDB dưới dạng một nguồn để kết hợp và sao chép liên tục dữ liệu trên nhiều cơ sở dữ liệu sát thời gian thực.
* Sử dụng PartiQL, một loại ngôn ngữ truy vấn tương thích SQL để truy vấn, chèn, cập nhật và xóa dữ liệu bảng trong DynamoDB.
* Sử dụng Amazon Kinesis Data Streams để nắm bắt các thay đổi cấp độ mục trong các bảng DynamoDB của bạn.
* Khôi phục các bảng DynamoDB nhanh hơn nữa.
* Công cụ tính giá AWS hiện đã hỗ trợ DynamoDB.
* Xuất dữ liệu từ DynamoDB sang Dịch vụ lưu trữ đơn giản của Amazon (Amazon S3) và sử dụng các dịch vụ AWS khác như Amazon Athena để phân tích dữ liệu của bạn cũng như trích xuất thông tin chuyên sâu có ý nghĩa.

AWS Glue Views cho phép các lập trình viên tạo nên các giao diện trực quan cho phép dễ dàng tìm kiếm, kết hợp, nhân bản giữa nhiều kho dữ liệu khác nhau theo thời gian thực mà không cần phải tự viết code tùy biến, dữ liệu tổng hợp có thể được chuyển đến các dịch vụ khác như: Redshift, S3, Aura … phục vụ nhiều mục đích khác nhau.

PartiSQL là một ngôn ngữ truy vấn tương thích với SQL, có thể liên kết với nhiều cơ sở dữ liệu với nhiều định dạng dữ liệu khác nhau như dữ liệu có cấu trúc, dữ liệu bán cấu trúc, dữ liệu dạng lưới. Ngôn ngữ truy vấn này được áp dụng trong nhiều dịch vụ của AWS, trong đó có Dynamodb.

Amazon Kinesis được sử dụng để tổng hợp và xử lý các luồng dữ liệu lớn theo thời gian thực. Các dữ liệu này được lưu trong các data record và có thể được sử dụng cho nhiều mục đích khác nhau: tạo thông báo, tạo các bảng báo cáo, các bảng giá có giá trị thay đổi theo thời gian,…

Công cụ tính giá tiền AWS dùng dể tính tiền các dịch vụ hoặc các tài nguyên đã sử dụng trên AWS.

* 1. **. Tại sao nên sữ dụng Dynamodb?**

Tính tự mở rộng: Chắc hẳn tất cả các lập trình viên đều biết rằng khi một trang web hoặc ứng dụng đưa vào thực tế một thời gian thì nó sẽ phát triển đến mức mà luồng dữ liệu ra vào hệ thống sẽ lớn mức gần như không điểm soát được. Đối với cơ sở dữ liệu có quan hệ (SQL Database) thì việc nâng cấp, mở rộng cơ sở dữ liệu sẽ rất khó thực hiện đi kèm với nhiều rủi ro, do hầu hết dữ liệu được lưu trữ là dữ liệu của khách hàng. Còn nếu cài đặt cơ sở dữ liệu quá lớn so với nhu cầu sẽ dễ đến hệ quả là tốn những chi phí không cần thiết. Bài toán này đã được DynamoDB giải quyết bằng chức năng tự scale theo nhu cầu sử dụng, từ đó mà xác định được ngưỡng trên của cơ sở dữ liệu, từ đó mà cân bằng được bài toán hiệu suất - chi phí.

Tính phân quyền: Khi cơ sở dữ liệu phát triển đến một mức độ nào đó thì việc cơ sở dữ liệu chỉ do duy nhất một lập trình viên quản lý là việc bất khả thi. Dynamodb cho phép phân quyền cho các lập trình, cho phép mỗi người có quyền truy cập vào một/ một số phân vùng nhất định.

Thời gian sống: Dynamodb cho phép lập trình viên cấu hình thời gian tồn tại của một table, khi thời gian tồn tại này hết thì table sẽ bị hủy. Việc này cho phép lập trình viên tự động hóa việc xóa dữ liệu thay vì xử lý bằng tay. Tính năng này cũng giúp tiết kiệm dung lượng sử dụng và chi phí.

Lưu trữ các dữ liệu không liên tục: Do đặc tính là một cơ sở dữ liệu không quan hệ mà DynamoDB có đặc tính rất phù hợp với những đối tượng không có cấu trúc cố định và thay đổi theo thời gian.

Tự động quản lý: Với mục đích bảo mật và an toàn mà dữ liệu trong DynamoDb được tự động liên tục cập nhật và backup lên cloud.

**CHƯƠNG 2: THỰC HIỆN**

## **2.1 Cài đặt**

### *2.1.1 Cài đặt Function Lamda*

#### CreateTable: Function dùng để tạo bảng trong DynamoDB

{

"TableName": "Employees",

"HashKeyName": "Name",

"HashKeyType": "S",

"RangeKeyName": "Email",

"RangeKeyType": "S",

"ReadCapacityUnits": 1,

"WriteCapacityUnits": 1

}

===

import json

import boto3

dynamodb = boto3.resource('dynamodb')

def lambda\_handler(event, context):

input = json.loads(event['Records'][0]['body'])

key\_schema = [

{

'AttributeName': input['HashKeyName'],

'KeyType': 'HASH'

}

]

attribute\_definitions = [

{

'AttributeName': input['HashKeyName'],

'AttributeType': input['HashKeyType']

}

]

if("RangeKeyName" in input.keys()):

key\_schema.append({

'AttributeName': input['RangeKeyName'],

'KeyType': 'RANGE'

})

attribute\_definitions.append({

'AttributeName':input['RangeKeyName'],

'AttributeType': input['RangeKeyType']

})

table = dynamodb.create\_table (

TableName = input['TableName'],

KeySchema = key\_schema,

AttributeDefinitions = attribute\_definitions,

ProvisionedThroughput={

'ReadCapacityUnits':input['ReadCapacityUnits'],

'WriteCapacityUnits':input['WriteCapacityUnits']

}

)

* **DeleteTable**: Function dùng để xóa bảng trong DynamoDB

{

"TableName": "Employees"

}

===

import json

import boto3

dynamodb = boto3.resource('dynamodb')

def lambda\_handler(event, context):

table\_name = json.loads(event['Records'][0]['body'])['TableName']

table = dynamodb.Table(table\_name)

res = table.delete()

return res

* **CreateItem:** Function dùng để tạo item trong DynamoDB

{

"TableName": "Students",

"uid": "012222222",

"Name": "Trung",

"Age": 20,

"Male": true

}

===

import json

import boto3

dynamodb = boto3.resource('dynamodb')

def lambda\_handler(event, context):

# input = event

input = json.loads(event['Records'][0]['body'])

table\_name = input['TableName']

items = {}

for key, val in input.items():

if key != "TableName":

items[key] = val

table = dynamodb.Table(table\_name)

response = table.put\_item(

Item = items

)

return response

* **UpdateItem:** Function dùng để cập nhật các item trong DynamoDB

{

"TableName": "Students",

"Key": {

"uid": "012222222"

},

"Age": 20,

"Male": true,

"New": "Trung"

}

===

import json

import boto3

dynamodb = boto3.resource('dynamodb')

client = boto3.client('dynamodb')

def lambda\_handler(event, context):

######################################

# Get table

input = event

table\_name = input['TableName']

table = dynamodb.Table(table\_name)

######################################

# Get keys

key\_schema = client.describe\_table(TableName=table\_name)['Table']['KeySchema']

hash\_key = ""

range\_key = ""

for key in key\_schema:

if(key['KeyType']=='HASH'):

hash\_key = key['AttributeName']

elif(key['KeyType']=="RANGE"):

range\_key = key['AttributeName']

#######################################

attr\_values = {}

attr\_names = {}

update\_expression = "SET "

i = 0

for key, val in input.items():

if ((key != "TableName") and (key != "Key")):

i+=1

k1 = '#name'+str(i)

k2 = ':val'+str(i)

attr\_names[k1] = key

attr\_values[k2] = val

update\_expression = update\_expression + k1 + " = " + k2 + " , "

update\_expression = update\_expression[:-2]

response = table.update\_item(

Key = input['Key'],

UpdateExpression = update\_expression,

ExpressionAttributeValues = attr\_values,

ExpressionAttributeNames = attr\_names

)

return response

* **DeleteItem:** Function dùng để xóa các item trong DynamoDB

{

"TableName": "Students",

"Key": {

"uid": "012222222"

}

}

===

import json

import boto3

dynamodb = boto3.resource('dynamodb')

client = boto3.client('dynamodb')

def lambda\_handler(event, context):

######################################

# Get table

# input = event

input = json.loads(event['Records'][0]['body'])

table\_name = input['TableName']

table = dynamodb.Table(table\_name)

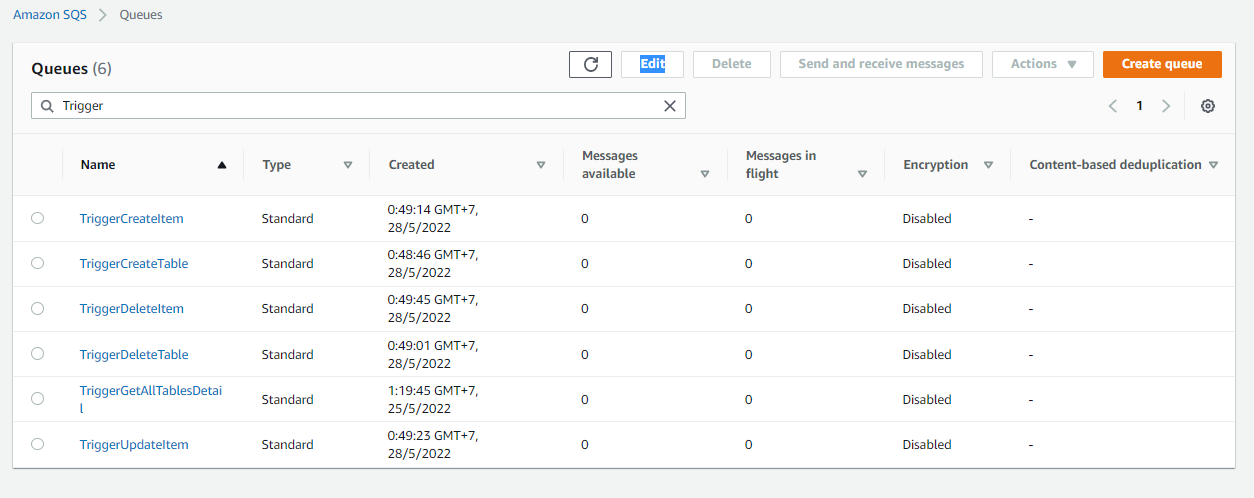
#######################################

response = table.delete\_item(Key = input['Key'])

return response

### *2.1.2 Cài đặt Simple Queue Service:*

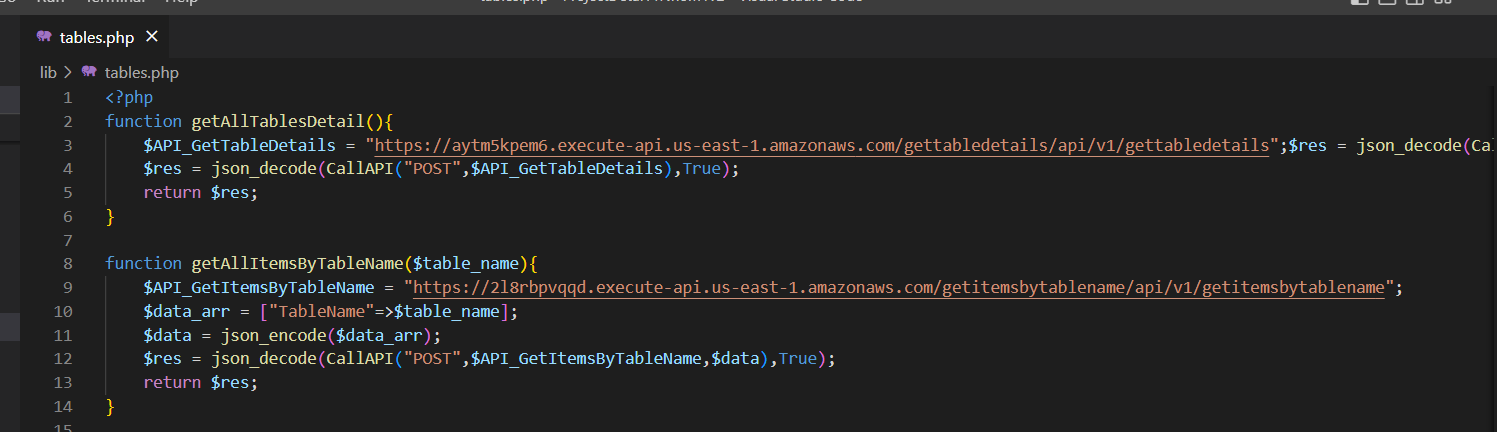
Lần Lượt tạo 5 SQS và kết nối đến các Function Lamda đã tạo:

* TriggerCreateItem: trigger đến Lamda tên CreateItem
* TriggerCreateTable: trigger đến Lamda tên CreateTable
* TriggerDeleteItem: trigger đến Lamda tên DeleteItem
* TriggerDeleteTable: trigger đến Lamda tên DeleteTable
* TriggerUpdateItem: trigger đến Lamda tên UpdateItem
* 

### *2.1.3 Setup Web (PHP):*

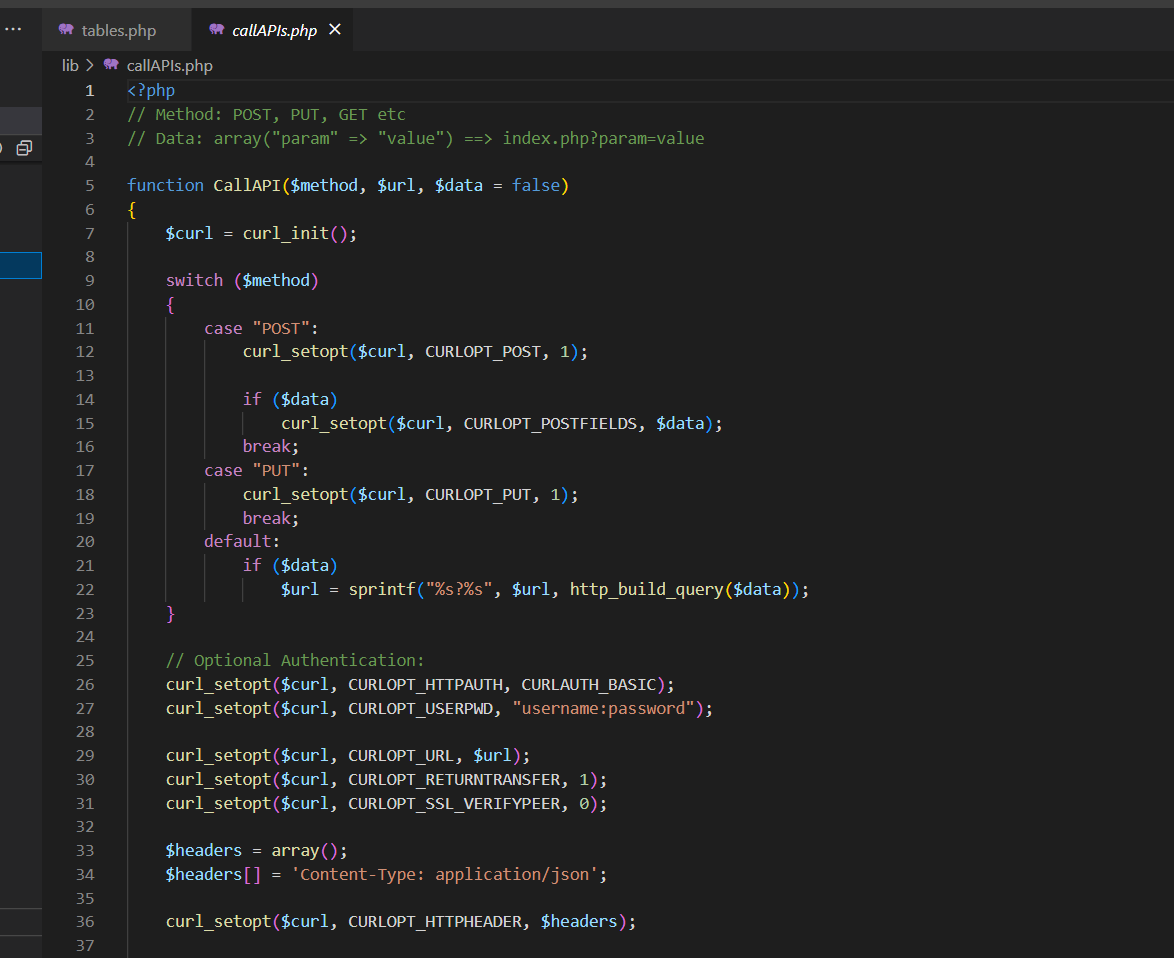
* Source Code:

**Tất cả các hàm sử dụng API để tương tác với Dynamodb**



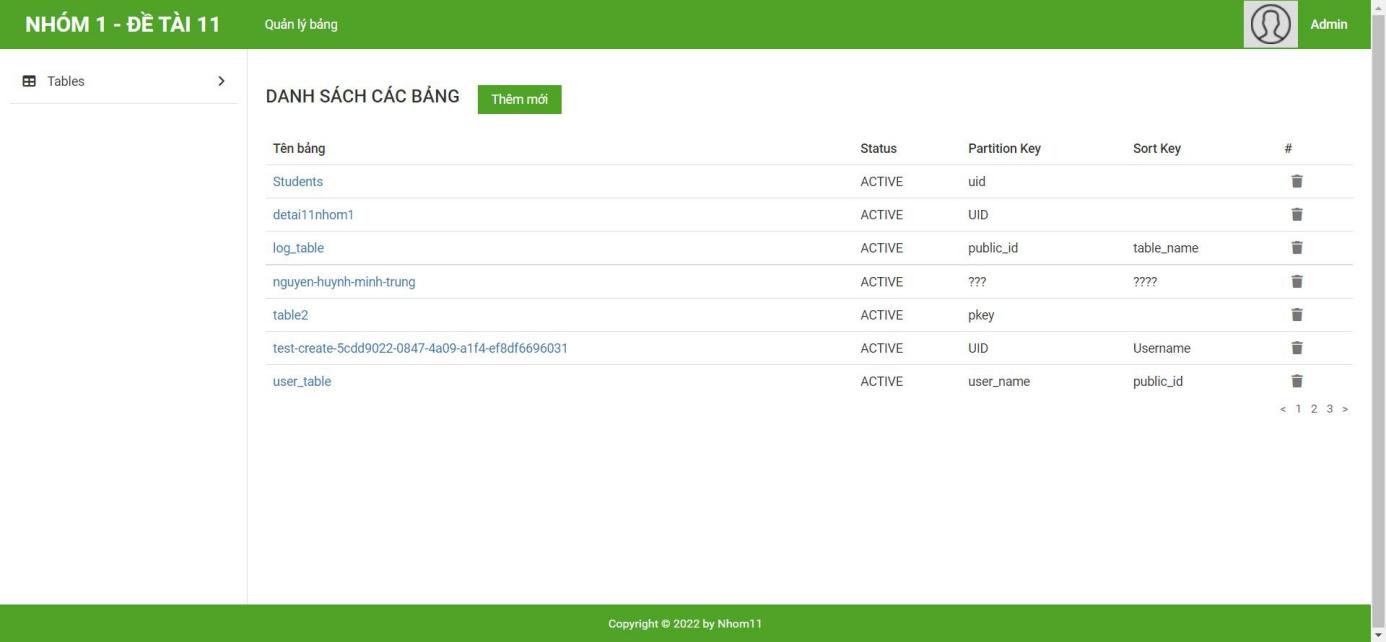


**Hàm Call API cho phép gửi request đến một API và nhận kết quả trả về của API đó**

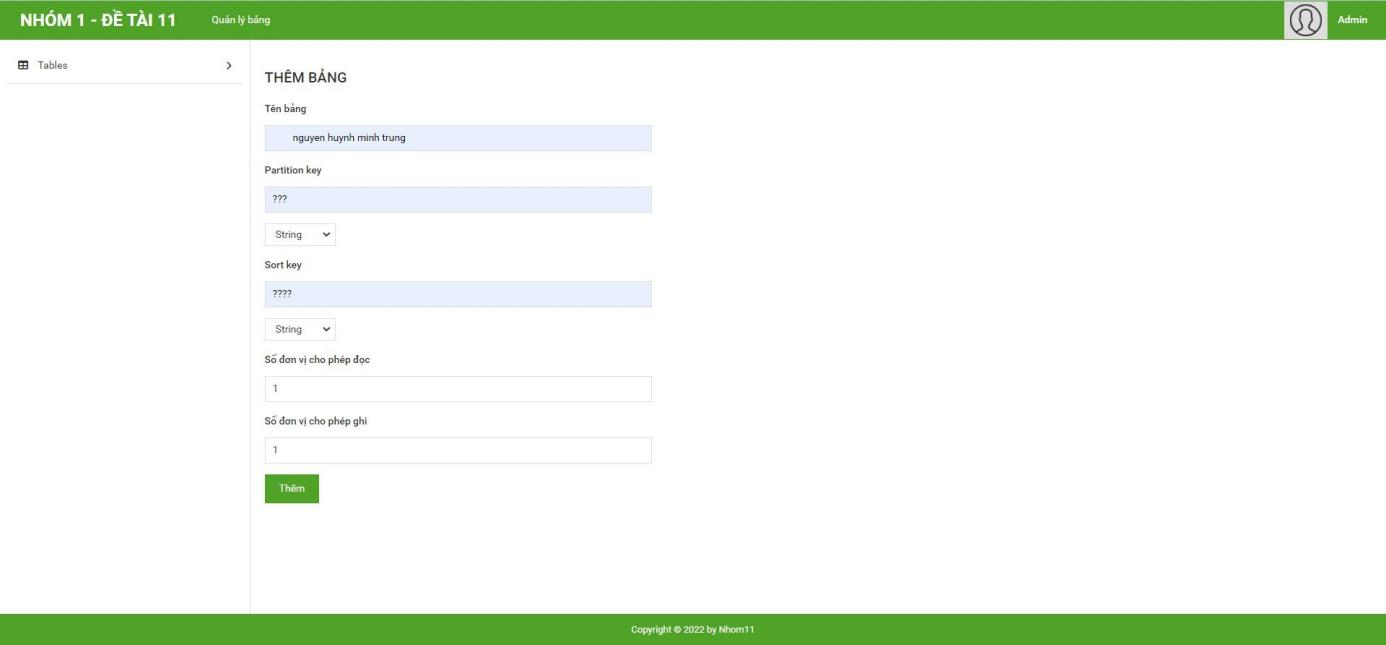


* Giao Diện:

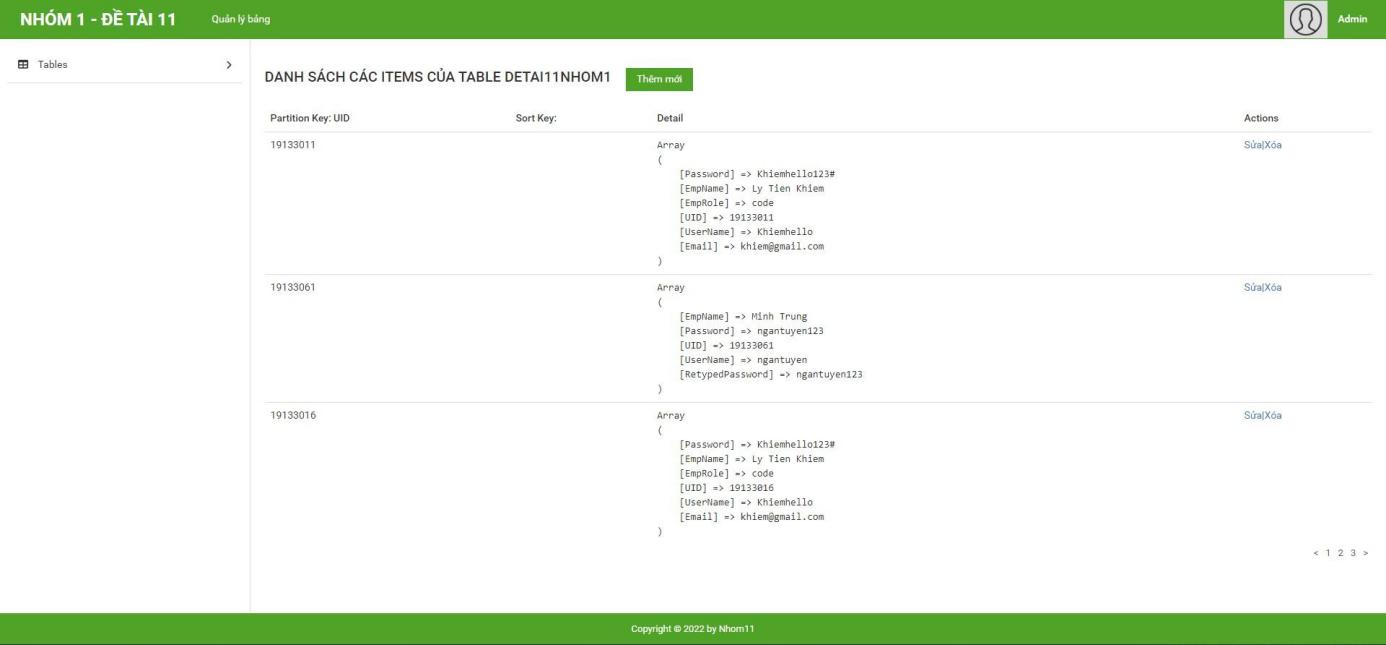
Giao diện hiển thị danh sách bảng



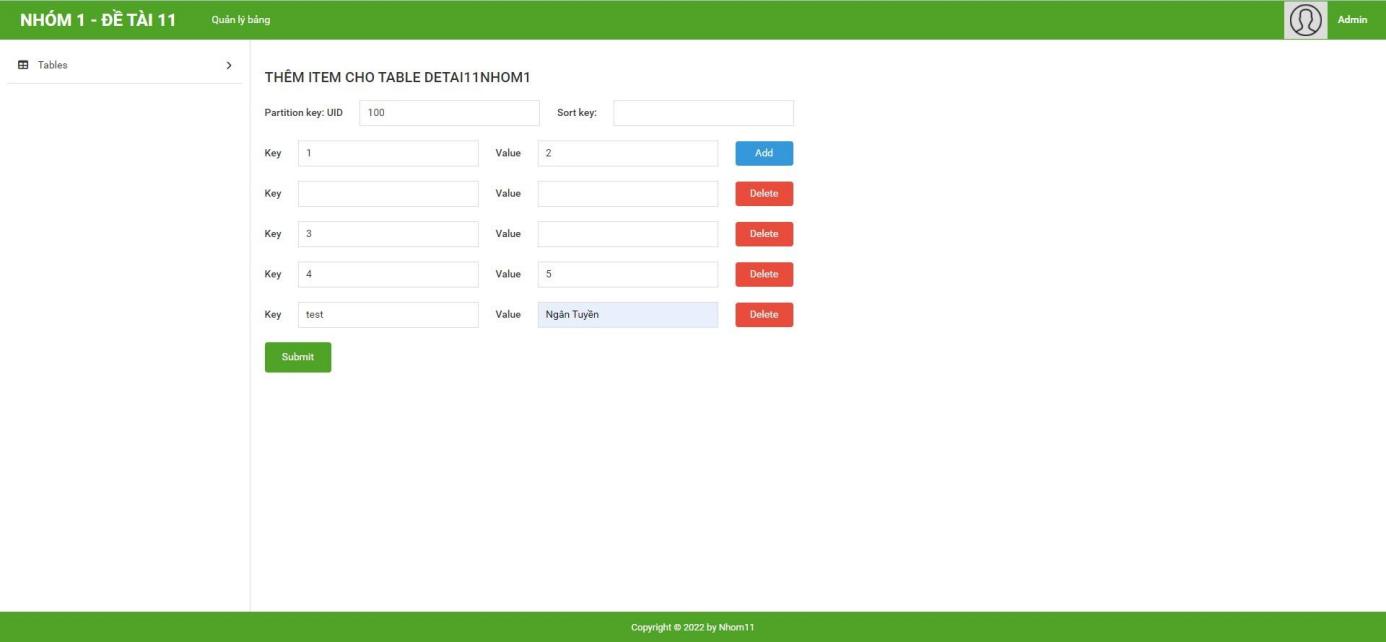
Giao diện thêm bảng



Giao diện hiển thị danh sách các Items trong bảng



Giao diện thêm/Sửa Item của bảng



## **2.2 Triển khai Project:**

### *2.2.1. Local:*

Bước 1: Chuẩn bị môi trường

Vào LearnLab--> start lab --> Aws Detail --> AWS CLI --> show copy bảng credentials

Vào thư mục C:\Users\tên máy\ .aws\credentials

Paste AWS CLI vừa copy file credentials

Tải và cài đặt XAMPP và Git

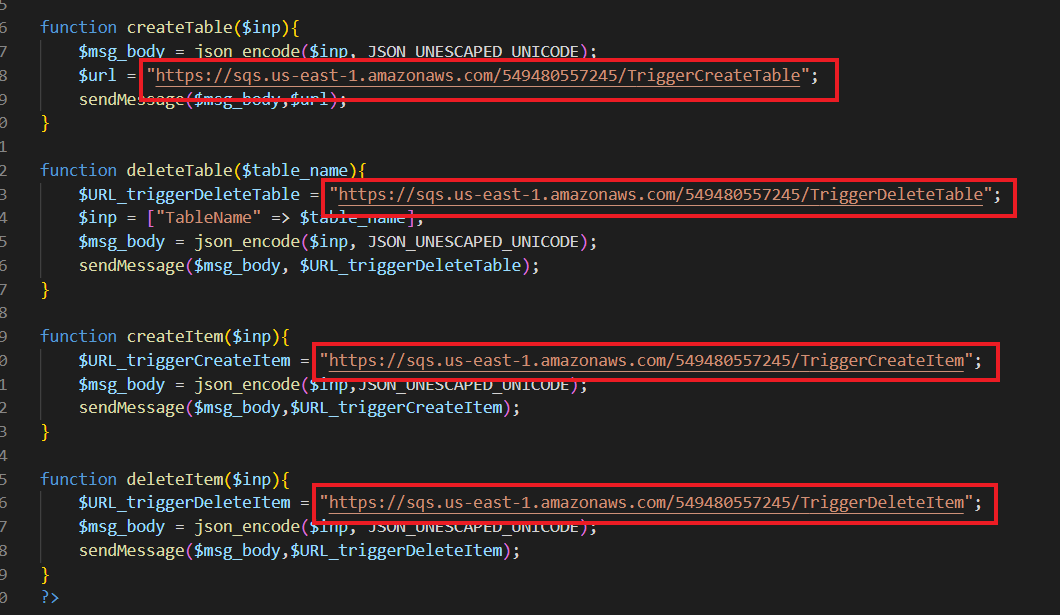
Bước 2: Cấu hình lại đồ án

git clone https://github.com/cr49/ProjectDetai11.git vào thư mục htdocs của XAMPP

**Tạo API bằng SQS:**

Vào AWS Tạo các SQS và Lamda theo file Lamda.txt

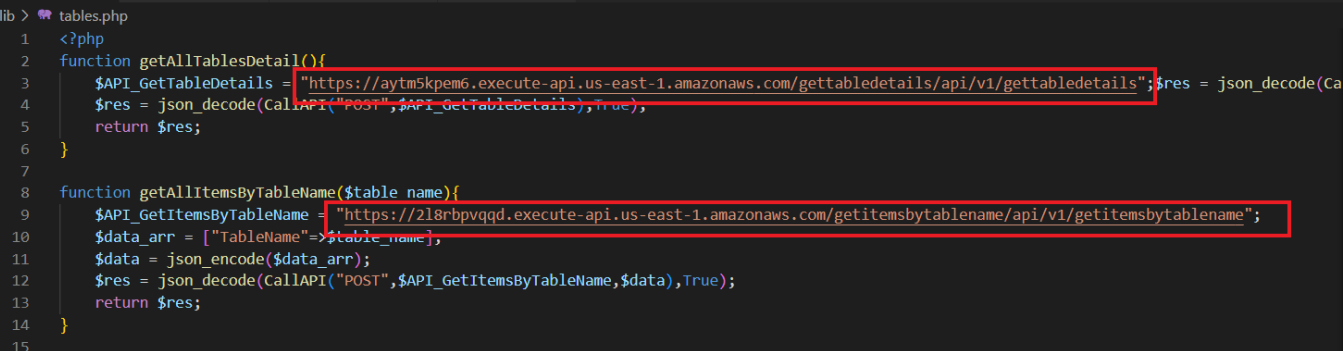
Chú ý Copy URL các SQS vừa tạo và past vào file lib/tables.php trong thư mục project để thay thế các URL của SQS



**Tạo API bằng API Gateway**

Vào AWS tạo các Lamda function theo API.txt

Sửa đường dẫn API tương tự SQS



Bước 3: Chạy

Mở XAMPP và start Apache

mở trình duyệt http://localhost/ProjectDetai11

### *2.2.2. EC2:*

* Các Bước thực hiện

Tạo EC2 với hệ điều hành window, mở port http và alltraffic

Tạo Elastic IP kết nối đến EC2

Connect EC2 bằng remote Desktop Connection

Trên window của EC2 vào Window Security --> Firewall & network protection --> Windows Defender Firewall Properties --> tag Public Profile --> Allow Inbound connections --> Apply--> ok

Dowload XAMPP và GIT

Mở git bash trong thư mục htdocs của XAMPP vừa tải về và

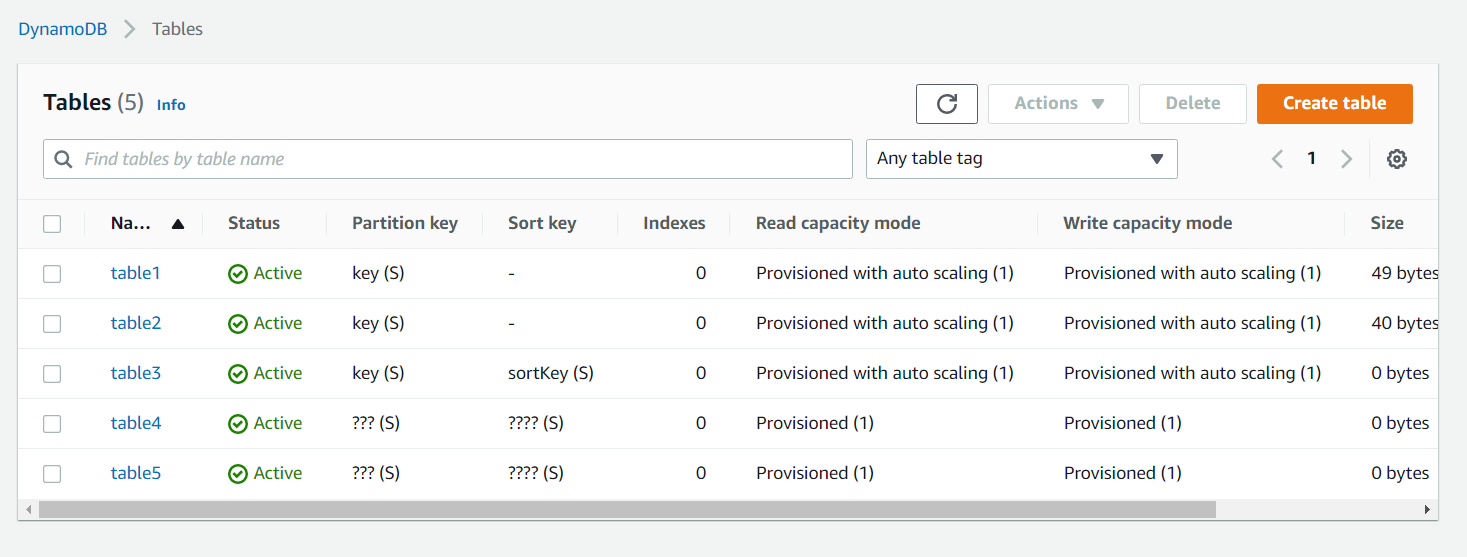
git clone https://github.com/cr49/ProjectDetai11.git

Copy .aws của local máy tính và past vào C:/user/Administrator của máy window EC2

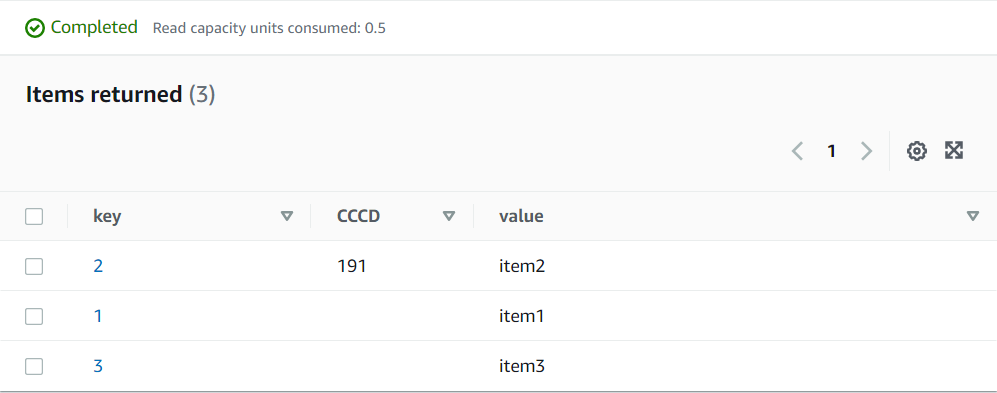
Mở XAMPP start apache và mở trình duyệt vào http://<ip máy EC2>/ProjectDetai11 => Deploy thành công

## **2.3 Thiết kế DynamoDB minh họa cho việc tương tác:**

# Danh sách các bảng



Danh sách Items của một bảng (table1)



# **PHẦN KẾT LUẬN**

## **Kết quả đạt được**

Sau một thời gian nghiên cứu và thực hiện đề tài “***Xây dụng ứng dụng trên AWS cho phép tạo database và cung cấp API để thêm, sửa, xóa trên Database***”, nhóm chúng em đã đạt được những kết quả như sau:

### *1.1. Kiến thức tìm hiểu được*

Nắm bắt được các kiến thức cũng như những cách hoạt động của các dịch vụ Lamda Function, SQS, API, DynamoDB và áp dụng kiến thức để thiết kế và xây dựng một mô hình tương tác với database.

Nắm bắt được mô hình hoạt động, cách các luồng Data được di chuyển, thiết kế và xây dựng ứng dụng để tương tác, thực hiện tích hợp API vào ứng dụng để tạo tương tác với Database.

### *1.2. Chương trình đã làm được*

Xây dựng hoàn chỉnh một ứng dụng tương tác với database (DynamoDB) bằng code PHP, Html, Css và các dịch vụ của AWS: SQS, Lamda Function (Python), API,DynamoDB với các chức năng cơ bản như:

* Thêm, xóa, đọc các table trong DynamoDB.
* Thêm, sửa, xóa, đọc các Item trong Table của DynamoDB.

## **Ưu điểm**

* Ứng dụng dễ tương tác, thân thiện với người sử dụng.
* luồng xử lý dữ liệu rõ ràng vì được xử lý thông qua các API
* Lưu trữ cũng như xử lý được lượng dữ liệu lớn nhanh chóng do cơ sở dữ liệu được quản lý bởi AWS.

## **Nhược điểm**

* Khó Setup.
* Các Function Lamda được theo một mẫu sẵn nên cần phải dành thời gian tìm hiểu.
* Tốn phí vì sử dụng dịch vụ của AWS và phụ thuộc phần lớn vào AWS trong vấn đề chạy code.

## **Hướng phát triển**

* Cải thiện giao diện
* Thêm các tính năng nâng cao khác