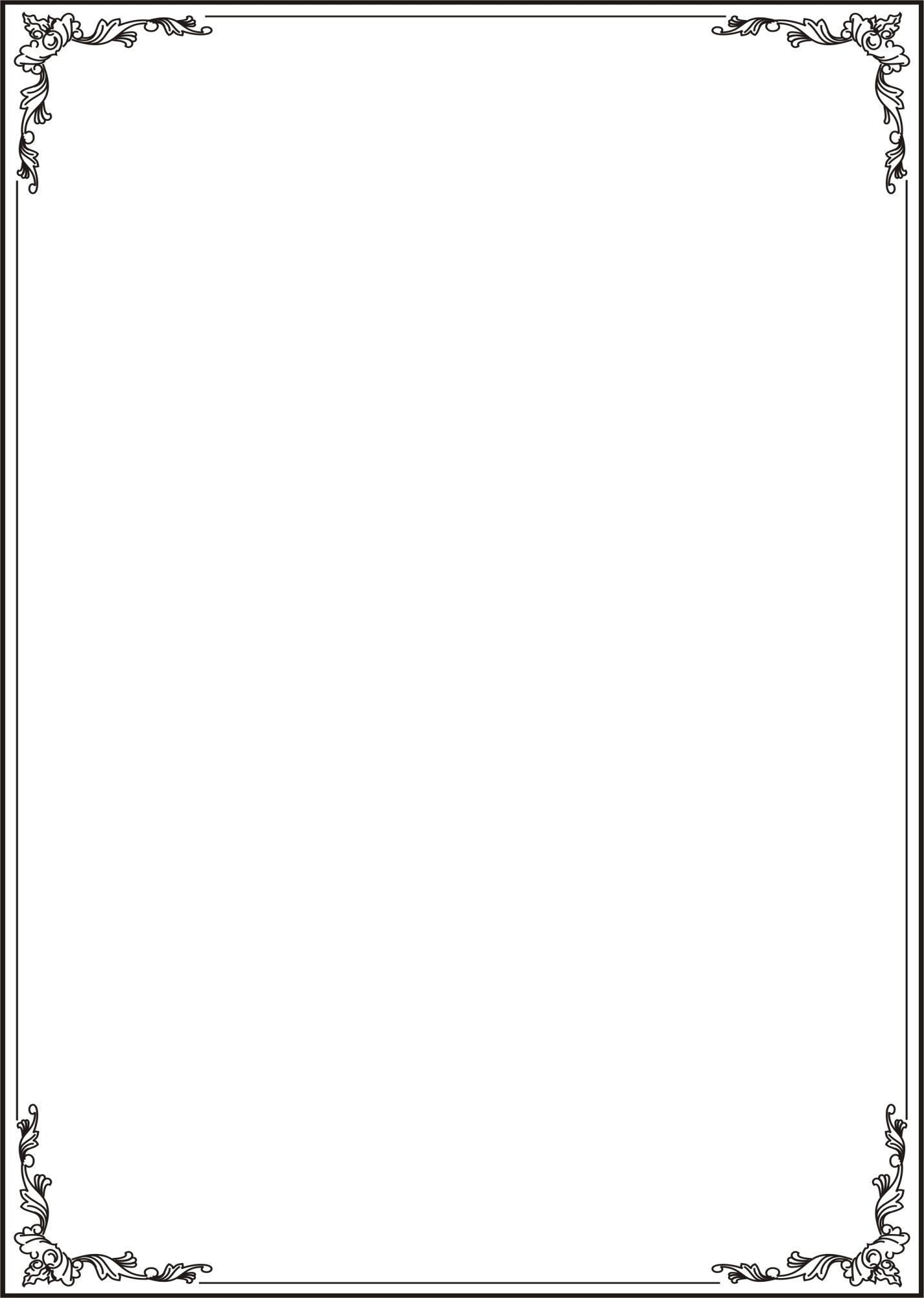
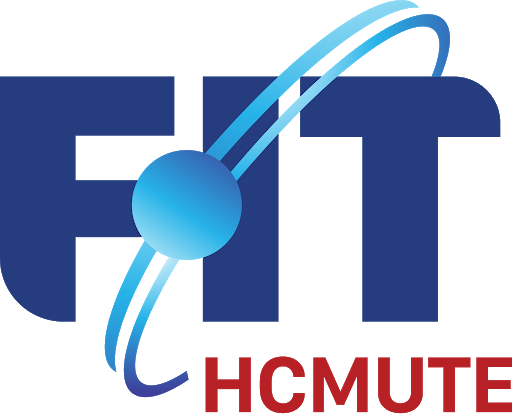
**.**

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

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN ĐIỆN TOÁN ĐÁM MÂY**

-----🙞🙜🕮🙞🙜-----

****

****



****

**KHÓA 2020-2024**

**DANH SÁCH NHÓM**

HỌC KÌ I, NĂM HỌC: 2022-2023

**Đề tài 11:** xây dựng ứng dụng trên aws cho phép tạo database và cung cấp api để thêm, xóa sửa trên database.

| STT | Họ và tên | MSSV | Mức độ đóng góp |
| --- | --- | --- | --- |
| 1 | Lê Hải Đăng | 20110234 | 100% |
| 2 | Trần Minh Mẫn | 20110301 | 100% |
| 3 | Trần Công Minh | 20110677 | 100% |

**Nhận xét của giảng viên:**

...................................................................................................................................................................................................................................................................................... ...................................................................................................................................................................................................................................................................................... ...................................................................................................................................................................................................................................................................................... ...................................................................................................................................................................................................................................................................................... ...................................................................................................................................................................................................................................................................................... ......................................................................................................................................................................................................................................................................................

*Ngày ........tháng........năm.......*

Giáo viên chấm điểm

**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**



# **MỤC LỤC**

[**MỤC LỤC**](#_heading=h.26in1rg) **4**

[**PHẦN MỞ ĐẦU**](#_heading=h.n7wdlh6wt3ay) **5**

[**1. Tính cấp thiết của đề tài**](#_heading=h.1y810tw) **5**

[**2. Đối tượng nghiên cứu**](#_heading=h.pamn9ef1kb9) **5**

[**3. Phạm vi nghiên cứu**](#_heading=h.4i7ojhp) **5**

[4. Mục tiêu đề tài](#_heading=h.3jldb52pkcm8) 5

[**5. Kết quả dự kiến đạt được**](#_heading=h.2xcytpi) **5**

[**PHẦN NỘI DUNG**](#_heading=h.1ci93xb) **6**

[**CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**](#_heading=h.147n2zr) **6**

[**1. Tổng quan về Lambda**](#_heading=h.3o7alnk) **6**

[**2. Các tính năng của Lambda**](#_heading=h.23ckvvd) **6**

[**3. Các dịch vụ liên quan đến Lambda**](#_heading=h.ihv636) **7**

[**CHƯƠNG 2: THỰC HIỆN ĐỒ ÁN**](#_heading=h.hhve4vulaixm) **10**

[**1. Phần Database**](#_heading=h.gnrdodjv1tev) **10**

[**2. Phần Back-End**](#_heading=h.dt90q9lh5dg8) **10**

[**3. Phần Front-End**](#_heading=h.yjmjv6581mlo) **24**

[**4. Hướng dẫn cài đặt và chạy dự án trên Local**](#_heading=h.ty2h6zxaj5pp) **31**

[**5. Hướng dẫn cài đặt và chạy dự án trên AWS**](#_heading=h.1ifft2np2oyp) **32**

[**PHẦN KẾT LUẬN**](#_heading=h.3tbugp1) **33**

[**1. Kết quả đạt được**](#_heading=h.28h4qwu) **33**

[1.1. Kiến thức tìm hiểu được](#_heading=h.nmf14n) 33

[1.2. Chương trình đã làm được](#_heading=h.37m2jsg) 34

[**2. Đánh giá và nhận xét**](#_heading=h.46r0co2) **34**

[2.1. Phần Back-End và Database](#_heading=h.ppo1pvjxzt6h) 34

[2.2. Phần Front-End](#_heading=h.xbchle1cnr7g) 34

[**Tài liệu tham khảo**](#_heading=h.p6ulai49dgvq) **35**

# PHẦN MỞ ĐẦU

## 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. Mục tiêu đề tài

Sử dụng được các dịch vụ AWS Lambda để xây dựng phần Back-End, dịch vụ AWS DynamoDB để tạo ra được phần Database cho web với mục tiêu là sử dụng các API để thực hiện các nghiệp vụ tạo, thêm, xóa, sửa cho các Table và các Record của table trong Database. Về phía Front-End sử dụng công nghệ ReactJS để xây dựng giao diện giúp người dùng dễ dàng thao tác, thực hiện trên website. Ngoài ra, nhóm sẽ sử dụng AWS S3 để lưu trữ phần giao diện.

## 5. Kết quả dự kiến đạt được

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

# 

# PHẦN NỘI DUNG

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## Tổng quan về Lambda

AWS Lambda là nơi để cho lập trình viên upload các function code của mình dưới dạng các ngôn ngữ được hỗ trợ, và AWS Lambda sẽ giúp người lập trình chạy các đoạn code đó dựa trên các thư viện có sẵn của AWS, đồng thời AWS Lambda sẽ lưu lại các tiến trình thay đổi của function và lưu vào CloudWatch.

AWS Lambda được hỗ trợ và quản lý bởi đội ngũ AWS nên các function sẽ thực hiện code của lập trình viên khi cần, và có khả năng tự động mở rộng, từ một vài request đến hàng trăm ngàn request trên giây. Nhờ vào các ưu điểm này mà Lamda có thể dễ dàng sử dụng để xây dựng các ứng dụng cùng với các dịch vụ khác của Lamda như S3, DynamoDB,... với các hiệu suất và độ bảo mật cao

## 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:

* Khả năng mở rộng

AWS Lambda cho phép lập trình viên dễ dàng kết hợp với các dịch vụ khác như S3, DynamoDB,.... vì vậy có thể dễ dàng kích hoạt các Lambda Function, khi có các luồng dữ liệu hoặc tài nguyên có sự thay đổi giúp lập trình viên hiểu được các tiến trình của các Request

* Xây dựng Back-end

AWS Lambda cho phép lập trình viên có thể tạo các backend cho ứng dụng bằng cách sử dụng các API Lambda hoặc API endpoint của Amazon API Gateway. Lamda sẽ tiến thành xử lý các event thay vì cung cấp các event lên client, nhờ đó giúp giảm tiêu hao các tài nguyên, dễ dàng bảo trì cũng như cập nhật hơn.

Bên cạnh đó còn có tính năng Lambda Function URL cho phép thêm điểm cuối HTTPS vào bất kỳ hàm Lambda nào và tùy chọn định cấu hình. Dễ hiểu hơn thì trước đây chúng ta sử dụng mô hình rất phổ biến trước đó là chúng ta có API Gateway sau đó đên Lambda rồi từ Lambda này chúng ta kế nối đến các dịch vụ khác của AWS hay database (DynamoDb, Aurora,…) nhưng thay vì như vậy chúng ta xây dựng một mô hình có Lambda và Lambda này được truy xuất thẳng từ bên ngoài không phải thông qua API Gateway.

* Sử dụng code riêng

AWS Lambda hỗ trợ đa dạng các ngôn ngữ cũng như cung cấp đầy đủ các API Runtime, ngoài ra AWS còn hỗ trợ lập trình viên có thể sử dụng bất kỳ thư viên, kể cả các thư viện thuần, đa dạng các framework, lập trình viên cũng có thể đóng gói các source code dưới dạng các class của Lam Dam, dễ dàng chia sẻ cũng như quản lý các code trên nhiều Function khác nhau.

* Các quản lý được tự động hoàn toàn

Lamda được quản lý hoàn toàn bởi các đội ngũ AWS nên có độ sẵn sàng và khả năng chịu lỗi cao, ngoài ra lập trình viên sẽ không cần cập nhật các bản vá cũng như cập nhật dung lượng, bổ sung máy chủ mới. AWS sẽ thực hiện tất cả các hoạt động bảo trì, vá lỗi, cung cấp chức năng lưu trữ nhật ký của các quá trình Lamda thông qua Amazon CloudWatch

## 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 HTTP và kích hoạt đến các Lambda 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 là cơ sở dữ liệu NoSQL, được lưu trữ trong SSD thực hiện nhanh và linh hoạt trên các ứng dụng khác nhau có yêu cầu độ trễ mili giây ở bất kỳ quy mô nào, những ứng dụng này có thể hoạt động trên quy mô nhỏ sau đó mở rộng ra để hỗ trợ nhiều petabyte dữ liệu, xử lý hàng ngàn request credit hoặc ghi mỗi giây.

DynamoDB được quản lý đầy đủ hỗ trợ cả mô hình dữ liệu dưới dạng tài liệu và cặp value-key.

Mô hình dữ liệu rất linh hoạt và hiệu suất cao phù hợp cho nhiều loại ứng dụng, nền tảng khác nhau.

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

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 để tính tiền các dịch vụ hoặc các tài nguyên đã sử dụng trên AWS.

**Tại sao nên sử 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.

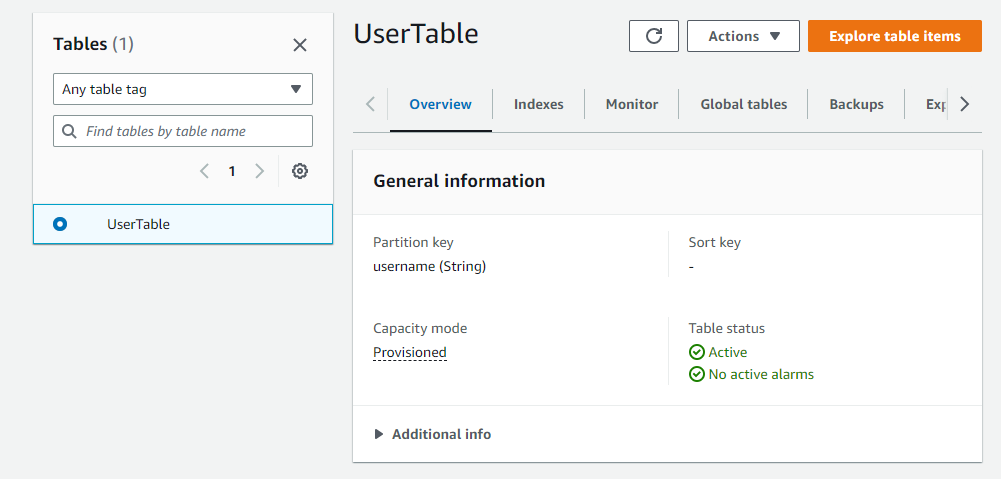
Lưu trữ các dữ liệu không liên tục: Do đặc tính là một cơ sở dữ liệu không quan hệ mà DynamoDB có đặc tính rất phù hợp với những đối tượng không có cấu trúc cố định và thay đổi theo thời gian.

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 ĐỒ ÁN

# Phần Database

Database sẽ được tạo sẵn một bảng có tên là UserTable với Partition Key là username với kiểu dữ liệu String và Sort Key là password. Table này lưu những tài khoản đã đăng ký, những tài khoản này sẽ tiến hành các thao tác thêm xóa table, thêm sửa xóa các item của table mà tài khoản này sẽ tạo.



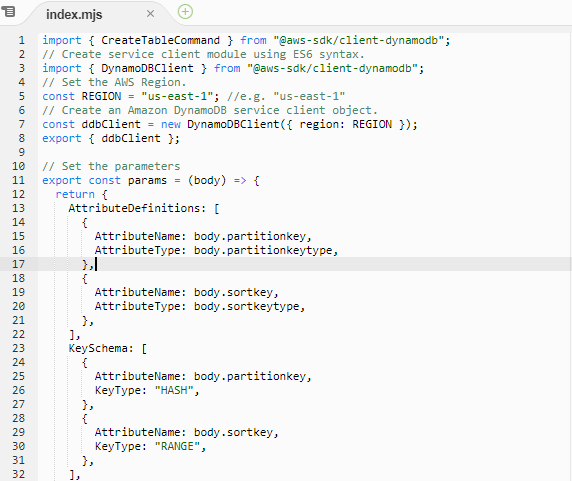
# Phần Back-End

Phần Back-End sử dụng dịch vụ Lambda của AWS, cụ thể là tính năng Lambda URL được viết bằng JavaScript

Các API hỗ trợ thêm sửa xóa dữ liệu trên DynamoDB

Source code của các chức năng:

* Chức năng tạo table: **create\_table:**



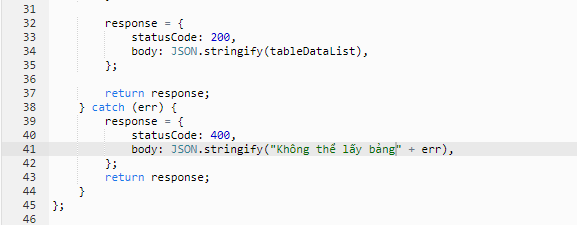


* import các module hỗ trợ tương tác với DynamoDb
* **REGION**: đặt khu vực, tại đây nhóm chọn khu vực “us-east-1”
* **AttributeDefinitions**: định nghĩa các thuộc tính cho khóa chính, ở đây khóa chính được được định nghĩa là sự kết hợp giữa Partition key và Sort key đều này giúp cho trong 1 table giá trị của Partition key trùng nhau là hoàn toàn bình thường nhưng phải có Sort key để phân biệt các item có nghĩa là khi truy xuất 1 item ta cần có Partition key và Sort Key.
* **KeySchema**: chỉ định các thuộc tính tạo nên khóa chính của table, ở đây 2 thuộc tính Partition key và Sort key được chỉ định làm khóa chỉnh của table, Partition key có vai trò HASH còn Sort key có vai trò RANGE
* **ProvisionedThroughput**:
* 1 RCUs: thông lượng đọc dữ liệu là 4KB/s
* 1 WCUs: thông lượng ghi dữ liệu là 1KB/s
* **TableName**: tên của table
* **StreamSpecification**: cấu hình Dynamodb Stream cho 1 bảng trong Dynamodb
* Sau khi nhận thông tin từ phía Client xử lý tạo table name cho table và tiến hành tạo table trên DynamoDB. Nếu thành công trả về table name đã được tạo thành công, ngược lại nếu tên table đã tồn tại thì trả về “Tên bảng đã tồn tại”
* Chức năng **delete\_table:**

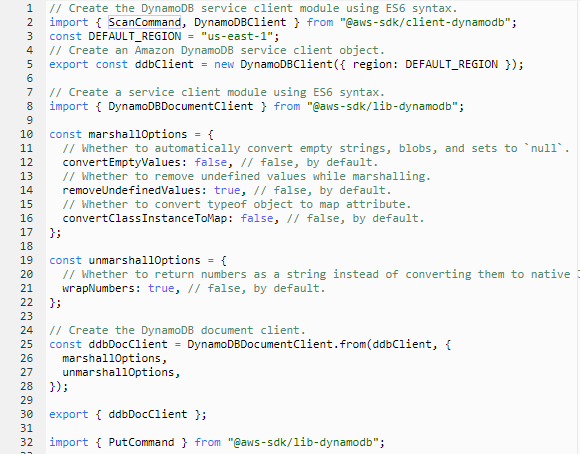
****

* Xóa table dựa vào tablename, sau khi xóa table thì toàn bộ các item trong table này đều sẽ bị xóa. Nếu xóa thành công thì trả về “Xóa thành công bảng + tên bảng”, ngược lại thì trả về “Không thể xóa bảng + tên bảng“.
* Chức năng **get\_all\_table\_by\_username:**

****

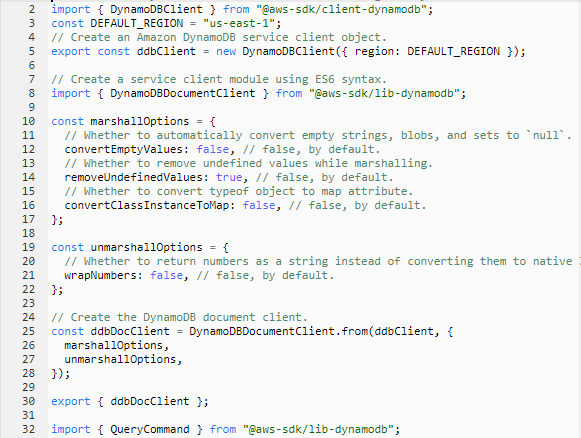
****

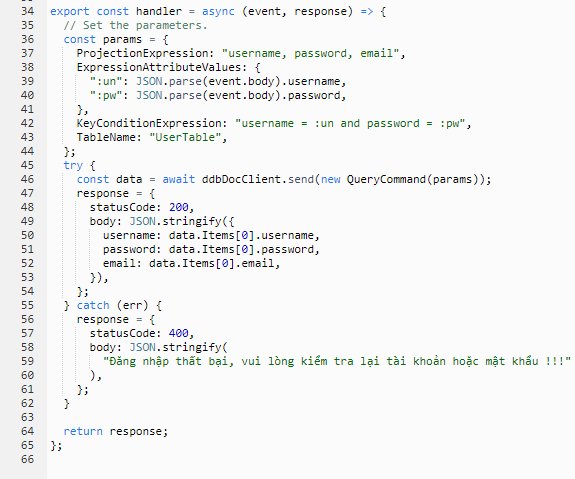
* import các module hỗ trợ tương tác với DynamoDb
* **ListTablesCommand:** dùng lấy ra các table tồn tại trong tài khoản AWS hiện tại
* Sau khi lấy được tất cả các table hiện tại kiểm tra xem table nào có chứa username của tài khoản chúng ta đăng nhập vào app bởi vì khi tạo table table được tạo có tên kèm theo username của tài khoản đã tạo tận dụng việc này để có thể lấy được các table theo từng tài khoản
* **DescribeTableCommand:** trả về thông tin của table bao gồm status, created, primary key và các indexes của table
* Nếu thành công sẽ trả về danh sách các table bao gồm tên, các thông tin liên quan đến table, ngược lại thì trả về “Không thể lấy bảng”
* Chức năng: **register:**

****

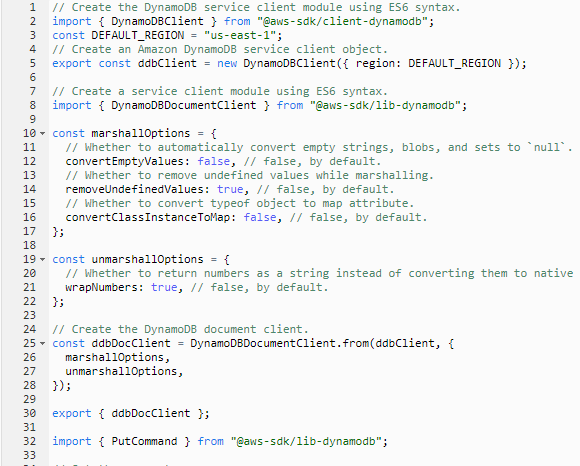
****

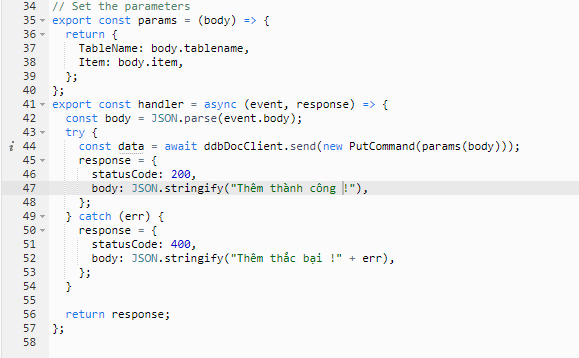
* import các module hỗ trợ tương tác với DynamoDb
* **PutCommand:** dùng để thêm hoặc cập nhật một item vào table, ở đây do đã chặn ở ngoài giao diện nên trong trường hợp này chỉ có thể thêm item và table không xảy ra trường hợp cập nhật lại table
* Nếu thành công thông tin tài khoản đăng ký sẽ được lưu và table UserTable, ngược lại thì báo lỗi
* Chức năng **login:**

****

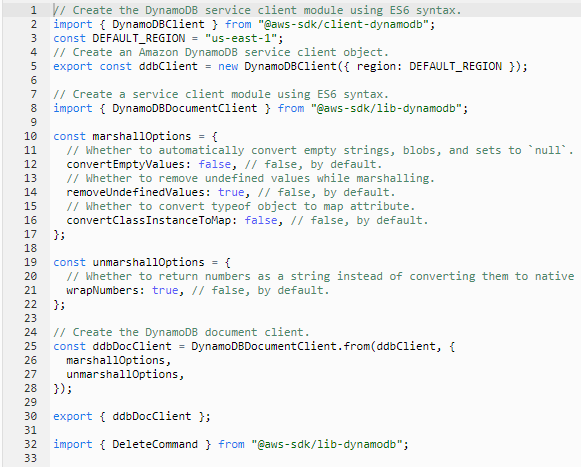
****

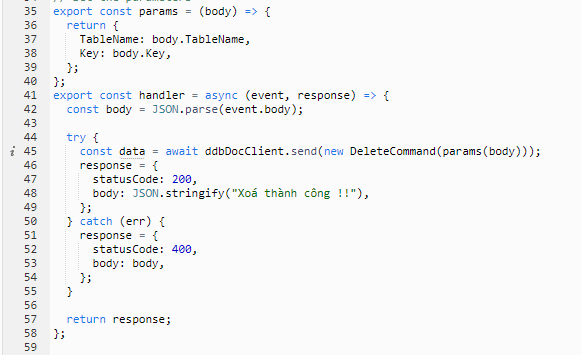
* import các module và cấu hình để hỗ trợ tương tác với DynamoDb
* **ProjectionExpression**: các thuộc tính cần lấy ra sau khi truy vấn, tại đây lấy ra username, password, email
* **ExpressionAttributeValues**: gán các giá trị nhận vào từ client vào các giá trị thuộc tính trong Dynamodb để tiến hành so sánh với các thuộc tính trong table
* **KeyConditionExpression**: cung cấp giá trị cụ thể cho username và password để tính hành so sánh và trả về kết quả
* **TableName**: thực hiện truy vấn trên bảng UserTable
* Nếu thành công trả về 1 item thỏa mãn điều kiện, ngược lại trả về thông báo “Đăng nhập thất bại, vui lòng kiểm tra lại tài khoản hoặc mật khẩu !!!”
* Chức năng **add\_or\_update\_item:**

****

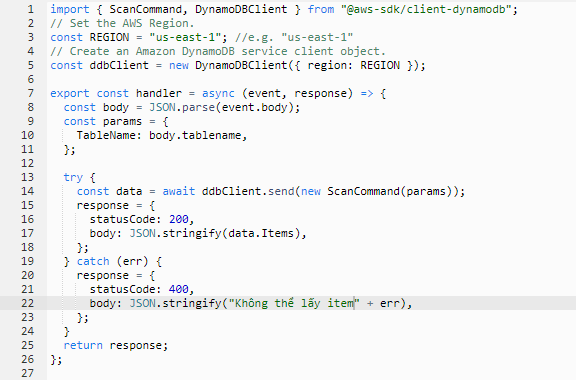
****

* import các module và cấu hình để hỗ trợ tương tác với DynamoDb
* **PutCommand:** dùng để thêm hoặc cập nhật lại item nếu primary key của item thêm vào trùng với item tồn tại trong table
* Dữ liệu nhận vào là tên table và item cần thêm vào database, item này chức các trường giá trị mà người dùng cần thêm vào.
* Nếu thành công item sẽ được lưu vào table có tên tương ứng, ngược lại thì báo lỗi
* Chức năng **delete\_item:**

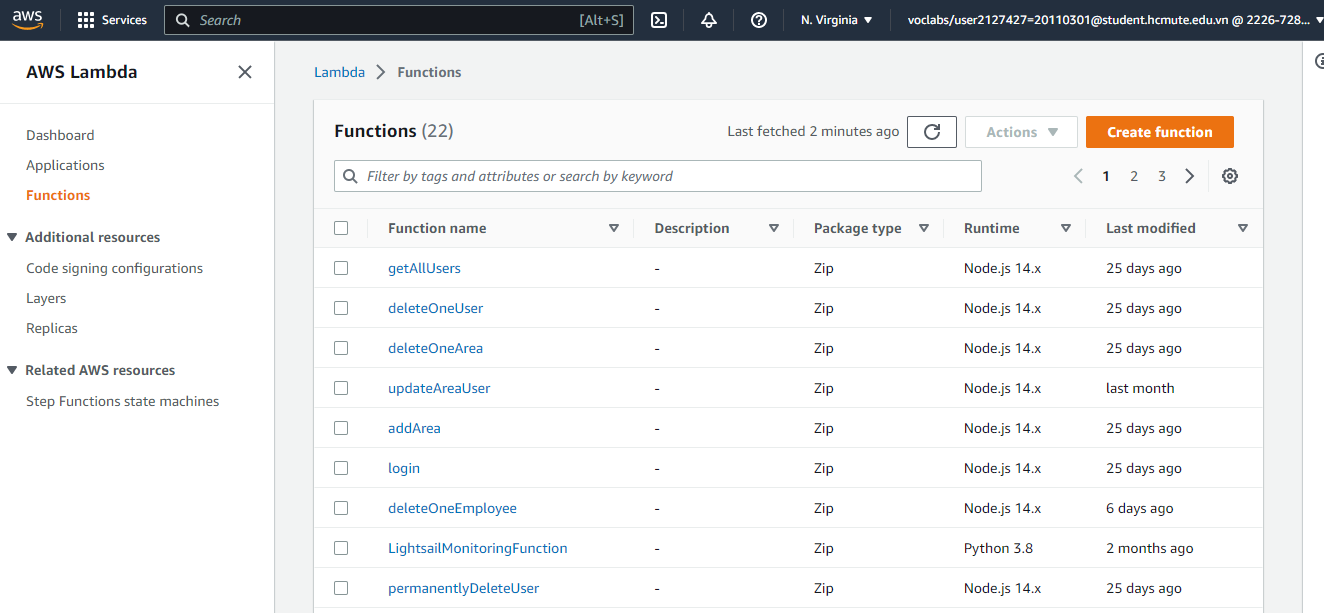
****

****

* import các module và cấu hình để hỗ trợ tương tác với DynamoDb
* Thực hiện xóa item trong table chỉ định dựa vào khóa chính của table đó
* Nếu thành công thì item trong table cần xóa sẽ được xóa, ngược lại sẽ báo lỗi
* Chức năng **get\_all\_item\_by\_table\_name:**

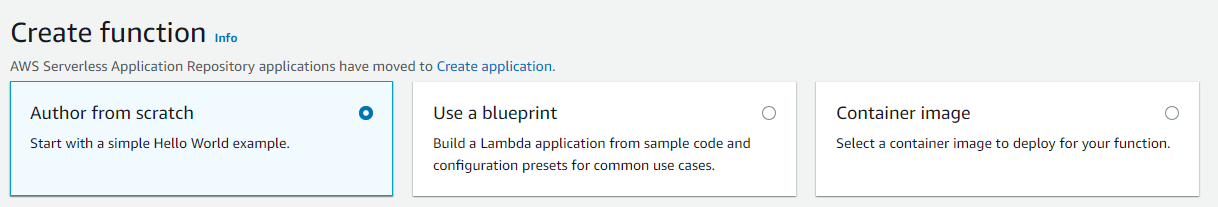
****

* import các module và cấu hình để hỗ trợ tương tác với DynamoDb
* **ScanCommand:** trả về một hoặc nhiều các item trong table dựa vào tên table mà chúng ta muốn lấy các item ra
* Thành công sẽ trả về danh sách các item có trong table mà chúng ta muốn truy xuất, ngược lại thì báo lỗi
* Cách cài đặt các Lambda Function trên AWS
* Vào AWS chọn Search, nhập “Lambda” rồi chọn chức năng Lambda

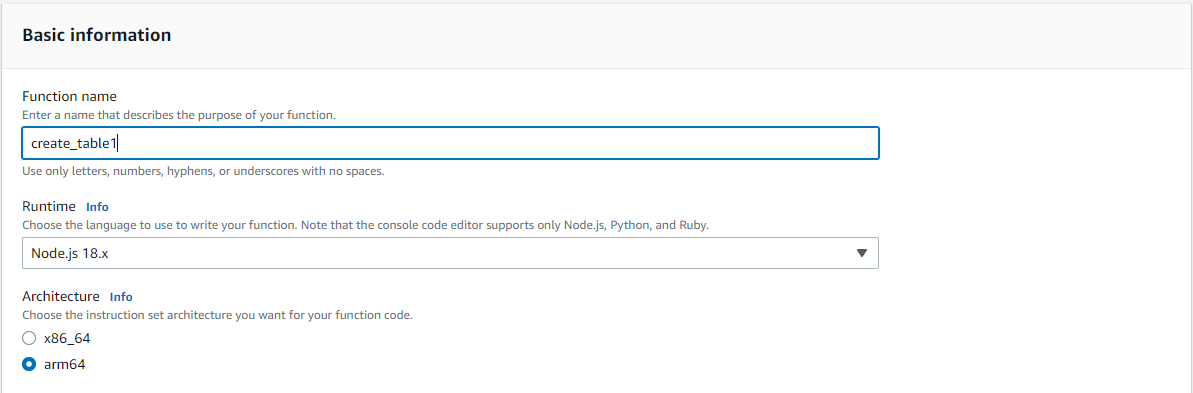


Chọn Create function, ta bắt đầu tạo function:

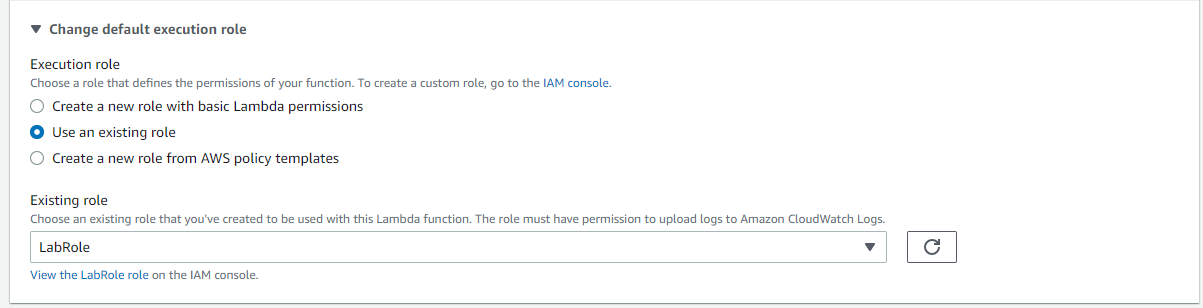
* Chọn **Author from scratch**

****

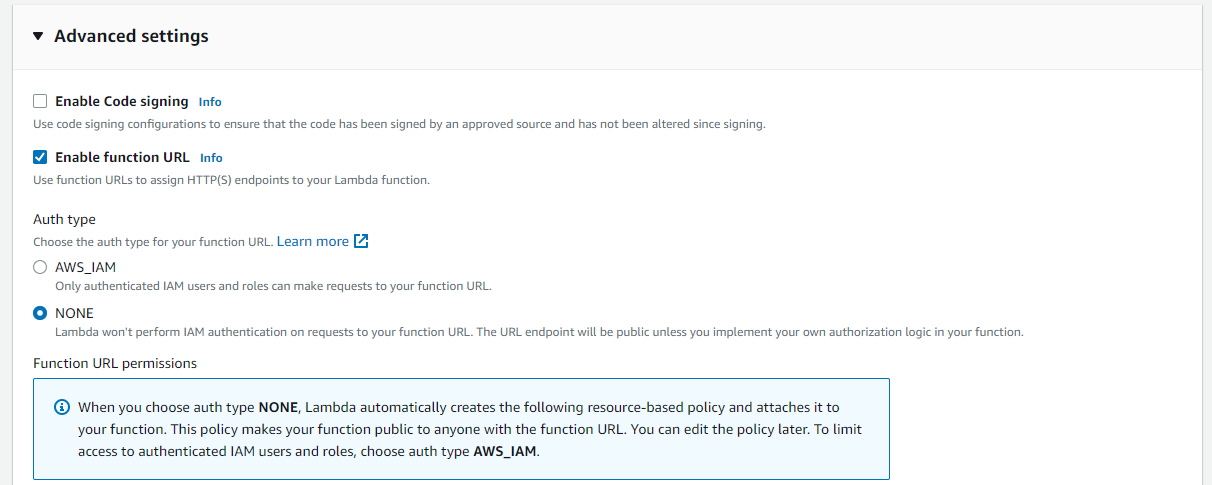
* Nhập **Function name**
* **Runtime** chọn Node.js 18.x
* **Architecture** chọn **arm64**



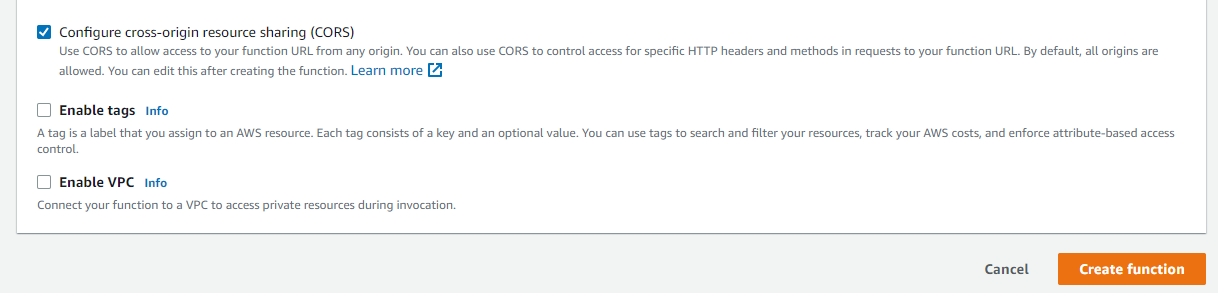
* Mở **Change default execution role**
* Chọn **Use an existing role**
* Existing role chọn **LabRole**

****

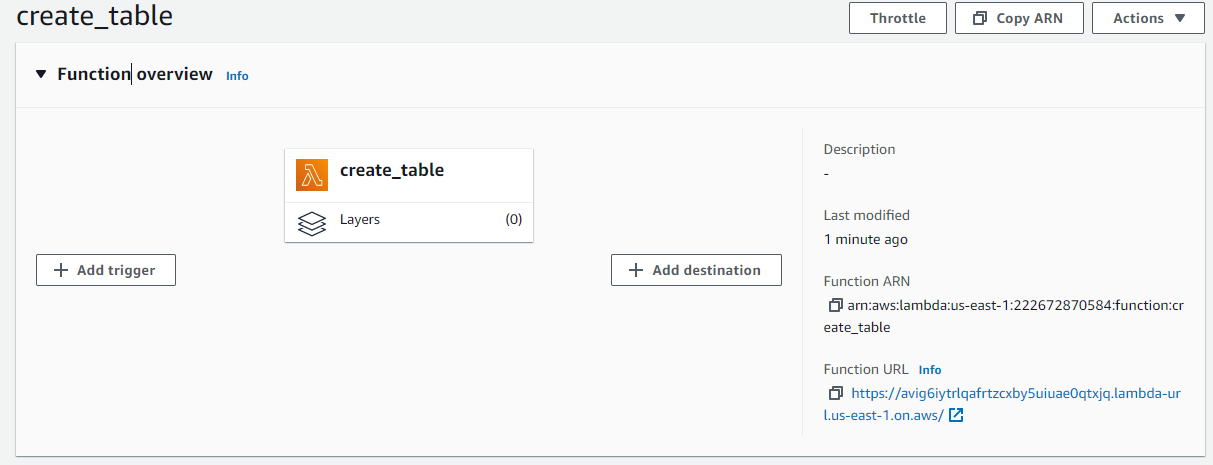
* Mở **Advanced Settings**
* Chọn **Enable function URL**
* **Auth type** chọn **None**

****

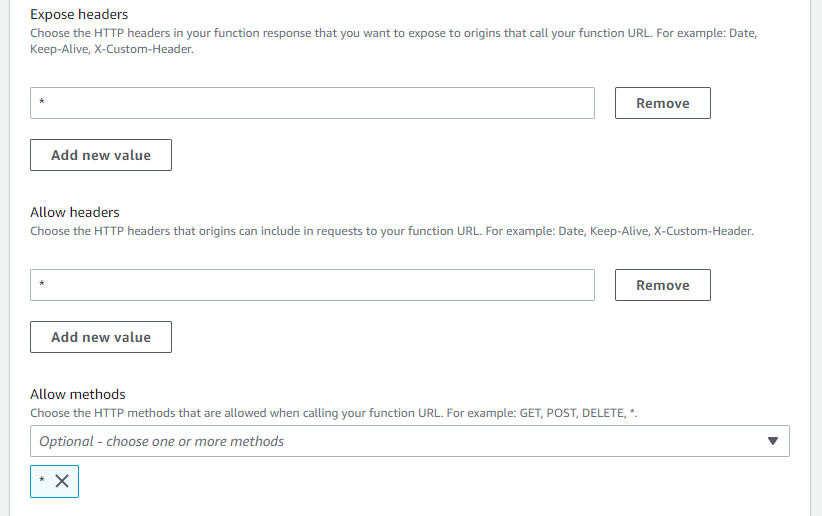
* Chọn **Configure cross-origin resource sharing (CORS)**
* Cuối cùng chọn **Create function**

****

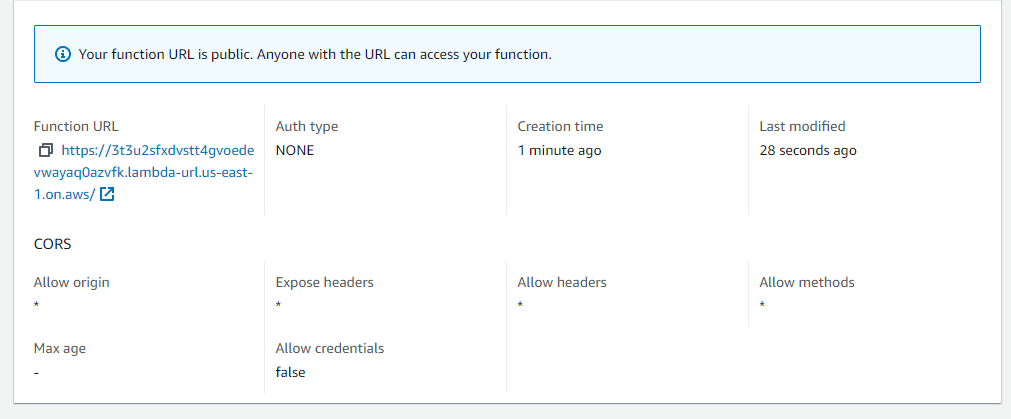
* Kết quả khi tạo một function



* Sau khi tạo function trên lambda ta cần config một số chỗ như sau: vào **Configuration -> Edit**



* Cuối cùng ấn **Save**, ta được kết quả như hình bên dưới



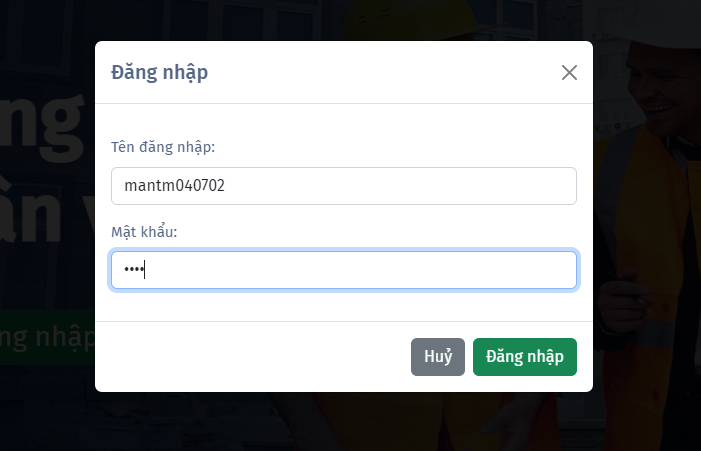
* Quá trình cài đặt và config một function trên lambda đã hoàn thành

# Phần Front-End

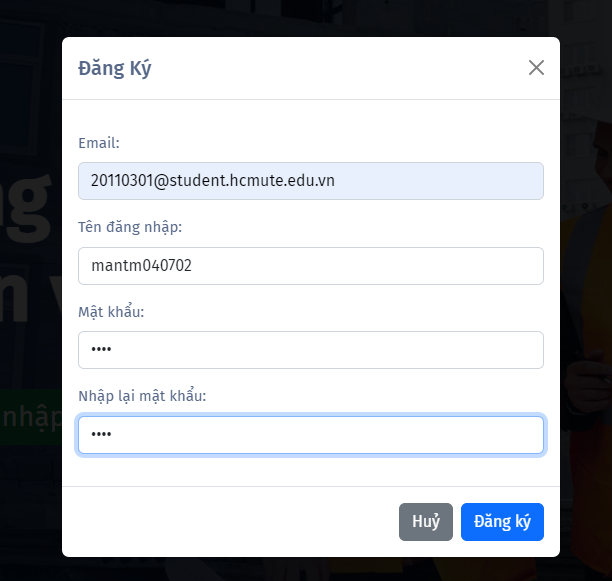
Phần Front-End sử dụng thư viện ReactJs



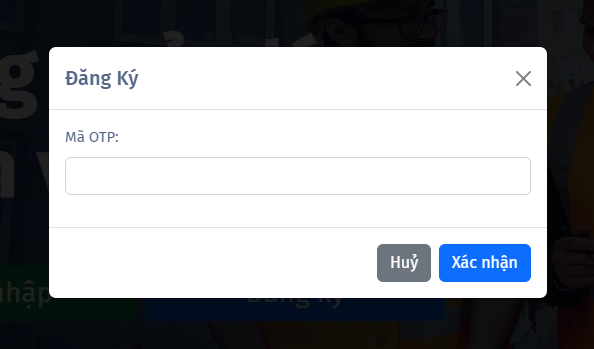
* Đây là phần giao diện khi chạy ứng dụng có thể đăng ký tài khoản sau khi đã đăng ký có thể đăng nhập vào và tiến hành các thao tác tương tác với DynamoDb



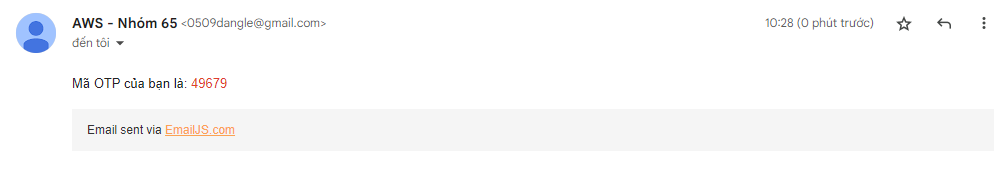
* Giao diện đăng nhập



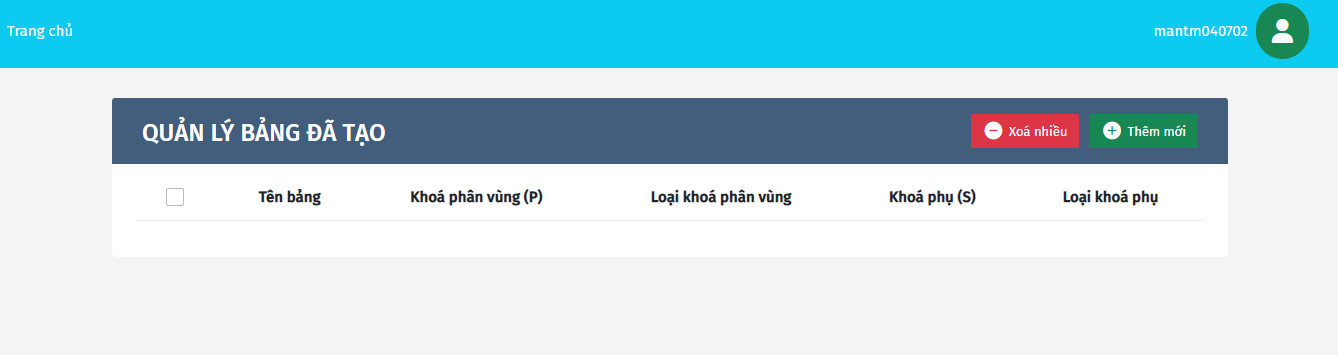
* Giao diện trên thực hiện việc đăng ký một tài khoản



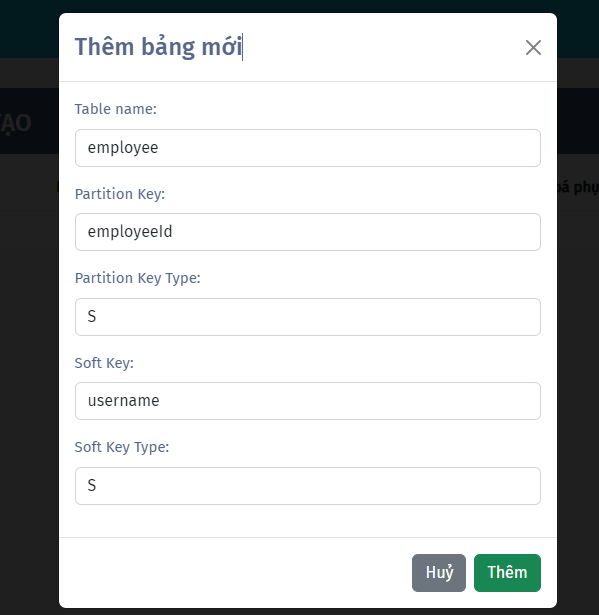
* Sao khi đăng ký hệ thống sẽ tiến hành gửi mã OTP để xác nhận đăng ký, kiểm tra email để có mã nhập mã vào để xác nhận



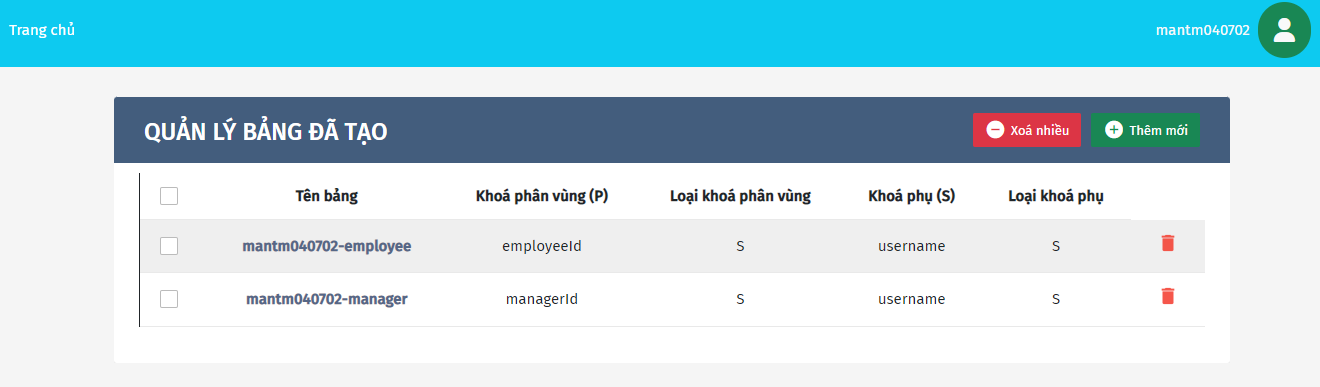
* Lấy mã đã nhận ở email tiến hành xác nhận



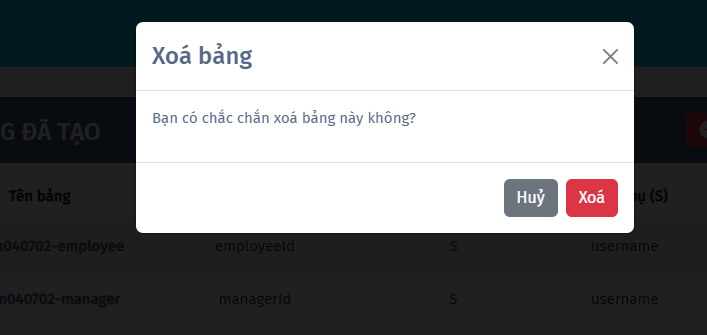
* Giao diện sau khi đăng nhập sẽ hiển thị danh sách các table mà tài khoản này có, hiện tại tài khoản chưa có table nào



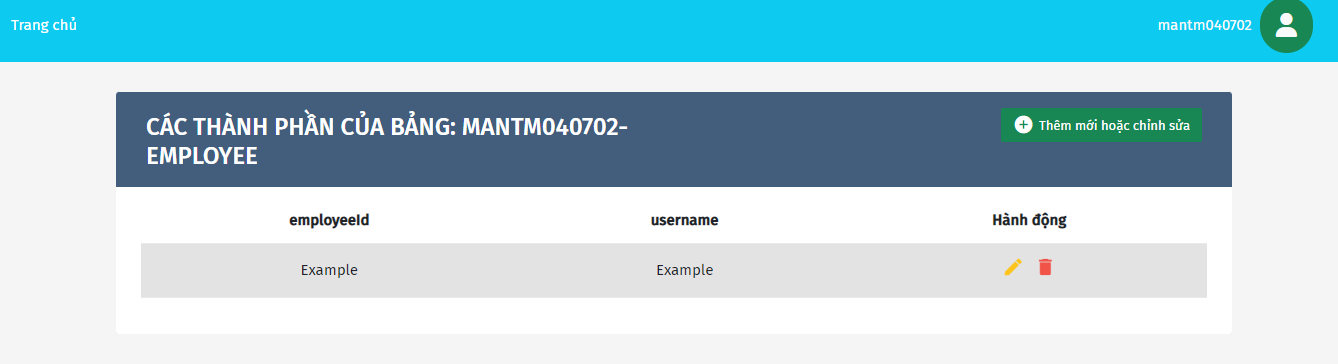
* Giao diện thực hiện việc tạo một bảng mới với các thuộc tính định nghĩa cho khóa chính bao gồm partition key và soft key



* Một table đã được tạo (nhóm tạo hai table), với table name là tên table cùng với tên đăng nhập giúp cho việc table không thể nào trùng trong hệ thống



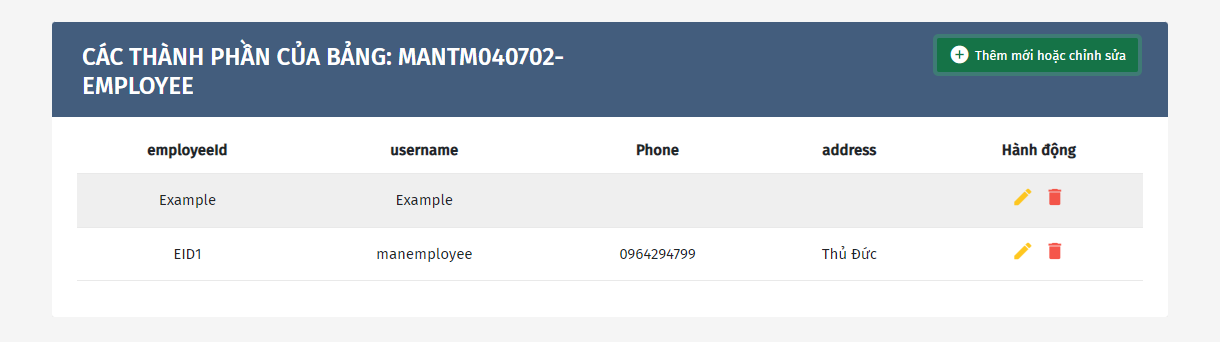
* Thông báo hiển thị khi chúng ta muốn xóa table, ở đây có thể xóa một lúc nhiều table bằng cách chọn nhiều table ấn nút xóa nhiều
* Nếu muốn thêm các item và table ta chọn vào table



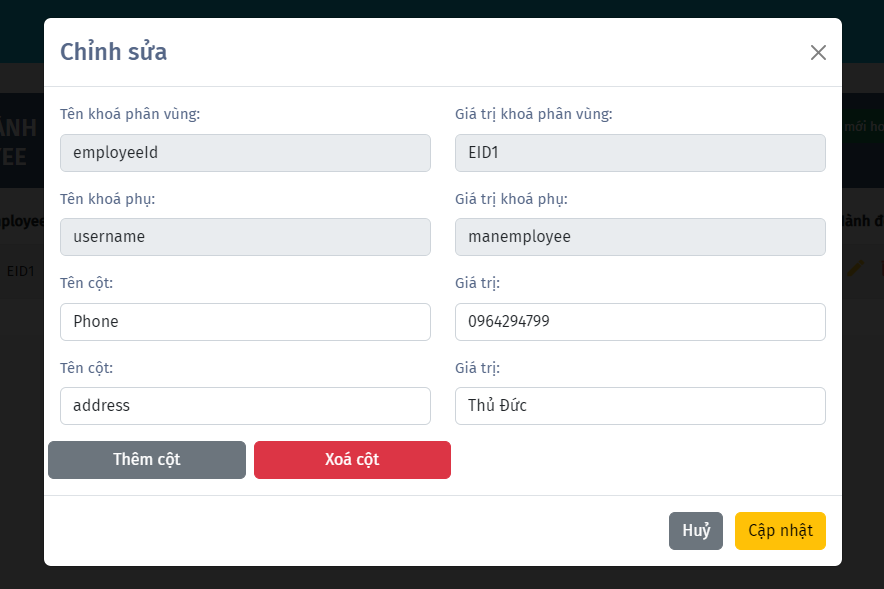
* Giao diện thể hiện các item có trong table
* Để thực hiện thêm các item vào table ấn nút Thêm hoặc chỉnh sửa



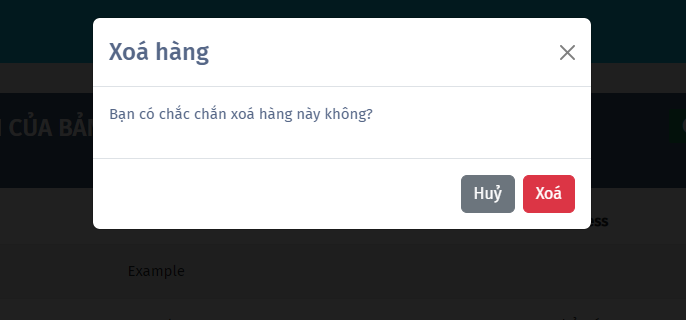
* Giao diện thực hiện việc thêm một item vào table, chúng ta có thể thêm một trường mới trong table ấn thêm cột để thêm



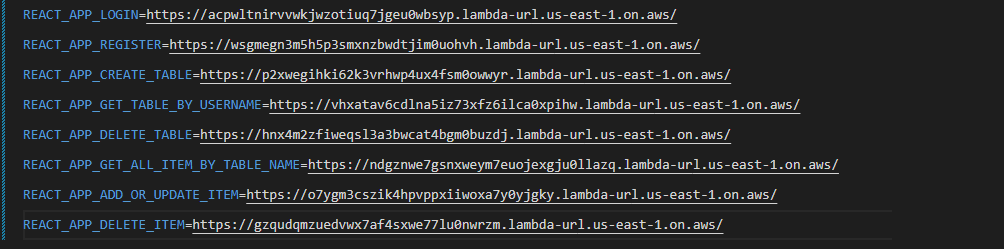
* Sau khi thêm item được thêm đã thể hiện trên danh sách các item trong table
* Muốn cập nhật item ấn icon cây bút để thực hiện việc chỉnh sửa



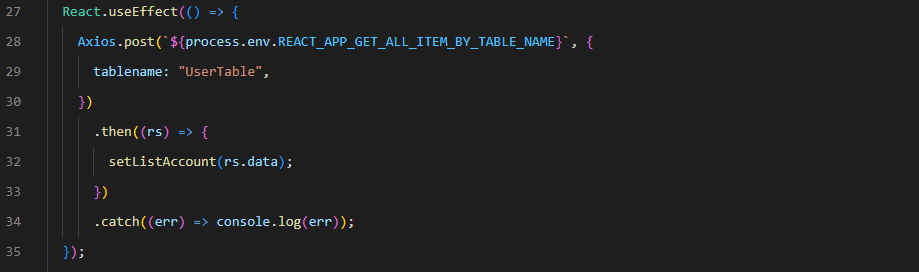
* Cập nhật hoặc thêm một trường mới vào item này cuối cùng ấn nút cập nhật
* Muốn xóa item trong table ấn icon thùng rác



* Thông báo hiện ra ấn xóa sẽ xóa item này khỏi table
* Các code call API:



* Các biến môi trường lưu các url của các function đã được tạo trên lambda

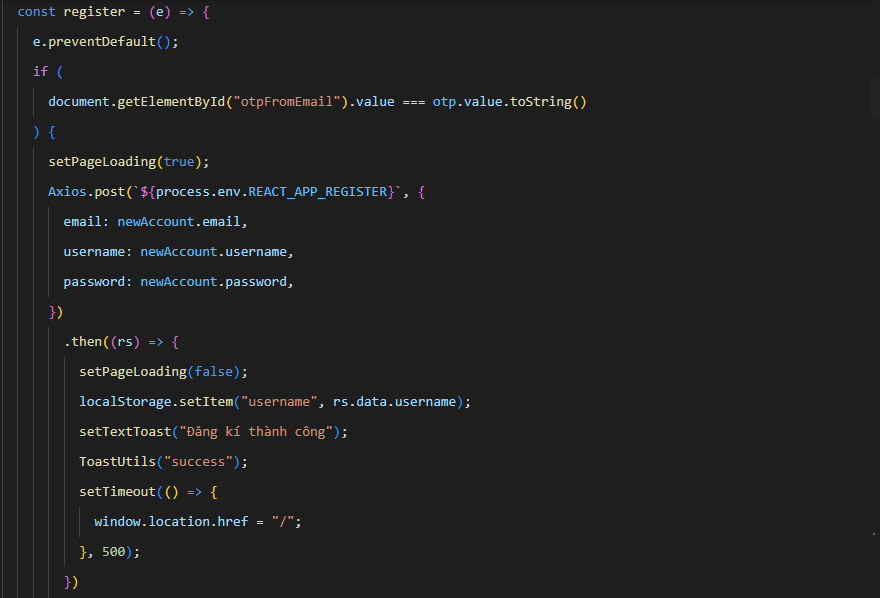


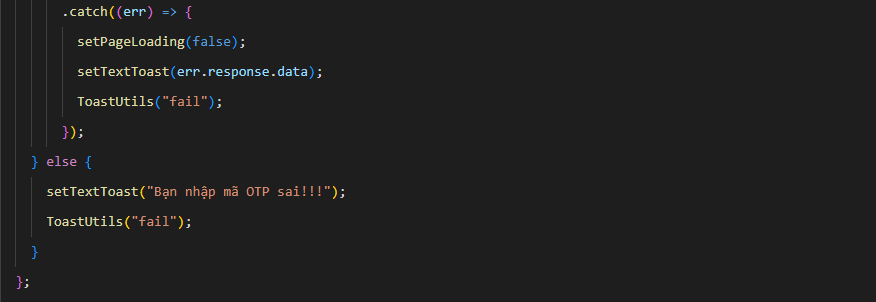
* Gọi API xử lý việc lấy các item trong table dựa vào table name



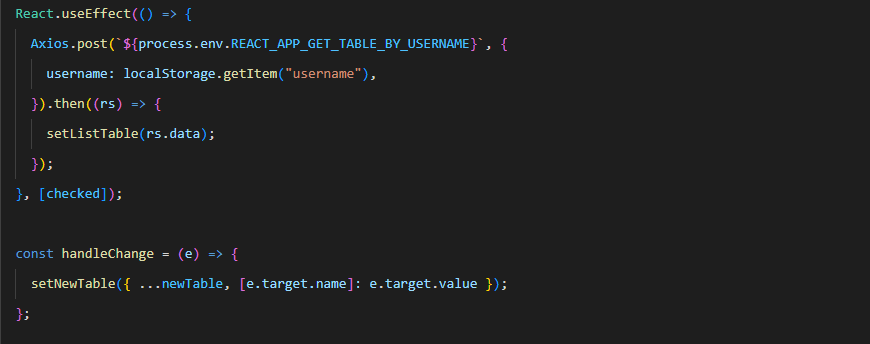


* Gọi API xử lý việc login

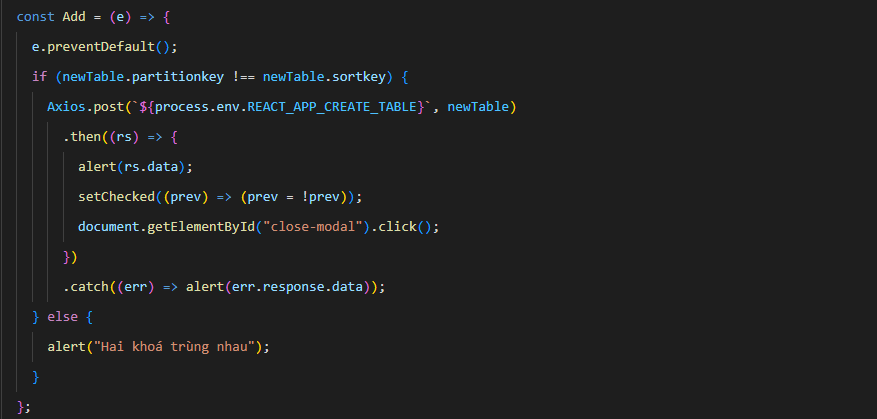




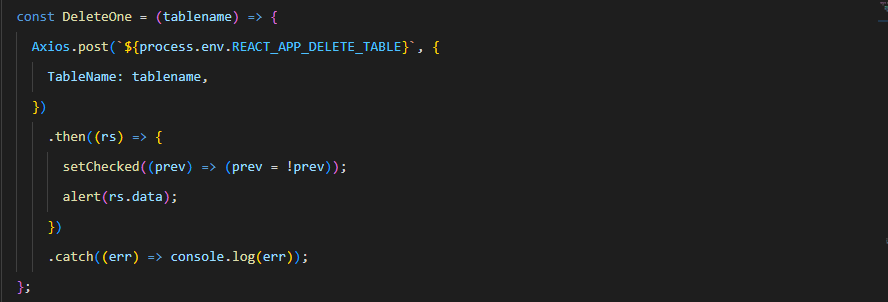
* Gọi API xử lý đăng ký tài khoản



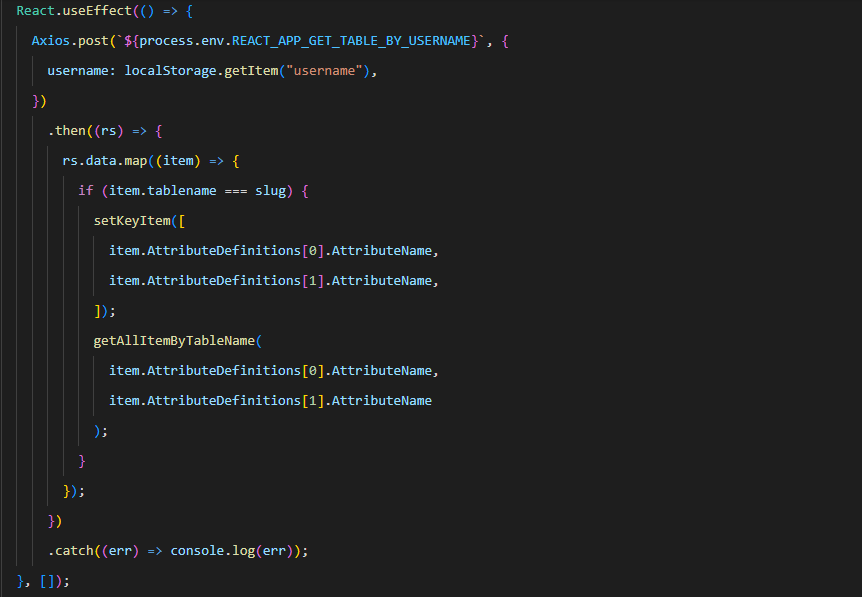
* Gọi API lấy tất cả các table dựa vào username của tài khoản đăng nhập hiện tại



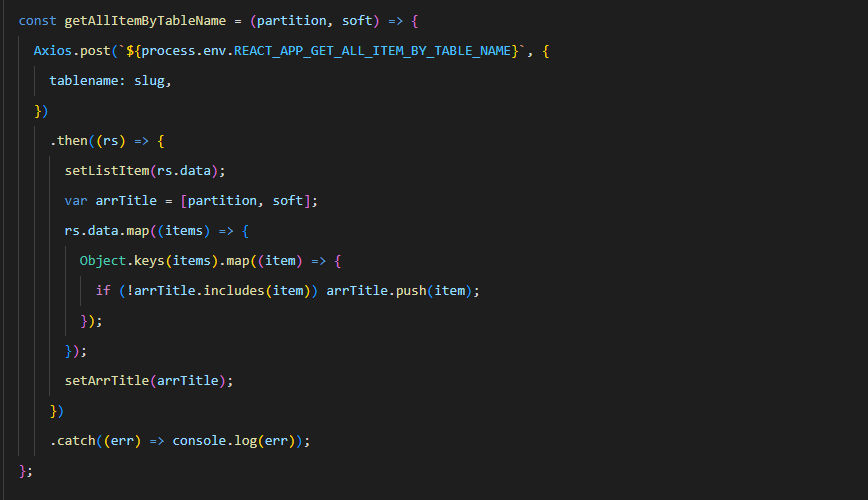
* Gọi API để thêm một table



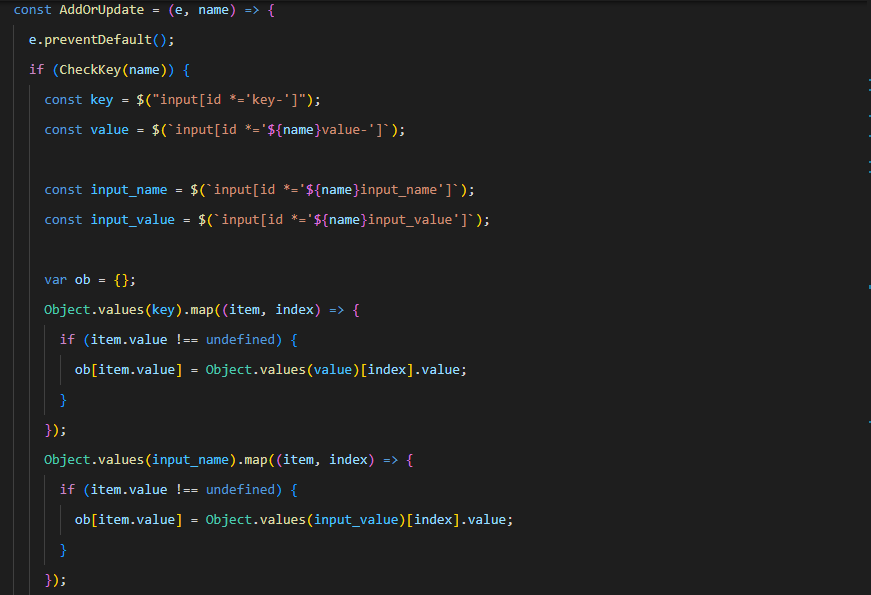
* Gọi API xóa table dựa vào tablename

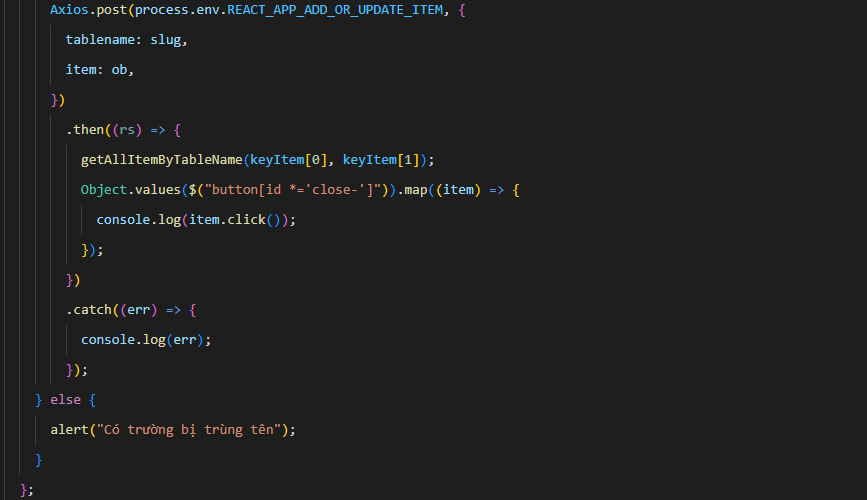


* Gọi API lấy tất cả các table dựa vào username của tài khoản hiện tại sau đó chọn ra table mà muốn quản lý (CRUD các item) dựa vào tên table đó (slug)

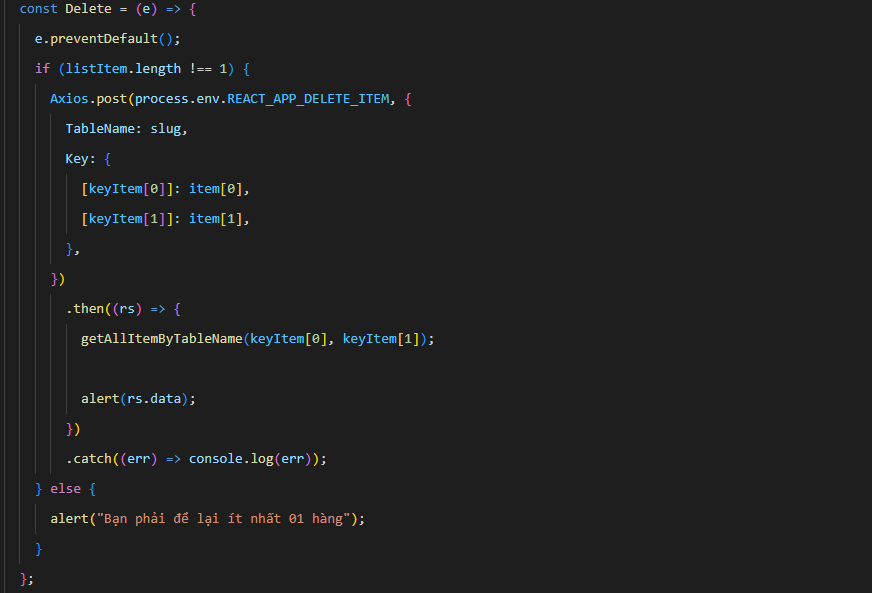


* Gọi API lấy tất cả các item trong table





* Xử lý dữ liệu từ form thêm item hoặc chỉnh sửa item sau đó gọi API để thêm hoặc chỉnh sửa các item trong một table



* Gọi API xử lý xóa một item trong table

# Hướng dẫn cài đặt và chạy dự án trên Local

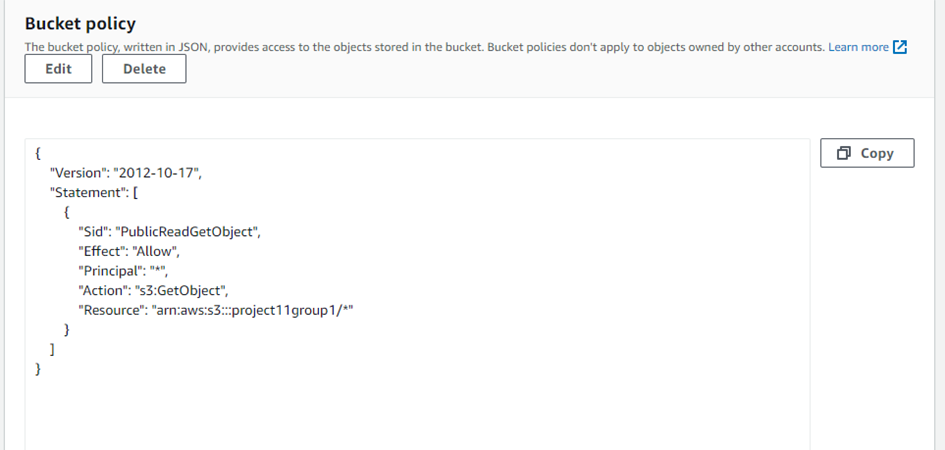
* Bước 1: Sử dụng link bên dưới để clone đề án về máy

<https://github.com/TranMinhMan040702/ProjectAWSDeTai11.git>

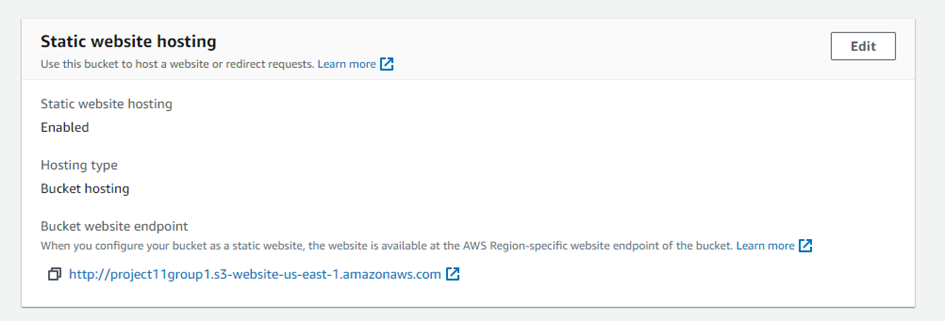
* Bước 2: Cài đặt NodeJs vào máy tính nếu chưa có, tham khảo link <https://nodejs.org/en> để cài đặt NodeJs
* Bước 3: Thực hiện tạo và thiết lập code cho các function ở trên Lambda như đã trình bày ở mục 2
* Bước 4: Vào thư mục **fe\_v2** mở file **.evn** bằng các IDE, thực hiện việc thay đổi các url tại đây thay thế các url có sẵn bằng các url lấy được khi tạo các function lambda.
* Bước 5: Mở Terminal tại **fe\_v2** và chạy các dòng lệnh sau:
* npm install (dùng để cài đặt các node\_module và các thư viện cần thiết)
* npm start (dùng để chạy web ở local)
* Bước 6: Nếu web không tự động mở lên thì có thể mở trình duyệt và dán đường dẫn <http://localhost:3000> để mở trang web.

# Hướng dẫn cài đặt và chạy dự án trên AWS

* Bước 1: Sau khi đã chạy được project ở local, ta mở terminal tại thư mục project chạy các lệnh sau:
* cd .\**fe\_v2** \ để di chuyển đến folder code front-end
* npm run build để thực hiện build project
* Bước 2: Deploy project lên AWS thực hiện như sau:
* Mở AWS S3
* Tại đây tạo bucket sau đó upload tất cả các file front-end vừa được build ở bước trên
* Cấu hình Permission -> Edit Bucket Policy, cấu hình như hình bên dưới



* Vào Properties kéo xuống phần Static Website hosting để lấy like chúng ta đã deploy



# PHẦN KẾT LUẬN

## Kết quả đạt được

Sau một thời gian nghiên cứu và thực hiện đề tài “***Xây dựng 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 Lambda Function URL, DynamoDB và áp dụng kiến thức để thiết kế và xây dựng một ứng dụng 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 với Database.

### 1.2. Chương trình đã làm được

Xây dựng hoàn chỉnh một ứng dụng sử dụng các công nghệ React JS cho FrontEnd, Lambda cho BackEnd và database sử dụng DynamoDB đồng thời sử dụng các dịch dịch của AWS: S3, Lambda Function URL (NodeJS), DynamoDB với các chức năng cơ bản thêm, sửa, xóa, đọc các item trong table của DynamoDB.

## Đánh giá và nhận xét

### 2.1. Phần Back-End và Database

Do kiến thức còn mới nên phần bài làm vẫn có một vài thiếu sót nhưng nhìn chung thì các chức năng gần như đầy đủ theo yêu cầu.

### 2.2. Phần Front-End

Phần giao diện tương đối đủ để phục vụ cho các thao tác và phù hợp với các chức năng nhưng vẫn còn chưa được đẹp.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Tài liệu tham khảo

[1]<https://github.com/trungnguyenhuynhminh46/ProjectDetai11Nhom1v2>: project các anh chị trước

[2]<https://docs.aws.amazon.com/>: tìm hiểu tham khảo về các dịch vụ thực hiện và hỗ trợ trong việc viết các function lambda