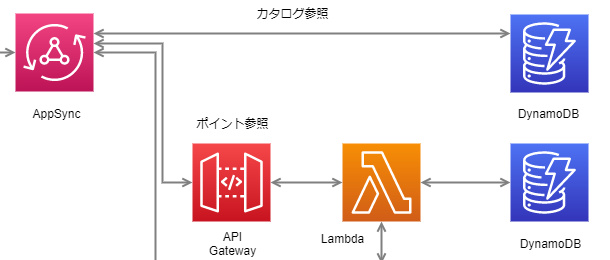
# 購入機能、決済機能

## 機能イメージ



　※Lambdaはポイント付与にも対応

## 機能概要

### ポイント参照機能

＃機能概要を記載する

＜インプット＞

　userid

＜アウトプット＞

　userid,username,point # Userマスタへもアクセスし、UserNameを取得するでも良いかも

### ポイント付与機能

＃機能概要を記載する

＜インプット＞

　userid,point（購入金額の10%）　＃切り上げ／切り捨てはお任せ

＜アウトプット＞

userid,point（加算後の総ポイント数）

## アクセスURL

### ポイント参照機能

　https://#作成時にセット.execute-api.ap-northeast-1.amazonaws.com/jgs2020ia01/getpoint

## APIリクエストパス

|  |  |  |
| --- | --- | --- |
| APIリソースパス | HTTPメソッド | サポートされているコンテンツタイプ |
| /jgs2020ia01/point | POST | application/xml |
| /jgs2020ia01/getpoint | GET | application/json |

## 設定

### Lambdaパラメータシート

|  |  |  |  |
| --- | --- | --- | --- |
| 設定　大項目 | 設定　小項目 | 設定　内容 | 対応する設計書 |
| Function |  | 取引DBに取引情報を格納 |  |
| Runtime |  | Python 3.8 |  |
| Role | アクセス許可 | AWSポジティブテンプレートから新しいロールを作成 |  |
|  | ロール名 | LambdaAccess2DynamoDB  (AmazonDynamoDBFullAccess, AWSLambdaDynamoDBExecutionRole) |  |
| Resource Name |  | arn:aws:lambda:us-west-2:275739154947:function:SetTransactionData |  |
| SetTransaction.py | githubで管理 | import json  import boto3  def lambda\_handler(event, context):    dynamodb = boto3.resource('dynamodb')  table = dynamodb.Table("TransactionTable")    table.put\_item(  Item = {  'TransactionID': event['TransactionID'],  'Amount': event['Amount'],  'PaymentID': event['PaymentID'],  'Price': event['Price'],  'ProductID': event['ProductID'],  'Status': event['Status'],  'UserID': event['UserID'],  'date': event['date'],  }  )  print("PutItem succeeded:")  return |  |

### Step Functions パラメータシート

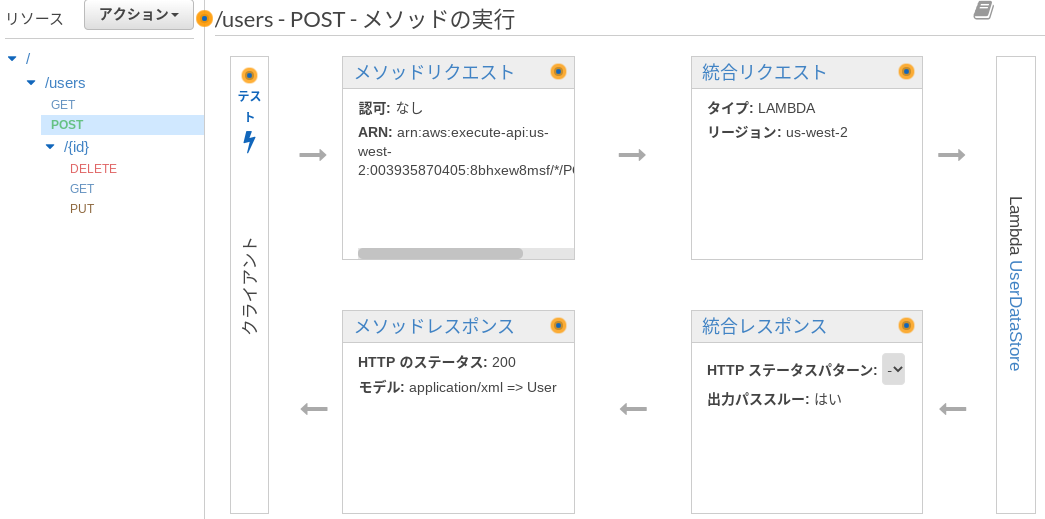
|  |  |  |  |
| --- | --- | --- | --- |
| 設定　大項目 | 設定　小項目 | 設定　内容 | 対応する設計書 |
| StateMachineName |  | MyStateMachine |  |
| JSON | githubで管理 | {  "Comment": "A simple AWS Step Functions state machine that automates a call center support session.",  "StartAt": "SetTransaction",  "States": {  "SetTransaction": {  "Type": "Task",  "InputPath":"$.dataset",  "Resource": "arn:aws:lambda:us-west-2:275739154947:function:SetTransactionData",  "Next":"GetTransaction"  },  "GetTransaction": {  "Type": "Task",  "Resource": "arn:aws:lambda:us-west-2:275739154947:function:TestFunction",  "End":true  }  }  } |  |
| Role | ロール名 | StepFunctions-MyStateMachine-role-96754991  (XRayAccessPolicy-b50e99ca-835f-43dc-8b34-db078a659e19, LambdaInvokeScopedAccessPolicy-d33bb3c2-8518-455b-8b8c-893135300d9a) |  |
| Resource Name |  | arn:aws:states:us-west-2:275739154947:stateMachine:MyStateMachine |  |

### Dynamo DB

|  |  |  |  |
| --- | --- | --- | --- |
| 設定　大項目 | 設定　小項目 | 設定　内容 | 対応する設計書 |
| コアコンポーネント | テーブル、項目、属性 | テーブル名、項目名、属性名、データ型、レングス | テーブル一覧、テーブル定義書 |
| プライマリーキー | ・パーティションキー  ・パーティションキーとソートキー |
| セカンダリインデックス | インデックス設定 |
| DynamoDBストリーム | 利用有無 |  |
| 読み込み整合性 | - | 結果整合性 | ・性能要件  　-DB処理要件  ・（仮称）DynamoDB論理設計  　・スループットキャパシティ設定 |
| 読み込み／書き込みキャパシティーモード | オンデマンドモード | - |
| プロビジョニングモード | ・Read Capacity Unit  ・Write Capacity Unit |
| DynamoDBのセットアップ | AWSアクセスキー | ・アクセスキー  ・シークレットキー  ・IAM権限設定（ユーザー、グループ、ロール、ポリシー） | ・AWSセキュリティ設計  　-認証・認可（IAM） |
| DynamoDBへのアクセス | アクセス方法 | ・WEBアクセス（HTTP、API）  ・AWS CLI  ・No SQLWorkbench  ・IPアドレスの範囲 | ・外部インターフェース一覧  ・保守環境設計  ・開発ツール  ・ネットワーク設計 |
| DynamoDB使用開始 |  |  |  |
| AWS SDKの開始方法 | ※アプリケーション側で考慮すべき要素 | | |
|  |  |  |  |

NoSQL Workbench は、DynamoDB テーブルの設計、作成、クエリ、管理に役立つデータモデリング、データ可視化、クエリ開発機能を提供する統合可視化ツールです。





### 