Instituto Tecnológico de Estudios Superiores de Monterrey
Campus Ciudad de México

Proyecto final

Diseño y Arquitectura de Software
TC3049.1

Equipo:

Anne Constanze Schreiber Brito	A01650066
Rafael E. Fierro Mendez	A01336685
Kevin Alejandro Ruvalcaba Pérez	A01652241
Saúl Neri Ortiz	A01652526

Descripción de la problemática

Durante este periodo de marzo-noviembre del presente año, se ha presentado una cuarentena causada por la pandemia del virus COVID-19. A causa de este encierro, las personas han experimentado diferentes problemas psicológicos y mentales, no necesariamente del término de volverse locos, pero algo que se le acerca. Las personas, a causa del encierro indefinido y del estrés contextual o estrés ambiental, han experimentado algunos sentimientos como enojo, tristeza, nervios, soledad, ansiedad entre otros. Al ser estos sentimientos los que gobiernan el día a día de muchas personas, ellos se sienten invadidos por temor, sentimiento respuesta natural del ser humano contra la incertidumbre, y este sentimiento la mayoría de veces los ha frenado de la productividad dentro del trabajo. Además al estar encerrado, muchas personas no salen a hacer las actividades recreacionales o hobbies que solían hacer para despejarse y sentirse relajados, provocando también que la productividad hacia el trabajo, o productividad escolar, sean afectadas a mal.

Para poder aliviar este problema de estrés y poder ayudarles a seguir con una vida más relajada, hemos creado una solución montada en una plataforma web, donde las personas podrán ingresar y navegar a través de ella, dándose una pequeña distracción de sus actividades cotidianas. Así alejando a las personas de otras opciones de distracción menos saludables, como el alcohol o las drogas, las cuales su consumo ha aumentado en los últimos meses, desde que se le dió inicio a la cuarentena.

Solución propuesta

La propuesta que tenemos para solucionar hasta una cierta extensión al problema presentado, es una aplicación montada en una plataforma web, que ofrecerá al usuario diferentes tipos de actividades para poder distraerse un rato y poder descansar la mente. Dentro de esta aplicación, los usuarios podrán subir sus propias actividades a la plataforma y así alimentar nuestras bases de datos, para poder ofrecer más actividades a diferentes usuarios que estén interesados en estas, como también poder ayudar a tener un contenido más variante. Esto ayudará a la plataforma a crecer y poder así mismo, ayudar a las personas que necesiten distraerse.

Con esta propuesta esperamos alivianar el estrés creado por el aislamiento social y el encierro de esta pandemia.

Historias de Usuario

En este segmento se mostrarán las diferentes historias de usuario que definen los requerimientos que tendrá este proyecto.

Número	01	Usuario:	Usuario
Historia:	Variedad de contenido dentro de la aplicación		
Prioridad	Alta		
Descripción: Quiero que la app me ofrezca diferentes actividades o rutinas para poder realizar diferentes ejercicios cada día y que estos no se repitan. Esto me ayudará a distraerme y no tener algo rutinario, sino algo variado y diferente.			
Validación: Tener dentro de la base de datos, diferentes actividades y ejercicios, que se muestren en la plataforma web.			

Número	02	Usuario:	Usuario
Historia:	Solicitud de una actividad de manera aleatoria		
Prioridad	Baja		
Descripción: Deseo poder solicitar que se me entregue una nueva actividad a realizar en cualquier momento cuando lo desee			
Validación: Desplegar botón para muestra una nueva actividad random a realizar y que este al ser seleccionado, muestre una actividad aleatoria de las que se encuentran dentro de la base de datos.			

Número	03	Usuario:	Desarrollador
Historia:	Login		
Prioridad	Media		
Deseo poder darles una medida de seguridad y acceso a su propia cuenta a través de un usuario y contraseña			
Validación: Acceder a la app mediante credenciales válidas asignadas por usuario			

Número	04	Usuario:	Usuario
Historia:	Realizar actividades en cualquier parte		
Prioridad	Baja		
Descripción: Quiero poder realizar las actividades en cualquier parte			
Validación: Poder acceder a la app en diferentes lugares			

Número	05	Usuario:	Usuario
Historia:	Poder realizar rutinas		
Prioridad	Media		
Descripción: poder realizar mis propios sets de rutinas con diferentes actividades			
Validación: Poder generar álbumes con actividades deseadas			

Número	06	Usuario:	Usuario
Historia:	Exploración de actividades		
Prioridad	Alta		
Descripción: Me gustaría navegar a través de la aplicación para encontrar nuevas actividades			
Validación: Desplegar varias actividades en el home screen			

Número	07	Usuario:	Usuario
Historia:	Editar Actividad		
Prioridad	Media		
Descripción: Deseo poder editar alguna actividad que yo haya subido			
Validación: Poder editar la actividad deseada			

Número	08	Usuario:	Usuario
Historia:	Eliminar Actividad		
Prioridad	Media		
Descripción: Deseo poder eliminar alguna actividad			
Validación: Eliminar una actividad y no desplegarla nuevamente			

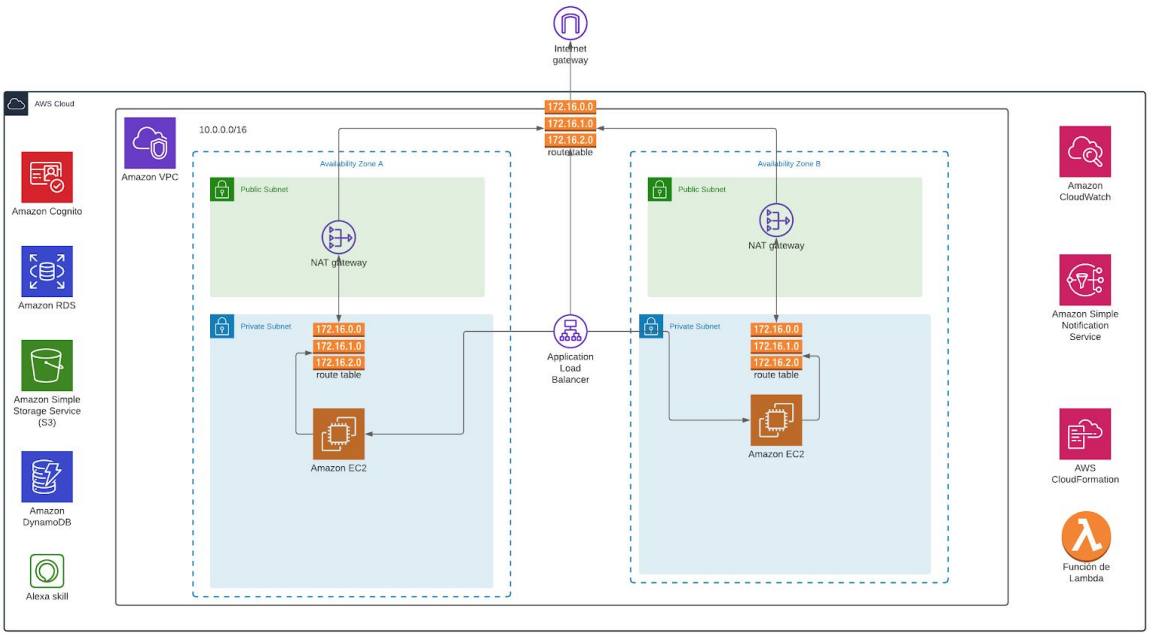
Número	09	Usuario:	Desarrollador
Historia:	Agregar Tag		
Prioridad	baja		
Como desarrollador deseo poder darles la opción a los usuarios de crear nuevos tags para que puedan poner sus actividades bajo el tag que desean			
Validación: Poder agregar tags			

Descripción de la arquitectura de alto nivel

Servicios contemplados a usar:

Los servicios que planeamos utilizar son EC2, el cual la utilizaremos para poder hostear nuestra aplicación web mediante la máquina virtual que este crea; Amazon Cognito, herramienta que servirá en realizar las autenticaciones por usuario creado; Amazon Simple Storage Service (también conocido como S3), el cual usaremos para almacenar y servir nuestras páginas estáticas, las cuales se desplegarán en nuestra aplicación web; Amazon CloudWatch, herramienta que sirve para monitorear el comportamiento de la infraestructura de nuestro proyecto; AWS CloudFormation, la cual nos ayudará a guardar y respaldar las especificaciones de nuestra infraestructura, el cual se guardará en un archivo que se podrá editar en caso de ser necesario; Amazon VPC, esta herramienta nos proporcionará una nube donde podemos lanzar los recursos de AWS necesarios para el proyecto; RDS, esta herramienta es la base de datos relacional de Amazon Web Services que usaremos, y dentro de las opciones que este da, utilizaremos MariaDB, el cual es parecido a MySQL. Dentro de esta base de datos se almacenarán las actividades que se mostrarán en la plataforma web; DynamoDB, esta es otra base de datos no relacional que ofrece Amazon Web Services, el cual se maneja con llaves. Esta base de datos se utilizará para guardar las categorías; AWS Lambda, este servicio

de informática sin servidor ayudará a ejecutar código en respuesta a eventos y administrar automáticamente los recursos informáticos subyacentes;

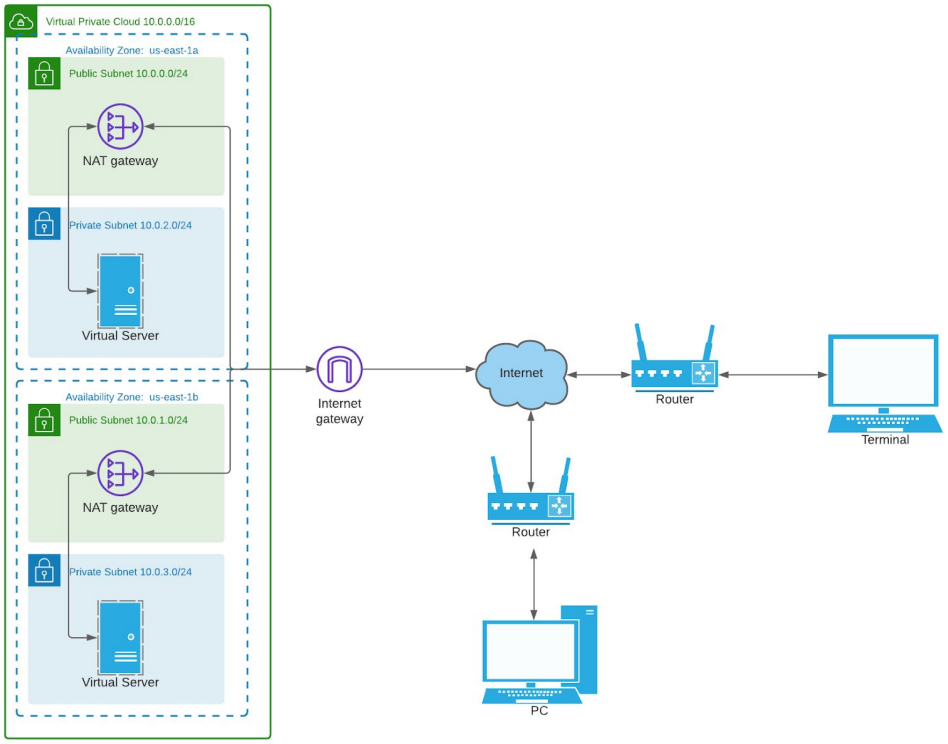


El diagrama anterior se observa que se está trabajando con el AWS Cloud y dentro de ella podemos encontrar diversos servicios que la componen. Dentro de nuestra cloud y para nuestra aplicación, haremos uso de Amazon Cognition, para poder detectar el usuario quien ingrese/ acceda a la cloud, la cual nos ayudará a identificarlo y a mostrarle el contenido debido, como es la aplicación y su contenido. Se hará uso de dos bases de datos para poder guardar los datos y contenidos de la app, las cuales es DynamoDB para aquellos datos que no son relacionales y se manejen con una llave, y la base de datos de MariaDB encontrada dentro de Amazon RDS. También dentro del AWS Cloud se encuentra el S3, el cual se utilizará para hostear la página estática de la aplicación. También encontraremos el Alexa Skill, el cual a través de ella podemos pedirle a Alexa que nos de una actividad aleatoria, y tenemos las lambdas, las cuales nos ayudarán a responder a eventos dentro de nuestro proyecto. Al entrar más a nuestra cloud, encontraremos el uso de la VPC, el cual nos ayudará a lanzar recursos que necesitemos del AWS, dentro podemos encontrar el uso de subnets privadas y públicas las cuales correrán en diferentes zonas disponibles. También podemos encontrar fuera de la VPC otros servicios que nos ayudarán durante la implementación de la aplicación, estas son CloudWatch, el cual nos ayudará a monitorear el comportamiento de nuestra infraestructura, Amazon Simple Notification Service, que nos ayudará a notificar al usuario de sus actividades, y el AWS CloudFormation, que se encargará de especificar nuestra infraestructura y a traducirla en un documento yaml fácil de editar.

Diseño de Redes para alta disponibilidad y escalamiento

La dirección de ip y máscara que se usará para este proyecto y para su VPC es la 10.0.0.0/16. La VPC tendrá dos pares de subnets públicos y privados. Las ip 's que se le asignan a las subnets públicas son 10.0.0.0/24 y 10.0.1.0/24, y para las privadas serán las ip' s 10.0.2.0/24 y 10.0.3.0/24. Las zonas de disponibilidad serían us-east-1a y us-east-1b dado que son las más cercanas a nosotros.

Topología de red



Routing Table

Destination	Target
0.0.0.0/0	internet gateway
10.0.0.0/16	Local

Diseño de NACL's

Inbound

Regla	Tipo	Protocolo	Puerto	Origen	Permitir/Rechazar
100	Https	TCP	80	0.0.0.0/0	Permitir
101	SSH	TCP	22	10.0.2.0/24 10.03.0/24	Permitir
*	Todo tráfico IPv4	Todos	Todos	0.0.0.0/0	Rechazar

Outbound

Reglas	Tipo	Protocolo	Puerto	Destino	Permitir/Rechazar
100	Todo tráfico IPv4	Todos	Todos	0.0.0.0/0	Permitir
*	Todo tráfico IPv4	Todos	Todos	0.0.0.0/0	Rechazar

Diseño de VPC Endpoints

Usaremos dos VPC Interface endpoints para comunicar nuestro EC2 con el internet gateway. También se hará uso de una serie de VPC Gateway endpoint para conectarnos con nuestra base de datos en MongoDB. Estos endpoints harán un CRUD para toda la información que guardemos en nuestra base de datos.

Otros aspecto de la VPC que usaremos, es un VPC Gateway endpoint para extraer y desplegar imágenes y videos.

Diseño de Security groups para EC2

Name	Regla-interior-http
Security group	inbound
Protocol	TCP
Port range	80
Origen	0.0.0.0/0

Name	Regla-interior-ssh
Security group	inbound
Protocol	TCP
Port range	22
Origen	10.0.2.0/24 y 10.0.3.0/24

Name	Regla-exterior
Security group	outbound
Protocol	All
Port range	All
Destination	0.0.0.0/0

S3

Archivos estáticos

Usaremos buckets de S3 par almacenar, compartir y distribuir archivos estáticos de diferentes formatos con múltiples objetivos, algunos de ellos se enlistan a continuación:

- Almacenamiento y acceso a contenido multimedia de la aplicación
    - Todos los videos, imágenes y audios relacionados con la aplicación estarán hosteados en buckets de S3 para acceder a ellos desde la aplicación del cliente.
    - El almacenamiento de archivos de texto relacionados con las rutinas creadas por la comunidad estarán almacenados en buckets.
    - En caso de que se permita al usuario cargar archivos multimedia, estos serán almacenados en este servicio de AWS.

Página Web

Dado que usaremos tecnologías de desarrollo de páginas web que nos permitan generar sitios estáticos. podremos hostear todo el contenido web que tengamos en AWS S3 buckets.

- La ventaja que esto tiene es que S3 tiene una funcionalidad integrada para este uso.

- Podemos asignar un dominio propio y hacer una configuración de DNS para que todo el tráfico se direccione al bucket de S3 que contiene nuestro sitio.
- S3 está muy optimizado para poder lidiar con cualquier cambio en la demanda de nuestro sitio, asegurando disponibilidad muy alta con tiempos de respuesta muy bajos.
- Nuestra página informativa así como el frontend de nuestra web app pueden estar hosteados de manera fácil y a un costo muy bajo

Bucket Policies

Para poder hacer que el contenido de nuestra bucket tenga permisos de lectura al público, esto es que desde internet podamos ver los objetos que contiene, necesitamos hacer efectiva la siguiente política:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "bucket-name": [
                "arn:aws:s3:::example.com/*"
            ]
        }
    ]
}
```

Nuestra política consta de los siguientes elementos:

- Sid: que es un identificador que nosotros hemos determinado.
- Efecto: que describe si es una política del tipo *allow* o del tipo *deny.*
- Principal, que enlista los usuarios que tendrán acceso a las acciones descritas en nuestra política. En nuestro caso deseamos dar acceso al público, entonces usamos el símbolo * como comodín.
- Action, enlista las acciones a las que tendrá acceso la política. En nuestro caso es un permiso de solo lectura, por lo que corresponde el valor de *s3:GetObject*
- Resources, enlista los buckets y objetos que se verán afectados por la política descrita.

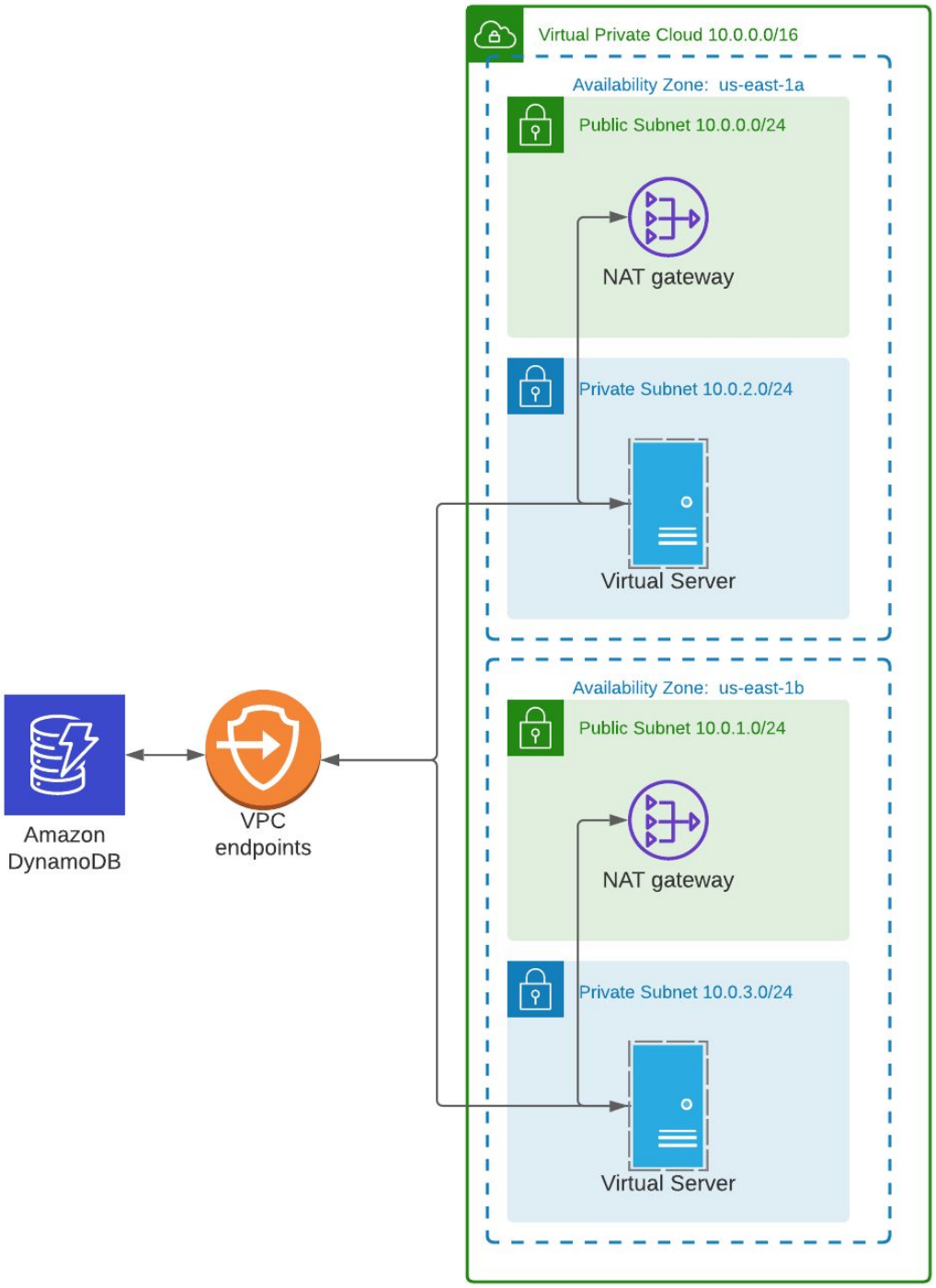
Diseño de Base de Datos

a. Selección y justificación de bases de datos

La base de datos seleccionada para este proyecto es RDS con MariaDB, que hace uso de MySQL, esto es debido a su uso de tablas relacionales. Además se utiliza DynamoDB, el cual es una base de datos no relacional. Ambas bases de datos se eligieron por su escalabilidad y su capacidad de guardado.

La primera base de datos se ha utilizado para guardar las actividades que se desplegarán dentro de la plataforma web, el cual guarda nombre, descripción corta e instrucciones de la actividad, mientras el segundo, DynamoDB, se ha utilizado para guardar los diferentes grupos o tags a los que pertenecen las actividades. Ambas tablas se relacionan por una llave.

Diseño de red para la base de datos



En el diagrama anterior vemos como es la comunicación entre nuestro EC2 y nuestra base de datos de DynamoDB. Con el uso de VPC endpoints podemos tener todo un CRUD para nuestras las necesidades de nuestra webapp sin tener que usar un gateway. Un mismo diagrama igual se podrá encontrar para nuestra base de datos que reside dentro del RDS.

Diseño de subnet groups si aplica

Subnet	DBSubnetGroup
10.0.2.0/24	1
10.0.3.0/24	1

Implementación de Cloudformation de lo anterior y demostración en cuenta AWS

Limitaciones del Diseño

Durante la implementación de AWS se encontraron con limitaciones de elementos que fueron planeados y no se lograron realizar dentro de la aplicación debido a limitaciones de permisos por la cuenta de Voraceum, esto limitó la creación de la VPC al no poder realizar los gateways de internet públicas manteniendo la VPC únicamente privada. Otras limitaciones existentes fueron la creación de EC2 dentro del cloudformation, la generación de usuarios de IAM y conexión del Code Pipeline por Github 2. El último elemento con el que encontramos limitación de implementación de cognito dada a causa de problemas de compatibilidad con Django, a pesar de todo esto se logro trabajar alrededor de la mayoría de las limitaciones para la realización de la aplicación.

YAML

```
Parameters:

  EnvironmentName:

      Description: Name of environment

      Type: String


  VpcAddress:

      Description: IP range for this VPC

      Type: String

      Default: 10.0.0.0/16


  PublicSubnet1Address:
```

```
    Description: IP range for the public subnet 1
    Type: String
    Default: 10.0.0.0/24

PublicSubnet2Address:
    Description: IP range for the public subnet 2
    Default: 10.0.1.0/24
    Type: String

PrivateSubnet1Address:
    Description: IP range for the private subnet 1
    Type: String
    Default: 10.0.2.0/24

PrivaateSubnet2Address:
    Description: IP range for the private subnet 2
    Type: String
    Default: 10.0.3.0/24
S3BucketName:
    Description: S3Bucket Name to call Lambda Functions
    Type: String
    Default: cf-templates-1k89fs48r6bc-us-east-1
#LambdaRole:
#  Description: IAM Lambda Role
#  Type: String
```

```
  InternalS3BucketName:
      Description: Bucket for all internal functions of the
CloudFormation
      Type: String



Resources:
#VPC Generation
  VPC:
      Type: AWS::EC2::VPC
      Properties:
      CidrBlock: !Ref VpcAddress
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
      - Key: Name
            Value: !Ref EnvironmentName
#VPC Endpoint


  S3Endpoint:
      Type: 'AWS::EC2::VPCEndpoint'
      Properties:
      PolicyDocument:
      Version: 2012-10-17
      Statement:
            - Effect: Allow
```

```
            Principal: '*'
            Action:
            - 's3:GetObject'
            Resource:
            - 'arn:aws:s3:::examplebucket/*'
      RouteTableIds:
      - !Ref PublicRouteTable
      ServiceName: !Sub 'com.amazonaws.${AWS::Region}.s3'
      VpcId: !Ref VPC


#GateWays
  InternetGateway:
      Type: AWS::EC2::InternetGateway
      Properties:
      Tags:
      - Key: Name
            Value: !Ref EnvironmentName


  InternetGatewayAttachment:
      Type: AWS::EC2::VPCGatewayAttachment
      Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
#Public Subnets
  PublicSubnet1:
      Type: AWS::EC2::Subnet
```

```
    Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PublicSubnet1Address
    MapPublicIpOnLaunch: true
    Tags:
    - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)


 PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 1, !GetAZs  '' ]
    CidrBlock: !Ref PublicSubnet2Address
    MapPublicIpOnLaunch: true
    Tags:
    - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ2)
#Private Subnets
 PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs  '' ]
    CidrBlock: !Ref PrivateSubnet1Address
```

```
      MapPublicIpOnLaunch: false
      Tags:
      - Key: Name
            Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

  PrivateSubnet2:
      Type: AWS::EC2::Subnet
      Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivaateSubnet2Address
      MapPublicIpOnLaunch: false
      Tags:
      - Key: Name
            Value: !Sub ${EnvironmentName} Private Subnet (AZ2)
#NAT Gateways
  NatGateway1EIP:
      Type: AWS::EC2::EIP
      DependsOn: InternetGatewayAttachment
      Properties:
      Domain: vpc

  NatGateway2EIP:
      Type: AWS::EC2::EIP
      DependsOn: InternetGatewayAttachment
      Properties:
```

```
      Domain: vpc


#  NatGateway1:
#     Type: AWS::EC2::NatGateway
#     Properties:
#     AllocationId: !GetAtt NatGateway1EIP.AllocationId
#     SubnetId: !Ref PublicSubnet1


#  NatGateway2:
#     Type: AWS::EC2::NatGateway
#     Properties:
#     AllocationId: !GetAtt NatGateway2EIP.AllocationId
#     SubnetId: !Ref PublicSubnet2
#Public Routing Tables
  PublicRouteTable:
     Type: AWS::EC2::RouteTable
     Properties:
     VpcId: !Ref VPC
     Tags:
     - Key: Name
           Value: !Sub ${EnvironmentName} Public Routes


  DefaultPublicRoute:
     Type: AWS::EC2::Route
     DependsOn: InternetGatewayAttachment
     Properties:
```

```
      RouteTableId: !Ref PublicRouteTable

      DestinationCidrBlock: 0.0.0.0/0

      GatewayId: !Ref InternetGateway


  PublicSubnet1RouteTableAssociation:

      Type: AWS::EC2::SubnetRouteTableAssociation

      Properties:

      RouteTableId: !Ref PublicRouteTable

      SubnetId: !Ref PublicSubnet1


  PublicSubnet2RouteTableAssociation:

      Type: AWS::EC2::SubnetRouteTableAssociation

      Properties:

      RouteTableId: !Ref PublicRouteTable

      SubnetId: !Ref PublicSubnet2


#Private Routing Tables

  PrivateRouteTable1:

      Type: AWS::EC2::RouteTable

      Properties:

      VpcId: !Ref VPC

      Tags:

      - Key: Name

            Value: !Sub ${EnvironmentName} Private Routes (AZ1)


#  DefaultPrivateRoute1:
```

```
#     Type: AWS::EC2::Route
#     Properties:
#     RouteTableId: !Ref PrivateRouteTable1
#     DestinationCidrBlock: 0.0.0.0/0
#     NatGatewayId: !Ref NatGateway1


  PrivateSubnet1RouteTableAssociation:
      Type: AWS::EC2::SubnetRouteTableAssociation
      Properties:
      RouteTableId: !Ref PrivateRouteTable1
      SubnetId: !Ref PrivateSubnet1


  PrivateRouteTable2:
      Type: AWS::EC2::RouteTable
      Properties:
      VpcId: !Ref VPC
      Tags:
      - Key: Name
            Value: !Sub ${EnvironmentName} Private Routes (AZ2)


#  DefaultPrivateRoute2:
#     Type: AWS::EC2::Route
#     Properties:
#     RouteTableId: !Ref PrivateRouteTable2
#     DestinationCidrBlock: 0.0.0.0/0
#     NatGatewayId: !Ref NatGateway2
```

```
  PrivateSubnet2RouteTableAssociation:
      Type: AWS::EC2::SubnetRouteTableAssociation
      Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2
#EC2 Creation
#  EC2Sub1:
#     Type: 'AWS::Cloud9::EnvironmentEC2'
#     Properties:
#     InstanceType: t2.micro
#     SubnetId: !Ref PrivateSubnet1

#  EC2Sub2:
#     Type: 'AWS::Cloud9::EnvironmentEC2'
#     Properties:
#     InstanceType: t2.micro
#     SubnetId: !Ref PrivateSubnet2

#Load Balancer
  LoadBalancer:
      Type: 'AWS::ElasticLoadBalancing::LoadBalancer'
      Properties:
      Listeners:
      - InstancePort: '80'
            LoadBalancerPort: '80'
```

```
            Protocol: HTTP
      Subnets:
      - !Ref PrivateSubnet1
      - !Ref PrivateSubnet1


#Document DB
  DocCluster:
      Type: 'AWS::DocDB::DBCluster'
      Properties:
      MasterUsername: "root"
      MasterUserPassword: "roottoor"
      StorageEncrypted: false
      DependsOn:
      - DocSubnet
  DocInstance:
      Type: 'AWS::DocDB::DBInstance'
      Properties:
      DBClusterIdentifier: !Ref DocCluster
      DBInstanceClass: "db.r5.large"
      DependsOn:
      - DocCluster
  DocSubnet:
      Type: 'AWS::DocDB::DBSubnetGroup'
      Properties:
      DBSubnetGroupDescription: DB Subnet Group
      SubnetIds:
```

```
      - !Ref PrivateSubnet1
      - !Ref PrivateSubnet2


#S3 Bucket
  S3Bucket:
      Type: AWS::S3::Bucket
      Properties:
      BucketName: !Ref InternalS3BucketName
      AccessControl: PublicRead
      WebsiteConfiguration:
      IndexDocument: index.html
      ErrorDocument: error.html
      RoutingRules:
      - RoutingRuleCondition:
            HttpErrorCodeReturnedEquals: '404'
            KeyPrefixEquals: app/
            RedirectRule:
            HostName: velezcloud-bucket.us-east-1.amazonaws.com
            ReplaceKeyPrefixWith: report-404/
      #DeletionPolicy: Retain



  S3BucketPolicy:
      Type: 'AWS::S3::BucketPolicy'
      Properties:
      Bucket: !Ref S3Bucket
```

```
      PolicyDocument:
      Statement:
            - Action:
            - 's3:GetObject'
            Effect: Allow
            Resource:
            'Fn::Join':
                  - ''
                  - - 'arn:aws:s3:::'
                  - Ref: S3Bucket
                  - /*
            Principal: '*'
            Condition:
            StringLike:
                  'aws:Referer':
                  - 'http://www.example.com/*'


#Amazon Cognito
  CognitoIdentity:
      Type: 'AWS::Cognito::IdentityPool'
      Properties:
      AllowUnauthenticatedIdentities: true


#ACL
  NetworkAcl:
      Type: AWS::EC2::NetworkAcl
```

```
      Properties:
      VpcId:
      Ref: VPC
#ACL Rules
  InboundRule100:
      Type: AWS::EC2::NetworkAclEntry
      Properties:
      NetworkAclId:
      Ref: NetworkAcl
      RuleNumber: 100
      Protocol: -1
      RuleAction: allow
      CidrBlock: 0.0.0.0/0
      PortRange:
      From: 80
      To: 80
  InboundRule101:
      Type: AWS::EC2::NetworkAclEntry
      Properties:
      NetworkAclId:
      Ref: NetworkAcl
      RuleNumber: 101
      Protocol: -1
      RuleAction: allow
      CidrBlock: !Ref PrivaateSubnet2Address
      PortRange:
```

```
    From: 22
    To: 22
InboundRule102:
    Type: AWS::EC2::NetworkAclEntry
    Properties:
    NetworkAclId:
    Ref: NetworkAcl
    RuleNumber: 102
    Protocol: -1
    RuleAction: allow
    CidrBlock: !Ref PrivateSubnet1Address
    PortRange:
    From: 22
    To: 22
InboundRule103:
    Type: AWS::EC2::NetworkAclEntry
    Properties:
    NetworkAclId:
    Ref: NetworkAcl
    RuleNumber: 103
    Protocol: -1
    RuleAction: deny
    CidrBlock: 0.0.0.0/0
OutboundRule:
    Type: AWS::EC2::NetworkAclEntry
    Properties:
```

```
      NetworkAclId:
      Ref: NetworkAcl
      RuleNumber: 100
      Protocol: -1
      Egress: true
      RuleAction: allow
      CidrBlock: 0.0.0.0/0
#Network Security Group
  SecurityGroupHTTP:
      Type: AWS::EC2::SecurityGroup
      Properties:
      GroupDescription: Client To HTTP
      VpcId:
            Ref: VPC
      SecurityGroupIngress:
      - IpProtocol: tcp
            FromPort: 80
            ToPort: 80
            CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
      - IpProtocol: tcp
            FromPort: 80
            ToPort: 80
            CidrIp: 0.0.0.0/0
  SecurityGroupHTTPS:
      Type: AWS::EC2::SecurityGroup
```

```
    Properties:
    GroupDescription: Client To HTTPS
    VpcId:
          Ref: VPC
    SecurityGroupIngress:
    - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
    SecurityGroupEgress:
    - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
SecurityGroupSSH:
    Type: AWS::EC2::SecurityGroup
    Properties:
    GroupDescription: Client To SSH
    VpcId:
          Ref: VPC
    SecurityGroupIngress:
    - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: !Ref PrivateSubnet1Address
    SecurityGroupEgress:
```

```
      - IpProtocol: tcp
            FromPort: 22
            ToPort: 22
            CidrIp: !Ref PrivaateSubnet2Address
  SecurityGroupExit:
      Type: AWS::EC2::SecurityGroup
      Properties:
      GroupDescription: Client To Exterior
      VpcId:
            Ref: VPC
      SecurityGroupIngress:
      - IpProtocol: -1
            CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
      - IpProtocol: -1
            CidrIp: 0.0.0.0/0
#Parcial 2
  LambdaCategories:
      Type: AWS::Lambda::Function
      Properties:
      Runtime: nodejs12.x
      Role: !GetAtt LambdaRootRole.Arn
      #arn:aws:iam::514815351447:role/LambdaCloudFormationTest
      #Edit To Role

#arn:aws:iam::139319357854:role/service-role/alexa-play-despacito-role-
t8fojru6
```

```
      Handler: index.handler
      Code:
      #Edit to Bucket
      S3Bucket: !Ref S3BucketName
      S3Key: get-all-categories.zip
      Description: Call for Function Categories
      #VpcConfig: !Ref VPC
  LambdaRandom:
      Type: AWS::Lambda::Function
      Properties:
      Runtime: nodejs12.x
      Role: !GetAtt LambdaRootRole.Arn
      #Edit To Role

#arn:aws:iam::139319357854:role/service-role/alexa-play-despacito-role-
t8fojru6
      Handler: index.handler
      Code:
      #Edit to Bucket
      S3Bucket: !Ref S3BucketName
      S3Key: random-rutine-lambda.zip
      Description: Call a Random Activity
      #VpcConfig: !Ref VPC
  LambdaAlexa:
      Type: AWS::Lambda::Function
      Properties:
      Runtime: nodejs12.x
```

```
      Role: !GetAtt LambdaRootRole.Arn
      #Edit To Role

#arn:aws:iam::139319357854:role/service-role/alexa-play-despacito-role-t8fojru6
      Handler: index.handler
      Code:
      #Edit to Bucket
      S3Bucket: !Ref S3BucketName
      S3Key: alexa-skills-code.zip
      Description: Call a Random Activity
      #VpcConfig: !Ref VPC
#DB Building
  DynamoDB:
      Type: AWS::DynamoDB::Table
      Properties:
      TableName: "categoria_rutinas"
      AttributeDefinitions:
      -
            AttributeName: "Id"
            AttributeType: "N"
      -
            AttributeName: "Tipo"
            AttributeType: "S"
      KeySchema:
      -
            AttributeName: "Id"
```

```
          KeyType: "HASH"
    -
          AttributeName: "Tipo"
          KeyType: "RANGE"
    ProvisionedThroughput:
          ReadCapacityUnits: 5
          WriteCapacityUnits: 5
RDS:
    Type: "AWS::RDS::DBInstance"
    Properties:
    AllocatedStorage: "50"
    Engine: "MariaDB"
    MasterUsername: "root"
    MasterUserPassword: "password"
    DBInstanceClass: "db.t2.small"
    #DBInstanceClass: "db.r5.large"
    DBSecurityGroups:
    - !Ref DBSecurityGroup
DBSecurityGroup:
    Type: AWS::RDS::DBSecurityGroup
    Properties:
    GroupDescription: All-Security-Group
    DBSecurityGroupIngress:
    - CIDRIP: 0.0.0.0/0
RandomApi:
    Type: AWS::ApiGateway::RestApi
```

```
    Properties:
    Name: AlexaDespacitoRandom
CategoriesApi:
    Type: AWS::ApiGateway::RestApi
    Properties:
    Name: AlexaDespacitoCategories
#VPCLink:
#   Type: AWS::ApiGateway::VpcLink
#   Properties:
#         Description: "VPC Link"
#         Name: "Alexa Despacito"
#         TargetArns:
#               - !Ref LoadBalancer


#IAM Lambda Role
LambdaRootRole:
    Type: 'AWS::IAM::Role'
    Properties:
    AssumeRolePolicyDocument:
    Version: 2012-10-17
    Statement:
          - Effect: Allow
          Principal:
          Service:
          - lambda.amazonaws.com
          Action:
```

```
            - 'sts:AssumeRole'
      Path: /
      Policies:
      - PolicyName: root
            PolicyDocument:
            Version: 2012-10-17
            Statement:
            - Effect: Allow
                  Action: '*'
                  Resource: '*'
  RootInstanceProfile:
      Type: 'AWS::IAM::InstanceProfile'
      Properties:
      Path: /
      Roles:
      - !Ref LambdaRootRole
#Outputs:
#  WebsiteURL:
#     Value: !GetAtt [S3Bucket, WebsiteURL]
#     Description: url del sitio en s3
#  S3BucketSecureURL:
#     Value: !Join ['', ['https://', !GetAtt [S3Bucket, DomainName]]]
#     Description: nombre del bucket de s3
```

Conclusiones

A pesar de que no pudimos implementar la función de Amazon Cognito por cuestiones limitaciones del proyecto, las demás funciones fueron implementadas con éxito. El proyecto que realizamos para poder dar una solución al estrés humano creado por el encierro y la pandemia, puede que tenga limitaciones de uso por el momento, pero con futuro trabajo y desarrollo de la plataforma, se espera que pueda ayudar a más gente y que se cree una comunidad más grande dentro de ella. Así mismo se sabe que existe varios competidores para esta plataforma, como es youtube, facebook, instagram u otras páginas web, cuyo fin es el de entretener a los usuarios y distraerlos del mundo real, pero lo que hace diferente a nuestro proyecto de ellas, es que queremos proporcionar actividades físicas y creativas, dejando los videos distractores hacia la salud mental y del cuerpo afuera de nuestra aplicación. Con esto esperamos que los usuarios no se queden sentados enfrente de una pantalla sin hacer nada, y por lo contrario realicen actividades que reten a la mente y cuerpo humano.

Referencias

(n.d.). VPC y subredes | AWS. Se recuperó el septiembre 8, 2020 de https://docs.aws.amazon.com/es_es/vpc/latest/userguide/VPC_Subnets.html

(n.d.). Amazon Cognito | AWS. Se recuperó el septiembre 8, 2020 de https://aws.amazon.com/es/cognito/

(n.d.). Amazon EC2 | AWS. Se recuperó el septiembre 8, 2020 de https://aws.amazon.com/es/ec2/

(n.d.). Amazon S3 | AWS. Se recuperó el septiembre 8, 2020 de https://aws.amazon.com/es/s3/

(n.d.). Amazon CloudWatch | AWS. Se recuperó el septiembre 8, 2020 de https://aws.amazon.com/es/cloudwatch/

(n.d.). Amazon CloudFormation | AWS. Se recuperó el septiembre 8, 2020 de https://aws.amazon.com/es/cloudformation/

(n.d.). Amazon Relational Database Service (RDS) | AWS. Se recuperó el noviembre 10, 2020 de https://aws.amazon.com/es/rds/

(n.d.). Amazon DynamoDB | AWS. Se recuperó el noviembre 10, 2020 de https://aws.amazon.com/es/dynamodb/

(n.d.). AWS Lambda | AWS. Se recuperó el noviembre 10, 2020 de https://aws.amazon.com/es/lambda/features/

Médica Sur .(2020). Salud emocional en aislamiento y cuarentena. Se recuperó el octubre 20, 2020 de https://www.medicasur.com.mx/es_mx/ms/covid_19_Salud_emocional_en_aislamiento_y_cuarentena_

OPS (2020). Salud Mental y COVID-19. Se recuperó el 17 de noviembre, 2020 de https://www.paho.org/es/salud-mental-covid-19

Marrero, E. (2020). Aumenta el uso de alcohol y las drogas durante la pandemia de la COVID-19. Se recuperó el 17 de noviembre, 2020 de https://www.paho.org/es/salud-mental-covid-19