



Cap 7 - Big Data na Nuvem - Azure - Criando um ambiente na cloud

ATIVIDADE – CRIANDO UM AMBIENTE NA CLOUD

Faça o upload de sua atividade na plataforma FIAP ON, na seção de atividades, e aguarde a nota e feedback do professor.

Nessa fase, conhecemos dois excelentes serviços de cloud: Azure e AWS, e tivemos a oportunidade de debater sobre as vantagens que esses serviços apresentam.

É chegada a hora de configurar um ambiente para o projeto que você vem desenvolvendo desde o início do ano.

Com a sua equipe, mapeie os recursos necessários para que o seu projeto funcione em um ambiente de cloud (não necessariamente os dois apresentados).

Vocês devem criar uma conta em um serviço de nuvem e configurar o servidor de acordo com as necessidades que a equipe mapeou.

Após isso, criem um documento no Word com um template semelhante ao seguinte:

Serviço escolhido: _____

Justificativa da escolha do serviço: _____

Prints das configurações feitas no serviço, acompanhados do uso pretendido para os recursos configurados.

Ao final, exporte o documento do Word em formato PDF e suba na plataforma FIAP ON!

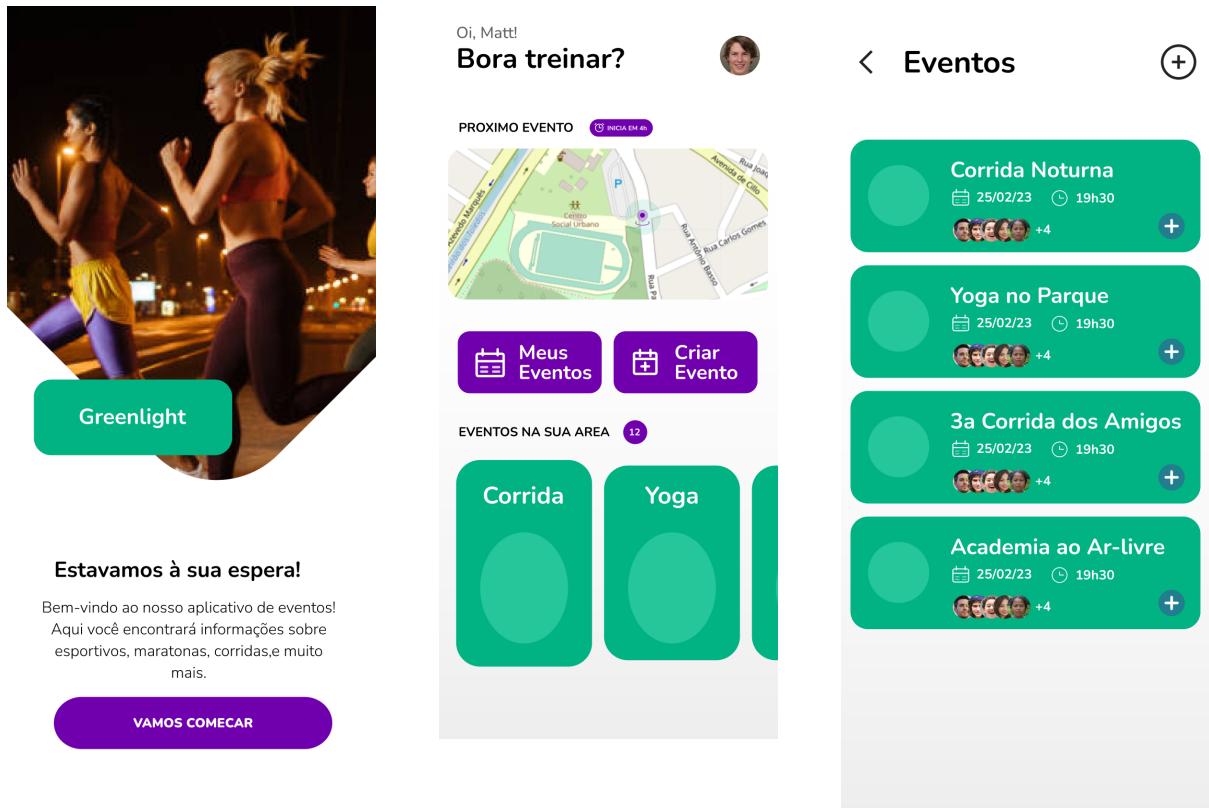
Se tiver dúvidas, consulte seus tutores!

Integrantes do grupo

- Andrei Vedovato - RM92217
- Elio Lima - RM92270
- Rafaela Rosso - RM92241

Sobre o Startup Greenlight

A startup criou um aplicativo de eventos de exercício que permite aos usuários se inscrever em eventos esportivos, maratonas, corridas de rua, torneios de tênis e muito mais. Com o aplicativo, é possível acompanhar as inscrições e resultados, receber notificações sobre novos eventos e interagir com outros participantes. O aplicativo também oferece recursos como mapas de percursos, informações sobre a localização dos eventos, horários e datas, além de dicas e informações úteis para ajudar os usuários a se prepararem para as suas participações. O objetivo da startup é atender a todos os amantes do esporte e da atividade física, independentemente do seu nível de experiência, proporcionando uma experiência única de participação em eventos esportivos.



Serviço Escolhido

Lambda AWS

Justificativa

Lambda é um serviço de computação sem servidor da AWS que permite executar código em resposta a eventos sem precisar gerenciar servidores ou infraestrutura. Ao usar o Lambda, não precisa se preocupar com capacidade, escalabilidade, disponibilidade ou manutenção de servidores. Em vez disso, a AWS cuida de tudo isso para você.

Razões pelas quais usar o Lambda na AWS:

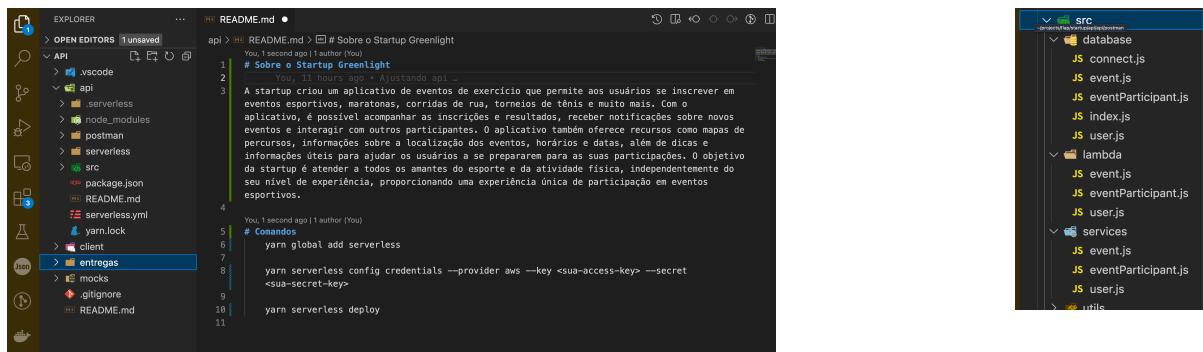
1. Custo: Com o Lambda, você paga apenas pelo tempo que sua função leva para ser executada, sem se preocupar com o tempo de inatividade ou capacidade não utilizada. Isso pode ser muito mais econômico do que manter sua própria infraestrutura ou usar servidores em nuvem tradicionais.
2. Escalabilidade: O Lambda se dimensiona automaticamente para lidar com a carga de trabalho, portanto, você não precisa se preocupar com a capacidade

ou escalabilidade de seus servidores.

3. Sem servidor: Ao não precisar gerenciar servidores, você pode se concentrar em escrever e melhorar o código de sua aplicação, em vez de se preocupar com a infraestrutura.
4. Integração com outros serviços da AWS: O Lambda se integra perfeitamente com outros serviços da AWS, permitindo que você construa aplicativos altamente escaláveis e resistentes.

Resumindo, o Lambda é uma excelente opção se você deseja executar código em resposta a eventos de maneira eficiente, econômica e sem se preocupar com a administração de servidores.

Demonstrativo da api na aws lambda em nodejs 14.xx



Arquivo de configuração serverless para deploy e manutenção do lambda.

```
api > serverless.yml > YAML > functions
      Serverless Framework Configuration - Schema for serverless framework configuration files (reference.json) | You, 11 hours ago | 1 author (You)
1  org: elioglima
2  app: aws-node-express-api-app
3  service: greenlight
4  provider:
5    name: aws
6    timeout: 30
7    memorySize: 128
8    runtime: nodejs14.x
9    region: us-east-1
10   environment:
11     MONGODB_URI: 'mongodb://localhost/greenlight'
12
13  functions: ${file(serverless/functions.yml)}      You, 12 hours ago • Ajustando api ...
14
15  plugins:
16    - serverless-offline
17
18  package:
19    excludeDevDependencies: false
```

Arquivo de função para configuração do api gateway POS, PUT, DELETE, GET

The screenshot shows the VS Code interface with the Explorer and Editor panes. The Explorer pane on the left displays the project structure under the 'API' folder, including '.vscode', 'api', 'serverless', 'src' (containing 'package.json', 'README.md', 'serverless.yml', and 'yarn.lock'), 'client', 'entregas', 'mocks', '.gitignore', and 'README.md'. The 'functions.yml' file is selected in the Explorer, highlighted with a blue border. The Editor pane on the right shows the contents of the 'functions.yml' file, which defines several AWS Lambda functions for managing users and events.

```
api > serverless > functions.yml > YAML > {} findEvent
      You, 12 hours ago | 1 author (You)

1 loginUser:
2   handler: src/lambda/user.login
3   events:
4     - http:
5       path: /usuario/cesso
6       method: POST
7
8 findUser:
9   handler: src/lambda/user.find
10  events:
11    - http:
12      path: /usuario
13      method: GET
14
15 insertUser:
16   handler: src/lambda/user.insert
17   events:
18     - http:
19       path: /usuario
20       method: PUT
21
22 updateUser:
23   handler: src/lambda/user.update
24   events:
25     - http:
26       path: /usuario
27       method: POST
28
29 removeUser:
30   handler: src/lambda/user.remove
31   events:
32     - http:
33       path: /usuario
34       method: DELETE
35
36 findEvent: You, 12 hours ago • Ajustando
37   handler: src/lambda/event.find
38   events:
39     - http:
40       path: /evento
41       method: GET
```

```
api > serverless > functions.yml > YAML > {} findEvent
    42
    43     insertEvent:
    44         handler: src/lambda/event.insert
    45         events:
    46             - http:
    47                 path: /evento
    48                 method: PUT
    49
    50     updateEvent:
    51         handler: src/lambda/event.update
    52         events:
    53             - http:
    54                 path: /evento
    55                 method: POST
    56
    57     removeEvent:
    58         handler: src/lambda/event.remove
    59         events:
    60             - http:
    61                 path: /evento
    62                 method: DELETE
    63
    64
    65     findEventParticipant:
    66         handler: src/lambda/eventParticipant.find
    67         events:
    68             - http:
    69                 path: /evento/participante
    70                 method: GET
    71
    72     insertEventParticipant:
    73         handler: src/lambda/eventParticipant.insert
    74         events:
    75             - http:
    76                 path: /evento/participante
    77                 method: PUT
    78
    79     updateEventParticipant:
    80         handler: src/lambda/eventParticipant.update
    81         events:
    82             - http:
    83                 path: /evento/participante
    84                 method: POST
    85
```

Arquivo de usuarios CRUD.

The screenshot shows a code editor window with the tab bar at the top. The active tab is 'JS user.js'. To its left is 'README.md' and to its right is a close button. Below the tabs, a breadcrumb navigation bar shows the file's location: 'api > src > services > JS user.js > [?] find > [?] searchDependences > [?] removePassword.ma...'. The main area of the editor displays the 'user.js' file content, which contains three asynchronous functions: 'update', 'insert', and 'remove'. Each function uses a try-catch block to handle database operations and return appropriate HTTP responses.

```
44
45 const update = async ({ queryStringParameters, body }) => {
46   try {
47     const params = queryStringParameters
48     const value = JSON.parse(body)
49     const filter = { id: params?.codigo }
50     const response = await db.user.update(filter, value)
51     return utils.httpHelper.ok(response)
52   } catch (error) {
53     return utils.httpHelper.badRequest(error)
54   }
55 }
56
57 const insert = async ({
58   body
59 }) => {
60   try {
61     const value = JSON.parse(body)
62     const response = await db.user.insert(value)
63     return utils.httpHelper.ok(response)
64   } catch (error) {
65     return utils.httpHelper.badRequest(error)
66   }
67 }
68
69 const remove = async ({ queryStringParameters }) => {
70   try {
71     const params = queryStringParameters
72     if (!params?.id) {
73       return utils.httpHelper.notFound()
74     }
75     const response = await db.user.remove(params?.id)
76     return utils.httpHelper.ok(response)
77   } catch (error) {
78     return utils.httpHelper.badRequest(error)
79   }
80 }
81
```

Arquivo de login do usuario

The screenshot shows a code editor window with the following details:

- File tabs: README.md, JS user.js, X
- Current file path: api > src > services > JS user.js > [e] insert
- Code content:

```
82  const login = async ({ body }) => {
83    try {
84      const { email, senha } = JSON.parse(body) || {}
85      if (!email || !senha) {
86        return utils.httpHelper.notAuthorized()
87      }
88
89      const response = await db.user.login(email, senha)
90      if (!response) {
91        return utils.httpHelper.notAuthorized()
92      }
93
94      return utils.httpHelper.ok(response)
95    } catch (error) {
96      return utils.httpHelper.badRequest(error)
97    }
98  }
99
100 module.exports = {
101   find,
102   insert,
103   update,
104   remove,
105   login
106 }
107 }
```

Mocker para desenvolvimento do aplicativo

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the project structure:

- .vscode
- api
- serverless
- src
 - backae.json
 - README.md
 - serverless.yml
 - yarn.lock
- client
- entregas
- mocks
 - eventos.json
 - participantes.json
 - usuarios.json
- .gitignore
- README.md

The Editor pane on the right shows the content of the `usuarios.json` file:

```
1 [ You, 11 hours ago | 1 author (You)
2 {
3     "_id": "59a6f9f6633aab53fa6309f1",
4     "nome": "elio.designer@hotmail.com",
5     "email": "elio.designer@hotmail.com",
6     "eventos": [
7         {
8             "_id": "642418901a3aca1c85f6b257",
9             "titulo": "Corrida Branca",
10            "descricao": "Encontre seus amigos",
11            "data": "2023-03-05T22:00:00.000Z",
12            "participantes": [
13                {
14                    "_id": "64241a7393bb4c99a8f0aed5",
15                    "descricao": "Eu quero ir 2",
16                    "data": "2023-03-29T23:43:59.687Z"
17                }
18            ]
19        }
20    ],
21 },
22 {
23     "_id": "59a6f810976eee51664ab55d",
24     "nome": "Andrei Vedovato",
25     "email": "andrei.vedovato@gmail.com",
26     "eventos": []
27 },
28 {
29     "_id": "5997676b741cf671f8b349fe",
30     "nome": "Rafaela Rosso",
31     "email": "rafa.rosso@gmail.com",
32     "eventos": []
33 }
34 ] You, 11 hours ago • Ajustando mocks ...
```

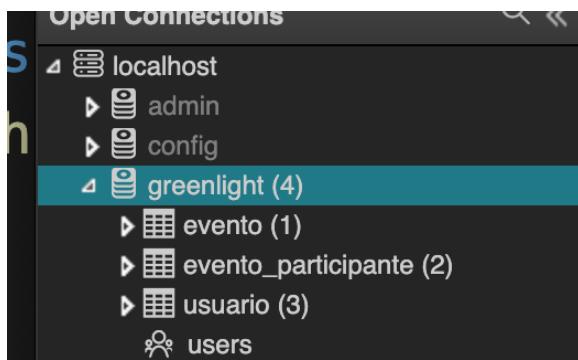
README.md ● { } eventos.json X

mocks > { } eventos.json > ...

You, 11 hours ago | 1 author (You)

```
1 [  
2 {  
3     "_id": "642418901a3aca1c85f6b257",  
4     "titulo": "Corrida Branca",  
5     "descricao": "Encontre seus amigos",  
6     "data": "2023-03-05T22:00:00.000Z",  
7     "usuario": {  
8         "_id": "59a6f9f6633aab53fa6309f1",  
9         "nome": "elio.designer@hotmail.com",  
10        "email": "elio.designer@hotmail.com",  
11        "senha": "Ab@123456"  
12    },  
13    "participantes": [  
14        {  
15            "_id": "64241a7393bb4c99a8f0aed5",  
16            "usuarioId": "59a6f9f6633aab53fa6309f1",  
17            "descricao": "Eu quero ir 2",  
18            "data": "2023-03-29T23:43:59.687Z",  
19            "nome": "elio.designer@hotmail.com",  
20            "email": "elio.designer@hotmail.com",  
21            "senha": "Ab@123456"  
22        },  
23        {  
24            "_id": "6424d7a3d5824bb0579e7d42",  
25            "usuarioId": "59a6f9f6633aab53fa6309f1",  
26            "descricao": "Eu quero ir",  
27            "data": "2023-03-30T00:28:19.853Z",  
28            "nome": "elio.designer@hotmail.com",  
29            "email": "elio.designer@hotmail.com",  
30            "senha": "Ab@123456"  
31        }  
32    ]  
33 }]  
34 ] You, 11 hours ago • Ajustando mocks ...
```

Demonstrativo mongodb com collections criadas (evento, evento_participante, usuario), segue abaixo o descritivo de cada uma.



Publicação

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    ...
zsh - api + ▾  ✖  ... ×

○ elioglima in api/api on [!] main took 3m 32.1s >
elioglima in api/api on [!] > yarn serverless deploy
yarn run v1.22.19
$ /Users/elioglima/projects/fiap/startup/api/api/node_modules/.bin/serverless deploy

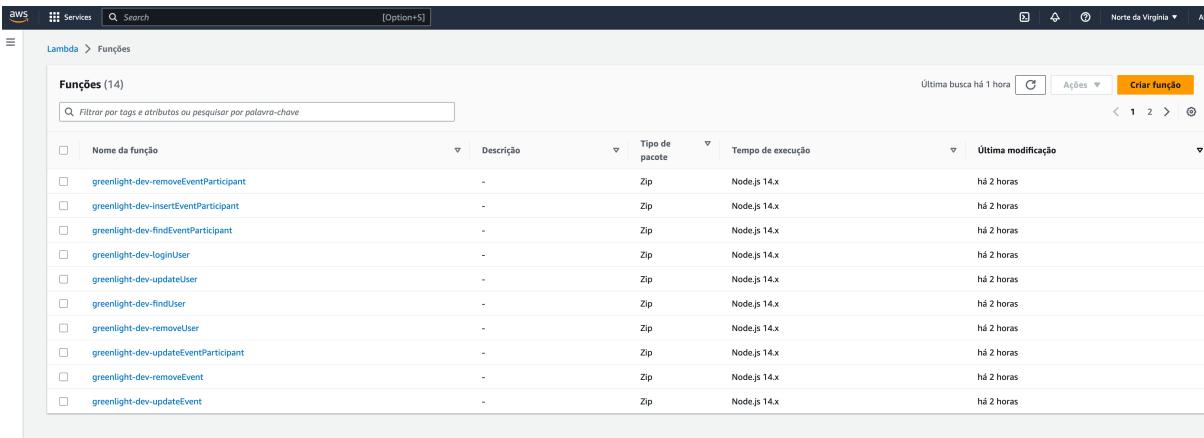
Deploying greenlight to stage dev (us-east-1)

✓ Service deployed to stack greenlight-dev (139s)

dashboard: https://app.serverless.com/elioglima/apps/aws-node-express-api-app/greenlight/dev/us-east-1
endpoints:
  POST - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/usuario/acesso
  GET - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/usuario
  PUT - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/usuario
  POST - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/usuario
  DELETE - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/usuario
  GET - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/evento
  PUT - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/evento
  POST - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/evento
  DELETE - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/evento
  GET - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/evento/participante
  PUT - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/evento/participante
  POST - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/evento/participante
  DELETE - https://4pwjf3y01j.execute-api.us-east-1.amazonaws.com/dev/evento/participante
functions:
  loginUser: greenlight-dev-loginUser (55 MB)
  findUser: greenlight-dev-findUser (55 MB)
  insertUser: greenlight-dev-insertUser (55 MB)
  updateUser: greenlight-dev-updateUser (55 MB)
  removeUser: greenlight-dev-removeUser (55 MB)
  findEvent: greenlight-dev-findEvent (55 MB)
  insertEvent: greenlight-dev-insertEvent (55 MB)
  updateEvent: greenlight-dev-updateEvent (55 MB)
  removeEvent: greenlight-dev-removeEvent (55 MB)
  findEventParticipant: greenlight-dev-findEventParticipant (55 MB)
  insertEventParticipant: greenlight-dev-insertEventParticipant (55 MB)
  updateEventParticipant: greenlight-dev-updateEventParticipant (55 MB)
  removeEventParticipant: greenlight-dev-removeEventParticipant (55 MB)
  ✨ Done in 144.94s.

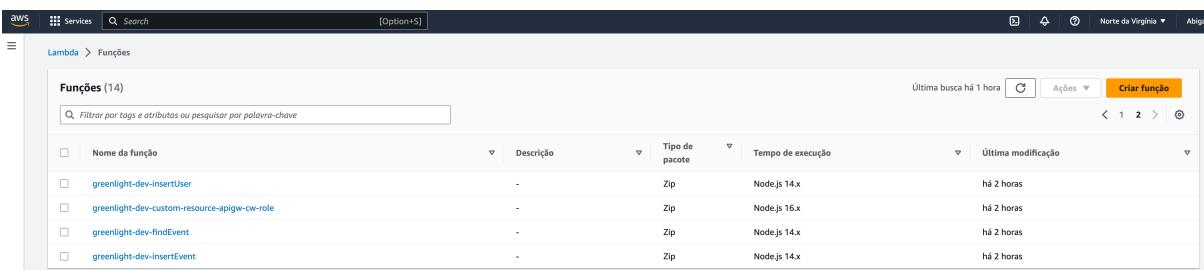
○ elioglima in api/api on [!] main took 2m 25.2s > [ ]
```

AWS Lambda



The screenshot shows the AWS Lambda service interface. In the top navigation bar, 'Lambda' is selected under 'Services'. The main title 'Funções (14)' is displayed above a search bar. A table lists 14 Lambda functions, each with a checkbox, name, description, package type (Zip), runtime (Node.js 14.x), execution time, and last modification date (all updated 2 hours ago). The table includes columns for Name, Description, Package Type, Execution Time, and Last Modification.

Nome da função	Descrição	Tipo de pacote	Tempo de execução	Última modificação
greenlight-dev-removeEventParticipant	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-insertEventParticipant	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-findEventParticipant	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-loginUser	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-updateUser	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-findUser	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-removeUser	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-updateEventParticipant	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-removeEvent	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-updateEvent	-	Zip	Node.js 14.x	há 2 horas



This screenshot shows the same AWS Lambda service interface as the first one, but with a smaller set of 4 functions listed. The table structure is identical, showing names like 'greenlight-dev-insertUser', 'greenlight-dev-custom-resource-apigw-cw-role', 'greenlight-dev-findEvent', and 'greenlight-dev-insertEvent', all updated 2 hours ago.

Nome da função	Descrição	Tipo de pacote	Tempo de execução	Última modificação
greenlight-dev-insertUser	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-custom-resource-apigw-cw-role	-	Zip	Node.js 16.x	há 2 horas
greenlight-dev-findEvent	-	Zip	Node.js 14.x	há 2 horas
greenlight-dev-insertEvent	-	Zip	Node.js 14.x	há 2 horas

AWS API Gateway

Screenshot of the AWS Lambda function configuration page for "greenlight-dev-findUser".

Visão geral da função

This function belongs to an application. [Click here to manage it.](#)

Funções relacionadas: Selecionar uma função

API Gateway

Adicionar gatilho

Descrição

Última modificação há 1 hora

ARN da função arn:aws:lambda:us-east-1:342061705349:function:greenlight-dev-findUser

Aplicativo greenlight-dev

URL da função [Informações](#)

Código | Testar | Monitor | Configuração | Aliases | Versões

Origem do código [Informações](#)

O pacote de implantação da função do Lambda "greenlight-dev-findUser" é muito grande para habilitar a edição de código em linha. Contudo, você ainda pode invocar a função.

Propriedades do código [Informações](#)

Tamanho do pacote 52,6 MB Hash SHA256 WqmWo63XSPu617ml2XfOa/t5rWiKh92EVD6TXCXdjxA= Última modificação 30 de março de 2023 às 12:23 BRT

Configurações de tempo de execução [Informações](#)

Tempo de execução Node.js 14.x Manipulador [Informações](#) s_findUser.handler Arquitetura [Informações](#) x86_64

Camadas [Informações](#)

Ordem de mesclagem Nome Versão da camada Tempos de execução compatíveis Arquiteturas compatíveis ARN da versão

Nenhum dado a ser exibido.

Lambda > Funções > greenlight-dev-findUser

greenlight-dev-findUser

This function belongs to an application. [Click here](#) to manage it.

Visão geral da função [Informações](#)

greenlight-dev-findUser [Funções relacionadas:](#) [Selecionar uma função](#)

Layers (0) [+ Adicionar destino](#)

API Gateway [+ Adicionar gatilho](#)

[+ Adicionar destino](#)

Descrição
-

Última modificação
há 1 hora

ARN da função
[arn:aws:lambda:us-east-1:342061705349:function:greenlight-dev-findUser](#)

Aplicativo
greenlight-dev

URL da função [Informações](#)
-

Código | **Testar** | **Monitor** | **Configuração** | **Aliases** | **Versões**

Variáveis de ambiente (1)
As variáveis de ambiente a seguir são criptografadas em repouso com a chave de serviço padrão do Lambda. [Editar](#)

Chave	Valor
MONGODB_URI	mongodb+srv://admin:AB102030@greenliht.z4jbtfo.mongodb.net/test

Variáveis de ambiente

- Tags
- VPC
- Ferramentas de monitoramento e operações
- Simultaneidade
- Invocação assíncrona
- Assinatura de código
- Proxies de banco de dados
- Sistemas de arquivos
- Máquinas de estado

Screenshot of the AWS Lambda function configuration page for 'greenlight-dev-findUser'.

Visão geral da função

Funções relacionadas: Selecionar uma função

API Gateway

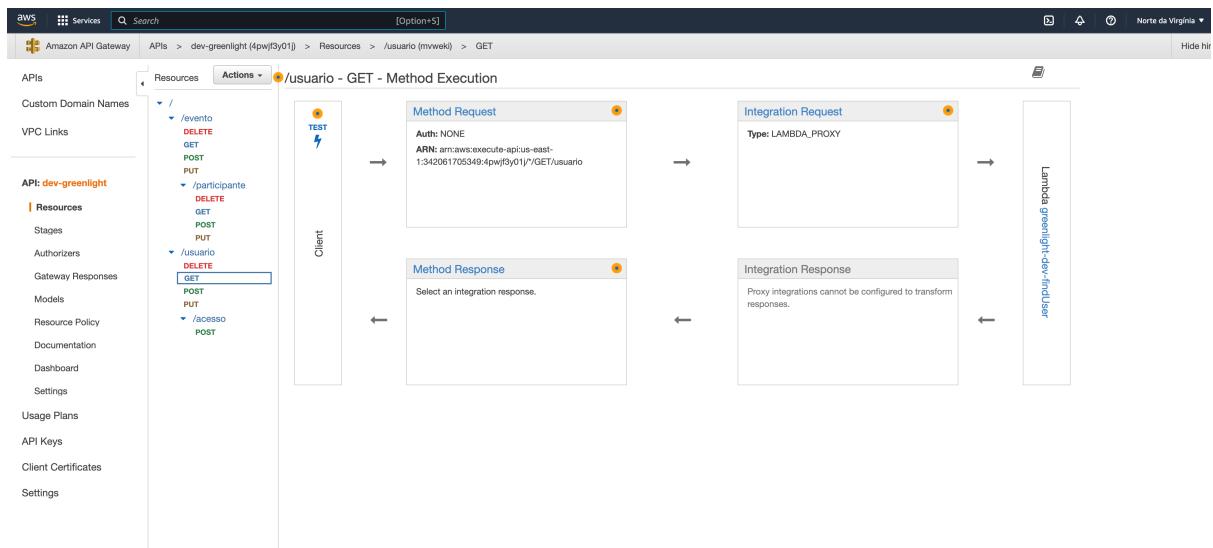
Gatilhos (1)

Nome	Descrição	ID	Protocolo	Tipo de endpoint	Criado(a)
API Gateway: dev-greenlight	amaws:execute-api:us-east-1:342061705349:4pwjf3y01j/*	4pwjf3y01j	REST	Edge	2023-03-30

Screenshot of the AWS API Gateway API list page.

APIs (1)

Nome	Descrição	ID	Protocolo	Tipo de endpoint	Criado(a)
dev-greenlight	4pwjf3y01j	REST	Edge	2023-03-30	



Chamadas para o server publicado

The screenshot shows the Postman application interface. On the left, there's a sidebar with various sections like Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named 'FIAP'. Within this collection, there's an 'API' section which contains a 'Usuarios' folder. Under 'Usuarios', there are several requests: 'GET Consultar Usuario por Codi...', 'GET Consultar Todos Usuarios' (which is selected), 'POST Atualizar Usuario', 'POST Acesso ao Sistema', 'PUT Inserir Usuario', and 'DELETE Remove Usuario'. Below the 'Usuarios' folder is another folder named 'Eventos'. The 'Consultar Todos Usuarios' request is expanded, showing its details. The 'Body' tab is selected, displaying the JSON response. The response is a list of users, each represented by an object with fields: '_id', 'nome', 'email', and 'eventos'. The 'eventos' field contains an array of event objects, each with fields: '_id', 'titulo', 'descricao', 'data', and 'participantes'. The JSON is formatted with line numbers from 1 to 34 on the left.

```
1 {
2   "_id": "59a6f9f6633aab53fa6309f1",
3   "nome": "elio.designer@hotmail.com",
4   "email": "elio.designer@hotmail.com",
5   "eventos": [
6     {
7       "_id": "642418901a3aca1c85f6b257",
8       "titulo": "Corrida Branca",
9       "descricao": "Encontre seus amigos",
10      "data": "2023-03-05T22:00:00.000Z",
11      "participantes": [
12        {
13          "_id": "64241a7393bb4c99a8f0aed5",
14          "descricao": "Eu quero ir 2",
15          "data": "2023-03-29T23:43:59.687Z"
16        }
17      ]
18    }
19  ],
20 },
21 },
22 {
23   "_id": "59a6f810976eee51664ab55d",
24   "nome": "Andrei Vedovato",
25   "email": "andrei.vedovato@gmail.com",
26   "eventos": []
27 },
28 {
29   "_id": "5997676b741cf671f8b349fe",
30   "nome": "Rafaela Rosso",
31   "email": "rafa.rosso@gmail.com",
32   "eventos": []
33 }
34 ]
```

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections like Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named 'FIAP'. Under 'API', there's a 'Usuarios' folder containing several endpoints: 'Consultar Usuario por Código', 'Consultar Todos Usuarios', 'Atualizar Usuario', 'Acesso ao Sistema', 'Inserir Usuario', and 'Remove Usuario'. Below 'Usuarios' is another folder 'Eventos' containing 'Consultar Evento por Código', 'Consultar Todos Eventos', 'Atualizar Eventos', 'Inserir Evento', and 'Remove Evento'. A specific endpoint, 'Atualizar Usuario', is selected and expanded. The details pane shows a POST request with the URL `({{server}})/usuario?codigo=59a6f9f6633aab53fa6309f1`. The 'Body' tab is active, displaying the following JSON payload:

```
1  {
2     "nome": "Elio Lima",
3     "email": "elio.designer@hotmail.com"
4 }
```

At the bottom, the response pane shows a successful 200 OK status with 169 ms latency and 322 B size. The response body is:

```
1 {
2     "error": false,
3     "length": 1,
4     "data": {
5         "acknowledged": true,
6         "modifiedCount": 0,
7         "upsertedId": null,
8         "upsertedCount": 0,
9         "matchedCount": 1
10    }
11 }
```

The screenshot shows the Postman application interface. On the left, there's a sidebar with various sections like Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named "FIAP". Under "API", there are several endpoints categorized under "Usuarios", "Eventos", and "Participantes". The selected endpoint is "POST Acesso ao Sistema". The request details show a POST method with the URL {{server}}/usuario/cesso. The "Body" tab is selected, showing a JSON payload:

```
1 {  
2   "email": "elio.designer@hotmail.com",  
3   "senha": "Ab@123456"  
4 }
```

Below the body, the response status is 200 OK with 197 ms and 335 B. The response body is also shown in JSON format:

```
1 {  
2   "error": false,  
3   "length": 1,  
4   "data": [  
5     {  
6       "_id": "59a6f9f663aab53fa6309f1",  
7       "nome": "elio.designer@hotmail.com",  
8       "email": "elio.designer@hotmail.com"  
9     }  
10   ]  
11 }
```

The screenshot shows the Postman application interface. On the left, the sidebar displays the 'FIAP' collection, which contains an 'API' section with 'Eventos' and 'Participantes' sub-sections. The 'Eventos' section is currently selected, showing four methods: GET Consultar Evento por Código, GET Consultar Todos Eventos, POST Atualizar Eventos, and PUT Inserir Evento. The 'Consultar Todos Eventos' method is highlighted. The main workspace shows the request details for this method: a GET request to `((server))/evento`. The 'Body' tab is selected, showing the response body as a JSON object. The response status is 200 OK, with a duration of 234 ms and a size of 955 B. The response body is as follows:

```
1  {
2   "id": "642418901a3aca1c85f6b257",
3   "titulo": "Corrida Branca",
4   "descricao": "Encontre seus amigos",
5   "data": "2023-03-05T22:00:00.000Z",
6   "usuario": {
7     "_id": "59a6f9f6633aab53fa6309f1",
8     "nome": "elio.designer@hotmail.com",
9     "email": "elio.designer@hotmail.com",
10    "senha": "Ab@123456"
11  },
12  "participantes": [
13    {
14      "_id": "64241a7393bb4c99a8f0aed5",
15      "usuarioId": "59a6f9f6633aab53fa6309f1",
16      "descricao": "Eu quero ir 2",
17      "data": "2023-03-29T23:43:59.687Z",
18      "nome": "elio.designer@hotmail.com",
19      "email": "elio.designer@hotmail.com",
20      "senha": "Ab@123456"
21    },
22    {
23      "_id": "6424d7a3d5824bb0579e7d42",
24      "usuarioId": "59a6f9f6633aab53fa6309f1",
25      "descricao": "Eu quero ir",
26      "data": "2023-03-30T00:28:19.853Z",
27      "nome": "elio.designer@hotmail.com",
28      "email": "elio.designer@hotmail.com",
29      "senha": "Ab@123456"
30    }
31  ]
32 }
```

The screenshot shows the Postman interface with the following details:

- Request URL:** {{server}}/evento/participante
- Response Headers:** 200 OK, 177 ms, 599 B
- Response Body (Pretty JSON):**

```

1
2   "error": false,
3   "length": 2,
4   "data": [
5     {
6       "_id": "64241a7393bb4c99a8f0aed5",
7       "eventoId": "642418901a3aca1c85f6b257",
8       "usuarioId": "59a6f9f6633aab53fa6309f1",
9       "descricao": "Eu quero ir 2",
10      "data": "2023-03-29T23:43:59.687Z"
11    },
12    {
13      "_id": "6424d7a3d5824bb0579e7d42",
14      "eventoId": "59a6f9f6633aab53fa6309f1",
15      "usuarioId": "59a6f9f6633aab53fa6309f1",
16      "descricao": "Eu quero ir",
17      "data": "2023-03-30T00:28:19.853Z"
18    }
19  ]
20

```

The screenshot shows the MongoDB Compass interface with the following details:

- Query:**

```

1 db.evento.find({})
2   .projection({})
3   .sort({_id:-1})
4   .limit(100)

```

- Results:** 1 Doc

Key	Type
_id	ObjectId
usuarioId	ObjectId
titulo	String
descricao	String
data	Date

```

1 (1) 642418901a3aca1c85f6b257 { titulo : "Corrida Branca", descricao : "Encontre seus amigos" } (5 fields)
2 _id: 642418901a3aca1c85f6b257
3 usuarioId: 59a6f9f6633aab53fa6309f1
4 titulo: Corrida Branca
5 descricao: Encontre seus amigos
6 data: 05/03/2023 19:00:00 - 25 days ago

```

```

1 db.evento_participante.find({})
2   .projection({})
3   .sort({_id:-1})
4   .limit(100)

```

evento_participante | 0.041 s | 2 Docs

Key	Type
6424d7a3d5824bb0579e7d42	Document
_id	ObjectId
eventold	ObjectId
usuariold	ObjectId
descricaو	String
data	String
64241a7393bb4c99a8f0aed5	Document
_id	ObjectId
eventold	ObjectId
usuariold	ObjectId
descricaو	String
data	String

```

localhost > greenlight >
1 db.usuario.find({})
2   .projection({})
3   .sort({_id:-1})
4   .limit(100)

```

usuario | 0.030 s | 3 Docs

Key	Type
59a6f9f6633aab53fa6309f1	Document
_id	ObjectId
nome	String
email	String
senha	String
59a6fb10976eee51664ab55d	Document
_id	ObjectId
nome	String
email	String
senha	String
5997676b741cf671fb349fe	Document
_id	ObjectId
nome	String
email	String
senha	String