

2TDSOS-2022

- Cap 9 - Pode trazer a conta
Grand Finale

Elio Gonçalves de Lima
RM92270

Andrei Vedovato
RM92217

Rafaela Rosso Pacheco
RM92241



Objetivo

Se você fez as atividades anteriores, pode reaproveitar e utilizar como base para o desenvolvimento do projeto final, já que a proposta deste último desafio nada mais é do que unir tudo o que foi feito até agora!

- 1 Aplicação Web
- 1 Aplicação Mobile
- Integração entre os sistemas

Grand Finale

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.

Grand Finale sobre as tecnologias

API Rest
AWS Lambda
Node js / typescript / serverless.yml

Banco de Dados
MongoDB

Docker Container/ Postman
Ambiente de Test Local

Web
React JS / Node Js / Typescript

Aplicativo
React Native / Node Js / Typescript

Link Repositório GitHub

<https://github.com/elioglima/fiap-startup-greenlight>

Link Protótipo

<https://www.figma.com/proto/qUVFbvQ5pQ9GHWly1ktJuO/App-Exercicio?type=design&node-id=205-551&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=444%3A84>

App Exercício – Figma

figma.com/file/qUVFbv05pQ9CHWly1ktJuO/App-Exercício?type=design&node-id=0-1&t=b1G2JkkK3Dlf78IK-0

Any.do New chat Login - Educação... Indianápolis - Sta... Aliexpress StartUP Como Usar a Geol... Ultracon - CLARO

Layers Assets Page 1

Splash Login social Entregas

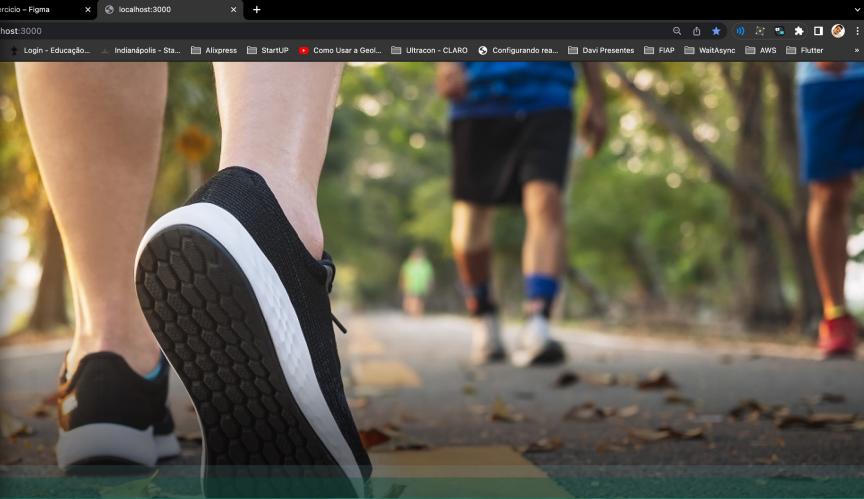
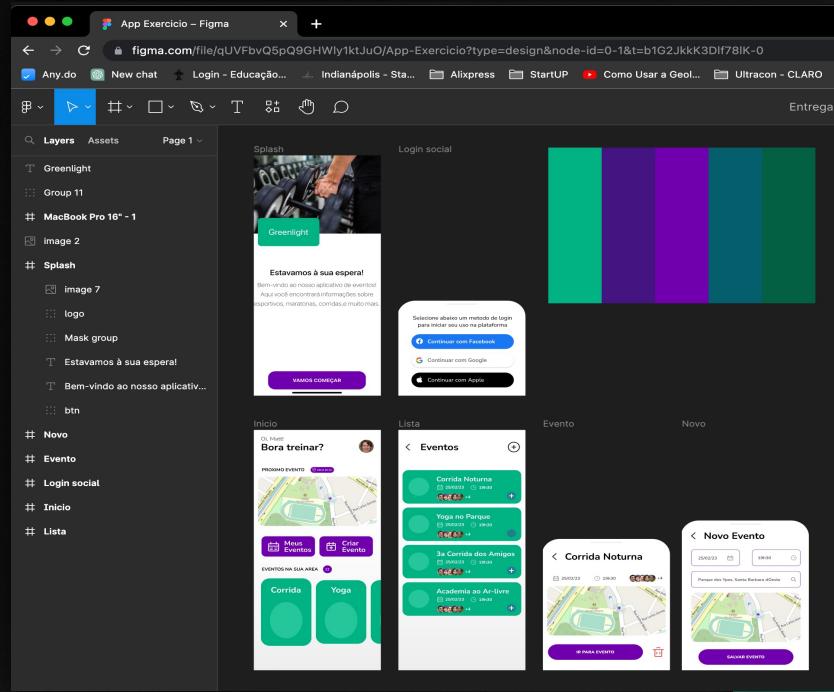
Greenlight Group 1 MacBook Pro 15" - 1 Image 2 Splash image 7 logo Mask group Estavamos à sua espera! Bem-vindo ao nosso aplicativo de eventos! Aqui você encontrará informações sobre esportes, maratona, corrida e muito mais! Seleção: escolha um método de login para iniciar seu uso na plataforma Continuar com Facebook Continuar com Google Continuar com Apple VAMOS COMEÇAR!

Novo Início Lista Evento Novo

Bora treinar? PROXIMO EVENTO Meus Eventos Criar Evento EVENTOS NA SUA ÁREA Corrida Yoga

Eventos < Corrida Noturna 2023-09-15 20:00 + 3a Corrida dos Amigos 2023-09-16 20:00 + Academia ao Ar-Livre 2023-09-17 20:00 +

< Novo Evento Corrida Noturna 2023-09-15 20:00 + Salvar Evento



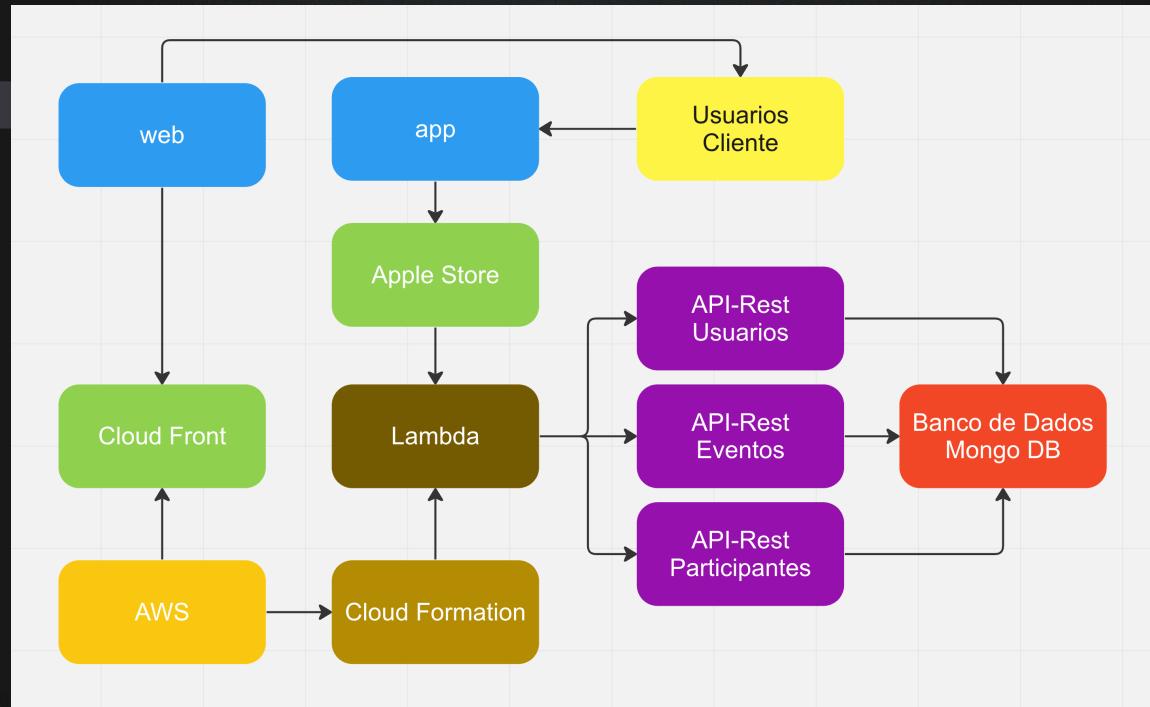
Greenlight



Grand Finale diagrama macro estrutura

FIAP-STARTUP-GREENLIGHT

- > .github
- > .vscode
- > api
- > client
- > docker_greenlight
- > docs
- > entregas
- > mocks
- > node_modules
- > .gitignore
- > package.json
- > README.md
- > yarn-error.log
- > yarn.lock



Grand Finale API

Tecnologia utilizadas

Uma API REST (Representational State Transfer) é uma abordagem popular para construir serviços web que seguem princípios e convenções bem definidos. Neste caso, a API é construída usando Node.js, uma plataforma de tempo de execução JavaScript, e TypeScript, que é um superconjunto de JavaScript com suporte a tipos estáticos. A API é implantada na AWS Lambda, um serviço de computação sem servidor fornecido pela Amazon Web Services.

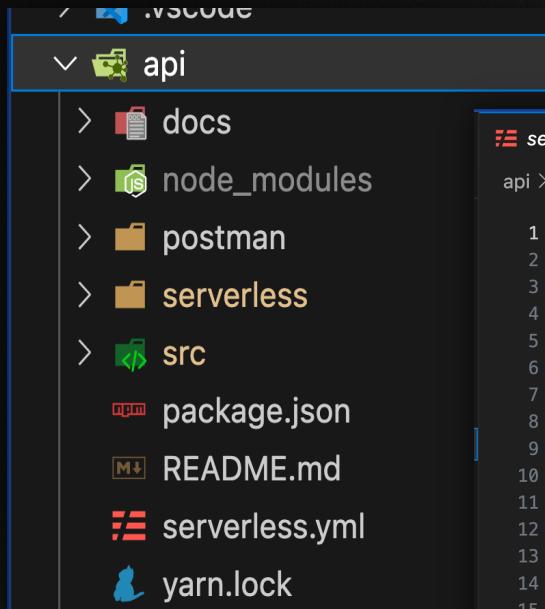
Node.js e TypeScript: Node.js é uma plataforma que permite executar código JavaScript no lado do servidor. TypeScript é uma linguagem que adiciona recursos de tipagem estática ao JavaScript. Juntos, eles permitem criar aplicativos backend robustos e escaláveis.

AWS Lambda: A AWS Lambda é um serviço de computação sem servidor que permite executar código sem precisar provisionar ou gerenciar servidores. Você pode criar uma função Lambda contendo seu código da API e a AWS cuidará da execução e escalabilidade.

Estrutura da API REST: A API REST é construída seguindo as práticas e convenções definidas pela arquitetura REST. Você definirá endpoints HTTP para manipular diferentes recursos. Por exemplo, você pode ter um endpoint GET /users para recuperar informações de usuários.

Grand Finale API

Estrutura do Projeto



Configurações de Deploy

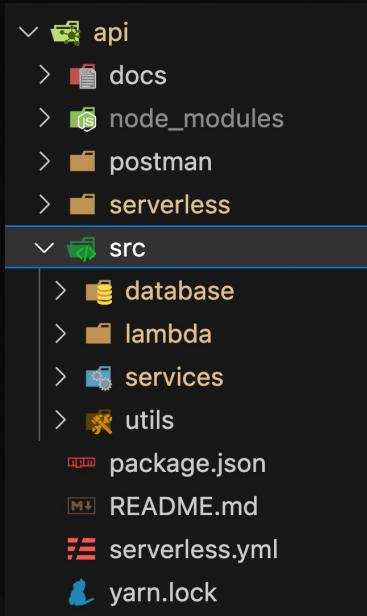
```
serverless.yml ×
api > serverless.yml > YAML > org
Serverless Framework Configuration - Schema for serverless framework configuration files
1 org: elioglima You, há 2 meses • Ajustando mocks ...
2 app: aws-node-express-api-app
3 service: greenlight
4 provider:
5   name: aws
6   timeout: 30
7   memorySize: 128
8   runtime: nodejs14.x
9   region: us-east-1
10  environment:
11    MONGODB_URI: "mongodb://localhost:27017/greenlight"
12
13  functions: ${file(serverless/functions.yml)}
14
15  plugins:
16    - serverless-offline
17
18  package:
19    excludeDevDependencies: false
```

Configurações das Funções

```
api > serverless > functions.yml > Serverless IDE > {} upload
You, há 11 segundos | 1 author (You)
1 loginUser:
2   handler: src/lambda/user.login
3   events:
4     - http:
5       path: /usuario/acesso
6       method: POST
7
8 refreshToken:
9   handler: src/lambda/user.refresh
10  events:
11    - http:
12      path: /usuario/acesso
13      method: PUT
14
15 findUser:
16   handler: src/lambda/user.find
17  events:
18    - http:
19      path: /usuario
20      method: GET
21
22 insertUser:
23   handler: src/lambda/user.insert
24  events:
25    - http:
26      path: /usuario
27      method: POST
28
29 updateUser:
30   handler: src/lambda/user.update
31  events:
32    - http:
33      path: /usuario
34      method: PUT
35
36 uploadPhotoUser:
37   handler: src/lambda/user.updatePhoto
38  events:
39    - http:
40      path: /usuario/foto
41      method: PUT
42
43 removeUser:
44   handler: src/lambda/user.remove
45  events:
46    - http:
47      path: /usuario
48      method: DELETE
```

Grand Finale API

Estrutura



Serviços

```

  event.js M
  api > src > services > JS event.js > find > listMount > list.map() callback > userData
  You, há 8 segundos | author (You)
  1 const httpHelper = require("../utils/httpHelper");
  2 const db = require("./database");
  3
  4 const find = async ({ queryStringParameters }) => {
  5   try {
  6     const params = queryStringParameters || {};
  7     const query = {
  8       ...(params.id ? { id: params.id } : {}),
  9       ...!(params.usuarioId ? { usuarioId: params.usuarioId } : {}),
  10      ...!(params.categoriaId ? { categoriaId: params.categoriaId } : {}),
  11    };
  12
  13    const response = await db.event.find(query);
  14    const list = response.data || [];
  15    const listMount = await Promise.all(
  16      list.map(async (m) => {
  17        const eventParticipantData = await db.eventParticipant.find({
  18          eventId: m._id,
  19        });
  20
  21        const userData = await db.user.findOne({ _id: m.usuarioId });
  22        const categoryData = await db.category.findOne({
  23          id: m.categoriaId,
  24        });
  25
  26        delete m.eventId;
  27        delete m.usuarioId;
  28        const participantes = await Promise.all(
  29          evenParticipantData.data.map((ev) => {
  30            const userData = await db.user.findOne({ _id: ev.usuarioId });
  31
  32            if (ev.eventId) delete ev.eventId;
  33            if (userData.data._id) delete userData.data._id;
  34
  35            const userResp = userData.data;
  36            delete userResp.senha;
  37            delete userResp.token;
  38            return {
  39              ...ev,
  40              ...{ ev },
  41              ...!(userResp.name ? userResp : { userNotFound: true }),
  42            };
  43          })
  44        );
  45
  46        return {
  47          ...m,
  48          usuario: userData.data,
  49          categoria: categoryData.data,
  50          participantes,
  51        };
  52      });
  53    }
  54  }
  55
  56  const find = async ({ queryStringParameters }) => {
  57    try {
  58      const params = queryStringParameters || {};
  59      const query = {
  60        ...(params.id ? { id: params.id } : {}),
  61      };
  62      const response = await db.user.find(query);
  63      if (response.length == 0) {
  64        return utils.httpHelper.ok(response.data);
  65      }
  66
  67      const removePassword = await Promise.all(
  68        response.data.map(async (user) => {
  69          delete user.senha;
  70          return user;
  71        })
  72      );
  73
  74      const searchDependences = await Promise.all(
  75        removePassword.map(async (user) => {
  76          const eventData = await db.event.find({ usuarioId: user._id });
  77
  78          const eventDependences = await Promise.all(
  79            eventData.data.map((sync) => {
  80              const participantData = await db.eventParticipant.find({
  81                eventId: sync._id,
  82              });
  83
  84              delete eventDependences.eventId;
  85              delete eventDependences.usuarioId;
  86              return participantData;
  87            })
  88          );
  89
  90          return {
  91            ...user,
  92            eventos: eventDependences,
  93          };
  94        })
  95      );
  96
  97      return {
  98        ...user,
  99        eventos: eventDependences,
  100      };
  101    }
  102  }
  103
  104  const removePassword = await Promise.all(
  105    response.data.map(async (user) => {
  106      delete user.senha;
  107      return user;
  108    })
  109  );
  110
  111  const searchDependences = await Promise.all(
  112    removePassword.map(async (user) => {
  113      const eventData = await db.event.find({ usuarioId: user._id });
  114
  115      const eventDependences = await Promise.all(
  116        eventData.data.map((sync) => {
  117          const participantData = await db.eventParticipant.find({
  118            eventId: sync._id,
  119          });
  120
  121          delete eventDependences.eventId;
  122          delete eventDependences.usuarioId;
  123          return participantData;
  124        })
  125      );
  126
  127      return {
  128        ...user,
  129        eventos: eventDependences,
  130      };
  131    })
  132  );
  133
  134  return {
  135    ...user,
  136    eventos: eventDependences,
  137  };
  138}
  
```

Modulo de usuário

```

  api > src > services > JS user.js > refresh
  You, há 1 segundo | author (You)
  1 const utils = require("../utils");
  2 const db = require("./database");
  3 const moment = require("moment");
  4
  5 const find = async ({ queryStringParameters }) => {
  6   try {
  7     const params = queryStringParameters || {};
  8     const query = {
  9       ...(params.id ? { id: params.id } : {}),
  10      };
  11      const response = await db.user.find(query);
  12      if (response.length == 0) {
  13        return utils.httpHelper.ok(response.data);
  14      }
  15
  16      const removePassword = await Promise.all(
  17        response.data.map(async (user) => {
  18          delete user.senha;
  19          return user;
  20        })
  21      );
  22
  23      const searchDependences = await Promise.all(
  24        removePassword.map(async (user) => {
  25          const eventData = await db.event.find({ usuarioId: user._id });
  26
  27          const eventDependences = await Promise.all(
  28            eventData.data.map((sync) => {
  29              const participantData = await db.eventParticipant.find({
  30                eventId: sync._id,
  31              });
  32
  33              delete eventDependences.eventId;
  34              delete eventDependences.usuarioId;
  35              return participantData;
  36            })
  37          );
  38
  39          return {
  40            ...user,
  41            eventos: eventDependences,
  42          };
  43        })
  44      );
  45
  46      return {
  47        ...user,
  48        eventos: eventDependences,
  49      };
  50    }
  51  }
  52
  53  const removePassword = await Promise.all(
  54    response.data.map(async (user) => {
  55      delete user.senha;
  56      return user;
  57    })
  58  );
  59
  60  const searchDependences = await Promise.all(
  61    removePassword.map(async (user) => {
  62      const eventData = await db.event.find({ usuarioId: user._id });
  63
  64      const eventDependences = await Promise.all(
  65        eventData.data.map((sync) => {
  66          const participantData = await db.eventParticipant.find({
  67            eventId: sync._id,
  68          });
  69
  70          delete eventDependences.eventId;
  71          delete eventDependences.usuarioId;
  72          return participantData;
  73        })
  74      );
  75
  76      return {
  77        ...user,
  78        eventos: eventDependences,
  79      };
  80    })
  81  );
  82
  83  return {
  84    ...user,
  85    eventos: eventDependences,
  86  };
  87}
  
```

Grand Finale API

Insert, Update, Find, Delete

```
api > src > services > JS user.js > [e] find > [e] removePassword > ↗ response.data.map()
  56  };
  57
  58 const update = async ({ queryParams, body }) => {
  59   try {
  60     const params = queryParams;
  61     const value = body;
  62     const filter = { id: params._id };
  63
  64     const response = await db.user.update(filter, value);
  65     return utils.httpHelper.ok(response);
  66   } catch (error) {
  67     return utils.httpHelper.badRequest(error);
  68   }
  69 };
  70
  71 const insert = async ({ body }) => {
  72   try {
  73     const value = JSON.parse(body);
  74     const response = await db.user.insert(value);
  75     return utils.httpHelper.ok(response);
  76   } catch (error) {
  77     return utils.httpHelper.badRequest(error);
  78   }
  79 };
  80
  81 const remove = async ({ queryStringParameters }) => {
  82   try {
  83     const params = queryStringParameters;
  84     if (!params?.id) {
  85       return utils.httpHelper.notFound();
  86     }
  87     const response = await db.user.remove(params?.id);
  88     return utils.httpHelper.ok(response);
  89   } catch (error) {
  90     return utils.httpHelper.badRequest(error);
  91   }
  92 };
  93
  94 const createAuth = async ({ usuarioId }) => {
  95   try {
  96     const secretKey = await utils.dataController.encryptObject({
  97       usuarioId,
  98     });
  99
 100    const token = await utils.dataController.encryptObject(
 101      {
 102        usuarioId,
 103        date: moment().add(10, "minute"),
 104      }
 105    );
 106
 107    const token = await utils.dataController.encryptObject(
 108      {
 109        usuarioId,
 110        date: moment().add(10, "minute"),
 111        secretKey
 112      }
 113    );
 114
 115    await update({
 116      queryParams: { _id: usuarioId },
 117      body: { token: token },
 118    });
 119
 120    return {
 121      usuarioId,
 122      token,
 123      refreshKey: secretKey,
 124    };
 125  } catch (error) {
 126    console.error(error);
 127    return undefined;
 128  }
 129
 130  const login = async ({ body }) => {
 131    try {
 132      const { email, senha } = JSON.parse(body) || {};
 133      if (!email || !senha) {
 134        return utils.httpHelper.notAuthorized();
 135      }
 136
 137      const response = await db.user.login(email, senha);
 138
 139      if (!response || !response.length) {
 140        return utils.httpHelper.notAuthorized();
 141      }
 142
 143      const usuarioId = response.data?._id;
 144      const authentication = await createAuth({
 145        usuarioId,
 146      });
 147
 148      if (!authentication) {
 149        return utils.httpHelper.notAuthorized();
 150      }
 151
 152    } catch (error) {
 153      console.error(error);
 154      return undefined;
 155    }
 156  }
 157
 158  return [
 159    ...response.data.map((user) => {
 160      const auth = await createAuth({ usuarioId: user._id });
 161
 162      return {
 163        ...user,
 164        auth
 165      };
 166    })
 167  ];
 168}
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
```

Autenticação e Login

```
api > src > services > JS user.js > [e] insert
  93
  94 const createAuth = async ({ usuarioId }) => {
  95   try {
  96     const secretKey = await utils.dataController.encryptObject({
  97       usuarioId,
  98     });
  99
 100
 101    const token = await utils.dataController.encryptObject(
 102      {
 103        usuarioId,
 104        date: moment().add(10, "minute"),
 105        secretKey
 106      }
 107    );
 108
 109    await update({
 110      queryParams: { _id: usuarioId },
 111      body: { token: token },
 112    });
 113
 114    return {
 115      usuarioId,
 116      token,
 117      refreshKey: secretKey,
 118    };
 119  } catch (error) {
 120    console.error(error);
 121    return undefined;
 122  }
 123
 124  const login = async ({ body }) => {
 125    try {
 126      const { email, senha } = JSON.parse(body) || {};
 127      if (!email || !senha) {
 128        return utils.httpHelper.notAuthorized();
 129      }
 130
 131      const response = await db.user.login(email, senha);
 132
 133      if (!response || !response.length) {
 134        return utils.httpHelper.notAuthorized();
 135      }
 136
 137      const usuarioId = response.data?._id;
 138      const authentication = await createAuth({
 139        usuarioId,
 140      });
 141
 142      if (!authentication) {
 143        return utils.httpHelper.notAuthorized();
 144      }
 145
 146    } catch (error) {
 147      console.error(error);
 148      return undefined;
 149    }
 150  }
 151
 152  return [
 153    ...response.data.map((user) => {
 154      const auth = await createAuth({ usuarioId: user._id });
 155
 156      return {
 157        ...user,
 158        auth
 159      };
 160    })
 161  ];
 162}
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
```

Grand Finale API

Executando

```

POST | http://localhost:3000/2015-03-31/functions/findEventPhoto/invocations
POST | http://localhost:3000/dev/evento/foto
POST |
http://localhost:3000/2015-03-31/functions/insertEventPhoto/invocations
PUT | http://localhost:3000/dev/evento/foto
POST |
http://localhost:3000/2015-03-31/functions/updateEventPhoto/invocations
DELETE | http://localhost:3000/dev/evento/foto
POST |
http://localhost:3000/2015-03-31/functions/removeEventPhoto/invocations
GET | http://localhost:3000/dev/categoria
POST |
http://localhost:3000/2015-03-31/functions/findCategory/invocations
PUT | http://localhost:3000/dev/categoria
POST |
http://localhost:3000/2015-03-31/functions/insertCategory/invocations
POST | http://localhost:3000/dev/categoria
POST |
http://localhost:3000/2015-03-31/functions/updateCategory/invocations
DELETE | http://localhost:3000/dev/categoria
POST |
http://localhost:3000/2015-03-31/functions/removeCategory/invocations

```

Server ready: http://localhost:3000 🎉

```

POST | http://localhost:3000/dev/usuario/acesso
POST |
http://localhost:3000/2015-03-31/functions/loginUser/invocations
PUT | http://localhost:3000/dev/usuario/acesso
POST |
http://localhost:3000/2015-03-31/functions/refreshUser/invocations
GET | http://localhost:3000/dev/usuario
POST | http://localhost:3000/2015-03-31/functions/findUser/invocations
POST | http://localhost:3000/dev/usuario
POST |
http://localhost:3000/2015-03-31/functions/insertUser/invocations
PUT | http://localhost:3000/dev/usuario
POST |
http://localhost:3000/2015-03-31/functions/updateUser/invocations
PUT | http://localhost:3000/dev/usuario/foto
POST |
http://localhost:3000/2015-03-31/functions/uploadPhotoUser/invocations
DELETE | http://localhost:3000/dev/usuario
POST |
http://localhost:3000/2015-03-31/functions/removeUser/invocations
GET | http://localhost:3000/dev/evento
POST |
http://localhost:3000/2015-03-31/functions/findEvent/invocations
GET | http://localhost:3000/dev/evento/contagem
POST |
http://localhost:3000/2015-03-31/functions/findCountEvent/invocations
POST | http://localhost:3000/dev/evento
POST |
http://localhost:3000/2015-03-31/functions/insertEvent/invocations
PUT | http://localhost:3000/dev/evento
POST |
http://localhost:3000/2015-03-31/functions/updateEvent/invocations
DELETE | http://localhost:3000/dev/evento
POST |

```

```

http://localhost:3000/2015-03-31/functions/findCountEvent/invocations
POST | http://localhost:3000/dev/evento
POST |
http://localhost:3000/2015-03-31/functions/insertEvent/invocations
PUT | http://localhost:3000/dev/evento
POST |
http://localhost:3000/2015-03-31/functions/updateEvent/invocations
DELETE | http://localhost:3000/dev/evento
POST |
http://localhost:3000/2015-03-31/functions/removeEvent/invocations
GET | http://localhost:3000/dev/evento/participante
POST | http://localhost:3000/2015-03-31/functions/findEventParticipant/invocations
POST | http://localhost:3000/dev/evento/participante
POST | http://localhost:3000/2015-03-31/functions/insertEventParticipant/invocations
PUT | http://localhost:3000/dev/evento/participante
POST | http://localhost:3000/2015-03-31/functions/updateEventParticipant/invocations
DELETE | http://localhost:3000/dev/evento/participante
POST | http://localhost:3000/2015-03-31/functions/removeEventParticipant/invocations
GET | http://localhost:3000/dev/evento/foto
POST |
http://localhost:3000/2015-03-31/functions/findEventPhoto/invocations
POST | http://localhost:3000/dev/evento/foto
POST |
http://localhost:3000/2015-03-31/functions/insertEventPhoto/invocations
PUT | http://localhost:3000/dev/evento/foto
POST |
http://localhost:3000/2015-03-31/functions/updateEventPhoto/invocations
DELETE | http://localhost:3000/dev/evento/foto
POST |
http://localhost:3000/2015-03-31/functions/removeEventPhoto/invocations
GET | http://localhost:3000/dev/categoria
POST |
http://localhost:3000/2015-03-31/functions/findCategory/invocations
PUT | http://localhost:3000/dev/categoria
POST |
http://localhost:3000/2015-03-31/functions/insertCategory/invocations
POST | http://localhost:3000/dev/categoria
POST |
http://localhost:3000/2015-03-31/functions/updateCategory/invocations
DELETE | http://localhost:3000/dev/categoria
POST |

```

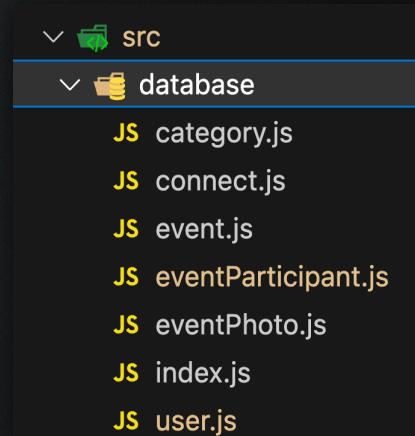
Grand Finale API

Conexão com banco de dados

```
JS connect.js M ×
api > src > database > JS connect.js > ...
You, há 1 segundo | 1 author (You)
1  const { MongoClient } = require("mongodb");
2
3  const MONGODB_URI = process.env.MONGODB_URI;
4  const connectDb = async () => {
5    try {
6      const config = {
7        useNewUrlParser: true,
8        useUnifiedTopology: true,
9        connectTimeoutMS: 2000,
10       serverSelectionTimeoutMS: 2000,
11     };
12
13     const client = new MongoClient(MONGODB_URI, config);
14     const eventName = "<event name>";
15     client.on(eventName, (event) => {
16       // console.log(`received ${eventName}: ${JSON.stringify(event, null, 2)}`);
17     });
18
19     if (!client) {
20       throw new Error("Banco de dados offline");
21     }
22
23     await client.connect();
24     return client;
25   } catch (error) {
26     throw new Error(error);
27   }
28 };
29
30 module.exports = { connectDb, MONGODB_URI };
31
```

NoSQL

Mongo DB



Grand Finale sobre API Mongo Modelos

```
api > src > database > JS event.js > [o] find > [o] params
You, anteontem | 1 author (You)
1 const utils = require("../utils");
2 const ObjectId = require("mongodb").ObjectId;
3 const user = require("./user");
4 const category = require("./category");
5
6 const collectionName = "evento";
7
8 const parseValues = (values) => {
9   return {
10     ...(values.usuarioId ? { usuarioId: new ObjectId(values.usuarioId) } : {}),
11     ...(values.categoriaId
12       ? { categoriaId: new ObjectId(values.categoriaId) }
13       : {}),
14     ...(values.titulo ? { titulo: values.titulo } : {}),
15     ...(values.local ? { local: values.local } : {}),
16     ...(values.data ? { data: new Date(values.data) } : {}),
17     ...(values.tempo ? { tempo: values.tempo } : {}),
18     ...(values.fotoBase64 ? { fotoBase64: values.fotoBase64 } : {}),
19   };
20 };
21
22 const queryMount = (params) => {
23   ... (params.id ? { _id: new ObjectId(params?.id) } : {}),
24   ... (params.categoriaId
25     ? { categoriaId: new ObjectId(params.categoriaId) }
26     : {}),
27   ... (params.usuarioId ? { usuarioId: new ObjectId(params.usuarioId) } : {}),
28   ... (params.titulo ? { titulo: params.titulo } : {}),
29   ... (params.local ? { local: params.local } : {}),
30 };
31 }
```

```
api > src > database > JS user.js > [o] update > [o] response > [o] utils.db.callBack() callback
You, há 2 segundos | 1 author (You)
1 const utils = require("../utils");
2 const ObjectId = require("mongodb").ObjectId;
3
4 const collectionName = "usuario";
5 const parseValues = (values) => {
6   return {
7     ... (values.nome ? { nome: values.nome } : {}),
8     ... (values.email ? { email: values.email } : {}),
9     ... (values.senha ? { senha: values.senha } : {}),
10    ... (values.data ? { data: values.data } : {}),
11    ... (values.token ? { token: values.token } : {}),
12    ... (values.fotoBase64 ? { fotoBase64: values.fotoBase64 } : {}),
13  };
14};
15
16 const queryMount = (params) => ({
17   ... (params.id ? { _id: new ObjectId(params?.id) } : {}),
18   ... (params.nome ? { nome: params?.nome } : {}),
19   ... (params.email ? { email: params?.email } : {}),
20   ... (params.token ? { token: params?.token } : {}),
21 });
22
23 const find = async (filter, limit) => {
24   try {
25     const params = filter || {};
26     const response = await utils.db.callBack(
27       collectionName,
28       async (collection) => {
29         return utils.db.find(collection, queryMount(params), { id: 1 }, limit);
30       }
31     );
32
33     return utils.returnDb.success(response);
34   } catch (error) {
35     return utils.returnDb.error(error.message || error.stack);
36   }
37};
```

Grand Finale AWS Lambda

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ...
zsh - api + ✓ ✎ ...
```

elioglima in api/api on main [!] took 3m 32.1s >
elioglima in api/api on main [!] > 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/cesso
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 > □

Lambda > Funções		Última busca há 1 hora	Ações	Criar função
Funções (14)				
<input type="text"/> Filtrar por tags e atributos ou pesquisar por palavra-chave				
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

Grand Finale AWS Lambda

The screenshot shows the AWS Lambda service interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, a search bar containing '[Option+S]', and account information for 'Norte da Virgínia'. Below the navigation is a breadcrumb trail: 'Lambda > Funções'. A search bar labeled 'Filtrar por tags e atributos ou pesquisar por palavra-chave' is present. On the right side of the header, there are buttons for 'Última busca há 1 hora', 'Ações', and 'Criar função'. The main area displays a table titled 'Funções (14)'. The table has columns: 'Nome da função', 'Descrição', 'Tipo de pacote', 'Tempo de execução', and 'Última modificação'. All functions listed are of type 'Zip' and written in 'Node.js 14.x', with the last modification time being 'há 2 horas'. The function names follow a pattern like 'greenlight-dev-[action]'. The table includes a header row with sorting icons and a footer row with page numbers (1, 2).

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

Grand Finale AWS Lambda

This screenshot shows the AWS Lambda function configuration page for 'greenlight-dev-findUser'. The top navigation bar includes 'AWS', 'Services', 'Search', and 'Options'. The main title is 'Lambda > Funções > greenlight-dev-findUser'. Below the title, it says 'greenlight-dev-findUser' and provides a note: 'This function belongs to an application. Click here to manage.' A 'Visão geral da função' section displays the ARN: arn:aws:lambda:us-east-1:1542061705149:function:greenlight-dev-findUser. It also shows the API Gateway integration and deployment stages.

Visão geral da função

- Funções relacionadas: Seleccionar uma função
- Descrição: Última modificação há 1 hora
- ARN da função: arn:aws:lambda:us-east-1:1542061705149:function:greenlight-dev-findUser
- Aplicativo: greenlight-dev
- URL da função: [Informações](#)

Origem do código

O pacote de implantação da função de 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

Tamanho do pacote: 52,6 MB Hash SHA256: WgnWw63XShu617mZ2fOu/t5/WKKh2EVdStXGdjwA= Última modificação: 30 de março de 2023 às 12:23 BRT

Configurações de tempo de execução

Tempo de execução: Node.js 14.x Manipulador: [informações](#) Arquitetura: [informações](#) x64

Camadas

Nenhum dado a ser exibido.

This screenshot shows the AWS API Gateway configuration page for the 'dev-greenlight' API. The top navigation bar includes 'AWS', 'Services', 'Search', and 'Options'. The main title is 'API Gateway > APIs > dev-greenlight'. The table lists one endpoint: 'dev-greenlight' with ARN arn:aws:execute-api:us-east-1:1542061705149:4pufly01/*'. The 'Configuração' tab is selected, showing environment variables and a detailed view of the 'Gatilhos' (Triggers) for the 'dev-greenlight' endpoint.

Nome	Descrição	ID	Protocolo	Tipo de endpoint	Criado em
dev-greenlight	4pufly01	REST	Edge	2023-03-30	

Configuração

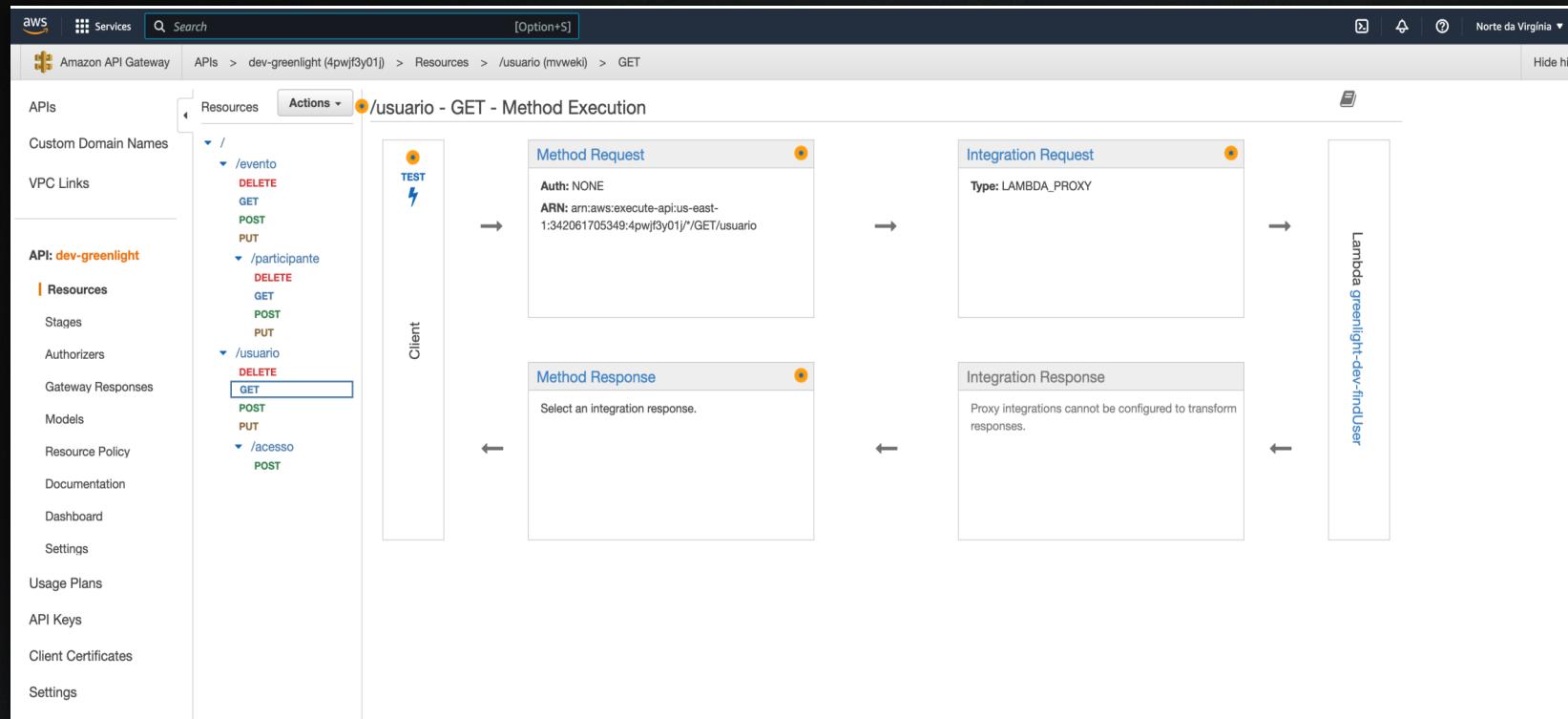
Variáveis de ambiente

Chave: MONGODB_URI Valor: mongodb+srv://admin:AB102030@greenlight.24jbfl0.mongodb.net/test

Gatilhos

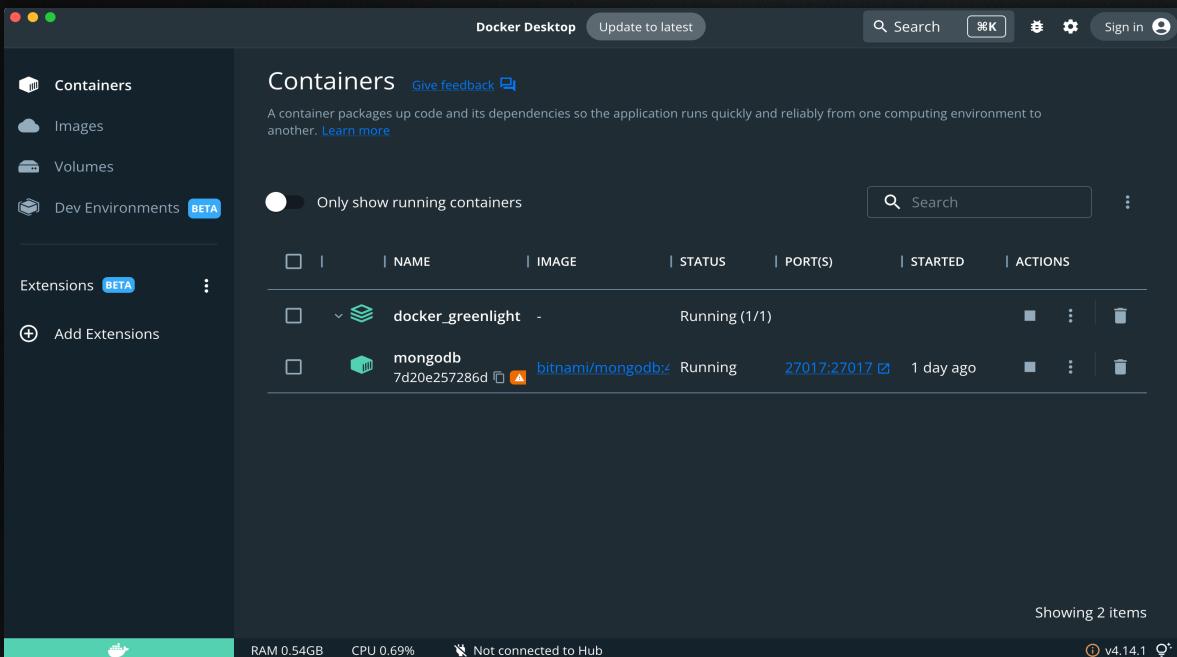
API Gateway: dev-greenlight
Arn: arn:aws:execute-api:us-east-1:1542061705149:4pufly01/*
Endpoint de API: <https://4pufly01.execute-api.us-east-1.amazonaws.com/dev/usuário>

Grand Finale AWS Lambda

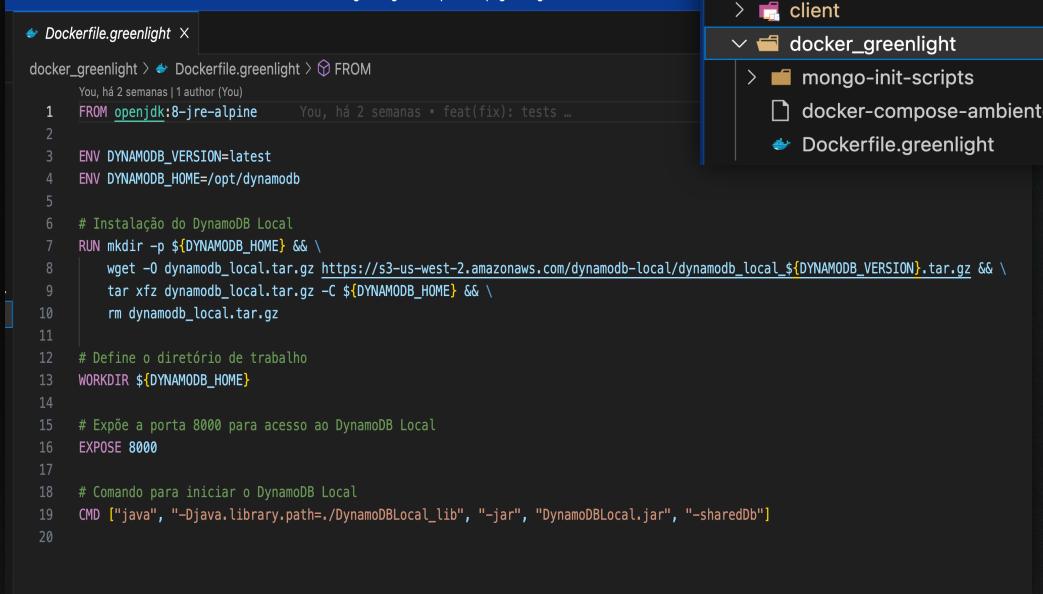


Grand Finale docker

Hoje gostaria de apresentar a vocês o Docker e como ele pode ser utilizado para simplificar o desenvolvimento, implantação e teste de aplicações. O Docker é uma plataforma de código aberto que permite empacotar e executar aplicativos em ambientes isolados chamados de contêineres.



Grand Finale docker



Dockerfile.greenlight

```
FROM openjdk:8-jre-alpine
ENV DYNAMODB_VERSION=latest
ENV DYNAMODB_HOME=/opt/dynamodb
# Instalação do DynamoDB Local
RUN mkdir -p ${DYNAMODB_HOME} && \
    wget -O dynamodb_local.tar.gz https://s3-us-west-2.amazonaws.com/dynamodb-local/dynamodb_local_${DYNAMODB_VERSION}.tar.gz && \
    tar xzf dynamodb_local.tar.gz -C ${DYNAMODB_HOME} && \
    rm dynamodb_local.tar.gz
# Define o diretório de trabalho
WORKDIR ${DYNAMODB_HOME}
# Expõe a porta 8000 para acesso ao DynamoDB Local
EXPOSE 8000
# Comando para iniciar o DynamoDB Local
CMD ["java", "-Djava.library.path=./DynamoDBLocal_lib", "-jar", "DynamoDBLocal.jar", "-sharedDb"]
```

FIAP-STARTUP-GREE... .github .vscode api client docker_greenlight mongo-init-scripts docker-compose-ambiente.l... Dockerfile.greenlight

mongo-init.js

```
const { ObjectId } = require("mongoose/lib/types");
db = db.getSiblingDB("greenlight");
db.createCollection("usuario");
db.createCollection("evento");
db.createCollection("evento_participante");
db.createCollection("categoria");

db.usuario.insert([
  {
    _id: new ObjectId("6466032822b9a867fd065db4"),
    nome: "Elio Gonçalves de Lima",
    email: "elio.designer@hotmail.com",
    senha: "Abc102030",
    data: "2023-05-18T10:51:20.809Z",
  },
  {
    _id: new ObjectId("6466032822b9a867fd065db5"),
    nome: "Andrei Vedovato",
    email: "andrei@vedovato@gmail.com",
    senha: "Abc102030",
    data: "2023-05-18T10:51:20.809Z",
  },
  {
    _id: new ObjectId("6466032822b9a867fd065db6"),
    nome: "Rafaela Rosso",
    email: "rafaelrosso@gmail.com",
    senha: "Abc102030",
    data: "2023-05-18T10:51:20.809Z",
  },
]);
db.categoria.insert([
  {
    _id: new ObjectId("6466032922b9a867fd065db7"),
    titulo: "Corrida",
    data: "2023-05-18T10:51:21.957Z",
  },
  {
    _id: new ObjectId("6466032922b9a867fd065db8"),
    titulo: "Yoga",
    data: "2023-05-18T10:51:21.957Z",
  },
  {
    _id: new ObjectId("6466032922b9a867fd065db9"),
    titulo: "Futebol",
    data: "2023-05-18T10:51:21.957Z",
  },
  {
    _id: new ObjectId("6466032922b9a867fd065dba"),
  }
]);
```

Grand Finale Client do Banco de Dados

The screenshot shows three instances of the NoSQLBooster for MongoDB interface, each displaying a different collection from a database named 'greenlight'.

- Left Window:** Shows the 'usuario' collection. It displays a query: `db.usuario.find({}).projection({}).sort({_id:-1}).limit(100)`. The results table shows one document with fields: _id, nome, email, senha, and data. The value for the first document is: {_id: 6466032822b9a867fd065db6, nome: 'Rafaela Rosso', email: 'rafarosso@gmail.com', senha: 'Ab@102030', data: '2023-05-18T10:51:20.809Z'}. The results table has 3 Docs and a duration of 0.053 s.
- Middle Window:** Shows the 'evento' collection. It displays a query: `db.evento.find({}).let(db: mongo.IdDatabase).sort({_id:-1}).limit(100)`. The results table shows two documents with fields: _id, usuarioId, categoriaId, titulo, local, data, tempo, and fotoBase64. The values for the first two documents are: {_id: 646d4d32e14eb19bdcf87a0, usuarioId: 6466032822b9a867fd065db4, categoriaId: 6466032922b9a867fd065db9, titulo: 'Jogar Juntos', local: 'Rua Alto Gárgas, 72', data: '31/12/1969 21:00:00 - 53 years ago', tempo: '08:17', fotoBase64: 'data:image/JPG;base64,9j/4AAQSkZJRgABAQAAQSABIAAD/4QB'} and {_id: 646d3f3c1dal761a308b85b7, usuarioId: 6466032922b9a867fd065db5, categoriaId: 6466032922b9a867fd065db9, titulo: 'Futebol', local: 'Estádio Olímpico', data: '31/12/1969 21:00:00 - 53 years ago', tempo: '08:17', fotoBase64: 'data:image/JPG;base64,9j/4AAQSkZJRgABAQAAQSABIAAD/4QB'}. The results table has 3 Docs and a duration of 0.068 s.
- Right Window:** Shows the 'evento_participante' collection. It displays a query: `db.evento_participante.find({})`. The results table shows one document with fields: _id, eventoId, and usuarioId. The value for the first document is: {_id: 646d3f3c1dal761a308b85b5, eventoId: 646d3f3c1dal761a308b85b7, usuarioId: 6466032922b9a867fd065db5}. The results table has 1 Doc and a duration of 0.068 s.

Grand Finale Testes no Postman

The image displays three separate windows of the Postman application, each showing a different API endpoint and its corresponding request and response.

Screenshot 1: POST /usuarios / Acesso ao Sistema

```

POST {{server}}/usuario/acesso
{
  "email": "elio.designer@hotmail.com",
  "rememberorigin": "true",
  "senha": "Ab@102030"
}
  
```

Screenshot 2: GET /usuarios / Consultar Todos Usuários

```

GET {{server}}/usuarios
  
```

Screenshot 3: GET /categorias / Consultar Todos

```

GET {{server}}/categorias
  
```

Grand Finale Testes no Postman

The image shows two side-by-side Postman application windows, both displaying successful API test results.

Left Window (API / Usuarios / Acesso ao Sistema):

- Method: POST
- URL: `((server))/usuario/acesso`
- Body (JSON):

```
1 ... "email": "elio.designer@hotmail.com",
2 ... "senha": "Ab@123456"
```
- Test Results: 200 OK, 197 ms, 335 B

Right Window (Eventos / Participantes / Participante Todos):

- Method: GET
- URL: `((server))/evento/participante`
- Body (JSON):

```
1 ...
2 ...
3 ...
4 ...
5 ...
6 ...
7 ...
8 ...
9 ...
10 ...
11 ...
12 ...
13 ...
14 ...
15 ...
16 ...
17 ...
18 ...
19 ...
20 ...
```
- Test Results: 200 OK, 177 ms, 599 B

Grand Finale WEB

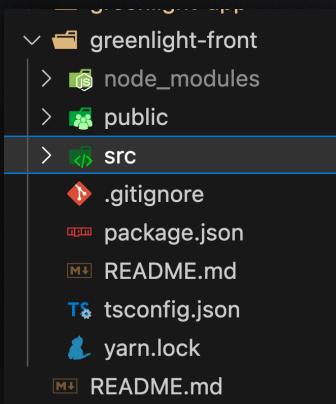
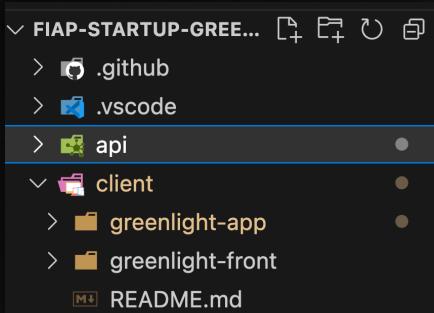
React JS / Node JS

O React.js é uma biblioteca JavaScript utilizada para criar interfaces interativas e responsivas em sites e aplicativos. Desenvolvida pelo Facebook, ela é amplamente adotada na indústria de desenvolvimento web.

A principal característica do React.js é o conceito de componentes. Os componentes são blocos de construção reutilizáveis que podem ser combinados para formar interfaces complexas. Cada componente é responsável por uma parte específica da interface e pode ser desenvolvido de forma independente dos outros componentes.

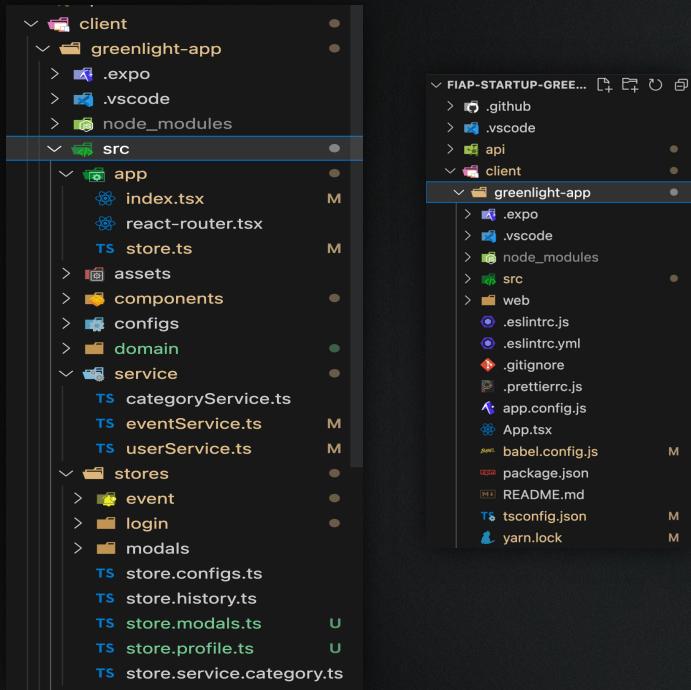
Para melhorar o desempenho das aplicações, o React.js utiliza o conceito de "Virtual DOM" (DOM virtual). O Virtual DOM é uma representação em memória do DOM (Document Object Model), que é a estrutura HTML do site. Ao realizar alterações na interface, o React compara o estado atual do Virtual DOM com o estado anterior e determina as mínimas mudanças necessárias para atualizar a interface real. Isso resulta em uma renderização eficiente e rápida das alterações, evitando a necessidade de atualizar a página inteira.

Grand Finale WEB



Grand Finale [github/worflow](#) – deploy web

Grand Finale Mobile



```

index.tsx M
client > greenlight-app > src > view > homeLogged > index.tsx > HomeView > useEffect() callback
You, há 12 segundos | 1 author (You)
1 import React, {useEffect, useState} from 'react';
2 import {TouchableOpacity} from 'react-native';
3
4 import {TAppState} from '@app/store';
5 import ControllerApp from '@components/controllerApp';
6 import HomeLoggedHeader from '@components/homeLoggedHeader';
7 import HorizontalMenu from '@components/horizontalMenu';
8 import {LoadApp} from '@components/loading';
9 import MapScreen from '@components/mapScreen';
10 import IconCalendarAddSVG from 'app/src/components/svg/iconCalendarSVG';
11 import IconCalendarSVG from '@components/svg/iconCalendarSVG';
12 import {TMenuItem} from '@domain/types/TMenuItem';
13 import {TStoreEventListCountState} from '@domain/types/TStates';
14 import {TStoreConfigState} from '@domain/types/states/TStatesConfigs';
15 import {TStoreLoginState} from '@domain/types/states/TStatesLogin';
16 import {ActionEventListCount} from '@stores/event/store.event.count';
17 import {ActionEventList} from '@stores/event/store.event.list';
18 import {ActionCategory} from '@stores/store.service.category';
19 import {useDispatch, useSelector} from 'react-redux';
20
21 import {ActionLoginRefresh} from '@stores/login/store.login';
22 import * as St from './styles';
23
24 const HomeView = () => {
25   const [openAddItem, setOpenAddItem] = useState<boolean>(false);
26   const [listDataCategory, setListDataCategory] = useState<TMenuItem[]>([]);
27
28   const dispatch = useDispatch();
29   const category = useSelector((state: TAppState) => state.category);
30   const stateLogin: TStoreLoginState = useSelector((state: TAppState) => state.login);
31   const stateConfigs: TStoreConfigState = useSelector((state: TAppState) => state.configs);
32   const eventListCount: TStoreEventListCountState = useSelector(
33     (state: TAppState) => state.eventListCount,
34   );
35
36   useEffect(() => {
37     dispatch(
38       ActionLoginRefresh({
39         routeCurrent: 'HomeLogged',
40         routeRedirect: '/HomeStart',
41         user: stateLogin.response?.user,
42       }),
43     );
44   }, []);
45
46   You, há 6 dias + configurado o persist ...
47
48   dispatch(ActionEventListCount({
49     userId: stateLogin.response?.user?._id || '-1',
50   }));

```

Grand Finale Mobile

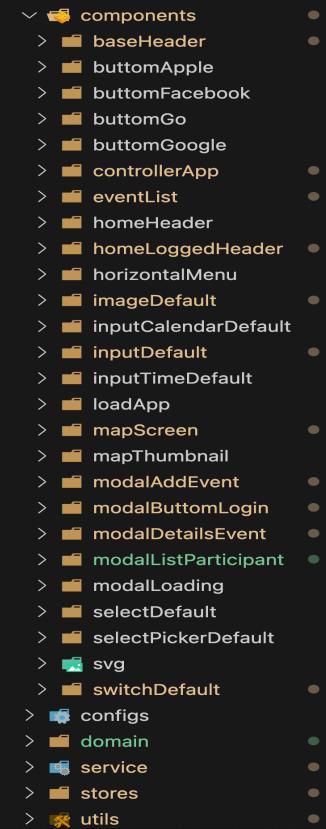
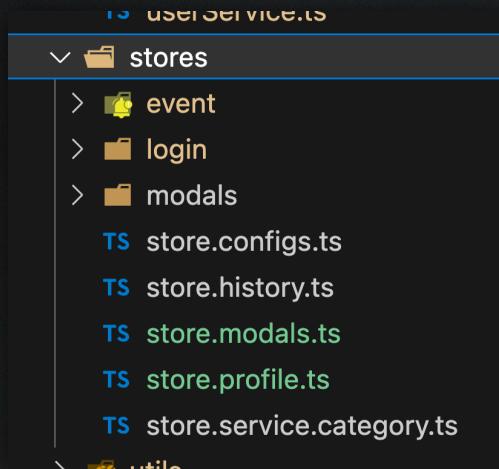
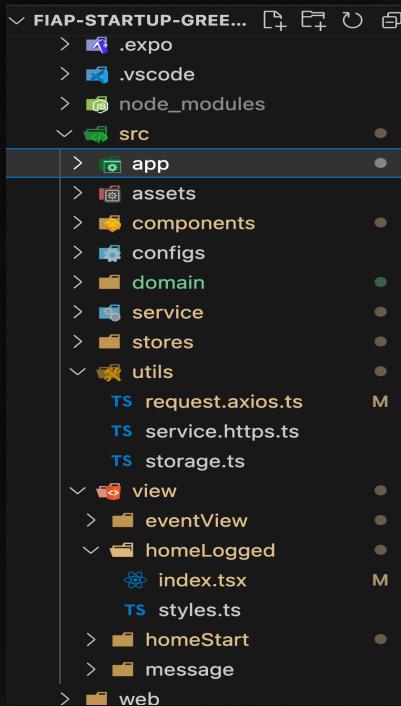
React Native / Node JS / Typescript

Em resumo, o React Native é um framework de desenvolvimento de aplicativos móveis que permite criar aplicativos nativos para iOS e Android usando JavaScript. Ele oferece reutilização de código, desenvolvimento em JavaScript, componentes reutilizáveis e um desempenho próximo ao de aplicativos nativos. O React Native é uma escolha popular para desenvolvedores que desejam criar aplicativos móveis eficientes, com uma base de código compartilhada e uma experiência de desenvolvimento mais produtiva.

TypeScript com React Native é uma combinação poderosa para o desenvolvimento de aplicativos móveis. O TypeScript é uma linguagem de programação que adiciona recursos de tipagem estática ao JavaScript, enquanto o React Native é um framework para construir aplicativos nativos para iOS e Android usando componentes reutilizáveis.

Node.js é um ambiente de execução de código JavaScript do lado do servidor, baseado no motor JavaScript V8 do Google Chrome. Ele permite que os desenvolvedores criem aplicativos escaláveis e de alta performance, usando JavaScript tanto no lado do cliente quanto no servidor.

Grand Finale Mobile



Grand Finale Mobile

styles.ts — fiap-startup-greenlight

TS styles.ts ●

client > greenlight-app > src > view > eventView > TS styles.ts > ...

You, há 6 dias | 1 author (You)

```
1 import styled from 'styled-components/native'; You, há 4
2
3 export const Container = styled.View`margin: 0 20px 0 20px;
4 flex-direction: column;
5 width: 90%;`;
6
7 ;
8
9 export const BaseTitle = styled.View`display: flex;
10 border-bottom-width: 1px;
11 border-bottom-style: solid;
12 border-bottom-color: #7000ad;
13 margin-bottom: 5px;
14 `;
15
16
17 export const Title = styled.Text`font-size: 25px;
18 font-weight: 600;
19 color: #1c70e5;
20 `;
21
22
23 export const EventList = styled.View`display: flex;
24 margin-top: 10px;
25 `;
26
```

styles.ts — fiap-startup-greenlight

TS styles.ts .../eventView ● TS styles.ts .../homeLogged X

client > greenlight-app > src > view > homeLogged > TS styles.ts > [e] Box

```
33
34 export const Title = styled.Text`display: flex;
35 color: #000;
36 font-size: 16px;
37 font-weight: 500;
38 `;
39
40
41 export const MakerBase = styled.View`display: flex;
42 background-color: #7000ad;
43 margin-left: 5px;
44 border-radius: 25px;
45 `;
46
47
48 export const Maker = styled.Text`display: flex;
49 margin: 4px;
50 color: #fff;
51 font-weight: 900;
52 font-size: 12px;
53 `;
54
55
56 export const ButtonRow = styled.View`display: flex;
57 flex-direction: row;
58 justify-content: space-between;
59 align-items: center;
60 margin: 10px 0 30px 0;
61 `;
62
63
64 export const ButtonBase = styled.View`display: flex;
65 background-color: #7000ad;
66 flex-direction: row;
67 border-radius: 10px;
68 justify-content: flex-start;
69 align-items: center;
70 `;
71
72
73 export const ButtonLogo = styled.View`display: flex;
74 width: 70px;
75 height: 65px;
76 flex-direction: row;
77 justify-content: center;
78 align-items: center;
79 `;
80
```

Grand Finale Mobile

The Figma interface displays the following screens:

- Splash:** Shows a hand on a keyboard with the text "Estavamos à sua espera!" and "Bem-vindo ao nosso aplicativo de eventos! Aqui você encontrará informações sobre esportivos, maratonas, corridas e muito mais". Includes a "VAMOS COMEÇAR" button.
- Login social:** A modal for selecting a login method: "Continuar com Facebook", "Continuar com Google", and "Continuar com Apple".
- Novo:** A screen for creating a new event with a map and a "CRIAR EVENTO" button.
- Evento:** A screen showing details for an event like "Corrida Noturna" with a date of "26/03/23" and a time of "18h30". Includes a "SAIR DO EVENTO" button.
- Login social:** A modal for selecting a login method: "Continuar com Facebook", "Continuar com Google", and "Continuar com Apple".
- Inicio:** A main screen showing a map with event locations, a "Mais Eventos" button, and categories like "Corrida" and "Yoga".
- Lista:** A screen listing events such as "Corrida Noturna", "Yoga no Parque", "3a Corrida dos Amigos", and "Academia as Ar-bove".

The mobile application screens are as follows:

- Splash Screen:** Shows a hand on a keyboard with the text "Estavamos à sua espera!" and "Bem-vindo ao nosso aplicativo de eventos! Aqui você encontrará informações sobre esportivos, maratonas, corridas e muito mais". Includes a "VAMOS COMEÇAR" button.
- Login Screen:** A modal for selecting a login method: "Continuar com Facebook", "Continuar com Google", and "Continuar com Apple". It also includes fields for email ("elio.designer@hotmail.com") and password ("*****"), a "Manter conectado" toggle switch, and an "Entrar" button.
- Eventos Screen:** A list of events:
 - Corrida Por do Sol** (24/05/23, 19:15)
 - Yoga Alongando mais** (31/12/69, 08:12)

Equipe

- Elio Gonçalves de Lima
RM92270

Andrei Vedovato
RM92217

Rafaela Rosso Pacheco
RM92241



Obrigado!

Estamos prontos para maiores esclarecimentos

FIAP

