

TypeDefs

index

```
1 //Se importan los typeDef de cada submodulo
2 const userTypeDefs = require("./user_type_defs");
3 const coverTypeDef = require("./cover_type_def");
4 const proyectosTypeDefs = require("./proyecto_typeDef");
5
6 //Se agregan los typeDefs importados para exportarlos
7 const schemasArrays = [userTypeDefs, coverTypeDef, proyectosTypeDefs];
8
9 //Se exportan
10 module.exports = schemasArrays;
11
```

cover_type_def

```
1 const { gql } = require("apollo-server"); Jaider Andres Pinto Pinto, 38 minutes ago • merge con pr
2
3 const coverTypeDef = gql`
4   type Cover {
5     idCover: Int!
6     idUsuario: Int!
7     trabajo: String
8     contenido: String
9   }
10
11   input CoverInput {
12     idCover: Int!
13     idUsuario: Int!
14     trabajo: String
15     contenido: String
16   }
17
18   type Query {
19     coverByIdCover(idCover: Int!): Cover
20   }
21
22   extend type Mutation {
23     createCover(coverletter: CoverInput!): Cover
24   }
25 `;
26
27 module.exports = coverTypeDef;
28
```

proyecto_type_def

```
1  const { gql } = require("apollo-server");           Jaider
2
3  const proyectosTypeDefs = gql`
4    type Tecnologia {
5      nombre: String!
6      version: String
7    }
8    type Duracion {
9      inicio: String!
10     fin: String
11   }
12   type Enlaces {
13     Enlace: String!
14   }
15   input TecnologiaInput {
16     nombre: String!
17     version: String
18   }
19   input DuracionInput {
20     inicio: String!
21     fin: String
22   }
23   input EnlacesInput {
24     Enlace: String!
25   }
```

```
26 type Proyecto {
27   _id: String!
28   nombreProyecto: String!
29   tecnologiasUsadas: [Tecnologia!]!
30   tematica: String!
31   duracion: Duracion!
32   descripcionProyecto: String!
33   enlaces: [Enlaces!]!
34   usuario: String!
35 }
36 input inputProyecto {
37   nombreProyecto: String!
38   tecnologiasUsadas: [TecnologiaInput!]!
39   tematica: String!
40   duracion: DuracionInput!
41   descripcionProyecto: String!
42   enlaces: [EnlacesInput!]!
43   usuario: String!
44 }
45 type Query {
46   getProyectoById(idProyecto: String!): Proyecto
47 }
48 type Mutation {
49   createProyecto(proyecto: inputProyecto!): Proyecto
50   putProyectoById(idProyecto: String!, proyecto: inputProyecto!): Proyecto
51   deleteProyectoById(idProyecto: String!): Proyecto
52 }
53 ~;
54
55 module.exports = proyectosTypeDefs;
56
```

```
1  const { gql } = require("apollo-server");           Jaider Andres Pinto Pinto,
2
3  const proyectosTypeDefs = gql`
4    type Tecnologia {
5      nombre: String!
6      version: String
7    }
8    type Duracion {
9      inicio: String!
10     fin: String
11   }
12   type Enlaces {
13     Enlace: String!
14   }
15   input TecnologiaInput {
16     nombre: String!
17     version: String
18   }
19   input DuracionInput {
20     inicio: String!
21     fin: String
22   }
23   input EnlacesInput {
24     Enlace: String!
25   }
```

user_type_def

```
1  const { gql } = require("apollo-server");
2
3  const userTypeDefs = gql`
4    type UserDetails {
5      idUsuario: Int!
6      tipoDocIdentidad: String!
7      numeroDocumento: String!
8      nombre: String!
9      correo: String!
10     telefono: String!
11     contrasena: String
12     idTipoEntidad: Int!
13   }
14
15   type Access {
16     exitoso: String!
17     mensajeError: String
18     body: String!
19   }
20
21   input CredentialsInput {
22     correo: String!
23     contrasena: String!
24   }
25
26   type RegistUser {
27     exitoso: String!
28     mensajeError: String
29     body: String
30   }
31`
```

Jaider Andres

```
31
32   input UserInput {
33     idUsuario: Int
34     tipoDocIdentidad: String!
35     numeroDocumento: String!
36     nombre: String!
37     correo: String!
38     telefono: String!
39     contrasena: String!
40     idTipoEntidad: Int!
41   }
42
43   type Perfil {
44     usuario: Int!
45     descripcion: String!
46     metodologia: String
47     formacion: [Formacion!]
48     trabajo: Trabajo
49     intereses: String
50   }
51
52   input PerfilInput {
53     usuario: Int!
54     descripcion: String!
55     metodologia: String
56     formacion: [FormacionInput!]
57     trabajo: TrabajoInput
58     intereses: String
59   }
60
```

```
61 type Formacion {
62     institucion: String!
63     anio: String!
64     materia: String!
65 }
66 input FormacionInput {
67     institucion: String!
68     anio: String!
69     materia: String!
70 }
71
72 type Trabajo {
73     empresa: String!
74     cargo: String!
75     funciones: [String!]!
76 }
77 input TrabajoInput {
78     empresa: String!
79     cargo: String!
80     funciones: [String!]!
81 }
82
83 input UserUpdateInput {
84     user: UserInput!
85     profile: PerfilInput!
86     id: Int!
87 }
88
89 type UserProfile {
90     user: UserDetail!
91     profile: Perfil!
92 }
93
```

```

94     type Mutation {
95       |   registerUser(user: UserInput!): RegisterUser!
96       |   login(credentials: CredentialsInput!): Access!
97       |   updateUser(user: UserUpdateInput!): UserProfile!
98     }
99
100     type Query {
101       |   userDetails(userId: Int!): UserProfile!
102     }
103   ~;
104
105   module.exports = userTypeDefs;
106

```

Resolver

Cover_Resolver

```

1  v const coverResolver = {
2  v   Query: {
3  v     |   coverByIdCover: async (_, { idCover }, { dataSources }) => {
4     |     |   return await dataSources.portafoliosAPI.coverByIdCover(idCover);
5     |   },
6   },
7  v   Mutation: {
8  v     |   createCover: async (_, { coverletter }, { dataSources }) => {
9     |     |   return await dataSources.portafoliosAPI.createCover(coverletter);
10    |   },
11   },
12 };
13
14 module.exports = coverResolver;
15

```


Index

```
1 //Se importan los resolvers de cada submodulo
2 const coverResolver = require("../cover_resolver");
3 const lodash = require("lodash");
4 const userResolver = require("../user_resolver");
5 const proyectoResolver = require("../proyecto_resolver");
6
7 //Se agregan los resolver importados para exportarlos
8 const resolvers = lodash.merge(userResolver, coverResolver, proyectoResolver);
9
10 module.exports = resolvers;
11
```

Proyecto_Resolver

```
1 const proyectoResolver = {
2   Query: {
3     getProjectById: async (_, { idProyecto }, { dataSources }) => {
4       return await dataSources.portafoliosAPI.getProjectById(idProyecto);
5     },
6   },
7   Mutation: {
8     createProject: async (_, { proyecto }, { dataSources }) => {
9       return await dataSources.portafoliosAPI.createProject(proyecto);
10     },
11     putProjectById: async (_, { idProyecto, proyecto }, { dataSources }) => {
12       return await dataSources.portafoliosAPI.putProjectById(
13         idProyecto,
14         proyecto
15       );
16     },
17     deleteProjectById: async (_, { idProyecto }, { dataSources }) => {
18       return await dataSources.portafoliosAPI.deleteProjectById(idProyecto);
19     },
20   },
21 };
22
23 module.exports = proyectoResolver;
24
```

User_Resolver

1? master 4a-apigateway / src / resolvers / user_resolver.js / <> Jump to -

Go to file ...

Jaider0111 mas ajustes Latest commit 6acde63 5 minutes ago History

R: 2 contributors

53 lines (50 sloc) 1.6 KB Raw Blame

```
1 const userResolver = {
2   Query: {
3     userDetails: async (_, { userId }, { dataSources, userIdToken }) => {
4       if (userId == userIdToken) {
5         let ans = {};
6         ans.user = await dataSources.userAPI.userDetail(userId);
7         ans.profile = await dataSources.portafoliosAPI.profileDetail(userId);
8         return ans;
9       } else return null;
10    },
11  },
12  Mutation: {
13    registUser: async (_, { user }, { dataSources }) => {
14      let userAns = await dataSources.userAPI.registUser(user);
15      await dataSources.portafoliosAPI.createProfile({
16        usuario: userAns.body,
17        formacion: [],
18        trabajo: [],
19      });
20      return userAns;
21    },
22  },
23  login: async (_, { credentials }, { dataSources }) => {
24    return await dataSources.userAPI.login(credentials);
25  },
26  },
27  updateUser: async (_, { user }, { dataSources }) => {
28    let ans = {};
29    ans.user = JSON.parse(
30      JSON.stringify(await dataSources.userAPI.updateUser(user.user))
31    );
32    ans.user = ans.user.body;
33    ans.profile = JSON.parse(
34      JSON.stringify(
35        await dataSources.portafoliosAPI.updateProfile(user.profile, user.id)
36      )
37    );
38    ans.profile.formacion = JSON.parse(ans.profile.formacion);
39    ans.profile.trabajo = JSON.parse(ans.profile.trabajo);
40    for (let tr in ans.profile.trabajo) {
41      let st = JSON.stringify(ans.profile.trabajo[tr].funciones).replace(
42        /\n/g,
43        ""
44      );
45      st = st.substring(1, st.length - 1);
46      ans.profile.trabajo[tr].funciones = JSON.parse(st);
47    }
48    return ans;
49  },
50  },
51  };
52
53 module.exports = userResolver;
```

DataSources

portafolios_api

```

1  const { RESTDataSource } = require("apollo-datasource-rest");
2  const serverConfig = require("../server");
3
4  Jaider Andres Pinto Pinto, 2 hours ago | 3 authors (Jaider Andres Pinto Pinto and others)
5  class PortafoliosAPI extends RESTDataSource {
6      constructor() {
7          super();
8          this.baseURL = serverConfig.portafolios_api_url;
9      }
10     //cover;
11     async createCover(cover) {
12         cover = new Object(JSON.parse(JSON.stringify(cover)));
13         return await this.post(`/cover/create/`, cover);
14     }
15     async coverByIdCover(idCover) {
16         return await this.get(`/cover/detail/${idCover}`);
17     }
18     async createProfile(profile) {
19         profile = new Object(JSON.parse(JSON.stringify(profile)));
20         console.log(profile);
21         return await this.post(`/perfil`, profile);
22     }
23     async updateProfile(profile, userId) {
24         profile = new Object(JSON.parse(JSON.stringify(profile)));
25         return await this.put(`/perfil/${userId}`, profile);
26     }
27
28     async profileDetail(userId) {
29         return await this.get(`/perfil/${userId}`);
30     }
31

```

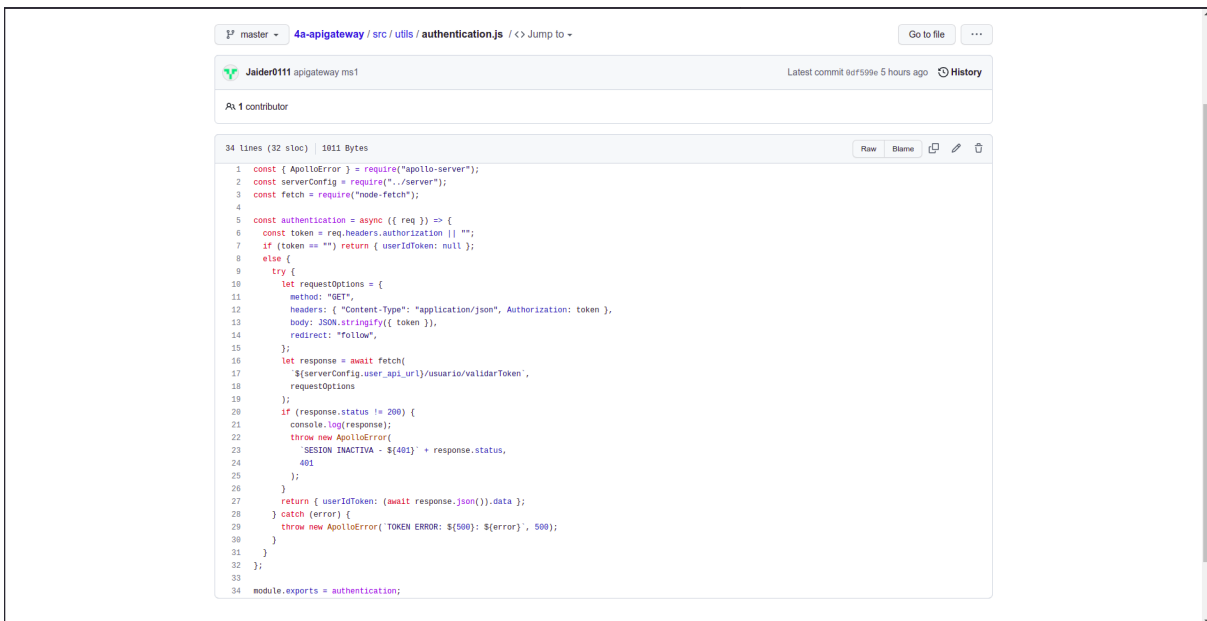
```
32 //proyecto's dataSources
33
34 async getAllProjects() {
35   return await this.get("/proyectos");
36 }
37 async createProject(proyecto) {
38   return await this.post("/proyectos", proyecto);
39 }
40 async getProjectById(idProyecto) {
41   return await this.get(`/ms2App/proyectos/${idProyecto}`);
42 }
43 async putProjectById(idProyecto, proyecto) {
44   return await this.put(`/ms2App/proyectos/${idProyecto}`, proyecto);
45 }
46 async deleteProjectById(idProyecto) {
47   return await this.delete(`/ms2App/proyectos/${idProyecto}`);
48 }
49 }
50
51 module.exports = PortafoliosAPI;
52
```

user_api

```

1 const { RESTDataSource } = require("apollo-datasource-rest");
2 const serverConfig = require("../server");
3
4 Jaider Andres Pinto Pinto, 3 hours ago | 1 author (Jaider Andres Pinto Pinto)
5
6 class UserAPI extends RESTDataSource {
7   constructor() {
8     super();
9     this.baseURL = serverConfig.user_api_url;
10   }
11
12   async registUser(user) {
13     user = new Object(JSON.parse(JSON.stringify(user)));
14     return await this.post(`/usuario/registro`, user);
15   }
16
17   async login(credentials) {
18     credentials = new Object(JSON.parse(JSON.stringify(credentials)));
19     return await this.post(`/login`, credentials);
20   }
21
22   async updateUser(user) {
23     user = new Object(JSON.parse(JSON.stringify(user)));
24     return await this.put(`/usuario/actualizar`, user);
25   }
26
27   async userDetails(userId) {
28     return await this.get(`/usuario/detalle/${userId}`);
29   }
30 }
31
32 module.exports = UserAPI;

```



Pruebas

URL: <https://tu-portafolio-apigateway.herokuapp.com/>

Login

The screenshot shows the GraphQL Playground interface. The query editor contains the following query:

```
1 mutation Mutation($credentials: CredentialsInput!) {  
2   login(credentials: $credentials) {  
3     exitoso  
4     mensajeError  
5     body  
6   }  
7 }  
8
```

The variables section shows the following JSON:

```
1 {  
2   "credentials": {  
3     "correo": "pruebapersona@yopmail.com",  
4     "contrasena": "Test1234*"  
5   }  
6 }
```

The response section shows the following JSON:

```
{  
  "data": {  
    "login": {  
      "exitoso": true,  
      "mensajeError": null,  
      "body": "eyJhbGciOiJIUzUxMiJ9.  
eyJ3YXQiOiJlZmZg3NjMwMjksImZcyI6Imh0dHBzOi8vd3d3LmF1dG8VudG1hLmVybS8iLCJz  
dWIiOiJwcnVlYmFwZjZib25hQHlvc6IhaWwY29tIiwiaXhwIjojM4NzYzOTI5fQ.  
GY8yXlRnyA8QEWXFrNhtfnQLC3uqcEakLx1bP10pyq74YKTYnuA0LcuaRBS1whFJWdp_6-D  
focQCHF7uMd9PQ"  
    }  
  }  
}
```

The status bar indicates a 200 status code, 1.03s execution time, and 314B response size.

Registro

(Contiene orquestación, Mutation a ambos microservicios)

The screenshot shows the GraphQL Playground interface. The query editor contains the following query:

```
1 mutation RegisterUser($user: UserInput!) {  
2   registerUser(user: $user) {  
3     exitoso  
4     mensajeError  
5     body  
6   }  
7 }
```

The variables section shows the following JSON:

```
1 {  
2   "user": {  
3     "contrasena": "Testtest1*",  
4     "correo": "test143@yopmail.com",  
5     "idTipoEntidad": 1,  
6     "nombre": "Prueba Gateway",  
7     "numeroDocumento": "101244545",  
8     "telefono": "312451712",  
9     "tipoDocIdentidad": "CC",  
10    "idUserario": null  
11  }  
12 }
```

The response section shows the following JSON:

```
{  
  "data": {  
    "registerUser": {  
      "exitoso": true,  
      "mensajeError": null,  
      "body": 23  
    }  
  }  
}
```

The status bar indicates a 200 status code, 961ms execution time, and 73B response size.

Actualizar perfil

(Contiene orquestación, Mutation a ambos microservicios)

SANDBOX https://tu-portafolio-apigateway

Mutation x RegistUser x Mutation x + Mutation

```
1 mutation Mutation($user: UserUpdateInput!) {
2   updateUser(user: $user) {
3     profile {
4       usuario
5       descripcion
6       intereses
    }
  }
}
```

Variables Headers Environment Variables

```
1 {
2   "user": {
3     "id": 22,
4     "user": {
5       "contrasena": "Testtest1*",
6       "correo": "test14@yopmail.com",
7       "idTipoEntidad": 1,
8       "nombre": "Prueba Gateway",
9       "numeroDocumento": "10124545",
10      "telefono": "312451712",
11      "tipoDocIdentidad": "CC",
12      "idUser": 22
13    },
14    "profile": {
15      "intereses": "Me gusta programar",
16      "descripcion": "Soy un gran programador",
17      "formacion": [
18        {
19          "anio": "2021",
20          "institucion": "UNAL",
21          "materia": "Programacion"
        }
      ]
    }
  }
}
```

Response

```
{
  "data": {
    "updateUser": {
      "profile": {
        "usuario": 22,
        "descripcion": "Soy un gran programador",
        "intereses": "Me gusta programar",
        "metodologia": "Codigo limpio",
        "trabajo": [
          {
            "cargo": "Desarrollador",
            "empresa": "Facebook",
            "funciones": [
              "Programar",
              "Desarrollar"
            ]
          }
        ],
        "formacion": [
          {
            "institucion": "UNAL",
            "anio": "2021",
            "materia": "Programacion"
          }
        ]
      },
      "user": {
        "contrasena": null,
        "correo": "test14@yopmail.com",
        "idTipoEntidad": 1,
        "idUser": 22
      }
    }
  }
}
```

STATUS 200 | 635ms | 521B

Ver Perfil

(Contiene orquestación, query a ambos microservicios)

Mutation x RegistUser x Mutation x Query x + Query

```
1 query Query($userId: Int!) {
2   userDetail(userId: $userId) {
3     user {
4       idUsuario
5       tipoDocIdentidad
6       numeroDocumento
7       nombre
8       correo
9       telefono
10      contrasena
11      idTipoEntidad
12    }
13    profile {
14      usuario
15      descripcion
16      metodologia
17      formacion {
18        institucion
19        anio
20        materia
21      }
22    }
23  }
24}
```

Variables Headers Environment Variables

```
1 {
2   "userId": 22
3 }
```

Response

```
{
  "data": {
    "userDetail": {
      "user": {
        "idUsuario": 22,
        "tipoDocIdentidad": "CC",
        "numeroDocumento": "10124545",
        "nombre": "Prueba Gateway",
        "correo": "test14@yopmail.com",
        "telefono": "312451712",
        "contrasena": "$2a$10$UBpBNC0GZ5LmCuSHU0vrS.
xmSpUEKwyLE37a8P53VDWjwm5r8po0",
        "idTipoEntidad": 1
      },
      "profile": {
        "usuario": 22,
        "descripcion": "Soy un gran programador",
        "metodologia": "Codigo limpio",
        "formacion": [
          {
            "institucion": "UNAL",
            "anio": "2021",
            "materia": "Programacion"
          }
        ]
      },
      "trabajo": [
        {
          "empresa": "Facebook",
          "cargo": "Desarrollador",
          "funciones": [
            "Programar"
          ]
        }
      ]
    }
  }
}
```

STATUS 200 | 249ms | 579B