TypeDefs

index

```
//Se importan los typeDef de cada submodulo
const userTypeDefs = require("./user_type_defs");
const coverTypeDef = require("./cover_type_def");
const proyectosTypeDefs = require("./proyecto_typeDef");

//Se agregan los typeDefs importados para exportarlos
const schemasArrays = [userTypeDefs, coverTypeDef, proyectosTypeDefs];

//Se exportan
module.exports = schemasArrays;

```

cover_type_def

```
const { gql } = require("apollo-server");

const coverTypeDef = gql`
  type Cover {
    idCover: Int!
    idUsuario: Int!
    trabajo: String
    contenido: String
  }

  input CoverInput {
    idCover: Int!
    idUsuario: Int!
    trabajo: String
    contenido: String
  }

  type Query {
    coverByIdCover(idCover: Int!): Cover
  }

  extend type Mutation {
    createCover(coverletter: CoverInput!): Cover
  }
`;

module.exports = coverTypeDef;

```

proyecto_type_def

```graphql
const { gql } = require("apollo-server");

const proyectosTypeDefs = gql`
  type Tecnologia {
    nombre: String!
    version: String
  }
  type Duracion {
    inicio: String!
    fin: String
  }
  type Enlaces {
    Enlace: String!
  }
  input TecnologiaInput {
    nombre: String!
    version: String
  }
  input DuracionInput {
    inicio: String!
    fin: String
  }
  input EnlacesInput {
    Enlace: String!
  }
```

```
  type Proyecto {
    _id: String!
    nombreProyecto: String!
    tecnologiasUsadas: [Tecnologia!]!
    tematica: String!
    duracion: Duracion!
    descripcionProyecto: String!
    enlaces: [Enlaces!]!
    usuario: String!
  }
  input inputProyecto {
    nombreProyecto: String!
    tecnologiasUsadas: [TecnologiaInput!]!
    tematica: String!
    duracion: DuracionInput!
    descripcionProyecto: String!
    enlaces: [EnlacesInput!]!
    usuario: String!
  }
  type Query {
    getProyectById(idProyecto: String!): Proyecto
  }
  type Mutation {
    createProyect(proyecto: inputProyecto!): Proyecto
    putProyectById(idProyecto: String!, proyecto: inputProyecto!): Proyecto
    deleteProyectById(idProyecto: String!): Proyecto
  }
`;

module.exports = proyectosTypeDefs;

```

```
const { gql } = require("apollo-server");        Jaider Andres Pinto Pinto,

const proyectosTypeDefs = gql`
  type Tecnologia {
    nombre: String!
    version: String
  }
  type Duracion {
    inicio: String!
    fin: String
  }
  type Enlaces {
    Enlace: String!
  }
  input TecnologiaInput {
    nombre: String!
    version: String
  }
  input DuracionInput {
    inicio: String!
    fin: String
  }
  input EnlacesInput {
    Enlace: String!
  }
```

user_type_def

```graphql
const { gql } = require("apollo-server");

const userTypeDefs = gql`
  type UserDetail {
    idUsuario: Int!
    tipoDocIdentidad: String!
    numeroDocumento: String!
    nombre: String!
    correo: String!
    telefono: String!
    contrasena: String
    idTipoEntidad: Int!
  }

  type Access {
    exitoso: String!
    mensajeError: String
    body: String!
  }

  input CredentialsInput {
    correo: String!
    contrasena: String!
  }

  type RegistUser {
    exitoso: String!
    mensajeError: String
    body: String
  }

```

```
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
```

```
type Formacion {
  institucion: String!
  anio: String!
  materia: String!
}
input FormacionInput {
  institucion: String!
  anio: String!
  materia: String!
}

type Trabajo {
  empresa: String!
  cargo: String!
  funciones: [String!]!
}
input TrabajoInput {
  empresa: String!
  cargo: String!
  funciones: [String!]!
}

input UserUpdateInput {
  user: UserInput!
  profile: PerfilInput!
  id: Int!
}

type UserProfile {
  user: UserDetail!
  profile: Perfil!
}

```

```
    type Mutation {
      registUser(user: UserInput!): RegistUser!
      login(credentials: CredentialsInput!): Access!
      updateUser(user: UserUpdateInput!): UserProfile!
    }

    type Query {
      userDetail(userId: Int!): UserProfile!
    }
  `;

  module.exports = userTypeDefs;

```

Resolver

Cover_Resolver

```
const coverResolver = {
  Query: {
    coverByIdCover: async (_, { idCover }, { dataSources }) => {
      return await dataSources.portafoliosAPI.coverByIdCover(idCover);
    },
  },
  Mutation: {
    createCover: async (_, { coverletter }, { dataSources }) => {
      return await dataSources.portafoliosAPI.createCover(coverletter);
    },
  },
};

module.exports = coverResolver;

```

Index

```
//Se importan los resolvers de cada submodulo
const coverResolver = require("./cover_resolver");
const lodash = require("lodash");
const userResolver = require("./user_resolver");
const proyectoResolver = require("./proyecto_resolver");

//Se agregan los resolver importados para exportarlos
const resolvers = lodash.merge(userResolver, coverResolver, proyectoResolver);

module.exports = resolvers;

```

Proyecto_Resolver

```
const proyectoResolver = {
  Query: {
    getProyectById: async (_, { idProyecto }, { dataSources }) => {
      return await dataSources.portafoliosAPI.getProyectsById(idProyecto);
    },
  },
  Mutation: {
    createProyect: async (_, { proyecto }, { dataSources }) => {
      return await dataSources.portafoliosAPI.createProyect(proyecto);
    },
    putProyectById: async (_, { idProyecto, proyecto }, { dataSources }) => {
      return await dataSources.portafoliosAPI.putProyectById(
        idProyecto,
        proyecto
      );
    },
    deleteProyectById: async (_, { idProyecto }, { dataSources }) => {
      return await dataSources.portafoliosAPI.deleteProyectById(idProyecto);
    },
  },
};

module.exports = proyectoResolver;

```

User_Resolver

master · 4a-apigateway / src / resolvers / user_resolver.js · Jump to · Go to file

Jaider0111 mas ajustes — Latest commit 6acde63 5 minutes ago · History

2 contributors

53 lines (50 sloc) · 1.6 KB · Raw · Blame

```
const userResolver = {
  Query: {
    userDetail: async (_, { userId }, { dataSources, userIdToken }) => {
      if (userId == userIdToken) {
        let ans = {};
        ans.user = await dataSources.userAPI.userDetail(userId);
        ans.profile = await dataSources.portafoliosAPI.profileDetail(userId);
        return ans;
      } else return null;
    },
  },
  Mutation: {
    registUser: async (_, { user }, { dataSources }) => {
      let userAns = await dataSources.userAPI.registUser(user);
      await dataSources.portafoliosAPI.createProfile({
        usuario: userAns.body,
        formacion: [],
        trabajo: [],
      });
      return userAns;
    },

    login: async (_, { credentials }, { dataSources }) => {
      return await dataSources.userAPI.login(credentials);
    },

    updateUser: async (_, { user }, { dataSources }) => {
      let ans = {};
      ans.user = JSON.parse(
        JSON.stringify(await dataSources.userAPI.updateUser(user.user))
      );
      ans.user = ans.user.body;
      ans.profile = JSON.parse(
        JSON.stringify(
          await dataSources.portafoliosAPI.updateProfile(user.profile, user.id)
        )
      );
      ans.profile = JSON.parse(
        JSON.stringify(
          await dataSources.portafoliosAPI.updateProfile(user.profile, user.id)
        )
      );
      ans.profile.formacion = JSON.parse(ans.profile.formacion);
      ans.profile.trabajo = JSON.parse(ans.profile.trabajo);
      for (let tr in ans.profile.trabajo) {
        let st = JSON.stringify(ans.profile.trabajo[tr].funciones).replace(
          /'/g,
          '"'
        );
        st = st.substring(1, st.length - 1);
        ans.profile.trabajo[tr].funciones = JSON.parse(st);
      }
      return ans;
    },
  },
};

module.exports = userResolver;
```

DataSources

portafolios_api

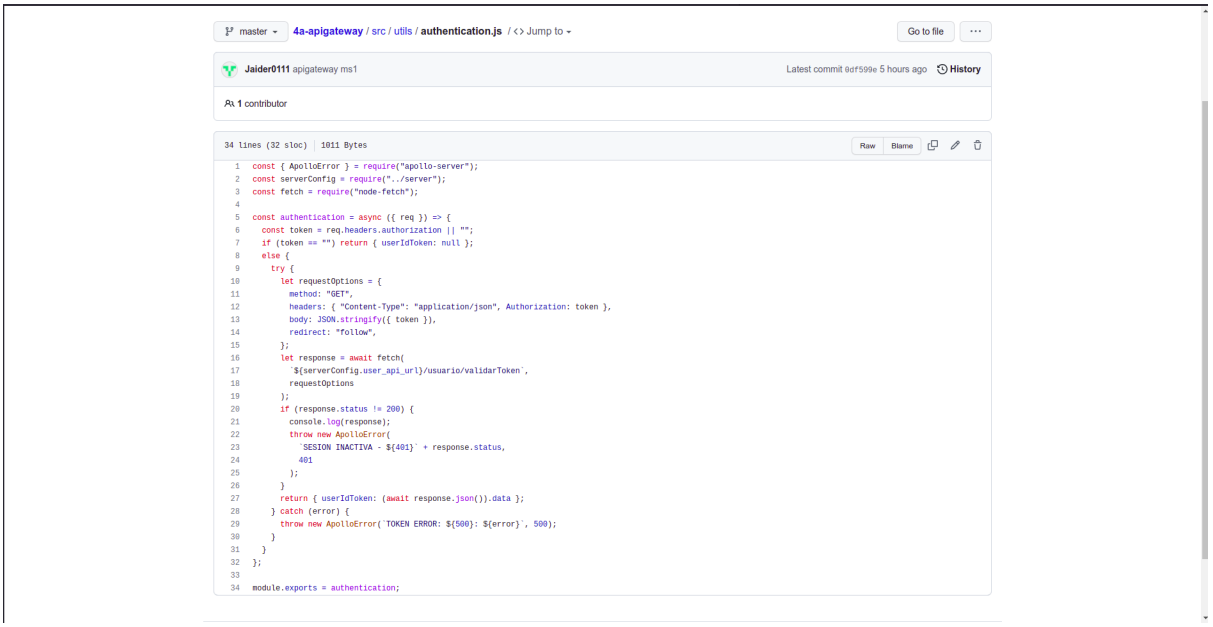
```
const { RESTDataSource } = require("apollo-datasource-rest");
const serverConfig = require("../server");

Jaider Andres Pinto Pinto, 2 hours ago | 3 authors (Jaider Andres Pinto Pinto and others)
class PortafoliosAPI extends RESTDataSource {
  constructor() {
    super();
    this.baseURL = serverConfig.portafolios_api_url;
  }
      Jorge681jorge, a day ago • cover
  //cover;
  async createCover(cover) {
    cover = new Object(JSON.parse(JSON.stringify(cover)));
    return await this.post(`/cover/create/`, cover);
  }
  async coverByIdCover(idCover) {
    return await this.get(`/cover/detail/${idCover}`);
  }
  async createProfile(profile) {
    profile = new Object(JSON.parse(JSON.stringify(profile)));
    console.log(profile);
    return await this.post(`/perfil`, profile);
  }
  async updateProfile(profile, userId) {
    profile = new Object(JSON.parse(JSON.stringify(profile)));
    return await this.put(`/perfil/${userId}`, profile);
  }

  async profileDetail(userId) {
    return await this.get(`/perfil/${userId}`);
  }
```

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

user_api

```
const { RESTDataSource } = require("apollo-datasource-rest");
const serverConfig = require("../server");

Jaider Andres Pinto Pinto, 3 hours ago | 1 author (Jaider Andres Pinto Pinto)
class UserAPI extends RESTDataSource {
  constructor() {
    super();
    this.baseURL = serverConfig.user_api_url;
  }

  async registUser(user) {
    user = new Object(JSON.parse(JSON.stringify(user)));
    return await this.post(`/usuario/registro`, user);
  }

  async login(credentials) {
    credentials = new Object(JSON.parse(JSON.stringify(credentials)));
    return await this.post(`/login`, credentials);      Jaider Andres Pinto Pint
  }

  async updateUser(user) {
    user = new Object(JSON.parse(JSON.stringify(user)));
    return await this.put(`/usuario/actualizar`, user);
  }

  async userDetail(userId) {
    return await this.get(`/usuario/detalle/${userId}`);
  }
}

module.exports = UserAPI;
```

Context



master · 4a-apigateway / src / utils / authentication.js · Jump to · Go to file

Jaider0111 apigateway ms1 — Latest commit 0df599e 5 hours ago · History

1 contributor

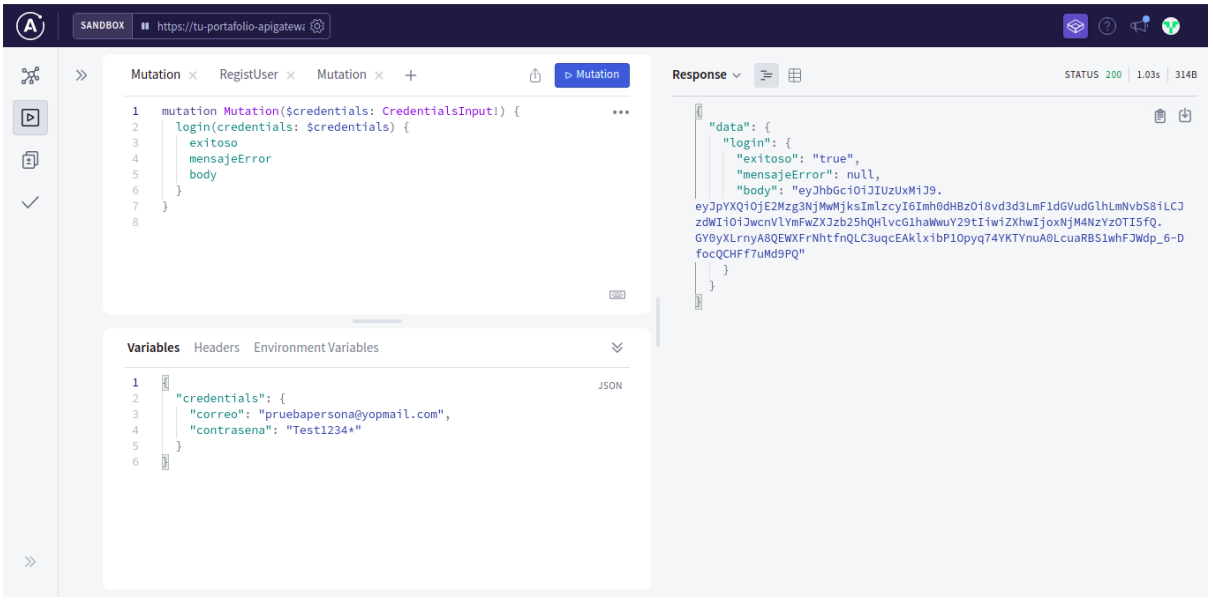
34 lines (32 sloc) | 1011 Bytes — Raw · Blame

```
const { ApolloError } = require("apollo-server");
const serverConfig = require("../server");
const fetch = require("node-fetch");

const authentication = async ({ req }) => {
  const token = req.headers.authorization || "";
  if (token == "") return { userIdToken: null };
  else {
    try {
      let requestOptions = {
        method: "GET",
        headers: { "Content-Type": "application/json", Authorization: token },
        body: JSON.stringify({ token }),
        redirect: "follow",
      };
      let response = await fetch(
        `${serverConfig.user_api_url}/usuario/validarToken`,
        requestOptions
      );
      if (response.status != 200) {
        console.log(response);
        throw new ApolloError(
          `SESION INACTIVA - ${401}` + response.status,
          401
        );
      }
      return { userIdToken: (await response.json()).data };
    } catch (error) {
      throw new ApolloError(`TOKEN ERROR: ${500}: ${error}`, 500);
    }
  }
};

module.exports = authentication;
```

Pruebas

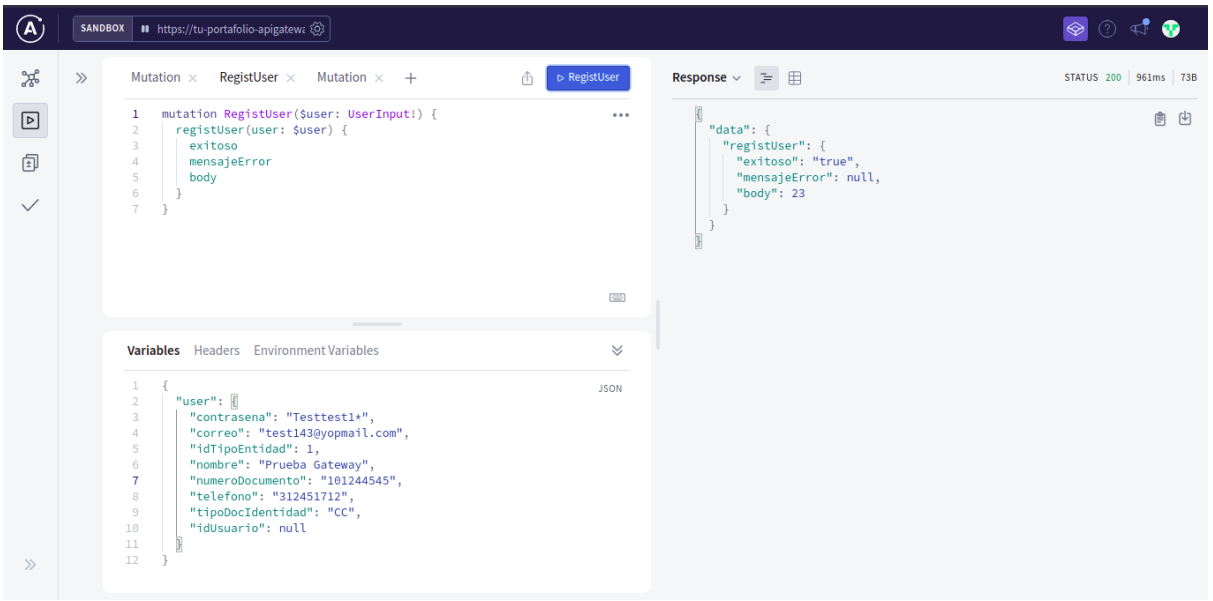
URL: https://tu-portafolio-apigateway.herokuapp.com/

Login



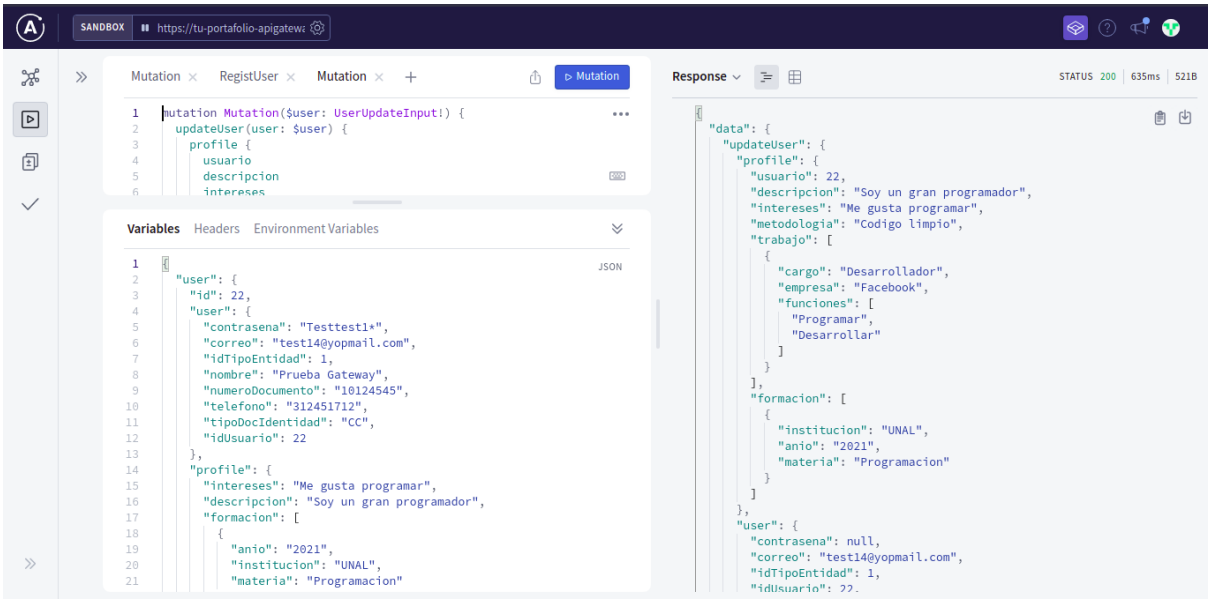
Registro

(Contiene orquestación, Mutation a ambos microservicios)



Actualizar perfil

(Contiene orquestación, Mutation a ambos microservicios)



Ver Perfil

(Contiene orquestación, query a ambos microservicios)