Documentação

1 - Funcionalidade

A aplicação tem como funcionalidade principal o gerenciamento de uma fila.

2 - Ferramentas e Linguagens

Para desenvolvimento da aplicação foi utilizado o Express (https://expressjs.com/pt-br/) , JavaScript e TypeScript, para validação foi utilizado o celebrate (https://www.npmjs.com/package/celebrate),para controle de versão o GitHub, para desenvolvimento do código Visual Studio Code e para teste de requisições HTTP o Insomnia.

3 - Parâmetros de Entrada

A aplicação utiliza arrays para fazer o armazenamento em memória, pois não utilizo nenhum tipo de banco de dados (SQL ou NoSQL) na aplicação.

Foram Utilizados os métodos HTTP :

`GET`

O método `GET` solicita a representação de um recurso específico. Requisições utilizando o método `GET` devem retornar apenas dados.

`HEAD`

O método `HEAD` solicita uma resposta de forma idêntica ao método `GET`, porém sem conter o corpo da resposta.

`POST`

O método `POST` é utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor.

`PUT`

O método `PUT` substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.

`DELETE`

O método `DELETE` remove um recurso específico.

3.1 - Rotas

As Rotas foram criadas com base nos Endpoints que foram solicitados e além disso foram criados rotas extras onde pode ser feita a busca do Usuário pelo e-mail sem contar posição na fila e deletar usuário cadastrado:

Rota para busca do usuário pelo E-mail

```
app.put('/search/:email', (request, response) => {
  const { email } = request.params;

  const userEmail = users.findIndex(user => user.email === email);

  if (userEmail < 0) {
    return response.status(400).json({ error: 'Email Not Found' })
  }


  return response.json(users[userEmail]);
});
```

Rota para deletar usuário cadastrado

```
app.delete('/delete/:email', (request, response) => {
  const { email } = request.params;

  const userEmail = users.findIndex(user => user.email === email);

  users.splice(userEmail, 1);

  return response.status(204).send();
});
```

A rota `/createUser` - recebe nome, e-mail e gênero do usuário e retornar id, nome e e-mail e gênero, além disso a rota conta com um validador que verifica se a tipagem de dados inserida é correta.

```
// I created the user constant to receive the insomnia
app.post('/createUser', celebrate({
  [Segments.BODY]: Joi.object().keys({
    name: Joi.string().required(),
    email: Joi.string().email().trim().required(),
    gender: Joi.string().required(),
  }),      You, 7 minutes ago • Uncommitted changes
}), (request, response) => {

  const { name, email, gender } = request.body;

  const user = { id: uuid(), name, email, gender }

  users.push(user);
  return response.json(users);
});
```

A rota `/addToLine` - recebe o id do usuário a ser adicionado à fila e deve retornar a posição em que ele está na fila.

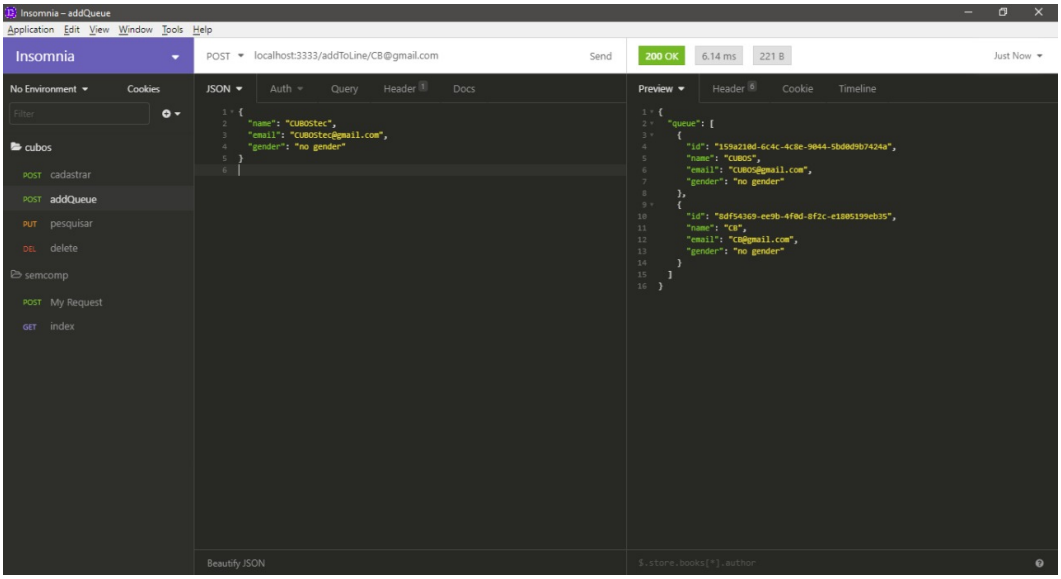
```
app.post('/addToLine/:email', (request, response) => {
  const { email } = request.params;

  const userEmail = users.findIndex(user => user.email === email);

  if (userEmail < 0) {
    return response.status(400).json({ error: 'Email Not Found' })
  }


  const UserAdd = queue.push(users[userEmail]);
  return response.json({ queue });

});
```



A Rota `/findPosition` - recebe o e-mail de um usuário e retornar a posição dele na fila.

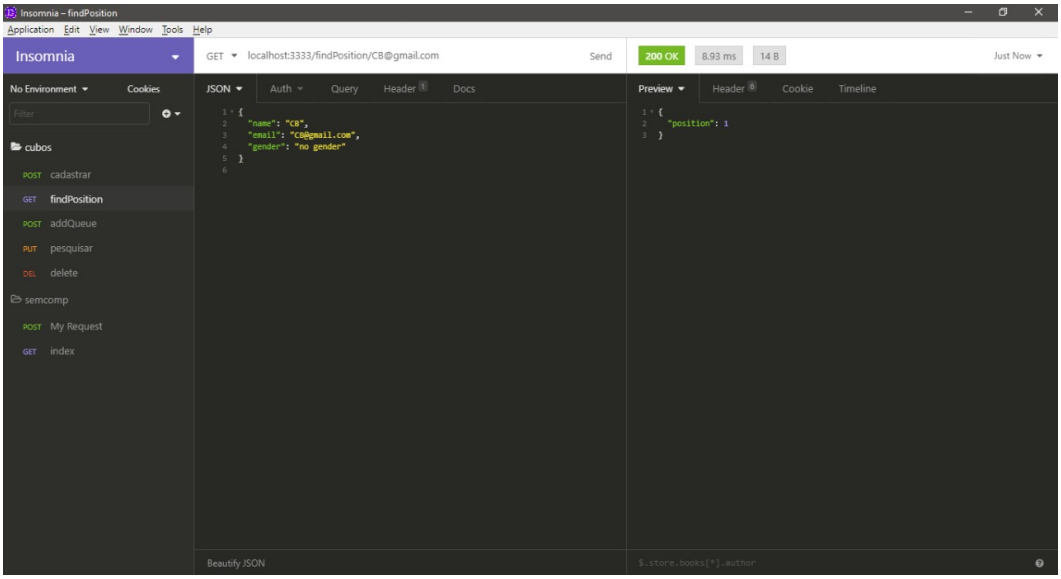
```
app.get('/findPosition/:email', (request, response) => {
  const { email } = request.params;

  const userEmail = users.findIndex(user => user.email === email);

  if (userEmail < 0) {
    return response.status(400).json({ error: 'Email Not Found' })
  }

  return response.json({ position: userEmail + 1 });

});
```

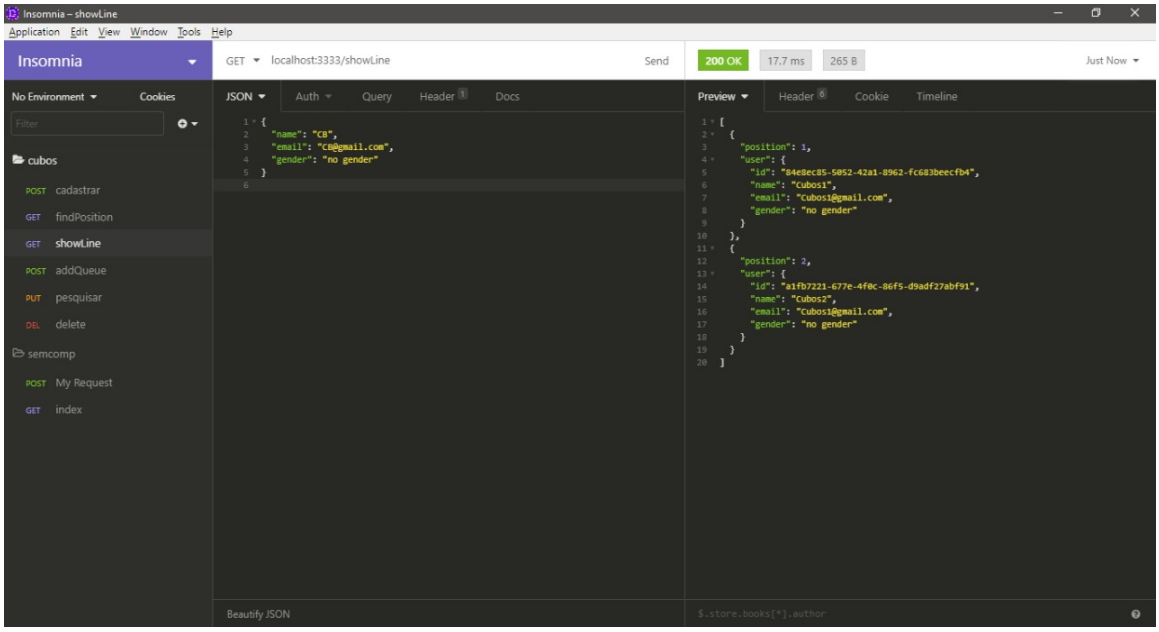


A Rota `/showLine -` retorna uma lista de usuários (nome, gênero e email), bem como a posição de cada um deles na fila (ordenando de primeira posição para última).

```
app.get('/showLine', (request, response) => {
  const { email } = request.params;

  const userEmail = users.map((user, position) => ({
    position: position + 1,
    user: user
  }));

  return response.json(userEmail);
});
```

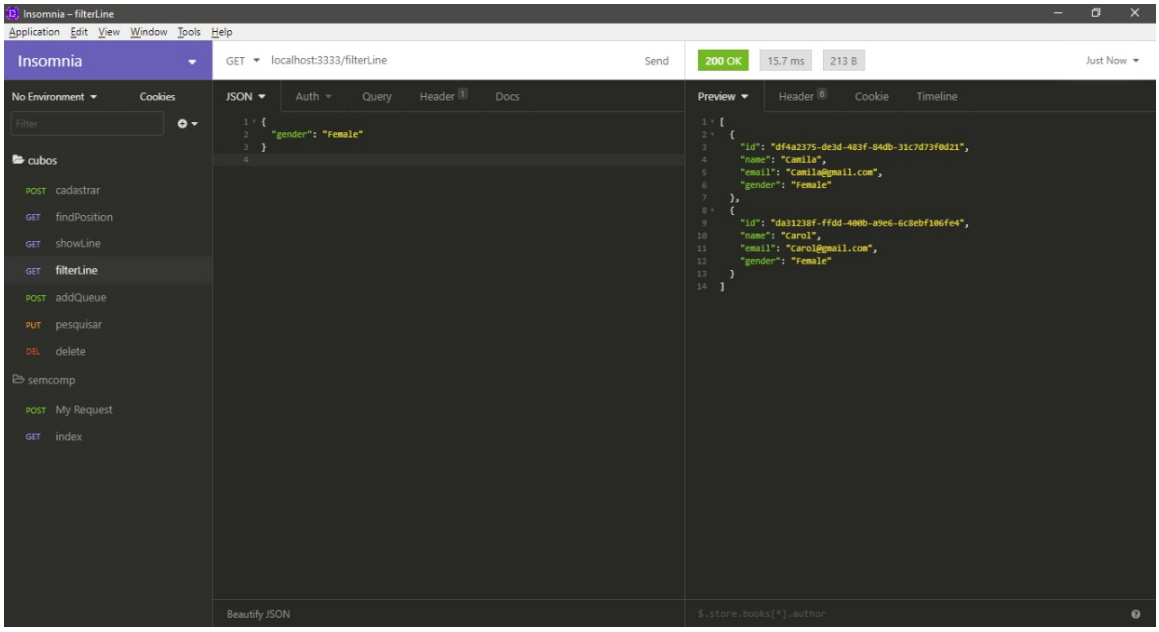


A Rota `/filterLine` - recebe um gênero de usuário e retornar uma lista de usuários com aquele gênero (nome, gênero e e-mail), bem como a posição de cada um deles na fila (ordenando de primeira posição para última).

```
app.get('/filterLine', (request, response) => {
  const { gender } = request.body;

  const filterGender = users.filter(user => user.gender === gender);

  return response.json(filterGender);
});
```

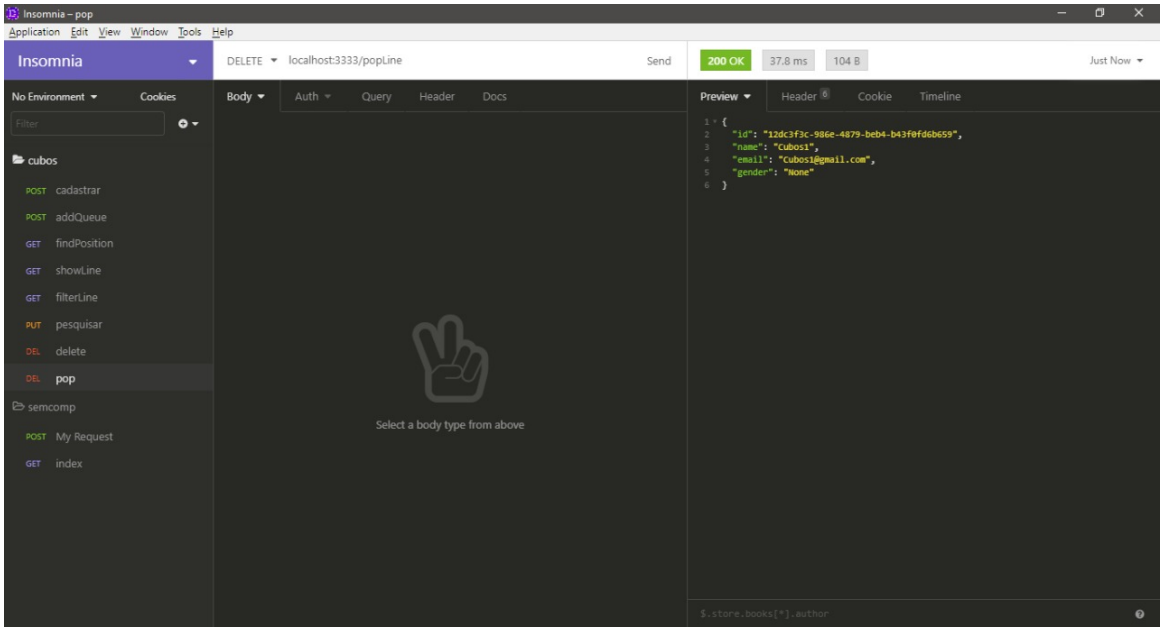


A Rota `/popLine -` retira da fila a primeira posição e retorna.

```
app.delete('/popLine', (request, response) => {
  console.log(queue);

  const deleteUser = queue.shift();
      You, 4 hours ago • deletar primeiro da fila
  console.log(deleteUser);
  return response.json(deleteUser);


});
```


Insomnia – pop
Application Edit View Window Tools Help
Insomnia
DELETE localhost:3333/popLine
Send
200 OK
37.8 ms
104 B
Just Now
No Environment
Cookies
Filter
cubos
POST cadastrar
POST addQueue
GET findPosition
GET showLine
GET filterLine
PUT pesquisar
DEL delete
DEL pop
semcomp
POST My Request
GET index
Body
Auth
Query
Header
Docs
Select a body type from above
Preview
Header
Cookie
Timeline
{
"id": "12dc3f3c-986e-4879-beb4-b43f0fd6b659",
"name": "Cubos1",
"email": "Cubos1@gmail.com",
"gender": "None"
}
$.store.books[*].author