

REST API - Think Toilet

Índice

- [REST API - Think Toilet](#)
 - [Índice](#)
 - [Introdução](#)
 - [Autenticação - Endpoints](#)
 - [Login](#)
 - [Registo](#)
 - [Usuários - Endpoints](#)
 - [Mostrar usuários](#)
 - [Mostrar usuários por ID](#)
 - [Editar e-mail do usuário](#)
 - [Editar ícone do usuário](#)
 - [Editar nome do usuário](#)
 - [Editar senha do usuário](#)
 - [Casas de Banho - Endpoints](#)
 - [Mostrar casas de banho](#)
 - [Mostrar casas de banho por bounding box](#)
 - [Mostrar casas de banho por proximidade](#)
 - [Mostrar casa de banho por ID](#)
 - [Mostrar casas de banho pelo ID do usuário](#)
 - [Pesquisar casas de banho](#)
 - [Visualizar imagem da casa de banho](#)
 - [Adicionar denúncia na casa de banho](#)
 - [Adicionar foto na casa de banho](#)
 - [Apagar denúncia na casa de banho](#)
 - [Comentários - Endpoints](#)
 - [Mostrar comentários de uma casa de banho](#)
 - [Mostrar comentários de um usuário](#)
 - [Mostrar reações de um comentário](#)
 - [Adicionar comentário](#)
 - [Adicionar reação em um comentário](#)
 - [Apagar reação em um comentário](#)
 - [Observações](#)
 - [State](#)
 - [Extras](#)
 - [Access](#)

Introdução

A API do Think Toilet oferece acesso eficiente a informações sobre usuários, casas de banho e comentários, permitindo consultas, visualizações e interações organizadas. Desenvolvida para integração com a aplicação móvel, a API fornece dados atualizados e é compatível com outras plataformas.

Os dados são entregues em formato JSON, garantindo respostas consistentes e facilitando a integração com sistemas diversos. A estrutura dos dados e os endpoints são flexíveis, projetados para suportar expansões futuras e melhorias contínuas na aplicação.

Autenticação - Endpoints

Login

- **URL:**
`/login`
- **METHOD:**
`POST`
- **DATA PARAMETERS:**

```
{
  "email": [string],
  "password": [string]
}
```

- **SUCCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": [integer],
  "iconId": [string],
  "numComments": [integer],
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 401,
  "message": "User with email [string] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 401,
  "message": "Invalid password.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("api/login")
suspend fun login(
    @Body login: LoginRequest
): Call<ResponseBody>
```

Registo

- **URL:**

/register

- **METHOD:**

POST

- **DATA PARAMETHERS:**

```
{
  "name": [string],
  "email": [string],
  "password": [string],
  "iconId": [string],
  "birthDate": [date]
}
```

- **SUCCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": [integer],
  "iconId": [string],
  "numComments": [integer],
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 409,
  "message": "Email already in use.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("api/register")
suspend fun register(
    @Body register: RegisterRequest
): Call<ResponseBody>
```

Usuários - Endpoints

Mostrar usuários

- **URL:**
`/users`
- **METHOD:**
`GET`
- **URL PARAMETHODS:**
 - Optional (Query):
`ids=[integers] (default: null)`
- **SUCCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "name": [string],
    "email": [string],
    "points": [integer],
    "iconId": [string],
    "numComments": [integer],
  },
]
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("api/users")
suspend fun getUsers(
    @Query("ids") ids: List<Int>
): Call<ResponseBody>
```

Mostrar usuários por ID

- **URL:**

/users/{id}

- **METHOD:**

GET

- **URL PARAMETHERS:**

- Required (Path):

id=[integer]

- **SUCCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": [integer],
  "iconId": [string],
  "numComments": [integer],
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("api/users/{id}")
suspend fun getUser(
    @Path("id") id: Int
): Call<ResponseBody>
```

Editar e-mail do usuário

- **URL:**

/users/{id}/edit/email

- **METHOD:**

POST

- **URL PARAMETHODS:**

- Required (Path):
id=[integer]
- Required (Query):
email=[string],
password=[string]

- **SUCCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": [integer],
  "iconId": [string],
  "numComments": [integer],
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 401,
  "message": "Invalid password.",
  "timestamp": [datetime]
}
```

```
{
  "status": 409,
  "message": "Email already in use.",
  "timestamp": [datetime]
}
```

```
{
  "status": 500,
  "message": "Could not save User.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("api/users/{id}/edit/email")
suspend fun editEmail(
    @Path("id") id: Int,
    @Query("email") email: String,
    @Query("password") password: String
): Call<ResponseBody>
```

Editar ícone do usuário

- **URL:**
/users/{id}/edit/icon
- **METHOD:**
POST
- **URL PARAMETHERS:**

- Required (Path):
`id=[integer]`
- Required (Query):
`iconId=[string]`

- **SUCCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": [integer],
  "iconId": [string],
  "numComments": [integer],
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 500,
  "message": "Could not save User.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("api/users/{id}/edit/icon")
suspend fun editIcon(
    @Path("id") id: Int,
    @Query("iconId") iconId: String
): Call<ResponseBody>
```


Editar nome do usuário

- **URL:**

`/users/{id}/edit/name`

- **METHOD:**

`POST`

- **URL PARAMETERS:**

- Required (Path):

`id=[integer]`

- Required (Query):

`name=[string],`

`password=[string]`

- **SUCCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": [integer],
  "iconId": [string],
  "numComments": [integer],
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 401,
  "message": "Invalid password.",
  "timestamp": [datetime]
}
```

```
{
  "status": 409,
  "message": "Name already in use.",
  "timestamp": [datetime]
}
```

```
{
  "status": 500,
  "message": "Could not save User.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("api/users/{id}/edit/name")
suspend fun editName(
    @Path("id") id: Int,
    @Query("name") name: String,
    @Query("password") password: String
): Call<ResponseBody>
```

Editar senha do usuário

- **URL:**

/users/{id}/edit/password

- **METHOD:**

POST

- **URL PARAMETHERS:**

- Required (Path):
id=[integer]
- Required (Query):
password=[string],
newPassword=[string]

- **SUCCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": [integer],
  "iconId": [string],
  "numComments": [integer],
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 401,
  "message": "Invalid password.",
  "timestamp": [datetime]
}
```

```
{
  "status": 500,
  "message": "Could not save User.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("api/users/{id}/edit/password")
suspend fun editPassword(
    @Path("id") id: Int,
    @Query("password") password: String,
    @Query("newPassword") newPassword: String
): Call<ResponseBody>
```

Casas de Banho - Endpoints

Mostrar casas de banho

- **URL:**

/toilets

- **METHOD:**

GET

- **URL PARAMETERS:**

- Optional (Query):
state=[string], (default: null)
userId=[integer], (default: null)
ids=[integers], (default: null)
pageable=[boolean], (default: false)
page=[integer], (default: 0)
size=[integer] (default: 20)

- **SUCCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "name": [string],
    "address": [string],
    "rating": {
      "avgClean": [double],
      "avgStructure": [double],
      "avgAccessibility": [double],
      "ratioPaper": [double],
    },
    "extras": [strings],
    "access": [string],
    "latitude": [double],
    "longitude": [double],
    "numComments": [integer],
    "placeId": [string]
  },
]
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "State with technical name [string] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("api/toilets")
suspend fun getToilets(
    @Query("state") state: String? = null,
    @Query("userId") userId: Int? = null,
    @Query("ids") ids: List<Int>? = null,
    @Query("pageable") pageable: Boolean = false,
    @Query("page") page: Int = 0,
    @Query("size") size: Int = 20
): Call<ResponseBody>
```

Mostrar casas de banho por bounding box

- **URL:**

/toilets/bounding

- **METHOD:**

GET

- **URL PARAMETHERS:**

- Required (Query):

minLat=[double],
maxLat=[double],
minLon=[double],
maxLon=[double]

- **SUCCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "name": [string],
    "address": [string],
    "rating": {
      "avgClean": [double],
      "avgStructure": [double],
      "avgAccessibility": [double],
      "ratioPaper": [double],
    },
    "extras": [strings],
    "access": [string],
    "latitude": [double],
```

```

        "longitude": [double],
        "numComments": [integer],
        "placeId": [string]
    },
]

```

- **ERROR RESPONSE:**

```

{
    "status": 500,
    "message": "An unexpected error occurred.",
    "timestamp": [datetime]
}

```

- **SAMPLE CALL:**

```

@GET("api/toilets/bounding")
suspend fun getToiletsBounding(
    @Query("minLat") minLat: Double,
    @Query("maxLat") maxLat: Double,
    @Query("minLon") minLng: Double,
    @Query("maxLon") maxLng: Double
): Call<ResponseBody>

```

Mostrar casas de banho por proximidade

- **URL:**

/toilets/nearby

- **METHOD:**

GET

- **URL PARAMETHERS:**

- Required (Query):
 - lat=[double],
 - lon=[double]
- Optional (Query):
 - state=[string], (default: null)
 - userId=[integer], (default: null)
 - pageable=[boolean], (default: false)
 - page=[integer], (default: 0)
 - size=[integer] (default: 20)

- **SUCCESS RESPONSE:**

```

[
    {
        "id": [integer],

```

```

    "name": [string],
    "address": [string],
    "rating": {
        "avgClean": [double],
        "avgStructure": [double],
        "avgAccessibility": [double],
        "ratioPaper": [double],
    },
    "extras": [strings],
    "access": [string],
    "latitude": [double],
    "longitude": [double],
    "numComments": [integer],
    "placeId": [string]
},
]

```

- **ERROR RESPONSE:**

```

{
    "status": 500,
    "message": "An unexpected error occurred.",
    "timestamp": [datetime]
}

```

```

{
    "status": 404,
    "message": "State with technical name [string] not found.",
    "timestamp": [datetime]
}

```

```

{
    "status": 404,
    "message": "User with id [integer] not found.",
    "timestamp": [datetime]
}

```

- **SAMPLE CALL:**

```

@GET("api/toilets/nearby")
suspend fun getToiletsNearby(
    @Query("lat") lat: Double,
    @Query("lon") lon: Double,
    @Query("state") state: String? = null,
    @Query("userId") userId: Int? = null,

```

```
@Query("pageable") pageable: Boolean = false,  
@Query("page") page: Int = 0,  
@Query("size") size: Int = 20  
) : Call<ResponseBody>
```

Mostrar casa de banho por ID

- **URL:**
`/toilets/{id}`
- **METHOD:**
`GET`
- **URL PARAMETERS:**
 - Required (Path):
`id=[integer]`
- **SUCCESS RESPONSE:**

```
{  
  "id": [integer],  
  "name": [string],  
  "address": [string],  
  "rating": {  
    "avgClean": [double],  
    "avgStructure": [double],  
    "avgAccessibility": [double],  
    "ratioPaper": [double],  
  },  
  "extras": [strings],  
  "access": [string],  
  "latitude": [double],  
  "longitude": [double],  
  "numComments": [integer],  
  "placeId": [string]  
}
```

- **ERROR RESPONSE:**

```
{  
  "status": 500,  
  "message": "An unexpected error occurred.",  
  "timestamp": [datetime]  
}
```

```
{  
  "status": 404,  
  "message": "Toilet with id [integer] not found.",  
}
```



```
"timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("api/toilets/{id}")
suspend fun getToilet(
    @Path("id") id: Int
): Call<ResponseBody>
```

Mostrar casas de banho pelo ID do usuário

- **URL:**

/toilets/users/{userId}

- **METHOD:**

GET

- **URL PARAMETHERS:**

- Required (Path):
userId=[integer]
- Optional (Query):
state=[string], (default: null)
pageable=[boolean], (default: false)
page=[integer], (default: 0)
size=[integer] (default: 20)

- **SUCCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "name": [string],
    "address": [string],
    "rating": {
      "avgClean": [double],
      "avgStructure": [double],
      "avgAccessibility": [double],
      "ratioPaper": [double],
    },
    "extras": [strings],
    "access": [string],
    "latitude": [double],
    "longitude": [double],
    "numComments": [integer],
    "placeId": [string]
  },
]
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "State with technical name [string] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("api/toilets/users/{userId}")
suspend fun getToiletsByUser(
    @Path("userId") userId: Int,
    @Query("state") state: String? = null,
    @Query("pageable") pageable: Boolean = false,
    @Query("page") page: Int = 0,
    @Query("size") size: Int = 20
): Call<ResponseBody>
```

Pesquisar casas de banho

- **URL:**

/toilets/search/{query}

- **METHOD:**

GET

- **URL PARAMETHODS:**

- Required (Path):
query=[string]
- Optional (Query):
pageable=[boolean], (default: false)

```
page=[integer], (default: 0)
size=[integer] (default: 20)
```

- **SUCCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "name": [string]
  },
]
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("api/toilets/search/{query}")
suspend fun searchToilets(
    @Path("query") query: String,
    @Query("pageable") pageable: Boolean = false,
    @Query("page") page: Int = 0,
    @Query("size") size: Int = 20
): Call<ResponseBody>
```

Visualizar imagem da casa de banho

- **URL:**

```
/toilets/{id}/image
```

- **METHOD:**

```
GET
```

- **URL PARAMETHERS:**

- Required (Path):

```
id=[integer]
```

- **SUCCESS RESPONSE:**

```
{
  "image": [image/jpeg]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "Toilet with id [integer] not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("api/toilets/{id}/image")
suspend fun getToiletImage(
    @Path("id") id: Int
): Call<ResponseBody>
```

Adicionar denúncia na casa de banho

- **URL:**

/toilets/reports

- **METHOD:**

POST

- **DATA PARAMETHERS:**

```
{
  "toiletId": [integer],
  "userId": [integer],
  "typeReport": [string]
}
```

- **SUCCESS RESPONSE:**

```
{
  "status": 201,
  "message": "Report added successfully.",
  "timestamp": [datetime]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "Toilet with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "TypeReport with technical name [string] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 500,
  "message": "Could not save Report.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("api/toilets/reports")
suspend fun addReport(
    @Body report: ReportRequest
): Call<ResponseBody>
```

Adicionar foto na casa de banho

- **URL:**
`/toilets/{id}/image`
- **METHOD:**
`POST`
- **URL PARAMETERS:**
 - Required (Path):
`id=[integer]`
- **DATA PARAMETERS:**

```
{  
  "image": [image/jpeg]  
}
```

- **SUCCESS RESPONSE:**

```
{  
  "status": 200,  
  "message": "Image added successfully.",  
  "timestamp": [datetime]  
}
```

- **ERROR RESPONSE:**

```
{  
  "status": 500,  
  "message": "An unexpected error occurred.",  
  "timestamp": [datetime]  
}
```

```
{  
  "status": 404,  
  "message": "Toilet with id [integer] not found.",  
  "timestamp": [datetime]  
}
```

- **SAMPLE CALL:**

```
@POST("api/toilets/{id}/image")  
suspend fun addImage(  
    @Path("id") id: Int,  
    @Body image: ImageRequest  
): Call<ResponseBody>
```

Apagar denúncia na casa de banho

- **URL:**
`/toilets/reports`
- **METHOD:**
`DELETE`
- **URL PARAMETHERS:**
 - Required (Query):
`toiletId=[integer],`
`userId=[integer]`
- **SUCCESS RESPONSE:**

```
{
  "status": 200,
  "message": "Report deleted successfully.",
  "timestamp": [datetime]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "Toilet with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "Interaction with toilet id and user id [integer], [integer]"
}
```

```
not found.",
    "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@DELETE("api/toilets/reports")
suspend fun deleteReport(
    @Query("toiletId") toiletId: Int,
    @Query("userId") userId: Int
): Call<ResponseBody>
```

Comentários - Endpoints

Mostrar comentários de uma casa de banho

- **URL:**
`/comments/toilets/{id}`
- **METHOD:**
`GET`
- **URL PARAMETHERS:**
 - Required (Path):
`id=[integer]`
 - Optional (Query):
`pageable=[boolean], (default: false)`
`page=[integer], (default: 0)`
`size=[integer] (default: 20)`
- **SUCCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "toiletId": [integer],
    "userId": [integer],
    "text": [string],
    "ratingClean": [integer],
    "ratingPaper": [boolean],
    "ratingStructure": [integer],
    "ratingAccessibility": [integer],
    "datetime": [datetime],
    "numLikes": [integer],
    "numDislikes": [integer],
    "score": [integer]
  },
]
```


- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "Toilet with id [integer] not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("api/comments/toilets/{id}")
suspend fun getComments(
    @Path("id") id: Int,
    @Query("pageable") pageable: Boolean = false,
    @Query("page") page: Int = 0,
    @Query("size") size: Int = 20
): Call<ResponseBody>
```

Mostrar comentários de um usuário

- **URL:**

/comments/users/{id}

- **METHOD:**

GET

- **URL PARAMETHODS:**

- Required (Path):
id=[integer]
- Optional (Query):
pageable=[boolean], (default: false)
page=[integer], (default: 0)
size=[integer] (default: 20)

- **SUCCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "toiletId": [integer],
    "userId": [integer],
```

```

        "text": [string],
        "ratingClean": [integer],
        "ratingPaper": [boolean],
        "ratingStructure": [integer],
        "ratingAccessibility": [integer],
        "datetime": [datetime],
        "numLikes": [integer],
        "numDislikes": [integer],
        "score": [integer]
    },
]

```

- **ERROR RESPONSE:**

```

{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}

```

```

{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}

```

- **SAMPLE CALL:**

```

@GET("api/comments/users/{id}")
suspend fun getCommentsByUser(
    @Path("id") id: Int,
    @Query("pageable") pageable: Boolean = false,
    @Query("page") page: Int = 0,
    @Query("size") size: Int = 20
): Call<ResponseBody>

```

Mostrar reações de um comentário

- **URL:**

/comments/reactions

- **METHOD:**

GET

- **URL PARAMETHODS:**

- Required (Query):

userId=[integer], commentIds=[integers]

- **SUCCESS RESPONSE:**

```
[
  {
    "commentId": [integer],
    "typeReaction": [string]
  },
]
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("api/comments/reactions")
suspend fun getReactions(
    @Query("userId") userId: Int,
    @Query("commentIds") commentIds: List<Int>
): Call<ResponseBody>
```

Adicionar comentário

- **URL:**

/comments

- **METHOD:**

POST

- **DATA PARAMETHODS:**

```
{
  "toiletId": [integer],
  "userId": [integer],
  "text": [string],
}
```

```
"ratingClean": [integer],
"ratingPaper": [boolean],
"ratingStructure": [integer],
"ratingAccessibility": [integer]
}
```

- **SUCCESS RESPONSE:**

```
{
  "id": [integer],
  "toiletId": [integer],
  "userId": [integer],
  "text": [string],
  "ratingClean": [integer],
  "ratingPaper": [boolean],
  "ratingStructure": [integer],
  "ratingAccessibility": [integer],
  "datetime": [datetime],
  "numLikes": [integer],
  "numDislikes": [integer],
  "score": [integer]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "Toilet with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 500,
  "message": "Could not save Comment.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("api/comments")
suspend fun addComment(
    @Body comment: CommentRequest
): Call<ResponseBody>
```

Adicionar reação em um comentário

- **URL:**

/comments/reactions

- **METHOD:**

POST

- **DATA PARAMETHERS:**

```
{
  "commentId": [integer],
  "userId": [integer],
  "typeReaction": [string]
}
```

- **SUCCESS RESPONSE:**

```
{
  "status": 201,
  "message": "Reaction added successfully.",
  "timestamp": [datetime]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "Comment with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "TypeReaction with technical name [string] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 500,
  "message": "Could not save Reaction.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("api/comments/reactions")
suspend fun addReaction(
    @Body reaction: ReactionRequest
): Call<ResponseBody>
```

Apagar reação em um comentário

- **URL:**

/comments/reactions

- **METHOD:**

DELETE

- **URL PARAMETHODS:**

- Required (Query):

commentId=[integer],
userId=[integer]

- **SUCCESS RESPONSE:**

```
{
  "status": 200,
  "message": "Reaction deleted successfully.",
  "timestamp": [datetime]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "Comment with id [integer] not found.",
  "timestamp": [datetime]
}
```

```
{
  "status": 404,
  "message": "User with id [integer] not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@DELETE("api/comments/reactions")
suspend fun deleteReaction(
    @Query("commentId") commentId: Int,
    @Query("userId") userId: Int
): Call<ResponseBody>
```

Observações

State

O state, visto em alguns endpoints, é um argumento que representa o estado de uma casa de banho. Ele pode ter os seguintes valores:

- **active** - Casa de banho ativa
- **inactive** - Casa de banho inativa
- **under-review** - Casa de banho em revisão
- **suggested** - Casa de banho sugerida
-

Extras

O campo extras, visto em alguns endpoints, é um array de strings que representa as características extras de uma casa de banho. Ele pode ter os seguintes valores:

- **WHEELCHAIR_ACCESSIBLE** - Acesso para cadeira de rodas
- **BABY_CHANGING_STATION** - Trocador de fraldas
- **DISABLED_PARKING** - Estacionamento para deficientes
- **ACCESSIBLE_FOR_VISUALLY_IMPAIRED** - Acessível para deficientes visuais

Esses valores apresentam diferenças com os presentes na base de dados, que basicamente são em lowercase, substituindo o `_` por `-`. Essa mudança serve para representações de classes Enum no JSON, facilitando o desenvolvimento pela parte do cliente.

Access

O campo access, visto em alguns endpoints, é uma string que representa o tipo de acesso de uma casa de banho. Ele pode ter os seguintes valores:

- **PUBLIC** - Acesso público
- **PRIVATE** - Acesso privado
- **CONSUMERS_ONLY** - Acesso apenas para consumidores

Esses valores diferem dos presentes na base de dados, pelo mesmo motivo dos extras.