

“How to Run” - Team 2

Garrett Davidson, Noah Maxey, Nate Ohlson, Riley Shaw

To run the server execute in the terminal `./HilingualServer`

Execute the server on a MacOSX 10 or Linux machine, and make sure to install MySQL before using.

This will start the server and listen for requests locally on the port 8180.

We recommend using Postman to send requests to the server. <https://www.getpostman.com/>

First, to create an account, you must use either Facebook or Google to get a test auth id and token. To do this with facebook visit the API documentation here -

<https://developers.facebook.com/docs/graph-api/reference/v2.8/test-user> To do this with google

visit the API documentation here - <https://developers.google.com/oauthplayground/> For google select the OAuth2 API v2 on that webpage and follow the instructions to get a test token. When getting a response from the OAuth2 test site, use the “`id_token`” parameter as the

“authorityToken” when registering with HiLingual, the “authorityAccountId” can be any unique id for google, so keep track of whatever id you pass.

Register:

http://localhost:8180/auth/register

Request type	authority	authorityAccountId	authorityToken	Body
POST	Either "FACEBOOK" or "GOOGLE"	ID from facebook that correlates the user. Google ID does not need to correlate to an ID, just has to be unique in the HiLingual DB.	Token given by Facebook or Google.	JSON containing authority, authorityAccountId and authorityToken.

Example:

localhost:8180/auth/register

POST

localhost:8180/auth/register

Params

Send

Authorization

Headers (1)

Body

Pre-request script

Tests

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

Text

```
1 {
2   "authority": "FACEBOOK",
3   "authorityAccountId": "140435239686268",
4   "authorityToken": "EAA0cGLfuCWEBAFjx42FyaUtT6kaXEdpcpxQa75cccbyi0gPZAQMqvNZAxwcDik0G0QEEYFTystZCoZAGhau0az
5 }
```

Login:

http://localhost:8180/auth/login

Request type	authority	authorityAccountId	authorityToken	Body
POST	Either "FACEBOOK" or "GOOGLE"	ID from facebook that correlates the user. Google ID does not need to correlate to an ID, just has to be unique in the HiLingual DB.	Token given by Facebook or Google.	JSON containing authority, authorityAccountId and authorityToken.

Example:


localhost:8180/auth/login

POST ▾

localhost:8180/auth/login

Params

Send ▾

 ▾

Authorization

Headers (1)

Body

Pre-request script

Tests

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

Text ▾

```
1 {
2   "authorityAccountId": "113901922344055",
3   "authorityToken": "EAA0cGlFuCWEBAPxsLTvNhhX8ZC6qu84pyo779518ECQkFfqpQpToonZAKBwSRudU0cEj0K7r8nt0iiIs1DU9B8",
4   "authority": "FACEBOOK"
5 }
```

Logout:

http://localhost:8180/auth/logout

Request type	authority	authorityAccountId	authorityToken	Body	UserId
POST	Either "FACEBOOK" or "GOOGLE"	ID from facebook that correlates the user. Google ID does not need to correlate to an ID, just has to be unique in the HiLingual DB.	Token given by Facebook or Google.	JSON containing authority, authorityAccountId, user_id and authorityToken.	HiLingual User id

Example:

localhost:8180/auth/logout

POST

localhost:8180/auth/logout

Params

Send

Authorization

Headers (1)

Body

Pre-request script

Tests

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

Text

```
1 {
2   "user_id": "1",
3   "authority": "FACEBOOK",
4   "authorityAccountId": "113901922344055",
5   "authorityToken": "EAA0cGlfuCWEBAPxSLTvNhhX8ZC6qu84pyo77951BECQkFfqPqToonZakBwSRudU0cEjOK7r8nt0iiIs1DU9B8
6 }
```

Messages:

Response: Should not say invalid message.

Sending a Message:

`http://localhost:8180/chat?auth={Auth-Token}&recipient={User-id}&message={Message}`

Request type	Auth-Token	User-id	Message
POST	An already logged in users auth token.	The user you want to send too.(they are incrementally increased based on when you added them so 1,2,3....) Is a number.	The message that you want to send.

Example:

The screenshot shows a REST client interface with a browser-like address bar at the top displaying 'http://localhost:8180/'. To the right of the address bar is a dropdown menu set to 'No Environment' and two icons (an eye and a gear). Below the address bar, there's a main configuration area. On the left, a dropdown menu is set to 'POST'. Next to it is the URL 'http://localhost:8180/translate?auth=EAAOcGifuCWEBALVFX&recipient=3&message=Hello'. To the right of the URL are tabs for 'Params', 'Send' (highlighted in blue), and 'Save'. Below these tabs are four sub-tabs: 'Authorization' (highlighted with an orange line), 'Headers', 'Body', 'Pre-request Script', and 'Tests'. At the bottom left, there's a 'Type' label and a dropdown menu set to 'No Auth'. On the far right, there's a 'Code' link in orange text.

Receiving Messages:

`http://localhost:8180/chat?auth={Auth-Token}&recipient={User-id}`

Request type	Auth-Token	User-id
GET	An already logged in users auth token.	The user you want to receive from.(they are incrementally increased based on when you added them so 1,2,3....) Is a number.

Example:

The screenshot shows a REST client interface with a browser-like address bar at the top displaying 'http://192.168.1.130:8180/'. To the right of the address bar is a dropdown menu set to 'No Environment' and two icons (an eye and a gear). Below the address bar, there's a main configuration area. On the left, a dropdown menu is set to 'GET'. Next to it is the URL 'http://192.168.1.130:8180/chat?auth=EAAOcGifuCWEBALVFXFo&recipient=1'. To the right of the URL are tabs for 'Params', 'Send' (highlighted in blue), and 'Save'. Below these tabs are four sub-tabs: 'Authorization' (highlighted with an orange line), 'Headers', 'Body', 'Pre-request Script', and 'Tests'. At the bottom left, there's a 'Type' label and a dropdown menu set to 'No Auth'. On the far right, there's a 'Code' link in orange text.

Sending a Picture Message:

`http://localhost:8180/picture?auth={Auth-Token}&recipient={User-id}`

Request type	Auth-Token	User-id	Picture
POST	An already logged in users auth token.	The user you want to send too.(they are incrementally increased based on when you added them so 1,2,3....) Is a number.	Should be a valid Image file.

Example:

The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:8180/picture?auth=SADefdaseE&recipient=1`
- Method: **POST**
- Body Type: **form-data**
- Key: **key**
- Value: **test.jpg** (selected via 'Choose Files')

Sending an audio message:

`http://localhost:8180/audio?auth={Auth-Token}&recipient={User-id}`

Request type	Auth-Token	User-id	Audio
POST	An already logged in users auth token.	The user you want to send too.(they are incrementally increased based on when you added them so 1,2,3....) Is a number.	Should be a valid m4a audio data.

Example:

The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:8180/audio?auth=adsf&recipient=1`
- Method: **POST**
- Body Type: **form-data**
- Key: **audio**
- Value: **under10.mp3** (selected via 'Choose Files')

Flashcards:

Sending flashcards:

<http://localhost:8180/flashcard?auth={Auth-Token}&setid{Set-id}>

Request type	Auth-Token	Set-id	Flashcards
POST	An already logged in users auth token.	What you want to set of flashcards to be called. Has to be Unique.	A valid JSon array with front and back fields.

Example:

The screenshot shows a REST client interface with a POST request to `http://localhost:8180/flashcard?auth=2&setid=set1`. The request body is a JSON array of two flashcard objects. The first object has a front of "Hello" and a back of "Hallo". The second object has a front of "I" and a back of "Ich".

```
1 {
2   "FlashcardRingName": [
3     {
4       "front": "Hello",
5       "back": "Hallo"
6     },
7     {
8       "front": "I",
9       "back": "Ich"
10    }
11  ]
12 }
```

Get flashcards:

<http://localhost:8180/flashcard?auth={Auth-Token}&setid{Set-id}>

Request type	Auth-Token	Set-id
Get	An already logged in users auth token.	What you want to set of flashcards to be called.

Example:

The screenshot shows a REST client interface with a GET request to `http://localhost:8180/flashcard?auth=asdfASDWewr&setid=set1`. The Authorization tab is selected, showing "No Auth" as the type.

Editing flashcards:

<http://localhost:8180/flashcard?auth={Auth-Token}&setid={Set-id}>

Request type	Auth-Token	Set-id	Flashcards
POST	An already logged in users auth token.	The set-id should match a previously submit set.	A valid JSon array with front and back fields.

Example:

The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:8180/flashcard?auth=adsfAFIiWJD&setid=set1`
- Method: `POST`
- Body Type: `JSON (application/json)`
- Body Content (JSON):

```
1 {
2   "NewFlashcardName": [
3     {
4       "front": "Hello",
5       "back": "Hallo"
6     },
7     {
8       "front": "I",
9       "back": "Ich"
10    },
11    {
12      "front": "NO",
13      "back": "NEIN"
14    }
15  ]
16 }
```

Translations:

Send a translation:

<http://localhost:8180/translate?auth={Auth-Token}&language={Language}&message={message}>

Request type	Auth-Token	Language	message
POST	An already logged in users auth token.	A language code for a language. http://www.science.co.il/Language/Locale-codes.asp	The message you want to be translated

Example:

The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:8180/translate?auth=EAAOcGlfuCW&language=de&message=Hello`
- Method: `POST`

User:

Update:

<http://localhost:8180/user/update>

Request type	Auth-Token	User Id	name	displayName
POST	An already logged in users auth token.	User's user id	Proper name of the user	The name that other users see

bio	gender	birthdate	Native languages	Learning languages
The bio of the person	Either all caps "MALE" or "FEMALE"	An integer of unix timestamp	Comma separated list of languages	Comma speparated list of languages

Example:


localhost:8180/user/update

POST ▾

localhost:8180/user/update

Params

Send ▾

 ▾

Authorization

Headers (1)

Body

Pre-request script

Tests

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

Text ▾

```
1 {
2   "user_id": "1",
3   "authorityToken": "EAAOcGl fuCWEBAPxsLTvNhhX8ZC6qu84pyo77951BECQkFfqPqToonZakBwSRudU0cEj0K7r8nt0iiIs1DU9f",
4   "name": "Bobby Smith",
5   "displayName": "bbob",
6   "bio": "I'm a cool cat",
7   "gender": "MALE",
8   "birthdate": "1477595593",
9   "nativeLanguages": "English",
10  "learningLanguages": "Spanish"
11 }
```

Matching

<http://localhost:8180/user/match>

Request type	authorityToken	Body
POST	Token given by Facebook or Google.	JSON containing authorityToken.

Example:

POST ▾

192.168.1.122:8180/user/match/

Params

Send ▾

Save ▾

AuthorizationHeadersBody ●Pre-request ScriptTestsManage CookiesGenerate Code

form-data

x-www-form-urlencoded

● raw

binary

Text ▾

1 {

2 "authorityToken": "EAA0cGlfuCWEBAEao5uIsn8qvShrEGYDiv8ZAE8gm3gym5iyQJBFS1CZCndJSP1zmUBPBb8U5rLxkkZAZ

3 AjZABV0sLorAAZBvuyudv33uNgekspgHggEt9yu8DQzIuzDaPv5ZCZCQLC7wVZC3uRLESoaNYYuR1Js3qVDhKa6QhxdzH8If

4 Gcj00SAIO"

5 }