Old Sharetribe API

NOTE: This is a deprecated documentation from an old version of Sharetribe Go. Currently Sharetribe Go doesn't have a public API, and while we will probably implement one at some point, there's currently no timeline for when that might happen.
If you need a marketplace platform with API access, you could take a closer look at Sharetribe Flex.

VERSION 2

For the API VERSION 1 (or alpha) see older documentation
See also information how to specify the version used

This is the Work-In-Progress documentation for the Sharetribe API . Feel free to modify, add stuff, make it more accurate and comment as much as there are thoughts coming.

Contents:

Servers

If you are connecting to the Sharetribe servers hosted by the Sharetribe Ltd, there are two options.

Test server (api.sharetribe.fi)

The test server is a sandbox like environment where you can play around without the fear of deleting critical data or confusing real users. There are few communities hosted in the test

server that are also visible in the web:

- http://alpha.sharetribe.com
- http://gamma.sharetribe.com
- http://translate.sharetribe.com

Any of these communities can be used to post any test comment and you can modify the existing ones. Don't expect any data to stay there as all can be changed. Part of the accounts are dummy accounts and part are made by developers or translators, so don't unnecessarily spam any account with messages or comments for example.

The test server has completely different data base, so the also user accounts are completely different. The server might be running newer version of the Sharetribe code than the production server depending on the situation of the development team. The test server response times are usually longer than the production server.

Few of the features are not always on at the test server, as they are needed only occasionally and cause separate costs in the could infrastructure:

- Search
- background job processing (some of the notification emails for example require this)

If you are doing testing where you need one of these to work, please notify core team at the Developers Flowdock.

Production server (api.sharetribe.com)

This is the real server. You may connect freely and authenticate as users existing in the Sharetribe platform. Be sure to inlcude client identification to your requests. You should do that on the test server too, but at least make sure those are filled when contacting production server. It's not currently enforced, but might become so in future versions of the API.

Formats

JSON (+ API version)

Possible MIME types:

```
application/json
application/vnd.sharetribe+json; version=2
```

The use of a MIME type that defines the version is **highly recommended** as the first one uses the newest version by default and returns the latest representation of the resource and as the API updates, that might break down the application which expects the response in old format. So specify version in the Accept header and you'll know what you get.

The generic JSON type is supported mainly to make exploring the API bit easier in a browser

RSS/ATOM

Possible MIME types:
`application/atom+xml`

Sharetribe currently supports only ATOM as the feed format, as it's bit more versatile than RSS2.0 and supported by all major feed readers.

Currently only /listings responds to ATOM requests. It supports the same parameters as [/listings GET JSON API]

GZIP

The API can return Gzipped responses if the request has *Accept-Encoding* header with "gzip".

Client identification

You should use HTTP header `Sharetribe-Client` that has your app's name, contact email and url. This makes it easier for us to measure usage from different sources and also gives us a possibility to easily contact our API users in case some bigger changes need to be informed.

example:

```
curl -H "Sharetribe-Client:
SharetribeAndroid,contact@greatdevteam.com,http://greatdevteam.com"
```

Thanks to Kippt API for this idea!

Authentication

Sharetribe API can be used with or without authentication. Authenticated requests must contain the API token and the response for authenticated requests can contain more information, for example listings that are visible only to the members of a specific community. Some requests are not possible without authentication, e.g. posting a new listing.

Sharetribe API uses token based authentication. You can request the token for the user by

submitting username or email and password. Do not store user passwords anywhere, store the token!

/tokens

POST

Parameters

login: User's username or email
password: User's password

Returns:


```
{"api_token":"6iKztUsbkemg6dLRopd1",
"person":{
        "username":"jacky",
        "phone_number":"1234567",
        "locale":"en",
        "given_name":"Proto",
        "description":null,
        "family_name":"Testro",
        "id":"drty-AWmWr4yxgUi0sbZZU"
        }
}
```


Example: curl -H "Accept: application/vnd.sharetribe+json; version=1" -H "Content-Type: application/json" -X POST --data '{"login": "example", "password": "secret"}' http://api.sharetribe.fi/tokens

Then you can provide the api_token in further API calls to make the server handle the calls as authenticated user.

You can provide the authentication token in the headers (recommended) or as parameter:

Examples:
curl -H "Sharetribe-API-Token: sTNEbgjeudi5BqGR3Hcy"
or
curl /listings?api_token=sTNEbgjeudi5BqGR3Hcy

MOST OF THE STUFF BELOW IS OVERSTRIKE AS IT MIGHT BE REMOVED/CHANGED SOON, SEE NOTE AT THE TOP OF THE DOCUMENT

~~Resources~~

~~People~~

~~The users. Each user is visible even to unauthenticated API requests, but the response contains possibly more information about the user if the request is authenticated.~~

~~Search users: /people~~

~~GET~~

~~Get the array of users who match the criteria given in parameters.~~

~~Parameters~~

~~**email:** returns the person that has the email, or empty array if no match found.~~
~~**community_id:** to get the members of that community (this can't be combined currently with email param)~~

~~example response:~~

~~{people":~~
~~[~~
~~{~~
~~"username":"kassi_tester3",~~
~~"id":"aGlUQW2q4r4yCYUi0sbZZU",~~
~~"given_name":"Proto",~~
~~"family_name":"Testro",~~
~~"thumbnail_url":"http://test.host/images/thumb/missing.png",~~
~~"picture_url":"http://test.host/images/medium/missing.png"~~
~~}~~
~~],~~
~~"per_page" :"50",~~
~~"page":"1",~~
~~"total_pages": "1"~~
~~}~~

Single user: /people/<id>

GET

Returns the single user specified by the <id>. Contact details are included only if request is authenticated. Email is included only if the requested user himself is authenticated.

example response:

{"given_name":"Proto",
"communities":[
{"name":"testcommunity_1","id":11,"domain":"test.sharetribe.com"},
{"name":"kassi3","id":12,"domain":"kassi3.sharetribe.com"}],
"family_name":"Testro",
"description":null,
"username":"tester1",
"id":"b_rYu0Wm0r4y8IUi0sbZZU",
"locale":"en",
"phone_number":null
"picture_url":"http://api.sharetribe.fi/system/images/bU8aHSBEKr3AhYaaWPEYjL/medium/holiday_in_canary.jpg?1336145277",
"thumbnail_url":"http://api.lvh.me:3000/system/images/bU8aHSBEKr3AhYaaWPEYjL/thumb/holiday_in_canary.jpg?1336145277"
}

PUT (not yet done)

Update user's profile. Only can modify the profile of the authenticated user

Single user's devices: /people/<id>/devices

Main usage is to store iOS devices for push notifications.

GET

Returns the list of devices of the user.

example response:

[
{
"device_type":"iPad",
"device_token":"738SDK2FFKD29D"

}
}

POST

Add a device to the user

parameters

device_type: The type of the device.
device_token: The device token needed for push notifications.

Single user's listings: /people/<id>/listings

GET

Returns the list of listings made by the user.

Parameters

(same as listing index below)

Listings

The offers and requests in the service. There are several categories and new ones are probably added in the future. You can expect every listing to contain the basic attributes, but should not rely on knowing all categories, and be able to show a listing also in unknown category with some default texts.

The listings currently have tags, and in future they might have subcategories. The solution how this is implemented is still open, and for that reason the API does not yet handle tags (or subcategories) at all.

/listings

All listings that match the criteria given in parameters. Results are ordered by created_at so that newest are first.

GET

parameters

community_id: returns listings only from single community (obligatory)
status: (optional) Possible values “open”, “closed” or “all”. Returns only listings with that status. Default is “open”.
listing_type: Possible values “request” or “offer”. Returns only listings with that type.
category: (optional) String for category name. Returns only listings with that category.
per_page: how many results you want to be in one response. Default 50.
page: When using pagination, define which page of results you want to get. Page numbers start from 0.
search: string that has word or part of word that is searched. Will match to title, description and tags. Currently can be combined only with status and listing_type parameters (category will be ignored if search parameter is given).
(Not yet implemented)
minLat
maxLat
minLon
maxLong

examples:

http://api.sharetribe.fi/listings.json?community_id=1&type=offer

example response:

{"per_page":"2",
"page":”1”,
“total_pages”: “2”,
"listings":[
{normal listing hash},
{normal listing hash}
]
}

~~**POST**~~

~~Creates a listing. Requires api token authorization.~~

~~**parameters**~~

~~**title:** The title of the listing. 2-90 characters. Not needed for rideshare listings as for them the title is created from origin and destination parameters.~~
~~**description (optional):** Longer description text. max 5000 characters.~~
~~**category:** Current options: item, favor, rideshare, housing. (Don’t rely in your code of this list~~

staying the same, as additions are coming)
share_type: (required) possible values depend on the commmunity, and can be figured out from the community's categorization_tree. On top level there are probably just offer/request but the the categories may include more sub-ShareTypes.
visibility: (optional) Means where the listing should be shown, i.e. in which communities it is relevant. Defaults to "all_communities" (meaning all the communities where the user is member). Other possible values are "this_community".
privacy: (optional) Selects if the listing is only shown to logged in members of the communities, or if it can be shown in public internet. Possible values "public" and "private". The default is "private".
community_id: Specify in which community is the listing added to. The authorized user must be member of that community. The communities are included in the data of the user.
image: (optional) The image for the listing as multipart file upload.
price_cents: (optional) The price in cents (meaning the 1/100s of the currency.) E.g. 15 EUR should have 1500.
currency: (optional) The official currency code, e.g. "EUR", "USD"
quantity: (optional) A short text explaining what quantity does the price concern.
latitude: (optional) Latitude of the listing location
longitude: (optional) Longitude of the listing location
address: (optional) Address for the listing location.

valid_until: (optional for some share types) How long is the listing get open. (Max 1 year.) Can be left out if offering items for lending or repeating ridesharing.
+Only for ridesharing listings:
origin: The text for the starting point of the ride. The title will de "origin - destination".
destination: The text for the destination of the ride. The title will de "origin - destination".
destination_latitude: (optional) Latitude of the rideshare listing destination location
destination_longitude: (optional) Longitude of the rideshare listing destination location
destination_address: (optional) Address for the rideshare listing destination location.

/listings/<id>

single listing

GET

Get the listing specified by <id>

e.g. http://api.sharetribe.fi/listings/1.json

example response:

{"type":"offer",

```
"thumbnail_url":"http://api.lvh.me:3000/system/images/17/thumb/pakki.jpg?1329352553",
"updated_at":"2012-06-29T05:50:42+03:00",
"times_viewed":19,
"comments":[
        {
        "created_at":"2012-06-29T05:50:56+03:00",
        "content":"cool tool",
        "author_id":"bU8aHSBEKr3AhYaaWPEYjL"
        },
        {
        "created_at":"2012-06-29T05:58:04+03:00",
        "content":"indeed",
        "author_id":"bU8aHSBEKr3AhYaaWPEYjL"
        }
],
"image_urls":[
        "http://api.lvh.me:3000/system/images/17/medium/pakki.jpg?1329352553"
],
"created_at":"2012-02-16T02:34:21+02:00",
"valid_until":null,
"description":"power drill, hammer etc.",
"tags":["hammer","tool"],
"title":"tools",
"category":"item",
"share_type":"sell",
"id":313,
"origin_location":{
        "address":"Provi tex, Laredo, TX 78043, USA",
        "latitude":27.4967,
        "google_address":"Provi tex, Laredo, TX 78043, USA",
        "longitude": 99.4497
        },
"author":{
        "given_name":"Antti",
        "family_name":"Vee",
        "username":"gekko",
        "description":"Nice guy",
        "communities":[
                {
                "domain":"test",
                "name":"test",
                "id":1
                }
```

```
        },
        "id":"bU8aHSBEKr3AhYaaWPEYjL",
        "phone_number":"3248923",
        "locale":"es"
},
"visibility":"all_communities",
“privacy”:”public”,
"price_cents":2900,
"currency":"EUR",
"quantity":"per piece"}
```


PUT (note yet done)

Update single listing with new details. (not yet implemented)

/listings/<id>/comments

POST

Creates a comment for the listing specified by <id>. Requires api-token authorization.

parameters

content: The text of the comment.
community_id: This is required to know which context was the comment made in.

returns

201 created

example response:


```
{"created_at":"2012-07-21T01:52:04+03:00",
"listing_id":333,
"content":"testing comments",
"author":{
        "username":"kassi_tester3",
        "thumbnail_url":"http://test.host/images/thumb/missing.png",
        "given_name":"Proto",
        "family_name":"Testro",
        "id":"aGlUQW2q4r4yCYUi0sbZZU",
        "picture_url":"http://test.host/images/medium/missing.png"
        }
}
```

Conversations & Messages

Conversation happens between two users and can have multiple messages. Conversations can be related to a listing or be a free form messages between two users.

/people/<id>/conversations

GET

Returns all the conversations of the user. Results have pagination.

parameters

per_page: how many results you want to be in one response. Default 50.
page: When using pagination, define which page of results you want to get.

returns

200 ok

example response:


```
{"page":1,
"per_page":50,
“total_pages”: “1”,
"conversations":[
{
        "title":"Item offer: Sledgehammer",
        "status":"pending",
        "listing_id":94,
        "updated_at":"2012-07-22T06:03:09+03:00",
        "participations":[
                {
                "last_received_at":"2012-07-29T02:48:46+03:00",
                "feedback_skipped":false,
                "person":
                {
                        "username":"kassi_tester27",
                        "thumbnail_url":"http://test.host/images/thumb/missing.png",
                        "given_name":"Proto",
```

```
                "family_name":"Testro",
                "id":"dfphUE2q4r4yZSUi0sbZZU",
                "picture_url":"http://test.host/images/medium/missing.png"
            },
            "is_read":true,"last_sent_at":null
        },
        {
            "last_received_at":"2012-07-29T02:48:46+03:00",
            "feedback_skipped":false,
            "person":
            {
                "username":"kassi_tester26",
                "thumbnail_url":"http://test.host/images/thumb/missing.png",
                "given_name":"Proto",
                "family_name":"Testro",
                "id":"dfoCMk2q4r4yZSUi0sbZZU",
                "picture_url":"http://test.host/images/medium/missing.png"
            },
            "is_read":true,
            "last_sent_at":null
        }
        },
        "created_at":"2012-07-22T06:03:09+03:00",
        "id":16,
        "last_message":
        {
            "created_at":"2012-07-30T05:39:44+03:00",
            "sender_id":"dfphUE2q4r4yZSUi0sbZZU",
            "content":"This is the last thing said"
        }
}
]
}


{
```


POST

Starts a new conversation.

parameters

target_person_id: The id of the target person, with whom the conversation is started.
listing_id: (optional) if the conversation is related to a listing, the id should be provided.
status: Can be “free” or “pending”. *Pending* means a direct request to the resource in the listing, and the other party is asked to accept or reject the pending request. *Pending* message needs to have a *listing_id. Free* message means discussion without explicit request to the resource. It may have *listing_id* if it’s stared from a listing page or go without if it is started on a profile page.
content: The textual content of the first message
title: (optional if listing_id provided) The title of the conversation. If listing_id is provided, this can be left empty and the title is generated automatically based on the listing title.
community_id: This is required to know which context was the conversations is started.

returns

201 created

example response:


```
{"status":"pending",
"listing_id":179,
"updated_at":"2012-07-22T22:48:59+03:00",
"created_at":"2012-07-22T22:48:59+03:00",
"participations":
[
        {
        "feedback_skipped":false,
        "is_read":true,
        "person_id":"bg_e5e1dyr4yQrUi0sbZZU",
        "last_sent_at":null,
        "last_received_at":"2012-07-22T22:48:58+03:00"
        },
        {
        "feedback_skipped":false,
        "is_read":true,
        "person_id":"bg_VQ41dyr4yQrUi0sbZZU",
        "last_sent_at":null,
        "last_received_at":"2012-07-22T22:48:58+03:00"
        }
],
"title":"Item offer: Sledgehammer",
"id":72,"messages":
[
        {
        "created_at":"2012-07-22T22:48:59+03:00",
```

```
        "sender_id":"bg_e5c1dyr4yQrUi0sbZZU",
        "content":"This will be the first message of the conversation"
        }
]
}
```


/people/<person_id>/conversations/<conversation_id>/

GET

Returns all the messages in the conversation specified by conversation_id.

parameters

returns

200 ok

example response:


```
{"title":"Item offer: Sledgehammer",
"status":"pending",
"participations":
[
        {
        "feedback_skipped":false,
        "is_read":false,
        "person_id":"bco6ls1c0r4A06Ui0sbZZU",
        "last_sent_at":"2012-07-22T21:44:24+03:00",
        "last_received_at":"2012-07-22T21:44:24+03:00"
        },
        {
        "feedback_skipped":false,
        "is_read":true,
        "person_id":"bcpL3S1c0r4A06Ui0sbZZU",
        "last_sent_at":"2012-07-22T21:44:24+03:00",
        "last_received_at":"2012-07-22T21:44:24+03:00"
        }
],
"messages":
[
        {
```

```
        "created_at":"2012-07-22T21:44:24+03:00",
        "sender_id":"bco6ls1c0r4A06Ui0sbZZU",
        "content":"Let's talk"
        },
        {
        "created_at":"2012-07-22T21:44:24+03:00",
        "sender_id":"bcpL3S1c0r4A06Ui0sbZZU",
        "content":"Ok! You start."
        }
],
"listing_id":96,
"updated_at":"2012-07-22T21:44:24+03:00",
"created_at":"2012-07-22T21:44:24+03:00",
"id":25
}
```


POST

Sends a new message to this conversation.

parameters

content: The textual content of the message
community_id: This is required to know which context was the conversations is started.

returns

Normal conversation JSON after the addition of the new message. (see GET for this URL)

PUT

Update the conversation. Mainly used if the other party accepts or rejects the request.
parameters
status: can be "free", “pending", "accepted", "rejected", "payed", "confirmed" or "canceled"
community_id: This is required for easier notifications to the conversation participants.

Feedbacks

Feedbacks that users give each other after a transaction.

/people/<id>/feedbacks

GET

Get the feedbacks of the person specified by <id>.

returns

200 ok

example response:


```
{
"feedbacks":[
        {
        "text":"Nice job!",
        "created_at":"2012-08-19T20:33:26+03:00",
        "grade":0.75,
        "author_id":"d7i1wU6lmr4y0SUi0sbZZU",
        "receiver_id":"d7gtlw6lmr4y0SUi0sbZZU",
        "converstation_id":56
        },
        {
        "text":"well done",
        "created_at":"2012-08-19T20:33:25+03:00",
        "grade":0.5,
        "author_id":"d7i1wU6lmr4y0SUi0sbZZU",
        "receiver_id":"d7gtlw6lmr4y0SUi0sbZZU",
        "converstation_id":55
        }
],
"grade_amounts":[
        ["exceeded_expectations",0,"5"],
        ["slightly_better_than_expected",1,"4"],
        ["as_expected",1,"3"],
        ["slightly_less_than_expected",0,"2"],
        ["less_than_expected",0,"1"]
        ],
"page":1,
"total_pages":1,
"per_page":50
}
```

POST (not yet done)

Add new feedback for the person specified by <id>.

Badges

Badges are virtual rewards that are automatically given to users based on their activity in the service.

/people/<id>/badges

GET

Get the badges of the person specified by <id>.

returns

200 ok

example response:

{"badges":[
{
"created_at":"2012-08-19T21:34:44+03:00",
"picture_url":"http://test.host/images/badges/rookie_large.png",
"description":"You have added an offer or a request in Sharetribe for the first time. Here we go!",
"name":"rookie",
"id":2
},
{
"created_at":"2012-08-19T21:34:44+03:00",
"picture_url":"http://test.host/images/badges/volunteer_bronze_large.png",
"description":"You like to put your skills in use by helping others. You have three open service offers in Sharetribe.",
"name":"volunteer_bronze",
"id":3
}
]}

Communities

Thir resouce contains the different communities in Sharetribe. In web they have separate subdomains, e.g https://oin.sharetribe.com/ for OIN tribe.

/communities

GET (not yet done)

Get the list of tribes

/communities/<id>

GET

Get the details of a single community.

Explanations to some fields:

custom_color1: May contain the hex color code e.g. “FF0042” which is the first customization color used by the community.

custom_color2: May contain a second custom color code. Many communties don’t have this even if they have the first color set. Then it is advisable to use the custom_color1 in all custom colored elements

service_name: Contains the name of the whole service that is displayed to the users of this tribe. The point here is that some tribes have been white label customized and don’t have the Sharetribe name visible. In most cases the service_name is “Sharetribe”, but exeeptions do exists and it’s advisable to use the service_name to the members of that tribe, as that’s probably the name they are used to see.

service_logo_style: Tells if a community has white label settings and whole sharetribe logo should not be shown. There are 3 possible values: “full-logo” = (default) show full sharetribe logo where needed, “icon-logo” = show only the S logo, “no-logo” = no Sharetribe or S logo should be shown when using this communitiy.

categories_tree: Contains the explanation of which share_types and categories are used in this community. First level in the JSON is the top level share_type (also called as listing_type) That key contains a hash, where the keys are the category names. If the value is empty there are no subcatgegories or sub share_types used for that category. If those are used, they are listed in the hash that is the value. Every listing **must** have a category and a sharetype, if there is no subcategory or share_type, the top level one is used (those exists always) e.g. “favor, offer” and if sub levels exists, those must be used always, e.g. “tools, lend” (not “item, offer”).

~~example response:~~

~~{"id"=>125,~~
~~"name"=>"sharetribe_testcommunity_1",~~
~~"slogan":"Test slogan",~~
~~"description":"Test description",~~
~~"custom_color1":“ff22dd”,~~
~~"custom_color2":null,~~
~~"payments_in_use":false,~~
~~"available_currencies":null,~~
~~"join_with_invite_only":false,~~
~~"members_count":138,~~
~~"domain":"sharetribe_testcommunity_2.lvh.me:9887",~~
~~"service_name":"Sharetribe",~~
~~“service_logo_style”:”full-logo”,~~
~~"logo_url":"http://test.sharetribe.com/logos/header/default.png",~~
~~"cover_photo_url":"~~http://test.sharetribe.com/cover_photos/header/default.jpg~~",~~
~~"location":{~~
~~"latitude":60.1904,~~
~~"longitude":24.8937,~~
~~"address":"Somecity",~~
~~"google_address":"Somecity, 12345, Finland"}~~
~~},~~
~~"categories_tree":{~~
~~"offer":{~~
~~"item":{~~
~~"subcategory":[~~
~~"tools","sports","music","books","games","furniture",~~
~~"outdoors","food","electronics","pets","film",~~
~~"clothes","garden","travel","other"~~
~~],~~
~~"share_type":["sell","rent_out","lend","offer_to_swap","give_away"]~~
~~},~~
~~"favor":{},~~
~~"rideshare":{},~~
~~"housing":{~~
~~"share_type":["sell","rent_out","share_for_free"]~~
~~}~~
~~},~~
~~"request":{~~
~~"item":{~~
~~"subcategory":[~~

~~"tools","sports","music","books","games","furniture",~~
~~"outdoors","food","electronics","pets","film",~~
~~"clothes","garden","travel","other"~~
~~],~~
~~"share_type":["buy","rent","borrow","request_to_swap","receive"]~~
~~},~~
~~"favor":{},~~
~~"rideshare":{},~~
~~"housing":{~~
~~"share_type":["buy","rent","accept_for_free"]~~
~~}~~
~~}~~
~~}~~
~~}~~

~~/communities/<id>/classifications~~

~~The classifications (categories and share_types) used in this community. The categories_tree is included in the community basic hash, but in order to get the translated display_names of the categories and to know whether price field should be used etc. this call is needed.~~

~~GET~~

~~Get more detailed information about the classifications that are used in this community: translated names, descriptions and payment information. Price is true if price field should be used with that share_type. Payment is true if payment is possible even if price is not used in the listing (e.g. buying). Price_quantity_placeholder is a helping text that can be used as example of what quantity does the price consider. If it's null, the quantity is not used, just the price field.~~

~~parameters~~

~~**locale**: The 2 letter code to define which translations should be returned, e.g. "en", "fi" etc.~~

~~example response:~~

```
~~{~~
~~"tools":~~
~~{"translated_name":"tools",~~
~~"description":"Tools",~~
~~"price":false,~~
~~"price_quantity_placeholder":null,~~
~~"payment":false~~
~~},~~
```

```
"rent_out":{
        "translated_name":"renting out",
        "description":"I'm renting it out for a fee",
        "price":true,
        "price_quantity_placeholder":"time",
        "payment":true
},
"buy":{
        "translated_name":"buying",
        "description":"I want to buy it",
        "price":null,
        "price_quantity_placeholder":null,
        "payment":true
}
}
```


/api_version

GET

Get the details of the current API version. This is intended to use e.g. in mobile clients so that the user can be prompted to update to the newest version if the client app is using a deprecated or non-supported API version. There may also be an optional message returned from the server, and it is a good idea to show that to the user if it exists. The request must contain a proper version definition in the MIME type.

The returned hash contains always a response to the API version used in the request MIME type. “your_version” has possible values “latest”, “deprecated” or “not_supported”.

example response:

{"your_version":"latest",
"message":null}

or

{"your_version":"not_supported",
"message":”The version you are using is too old, please update.”}

Questions & Discussion

- What method for authentication?
  - **token auth for now**
    - supported by devise
  - Good examples?
    - Twitter uses OAuth,
      - it seems bit tedious to get started with that
    - Kippt uses http-basic + token
      - simpler to do
    - Twilio (often recommended as good example)
      -
- Should we code the API responses inside our current controllers or separate
  - If inside we can benefit from existing filters and checks (but might need to bypass some)
  - if outside we need to write all and remember to update the API too if new checks added. But on the other hand code stays more clear and API versioning is easier
    - **Looks better to do separate controllers for API**
- How to do API versioning.
  - Should we already add /api/v1/ in urls or is it enough to do that for v2 onwards
    - probably no need for v1
  - read about best practices, should it be in url or headers
    - It seems putting it in the headers is a more correct way of doing things, as returning links with the api verion as part of the uri will get those broken or inconsistent when version changes, (at least if upgrading only part of API or if there's links store on client side and then there will mix of links to different versions)
    - however, requiring the header always can make the API harder to discover and play with, so it might be good to make it return also some sensible stuff if mime type is just application/json.
      - like github does it: http://developer.github.com/v3/mime/
    - Actually even **better format would be to use MIME type parameters for version**, e.g. `application/vnd.sharetribe; version=1`
- How to serve different languages in API?
  - having language in url is nice in web because people can copy/paste links to different language sites. In API it's usually not needed, as resource contents are rarely translated. So basic use is without language in the url and if need for specific language there is a header for that
    - although supporting similarurl part language as the service itself could make sense for the sake of consistency, but not urgent now as there's no immediate need for this at all. (The error messages is the first thing that

comes to mind)

- How to handle translation files in different mobile clients?
  - any best practices?
  - for two identical clients we could have the same translation file
- Should we add a client header when making API public, like Kippt:

  You should also use `X-Kippt-Client` HTTP header with app's name, contact email and url. This makes it easy for us to contact you if there's a case for that (i.e. deprecated features).

  Format: `curl -H "X-Kippt-Client: BookmarkingApp,contact@bookmarkingapp.com,http://bookmarkingapp.com"`
  - Yes we should

Resources

http://blog.steveklabnik.com/posts/2011-07-03-nobody-understands-rest-or-http good post about how to do things correctly, and follow up: http://blog.steveklabnik.com/posts/2011-08-07-some-people-understand-rest-and-http