1 Introduction

Build Kele, a Ruby Gem API client to access the Bloc API.

Overview and Purpose

In this project you'll create a Ruby Gem API client.

Objectives

After this project, you should be able to:

- Create a basic Ruby Gem.
- Explain authorization with JSON Web Tokens.
- Understand the reasons to call an API using a client.
- Explain how to update resources via an API.

Use Case

Bloc's API provides an external facing **JSON Web Token** authorized gateway to the Bloc application. You can access it via **cURL**, but an API client can manage the low-level details of making requests and handling responses. Build the **Kele** API Client to provide easy access to and use of **the student endpoints** of Bloc's API.

User Stories

User Story	Difficulty Rating
As a user, I want to initialize and authorize `Kele` with a Bloc username and password	3
As a user, I want to retrieve the current user as a JSON blob	2
As a user, I want to retrieve a list of my mentor's availability	3
As a user, I want to retrieve roadmaps and checkpoints	2
As a user, I want to retrieve a list of my messages, respond to an existing message, and create a new message thread	4
As a user, I want to submit checkpoint assignments	3

How would you rate this checkpoint and assignment? ☺ ☹

2 Initialize and authorize KeleClient

> As a user, I want to initialize and authorize `Kele` with a Bloc username and password
> **Difficulty Rating**: 3

Creating the `Kele` Gem

Create an empty repository on GitHub named `Kele` and clone it locally.

> You should do this from memory, but see **Git Checkpoint Workflow** if you need a refresher.

Creating this project as a **RubyGem** allows us to integrate `Kele` with other software.

At a minimum, a RubyGem needs a `.gemspec` file (typically `project_name.gemspec`) and one Ruby file (typically `lib/project_name.rb`).

A gemspec defines metadata about your RubyGem like its name, version, and author.

Create `kele.gemspec`:

```diff
+ Gem::Specification.new do |s|
+   s.name          = 'kele'
+   s.version       = '0.0.1'
+   s.date          = '2015-12-02'
+   s.summary       = 'Kele API Client'
+   s.description   = 'A client for the Bloc API'
+   s.authors       = ['Hannah McExample']
+   s.email         = 'hannah@example.com'
+   s.files         = ['lib/kele.rb']
+   s.require_paths = ["lib"]
+   s.homepage      =
+     'http://rubygems.org/gems/kele'
+   s.license       = 'MIT'
+   s.add_runtime_dependency 'httparty', '~> 0.13'
+ end
```

Replace `date`, `authors`, and `email` with your own information.

`files` is an array of files included in the gem. List them individually as you create them:

```
s.files = ['lib/kele.rb', 'lib/roadmap.rb']`
```

> A gemspec is called from a Ruby method — anything you can do in Ruby you can do in a gemspec.

We added a `httparty` dependency using `add_runtime_dependency`. This instructs `bundle` to install **httparty**, which provides a programmatic Ruby interface to make HTTP requests.

> The string `'~> 0.13'` indicates we want the latest possible version in the 0.13 minor range. That is, if we have version 0.13.6 and version 0.13.7 comes out, we want to use that, but we don't want version 0.14. This is called **semantic versioning**.

Create `kele.rb`:

```
$ md lib
$ touch lib/kele.rb
```

We place the code for a gem within the `lib` directory. Gem conventions are to have one Ruby file with the same name as the gem (in this case `kele.rb`), which gets loaded when we call `require './lib/kele'`.

Initialization

With the skeleton of `Kele` created, add an initialize method that creates a new `Kele` client authorized with a username and password. The client can be used as follows:

```
$ irb
>> require './lib/kele'
=> true
>> Kele.new("jane@gmail.com", "abc123")
```

In `initialize`, populate two instance variables:

- Bloc's base API URL: `https://www.bloc.io/api/v1`
- The user's authentication token, which can be retrieved from **the sessions endpoint**.

To retrieve the authentication token `include HTTParty` in `Kele`, use `self.class.post`, and pass in the sessions URL along with username and password. See the **HTTParty repository** for examples.

Test Your Code

Test initializing `Kele` in IRB to ensure that:

- You retrieve and store the authentication token when passing valid credentials
- An appropriate error is raised when passing invalid credentials

How would you rate this checkpoint and assignment? ☺ ☹

3 Retrieve Users

> As a user, I want to retrieve the current user as a JSON blob
> **Difficulty Rating**: 2

Retrieving the Current User

Retrieve the **current user** from the Bloc API by defining `get_me` which can be used as follows:

```
$ irb
>> require './lib/kele'
=> true
>> kele_client = Kele.new("jane@gmail.com", "abc123")
>> kele_client.get_me
```

Pass `auth_token` to the request to properly authenticate against the Bloc API. Pass the `auth_token` via HTTParty's `headers` option:

```
response = self.class.get(url, headers: { "authorization" => @auth_token })
```

HTTParty requests return a **response object** with the data accessible via the **#body method**. This information is a JSON String. Add the **json gem** as a runtime dependency and use the **#parse method** to convert the user data to a Ruby hash.

Test Your Code

Test `Kele` in IRB to ensure that:

- You retrieve your own user data
- You convert your user data to a Ruby hash

How would you rate this checkpoint and assignment? ☺ ☹

4 Mentor Availability

> As a user, I want to retrieve a list of my mentor's availability **Difficulty Rating**: 3

Body

Retrieve a list of **a mentor's available time slots** for the current user from the Bloc API by defining `get_mentor_availability` which can be used as follows:

```
$ irb
>> require './lib/kele'
=> true
>> kele_client = Kele.new("jane@gmail.com", "abc123")
>> mentor_id = 99
>> kele_client.get_mentor_availability(mentor_id)
```

You will need to include the mentor id in the URL. Find your mentor's id in the user data returned by `get_me`. Pass `auth_token` to the request to properly authenticate against the Bloc API.

Convert the JSON response to a Ruby array.

Test Your Code

Test `Kele` in IRB to ensure that:

- You retrieve a list of your mentor's available time slots
- You convert your mentor's available time slots data to a Ruby array

How would you rate this checkpoint and assignment? ☺ ☹

4. Mentor Availability

Assignment | **Discussion** | **Submission**

5 Roadmaps and Checkpoints

> As a user, I want to retrieve roadmaps and checkpoints **Difficulty Rating**: 3

Retrieving Roadmaps

Retrieve **roadmaps** with their associated sections and checkpoints by defining `get_roadmap` to be used as follows:

```
$ irb
>> require './lib/kele'
=> true
>> kele_client = Kele.new("jane@gmail.com", "abc123")
>> roadmap_id = 99
>> kele_client.get_roadmap(roadmap_id)
```

> Your roadmap id is included in the object returned from calling the `get_me` method defined in the "Retrieve Users" checkpoint of this project.

Convert the JSON response to Ruby.

Retrieving Checkpoints

Retrieve **checkpoints** with their associated body and assignment by defining `get_checkpoint` to be used as follows:

```
$ irb
>> require './lib/kele'
=> true
>> kele_client = Kele.new("jane@gmail.com", "abc123")
>> checkpoint_id = 99
>> kele_client.get_checkpoint(checkpoint_id)
```

Retrieve checkpoint ids from the `get_roadmap` response.

Refactor Roadmaps and Checkpoints

`lib/kele.rb` is starting to get cluttered. Clean it up by moving `get_roadmap` and `get_checkpoint` into a separate `lib/roadmap` module. `require` the file in `lib/kele.rb` and `include` the module.

Test Your Code

Test `Kele` in IRB to ensure that you can:

- Retrieve a roadmap and its associated sections and checkpoints
- Retrieve a checkpoint and its associated body and assignment
- Continue to use `get_roadmap` and `get_checkpoint` after moving them into a module

How would you rate this checkpoint and assignment? ☺ ☹

6 Messaging

> As a user, I want to retrieve a list of my messages, respond to an existing message, and create a new message thread
> **Difficulty Rating**: 4

Retrieving Messages

On the Bloc platform all messages belong to a message thread. Retrieve **all message threads** for the current user by defining `get_messages`. The all message threads endpoint returns message threads paginated with 10 threads per page and a total count of all threads. Either return the first page of messages or a specified page.

```
$ irb
>> require './lib/kele'
=> true
>> kele_client = Kele.new("Hannah.McExample@gmail.com", "abc123")
>> kele_client.get_messages(1) # returns the first page of message threads
>> kele_client.get_messages # returns all message threads
```

Convert the JSON responses to native Ruby objects.

Creating Messages

Add a `create_message` method that creates a new message on the Bloc platform. Use the **create message endpoint** to create a new message and thread.

Test Your Code

Test `Kele` in IRB to ensure that you can:

- Retrieve all messages for the current user
- Create a new message and thread

How would you rate this checkpoint and assignment? ☺ ☹

7 Checkpoint Submission

> As a user, I want to submit checkpoint assignments **Difficulty Rating**: 3

Checkpoint Submissions

Add a `create_submission` method that creates a new Bloc checkpoint submission on the Bloc platform. Use the **create checkpoint submission endpoint** to create a new submission. Checkpoint submissions are tied to your account via an `enrollment_id`, which is included in your user information. Implement `create_submission` so that it can be used as follows:

```
$ irb
>> require './lib/kele'
=> true
>> kele_client = Kele.new("Hannah.McExample@gmail.com", "abc123")
>> kele_client.create_submission(checkpoint_id, assignment_branch, assignment_commit_
```

Test Your Code

Test `Kele` in IRB to ensure that you can:

- Create a new checkpoint submission

How would you rate this checkpoint and assignment?

7. Checkpoint Submission

Assignment | Discussion | Submission