|  |  |  |
| --- | --- | --- |
|  |  | Crowd Microservices  Client Getting Started Guide |

4/9/2019

Crowd Microservices is the first microtask crowdsourcing system for implementing microservices, a novel behavior-driven development workflow for crowdsourcing programming which offers workers immediate feedback on their contributions, and a cloud-based IDE for building microservices.

In the rest of this document we offer a brief overview of Crowd Microservices and a guide for creating and deploying projects using Crowd Microservices. Throughout, we use “clients” to denote those who are creating and deploying Crowd Microservices projects.

## Client-Request

Crowd Microservices applies behavior-driven development to implementing microservices. Microservices are first described by a client through the Client-Request page. Clients define a set of endpoints describing HTTP requests which will be handled by the microservice. Each endpoint is defined as a function, specifying an identifier, parameters, and a description of its behavior. As endpoints may accept complex JSON data structures as input and generate complex JSON data structures as output, clients may also describe a set of abstract data types (ADTs). Each ADT describes a set of fields for a JSON object, assigning each field a type which may be either a primitive or another ADT. In defining endpoints, clients may specify the expected data by giving each parameter and return value a type.

Some hints for working with the Client-Request page:

1. To create a client request, use this link: <http://crowdcode5.herokuapp.com/clientRequest> . Since it needs authentication, the first landing it redirects you to authentication page. After logging in successfully, you may need to click the link a second time to reach the client request page (this is an issue we hope to address in a new version).
2. Here is an example of a function created through the Client Request and the resulting endpoint this will generate:

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

1. ADTs are specified using a JSON format. The “endpoint name” should consist only of a valid JavaScript function identifier. This will be used to generate an endpoint. For example, if you define the endpoint name as “fetchTodoBasedOnStatus” and parameters “userId”and “status”, the deployed endpoint will be “/fetchTodoBasedOnStatus” and “req.query.userId” and “req.query.status” will be parameters.
2. Third party APIs (a checkbox you can select for any function) are an advanced feature you should not need to use for simple projects. This feature lets you define functions in a GitHub project (e.g., functions for persisting data) and surface these functions to crowd workers. By default, there are 5 functions you may choose to expose to the crowd (see below) to enable persistence using a JSON datastore. Through this API, workers can store, update, and delete JSON objects in a persistence store. Workers may use any of these API functions when working with functions and unit tests in the Implement Function Behavior microtask. Any schema-less persistence store may be used as an implementation for this API. In our IDE, a development version used by workers simulates the behavior of a persistence store within the browser and clears the persistence store after every test execution. In the production version used after the microservice is deployed, the persistence API is implemented through a Firebase store. Their names and parameters are listed here. Please contact the Crowd Microservices developers if you’d like to create any API function other than the default functions for further instructions.

|  |
| --- |
| **SaveObject**  It inserts an object in the database. It returns 'Duplicate' if it is already persisted in the database.  **Parameters**  todo Todo  **Return:** Todo |
| **FetchObject**  Fetch the object from the database. It returns null if the desired object does not exist.  **Parameters**  objectId Number  **Return:** Todo |
| **DeleteObjectImplementation**  Delete an object from the database. It returns null if the desired object does not exist.  **Parameters**  todo Todo  **Return:** Todo |
| **UpdateObject**  Update an object in the database. It returns null if the desired object does not exist.  **Parameters**  todo Todo  **Return:** Todo |
| **FetchAllObjects**  Get the list of objects in the database.  **Parameters**  userId String  **Return:** Todo[] |

1. After you have finished defining the microservice in the Client-page, the project is created. After you landed in the <http://crowdcode5.herokuapp.com/> , In the welcome page at the end Crowd Microservices has a "Get Started" button which route you to the defined project which you defined in the client request (the name you choose in the first input text in the Client-request as your project name) . For instance, you created a project with name “todo” in your client request, so when you click on the "Get Started" button, you must route to URL like <http://crowdcode5.herokuapp.com/todo> . The pattern would be <http://crowdcode5.herokuapp.com/_PROJECT_NAME> . Every time you create a new project in the Client-request you must correct the URL that “Get Started” button routes you.
2. For working with the system, you need at least two workers. It makes sense for improving the quality of code and avoiding malicious activity the contribution reviewed by another worker. When testing the environment, we usually simulate two workers by using two browsers with two separate accounts.

## Microtasks

After a client has completed a client request, they may then submit the client request to generate a new Crowd Microservices project. Submitting a client request generates an initial set of microtasks, generating an Implement Function Behavior microtask for each endpoint function. The crowd may then log into the project to begin completing microtasks. As workers complete microtasks, additional microtasks are automatically generated by the system to review contributions, continue work on each function, and implement any new functions requested by crowd workers.

Crowd Microservices automatically generates microtasks based on the current state of submitted work. After a client request defines endpoints, the system automatically generates a function and microtask to begin work on each. After an Implement Function Behavior microtask is submitted, the system automatically creates a Review microtask. After a Review microtask is submitted, an Implement Function Behavior is generated to continue work, if the contributor has not indicated that work is complete. If a review of an Implement Function Behavior contribution indicates issues that need to be fixed, a new Implement Function Behavior microtask is generated, which includes the issue and an instruction to fix it. After a microtask is generated, it is added to a queue. When a worker fetches a microtask, the system automatically assigns the worker the next microtask and removes it from the queue.

## Microservices Deployment

After the crowd finishes the implementation of a microservice, the client may choose to create and deploy the microservice to a hosting provider. Invoking the Deploy command first creates a new node.js GitHub project which includes each function implemented by the crowd. For endpoint functions, the environment automatically generates an HTTP request handler function for the endpoint. Each endpoint then contains the implementation of the function defined by the crowd.

Next, this GitHub project is deployed to a hosting site. In Crowd Microservices, projects are deployed to the Heroku hosting site. A new project instance is created, and the project deployed. After this process has been completed, anyone may use the completed microservice by making HTTP requests against the publicly available microservice.

Here are some hints for deploying projects with Crowd Microservices:

1. After you finish defining the endpoints in the client request, Crowd Microservices needs to be restarted. You can contact Emad to restart the server, which is hosted on Heroku, or we can give you permission for doing that yourself through the GUI dashboard in Heroku. If you want to do it yourself, we would need your Heroku username to give you permissions. <https://devcenter.heroku.com/>
2. After the crowd finishes their work, you can push their contribution to a GitHub repo. Crowd Microservices offers a template Node.js project (similar to <https://github.com/eaghayi/crowd-todo-microservices>) to create as an empty project on your GitHub account. Before invoking the push command, you should clone this project to create a new repo. it needs your GitHub user name, repository name, and a token. You might send them to us, and we will add your information as a valid user in Crowd Microservices.
3. To push the code created by the crowd to GitHub, execute the following request [http://crowdcode5.herokuapp.com/{project\_name\_you\_selected\_in\_client\_Request}/deploy](http://crowdcode5.herokuapp.com/%7bproject_name_you_selected_in_client_Request%7d/deploy)

## Additional Resources

1. Paper: <http://mason.gmu.edu/~eaghayi/papers/Eaghayi2019TSE.pdf>
2. Demo video on Crowd Microservices: <https://youtu.be/qQeYOsRaxHc>
3. Crowd Microservices URL: <http://crowdcode5.herokuapp.com/>
4. Client-Request URL: <http://crowdcode5.herokuapp.com/clientRequest>

If you have any questions, please contact Emad at eaghayi@gmu.edu