Service Layer

Christ Mboungou 04/02/20222

Overview

Our application will use Django Rest Framework in the Backend. The Django Rest Framework is a powerful and flexible toolkit for building web APIs. Our front-end will make API calls to perform CRUD operations.

Our backend service will have four essential parts: APIView, Model, Serializer, and URL.

1. APIView: this class will allow us to use the REST framework's Request instances, not Django's HttpRequest instances. It will also return REST framework Response, instead of Django's HttpResponse.
2. Django's model is where we house information about our data. It contains field definitions and behaviors of the data we will store in the database. Each model maps to a single database table.
3. Serializer: this allows complex data such as model instances to be converted to native Python datatypes that can be quickly rendered into JSON. We are also using the Serializer to validate incoming data.
4. URL: This module helps map URL path expressions to Python functions.

Specifications:

I've listed all my routes and endpoints to my service layer. My API is live and working already, but It might not work when you test it because I'm still making changes. I'm using Heroku.

Stretch Features

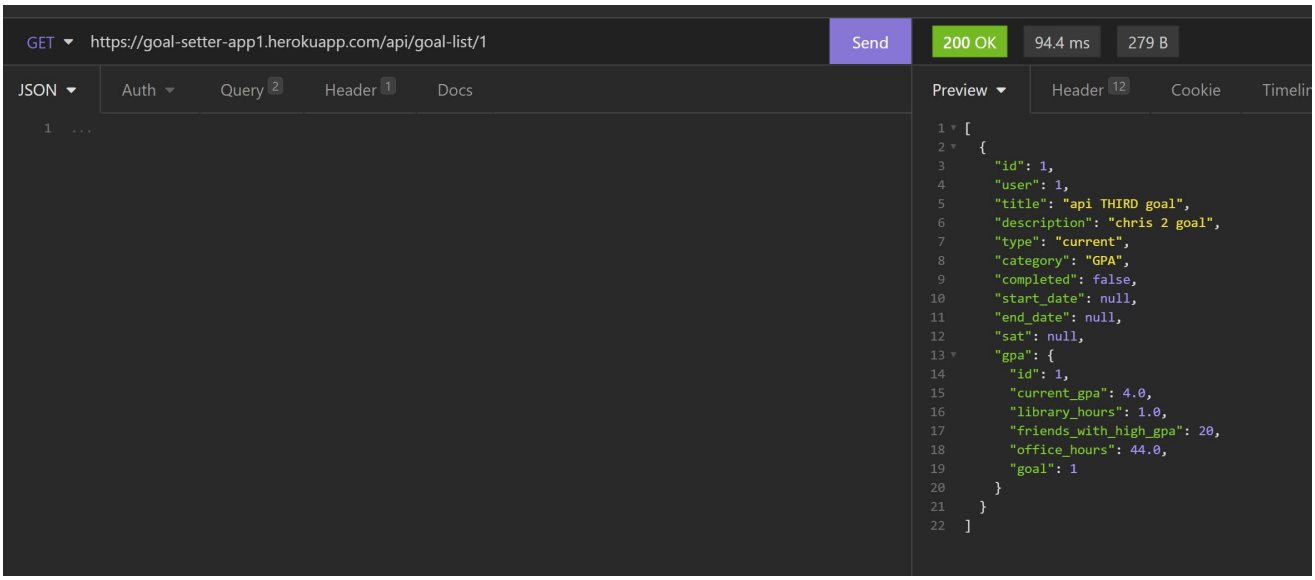
For our application, stretch features will benefit from the current architecture, and I will not be required to create a different endpoint. To scale this application, we will add more database tables and associate those tables to the goal table to query the data based on the one-to-one relationship that will be established. This is why selecting a category will be very important when creating a goal.

API Overview URL: https://goal-setter-app1.herokuapp.com/api/

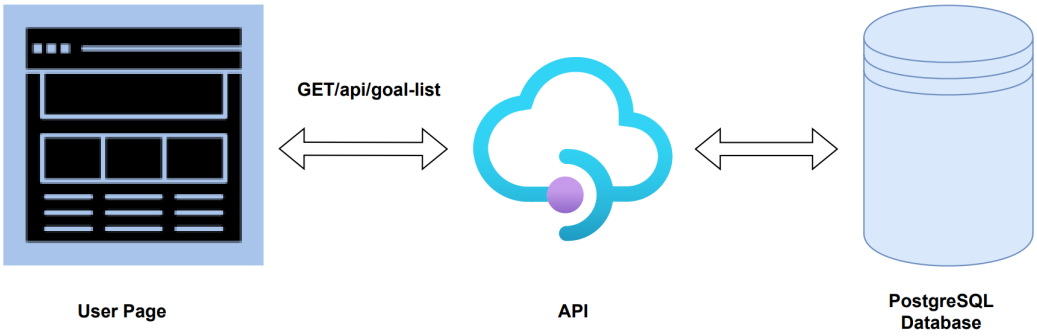
Goal, SAT, GPA Routes:

1. goal list:
   Method: GET
   URL: https://goal-setter-app1.herokuapp.com/api/goal-list/<user_id>/

   Purpose: This endpoint will be used in many different areas of our application. For instance, when a user successfully logs in, this endpoint will be called, and it will deliver all goals associated with that user. The image below will show all goals submitted by a particular user, in this case, we are using user id 1. We will also use this endpoint in the timeline. We will fetch all the goals created by the user and filter and show completed goals in a container. We will use the same endpoint and filter “type=future” indicating future goals, to show future goals in another container.
   Success response:

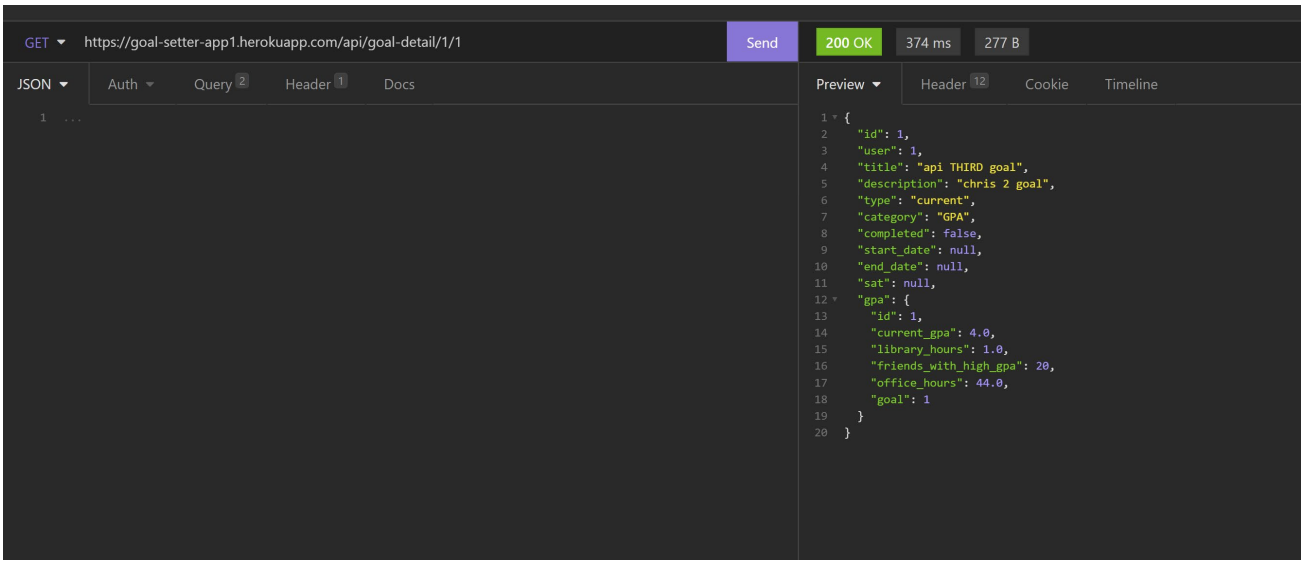


Error response: There's no way to break this URL because you will get an empty list if the user_id doesn't exist.

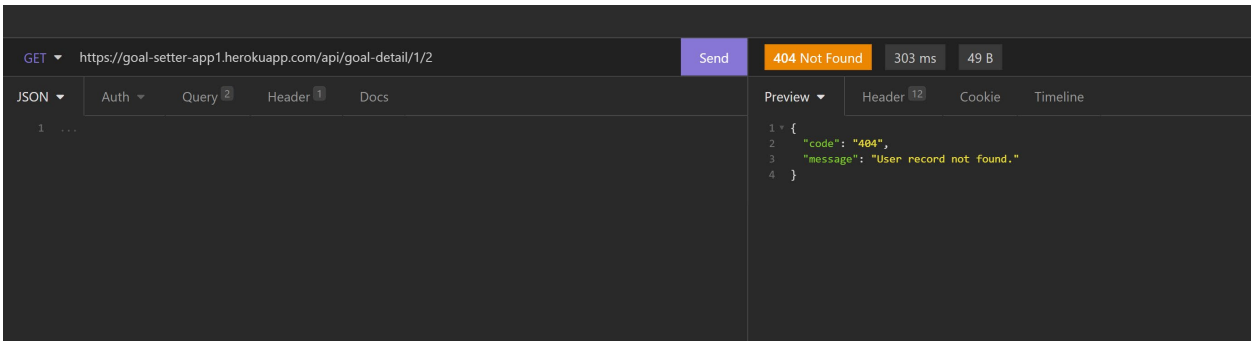


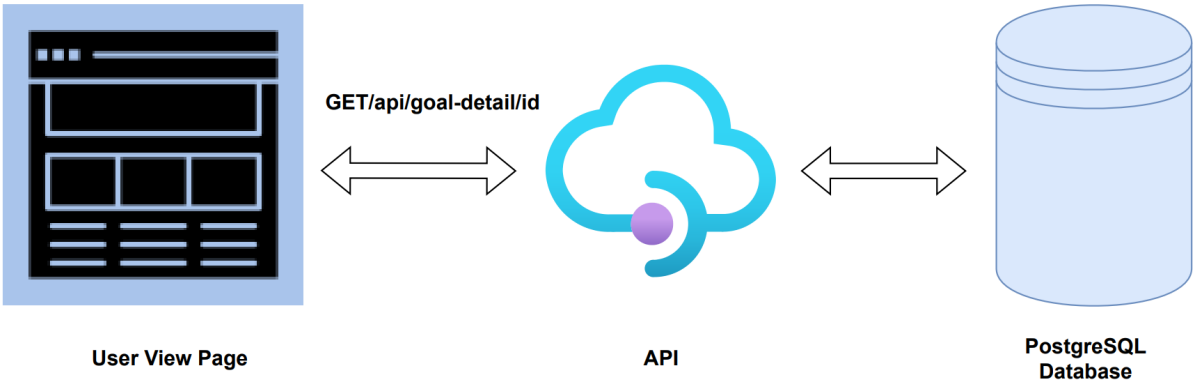
2. goal detail:
   Method: GET
   URL: https://goal-setter-app1.herokuapp.com/api/goal-detail/<user_id>/<goal_id>

   Purpose: This endpoint will allow us to fetch a single goal with all its details. Notice we include the goal_id and user_id to get back that specific goal in return. We will use this to show the goal in view mode.
   Success response:



Customized error response when trying to fetch a non-existing goal_id:

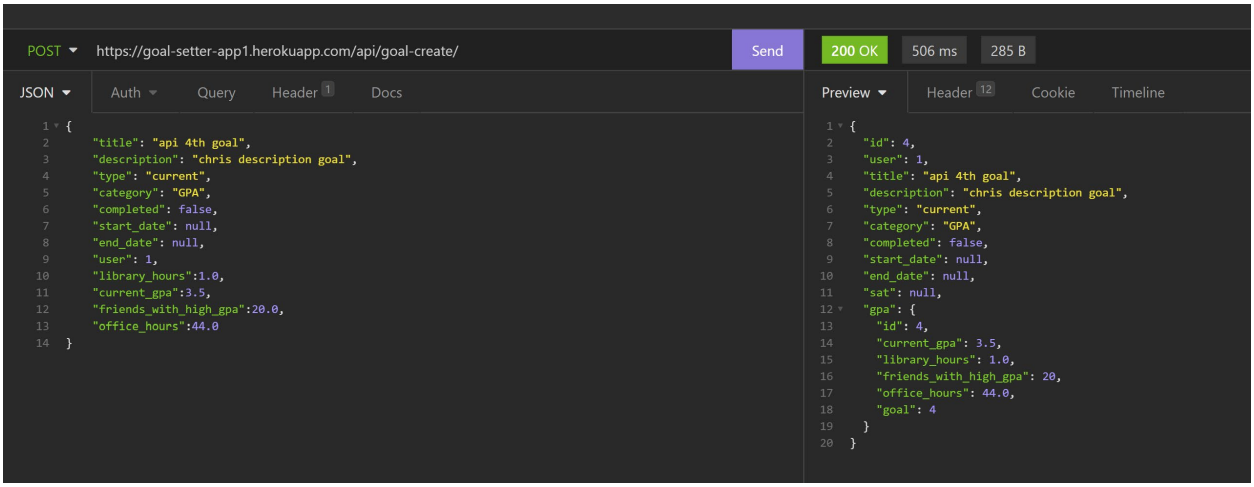




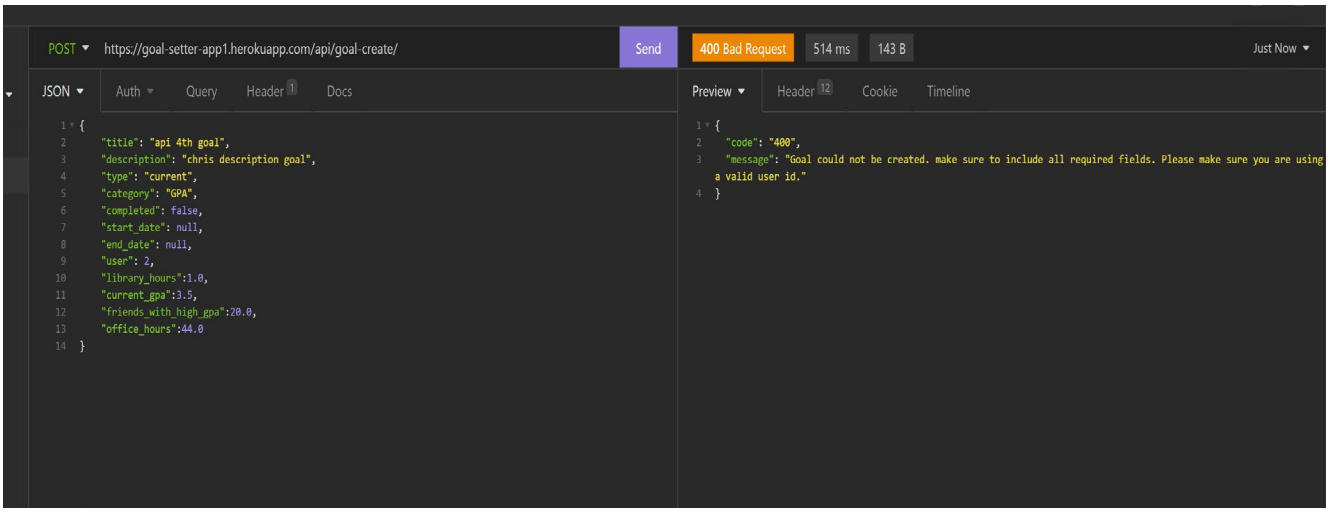
3. goal create:
   Method: POST
   URL: https://goal-setter-app1.herokuapp.com/api/goal-create/

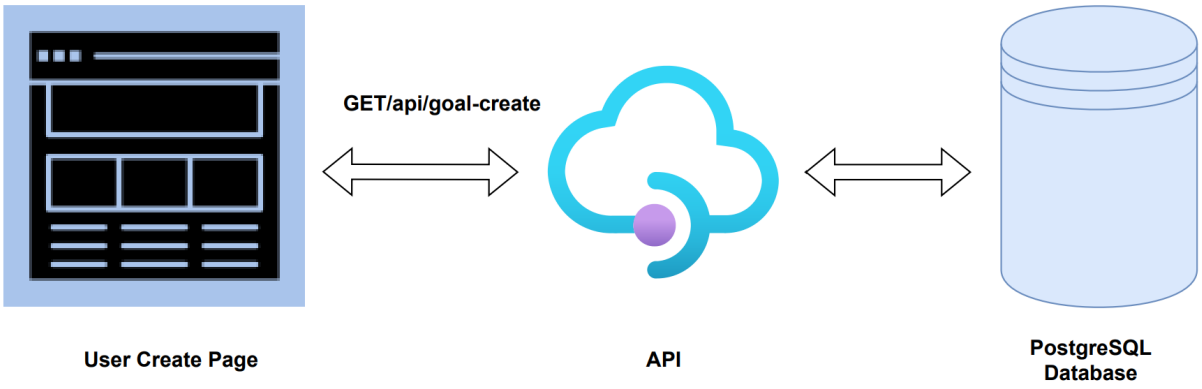
   Purpose: This endpoint will help us create new goals. Please note that the system will automatically generate a Goal id for us because we didn't pass a goal id. The system also handles the creation of SAT or a GPA record whenever a goal is created, depending on your selected category. For instance, in the picture below, the system automatically created a Goal record with id 4 and a GPA record with id 4. This is because I selected GPA as a category. This GPA data will delete on Cascade.

Success response: please notice the JSON body that I'm passing to the API.



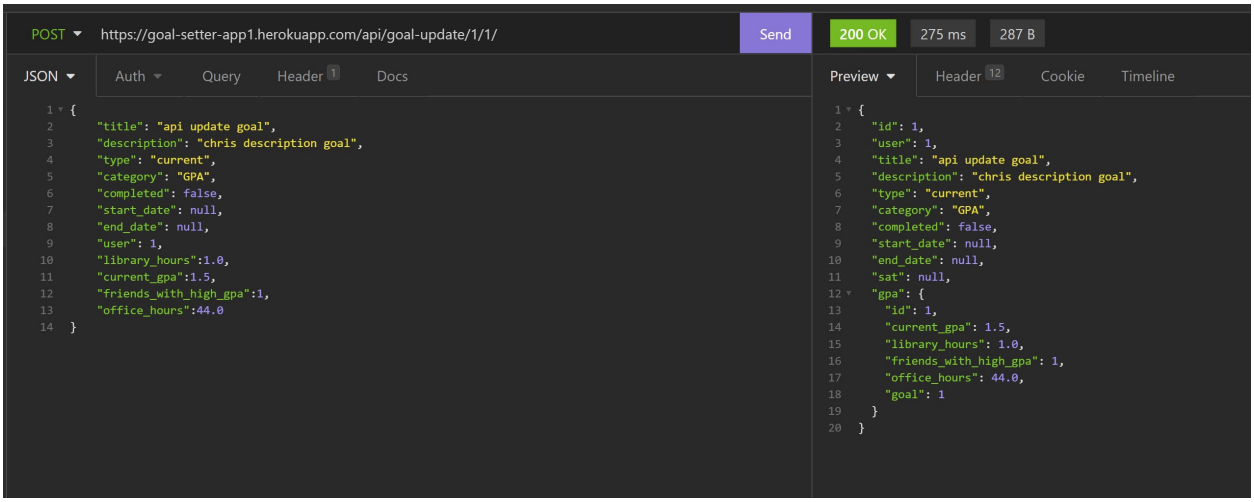
Customized error: I tried creating a goal with a nonexisting user_id.



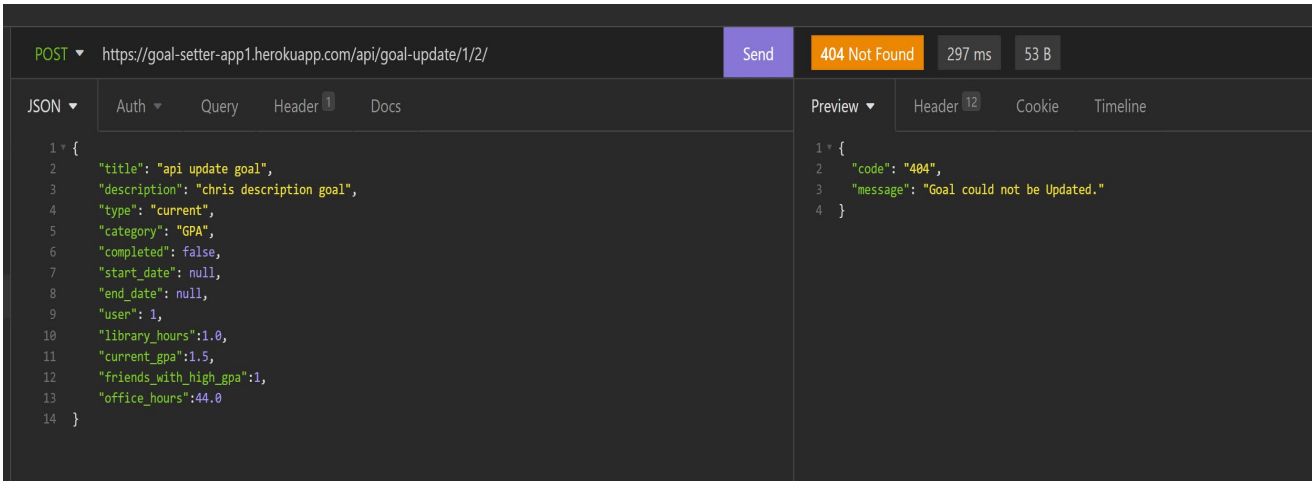


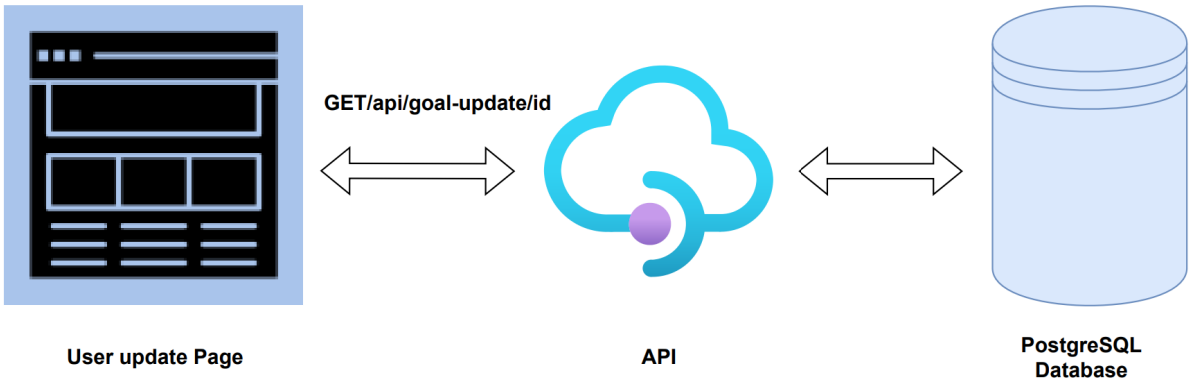
4. goal update:
   Method: POST
   URL: https://goal-setter-app1.herokuapp.com/api/goal-update/<user_id>/<goal_id>

   Purpose: This endpoint will be used to update goals, and the child records GPA or SAT. Notice we are using a POST, not a PUT or UPDATE, PATCH. I might change that later, but I'm using a post to update existing records for now.
   Success response: notice that I'm passing the user_id and the goal_id in the URL.



Custom error when trying to update a non-existing record. It’s a bad goal_id





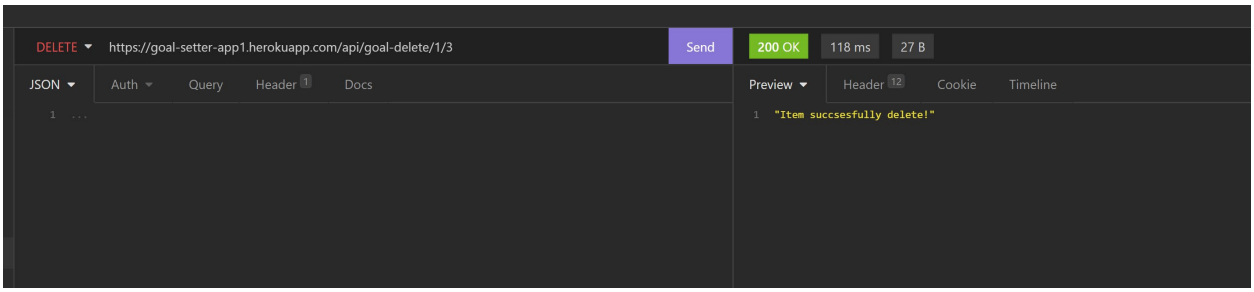
5. goal delete:

   Method: DELETE

   URL: https://goal-setter-app1.herokuapp.com/api/goal-delete/<user_id>/<goal_id>/

   Purpose: This endpoint will help delete Goal records and cascade to SAT and GPA.

Success response:



Custom error: when trying to delete a non-existing goal: