DRF PoC Web API

- https://django-poc-session-maheshbodas.herokuapp.com/auth/login/
- Use above link to access browsable Web API.
- Username :- admin (lower case) – Admin
- Password :- poctest#1 (lower case)
- Username :- editor (lower case) – Non Admin
- Password :- test#123 (lower case)

Unit testing.

- Added test cases related for Post and Get action for RiskType and Risks.
- Tested response for Admin and Non-admin users as API support role base authorization.
- Made use of Design patterns to avoid duplication of code while writing test cases.

Salient features

- Solution that allows insurers to define their own custom data model for their risks. There are no database tables called automobiles, houses, or prizes. Instead, insurers will be able to create their own risk types and attach as many different fields as they would like.
- Fields are bits of data like first name, age, zip code, model, serial number, Coverage A limit, or prize dollar amount. Basically any data the carrier would want to collect about the risk. Fields can also be of different types, like text, date, number, currency, and so forth.

Salient features (continued)

- As risk field types are user defined and not commited at time of table creation. Actual field/column value stored in table is string type. But field type sanctity is enforced using model validators. i.e User can not supply Date value where Currency is expected.
- Uniqness among RiskType names is maintained by enforcing unique constrain. But enforcing unique risk field name in similar fashion is somewhat restrictive and naive. I have tried to maintain unique field name within Risk Types and not across all Risk Type using mode validators.

Salient features (continued)

- During risk and risk field creation system checks that proper referential integrity is maintained among Risk Type and Risks. Risk Fields ensures that proper referential integrity is maintained with Risk Type and Risk Type fields.

- Risk API allows users to create RiskTypes and associated RiskTypeFields in one go with use of nested serializer.

- Risk API allows users to create Risk and associated RiskFields in one go with use of nested serializer.

- Various model validation errors within RiskType and RiskTypeFields, Risk and RiskFields are returned to client of RiskAPI in JSON format.

Salient features (continued)

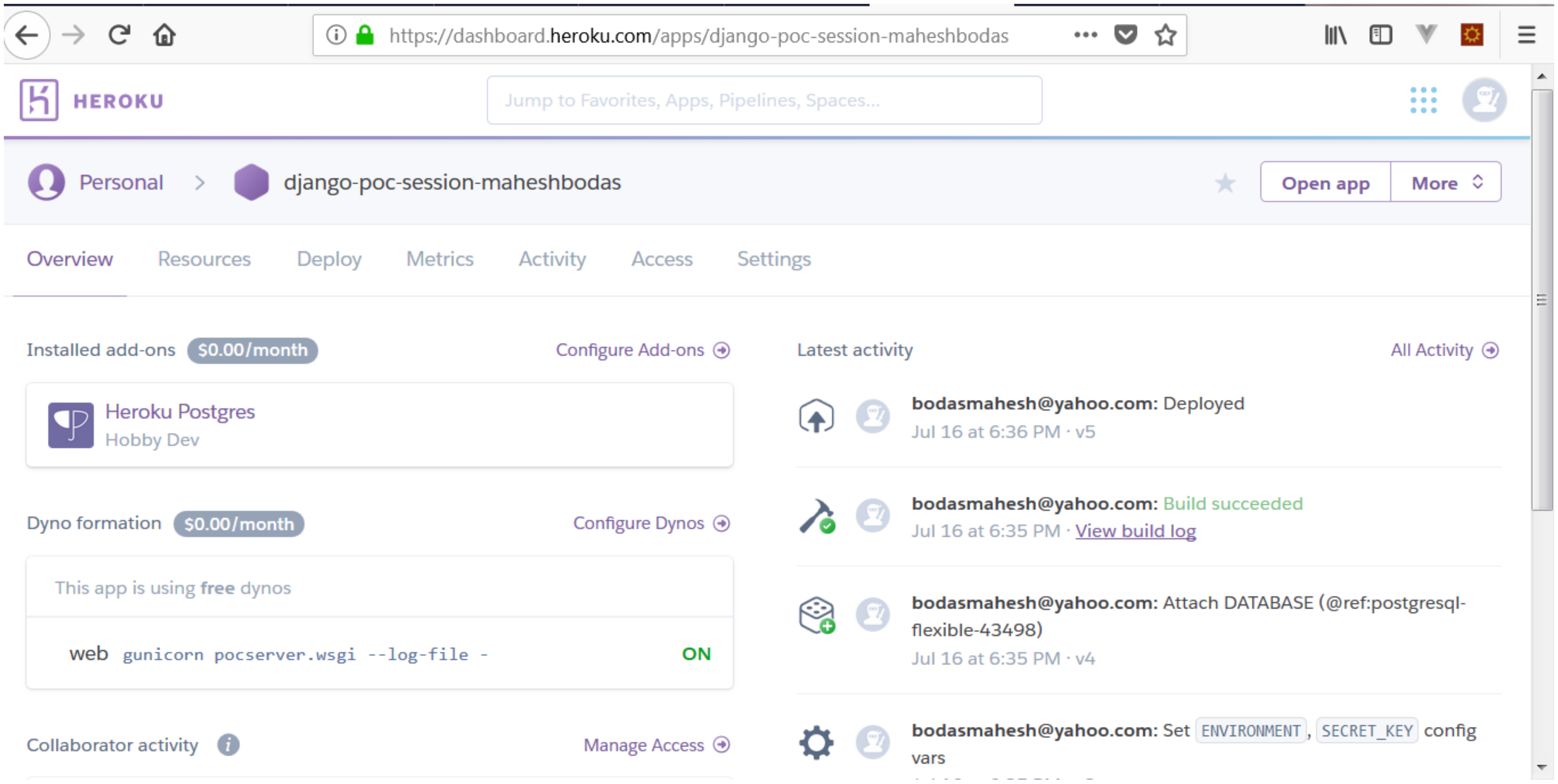
- Risk and RiskFields carry extra metadata related to RiskType and RiskTypeFields.
- On authentication front, only authenticated users can access RiskAPI. Role based authorization further enable only admin users to create/delete RiskType. Non admin users only have list and details option for RiskTypes. However they can create Risk instances based on RiskTypes.
- Session authentication is used for browsable API that runs within same session that of Web API
- Token authentication is used for seperate website that access Web API for retrieving and creating RiskTypes and Risk.

Salient features (continued)

- Used Default router to list all available links and Pastebin docs and schema information also available for API.

- One touch Heroku deploy button.

Screenshot Heroku deployment



https://dashboard.heroku.com/apps/django-poc-session-maheshbodas

Installed add-ons $0.00/month — Configure Add-ons

Heroku Postgres
Hobby Dev

Dyno formation $0.00/month — Configure Dynos

This app is using **free** dynos

```
web  gunicorn pocserver.wsgi --log-file -    ON
```

Collaborator activity — Manage Access

Latest activity — All Activity

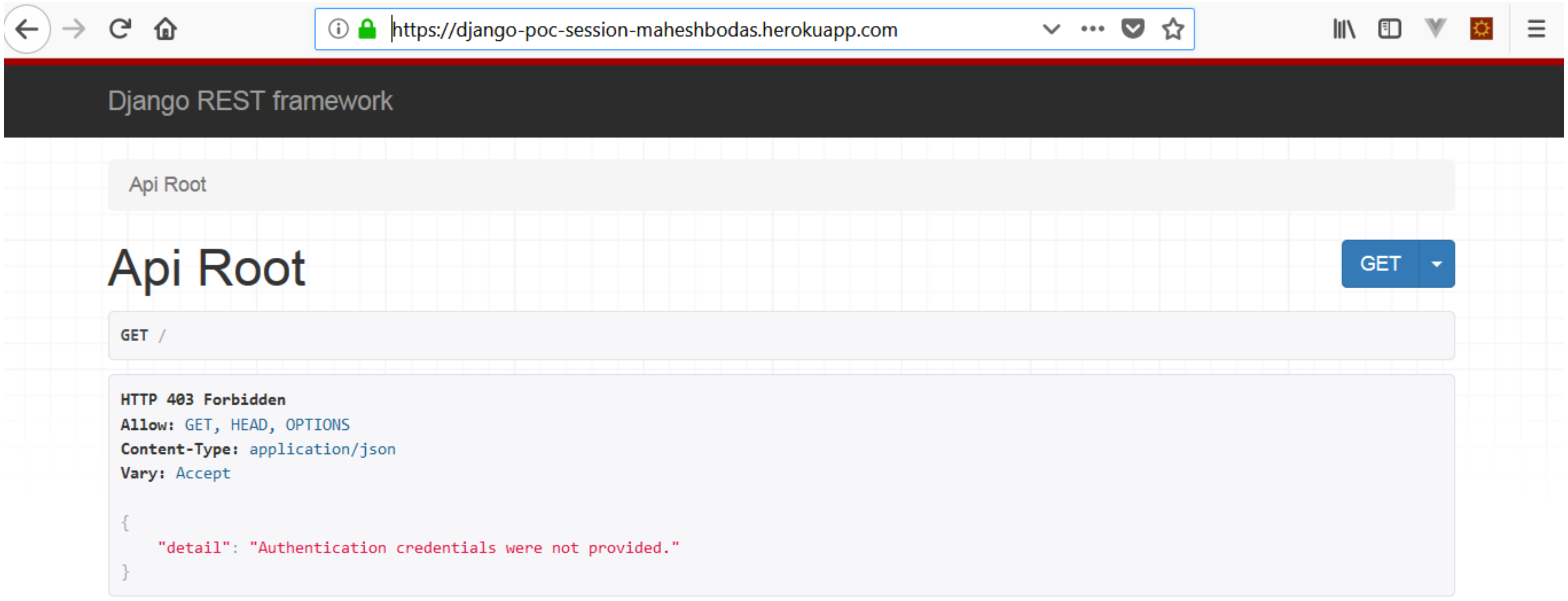
bodasmahesh@yahoo.com: Deployed
Jul 16 at 6:36 PM · v5

bodasmahesh@yahoo.com: Build succeeded
Jul 16 at 6:35 PM · View build log

bodasmahesh@yahoo.com: Attach DATABASE (@ref:postgresql-flexible-43498)
Jul 16 at 6:35 PM · v4

bodasmahesh@yahoo.com: Set `ENVIRONMENT`, `SECRET_KEY` config vars

API response when not Logged In



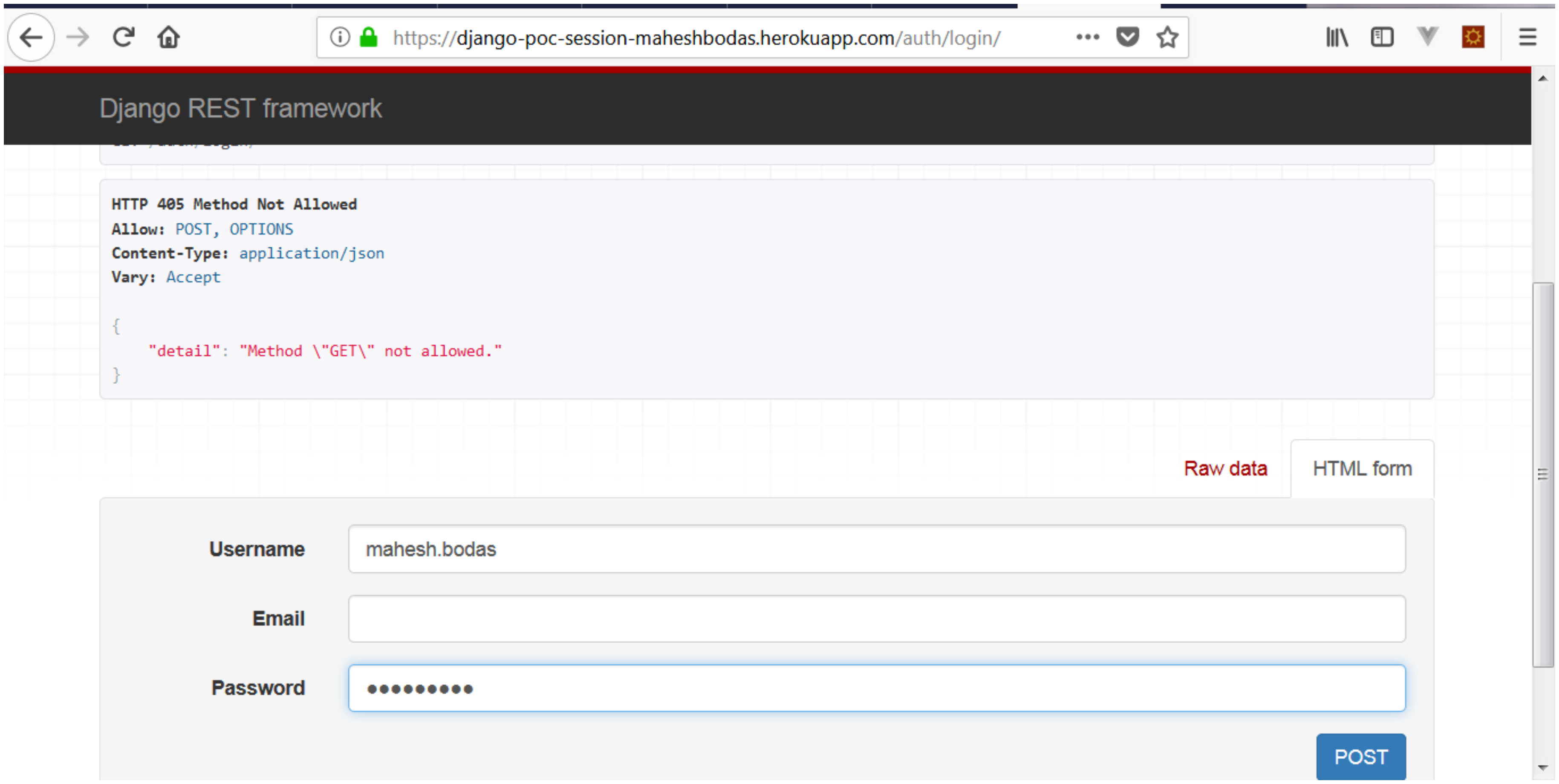
Api Root

```
GET /
```

```
HTTP 403 Forbidden
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Authentication credentials were not provided."
}
```

Enter credentials



https://django-poc-session-maheshbodas.herokuapp.com/auth/login/

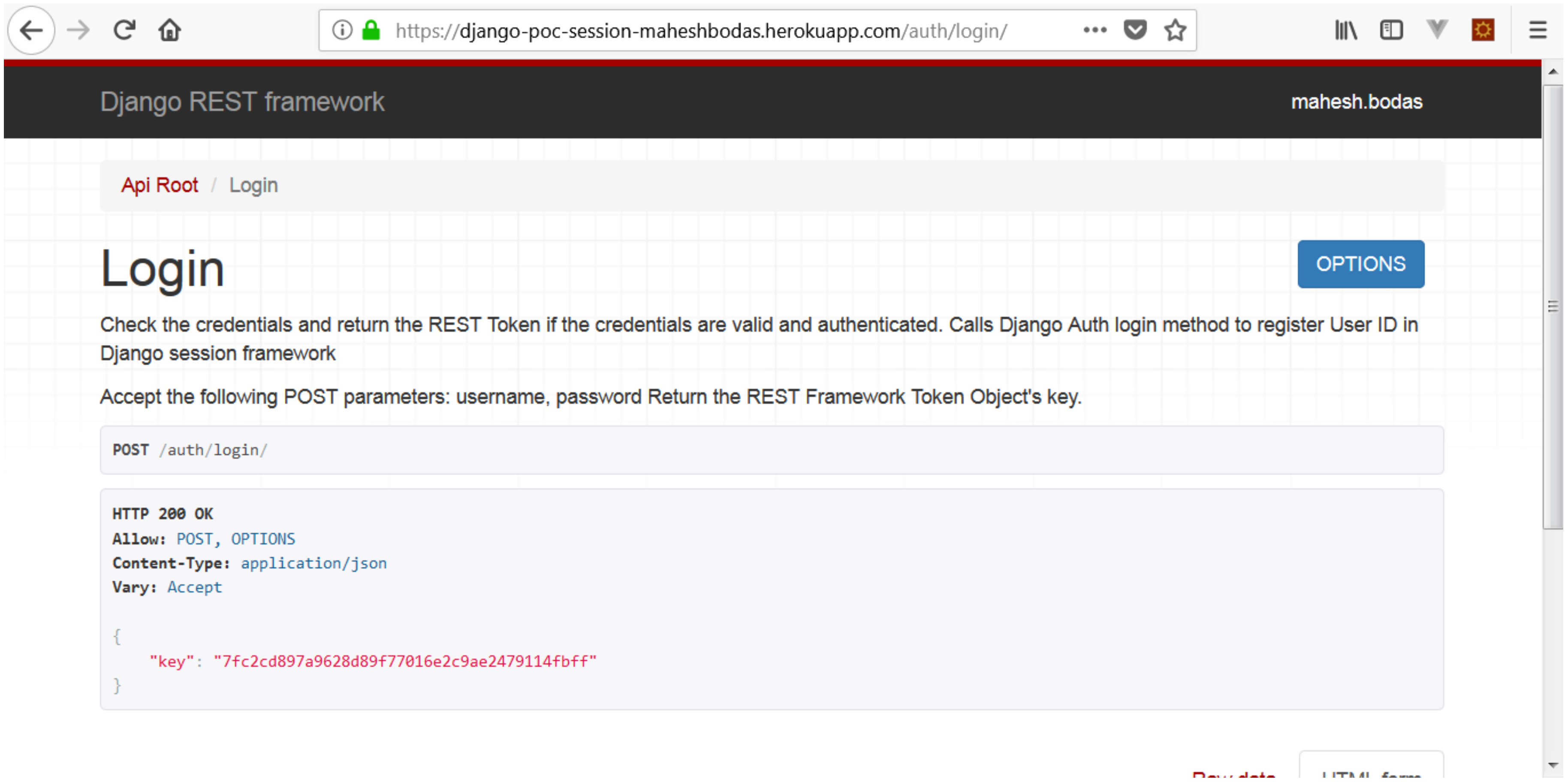
Django REST framework

```
HTTP 405 Method Not Allowed
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Method \"GET\" not allowed."
}
```

Username	mahesh.bodas
Email	
Password	●●●●●●●●●

Successful Login


https://django-poc-session-maheshbodas.herokuapp.com/auth/login/
Django REST framework
mahesh.bodas
Api Root / Login
OPTIONS

Login

Check the credentials and return the REST Token if the credentials are valid and authenticated. Calls Django Auth login method to register User ID in Django session framework

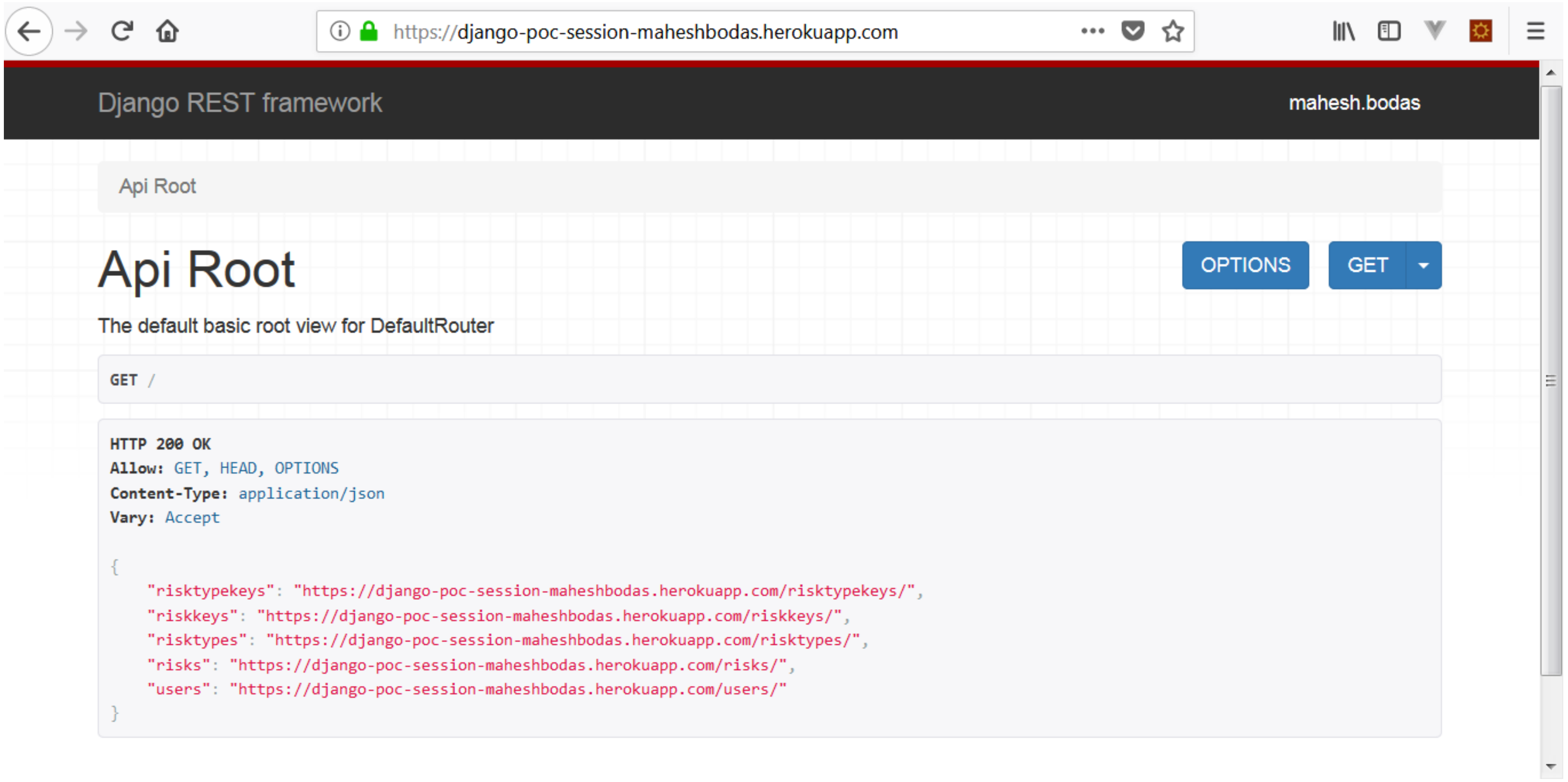
Accept the following POST parameters: username, password Return the REST Framework Token Object's key.

```
POST /auth/login/
```

```
HTTP 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "key": "7fc2cd897a9628d89f77016e2c9ae2479114fbff"
}
```

API Root shows Default Router



Api Root

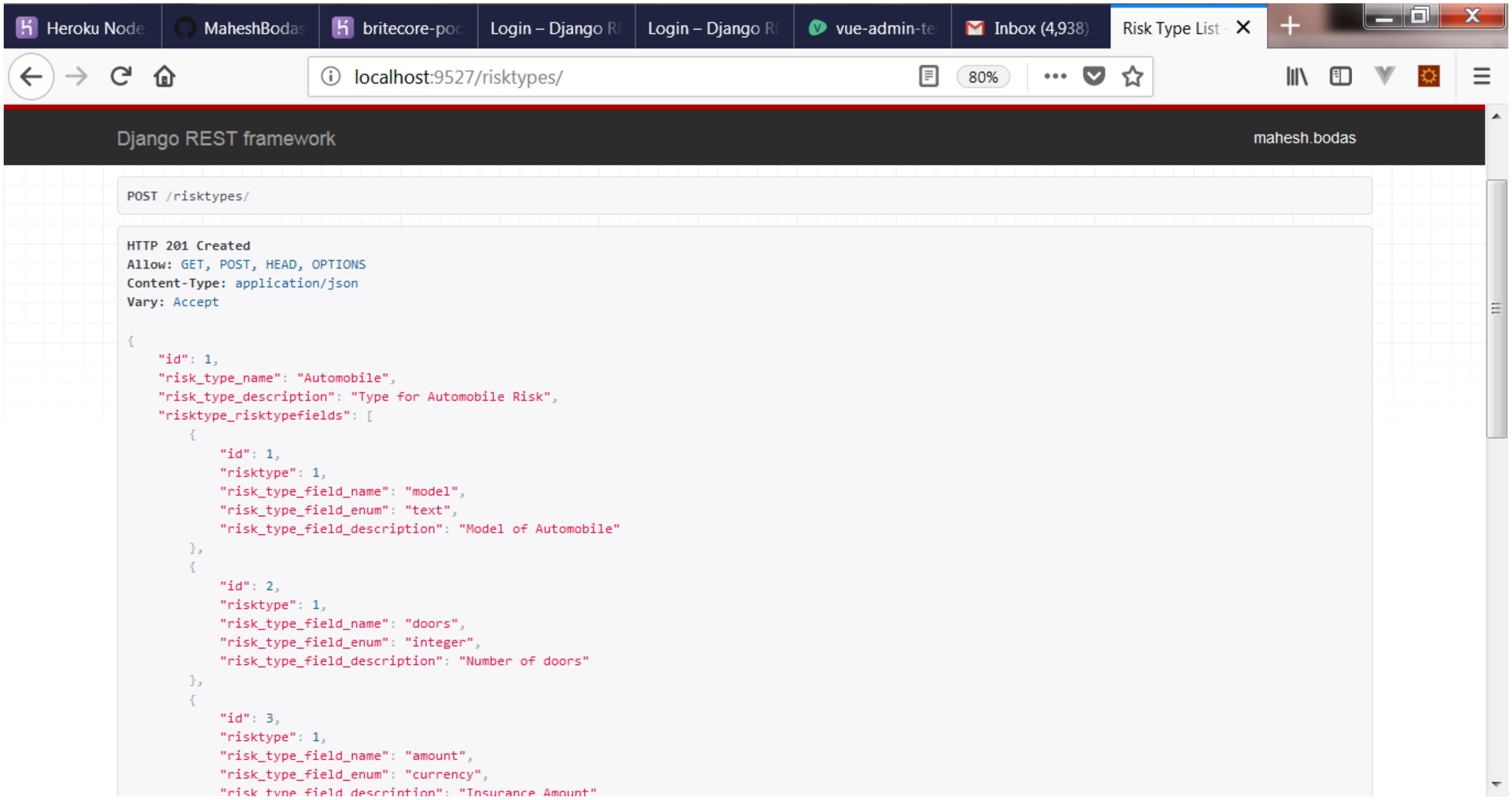
The default basic root view for DefaultRouter

```
GET /
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "risktypekeys": "https://django-poc-session-maheshbodas.herokuapp.com/risktypekeys/",
    "riskkeys": "https://django-poc-session-maheshbodas.herokuapp.com/riskkeys/",
    "risktypes": "https://django-poc-session-maheshbodas.herokuapp.com/risktypes/",
    "risks": "https://django-poc-session-maheshbodas.herokuapp.com/risks/",
    "users": "https://django-poc-session-maheshbodas.herokuapp.com/users/"
}
```

Posting RiskType successfully

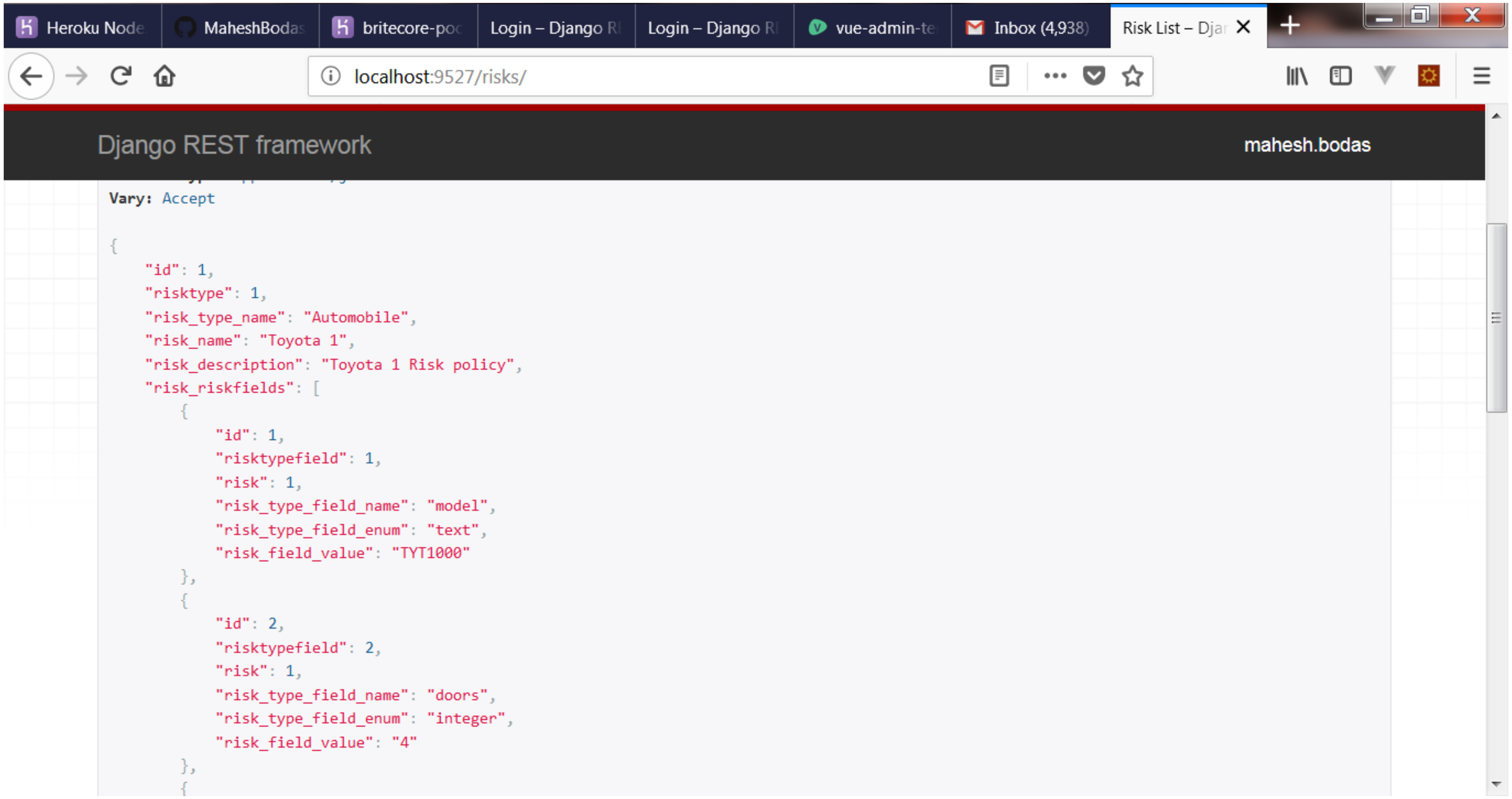


```
POST /risktypes/
```

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "risk_type_name": "Automobile",
    "risk_type_description": "Type for Automobile Risk",
    "risktype_risktypefields": [
        {
            "id": 1,
            "risktype": 1,
            "risk_type_field_name": "model",
            "risk_type_field_enum": "text",
            "risk_type_field_description": "Model of Automobile"
        },
        {
            "id": 2,
            "risktype": 1,
            "risk_type_field_name": "doors",
            "risk_type_field_enum": "integer",
            "risk_type_field_description": "Number of doors"
        },
        {
            "id": 3,
            "risktype": 1,
            "risk_type_field_name": "amount",
            "risk_type_field_enum": "currency",
            "risk_type_field_description": "Insurance Amount"
```

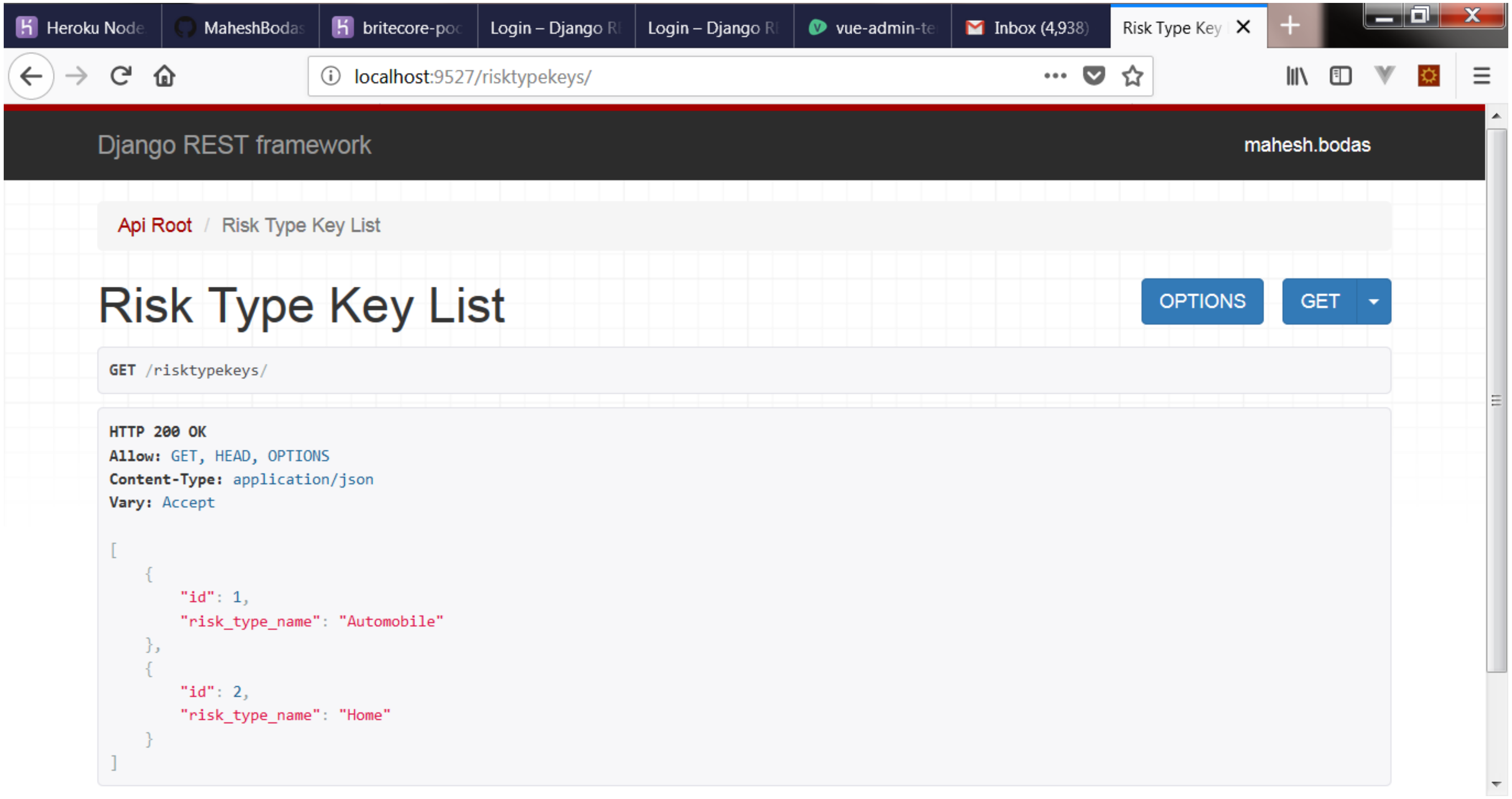
Posting Risk Instance



```
Vary: Accept

{
    "id": 1,
    "risktype": 1,
    "risk_type_name": "Automobile",
    "risk_name": "Toyota 1",
    "risk_description": "Toyota 1 Risk policy",
    "risk_riskfields": [
        {
            "id": 1,
            "risktypefield": 1,
            "risk": 1,
            "risk_type_field_name": "model",
            "risk_type_field_enum": "text",
            "risk_field_value": "TYT1000"
        },
        {
            "id": 2,
            "risktypefield": 2,
            "risk": 1,
            "risk_type_field_name": "doors",
            "risk_type_field_enum": "integer",
            "risk_field_value": "4"
        },
        {
```

Get RiskTypeKeys



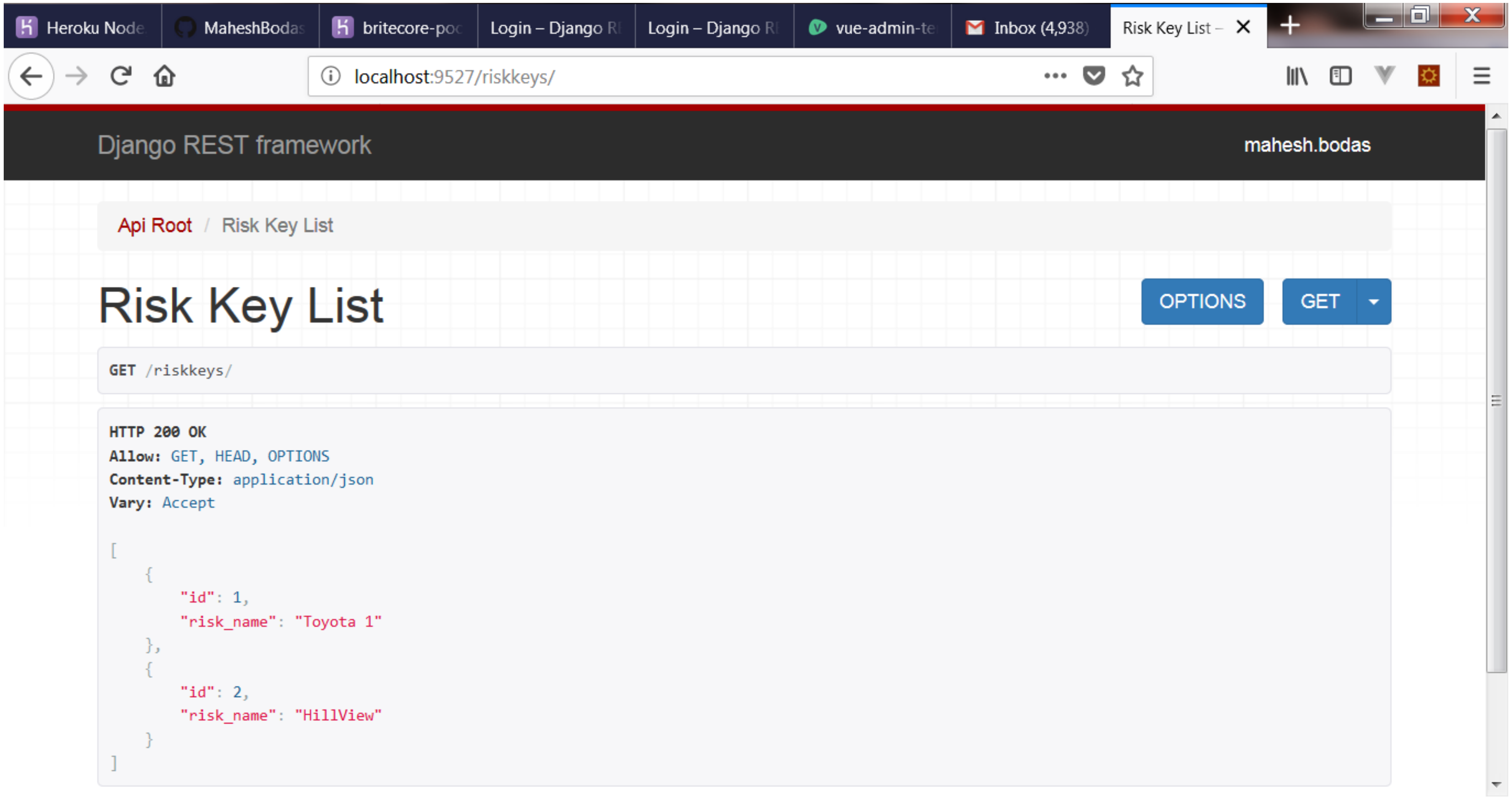
Risk Type Key List

```
GET /risktypekeys/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "risk_type_name": "Automobile"
    },
    {
        "id": 2,
        "risk_type_name": "Home"
    }
]
```

Get Risk Key List



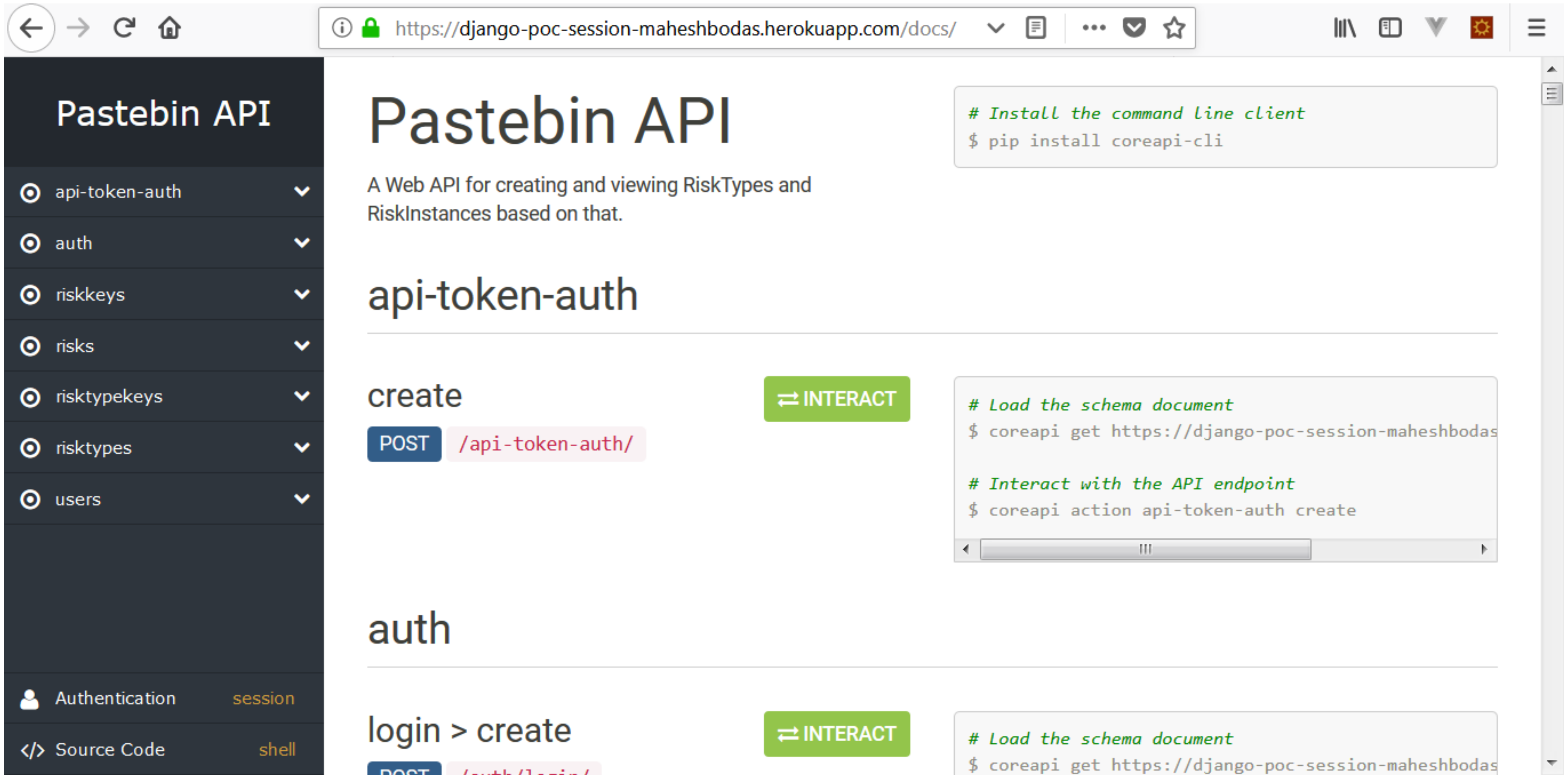
Risk Key List

```
GET /riskkeys/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "risk_name": "Toyota 1"
    },
    {
        "id": 2,
        "risk_name": "HillView"
    }
]
```

https://django-poc-session-maheshbodas.herokuapp.com/docs/



Pastebin API

A Web API for creating and viewing RiskTypes and RiskInstances based on that.

```
# Install the command line client
$ pip install coreapi-cli
```

api-token-auth

create

POST /api-token-auth/

```
# Load the schema document
$ coreapi get https://django-poc-session-maheshbodas

# Interact with the API endpoint
$ coreapi action api-token-auth create
```

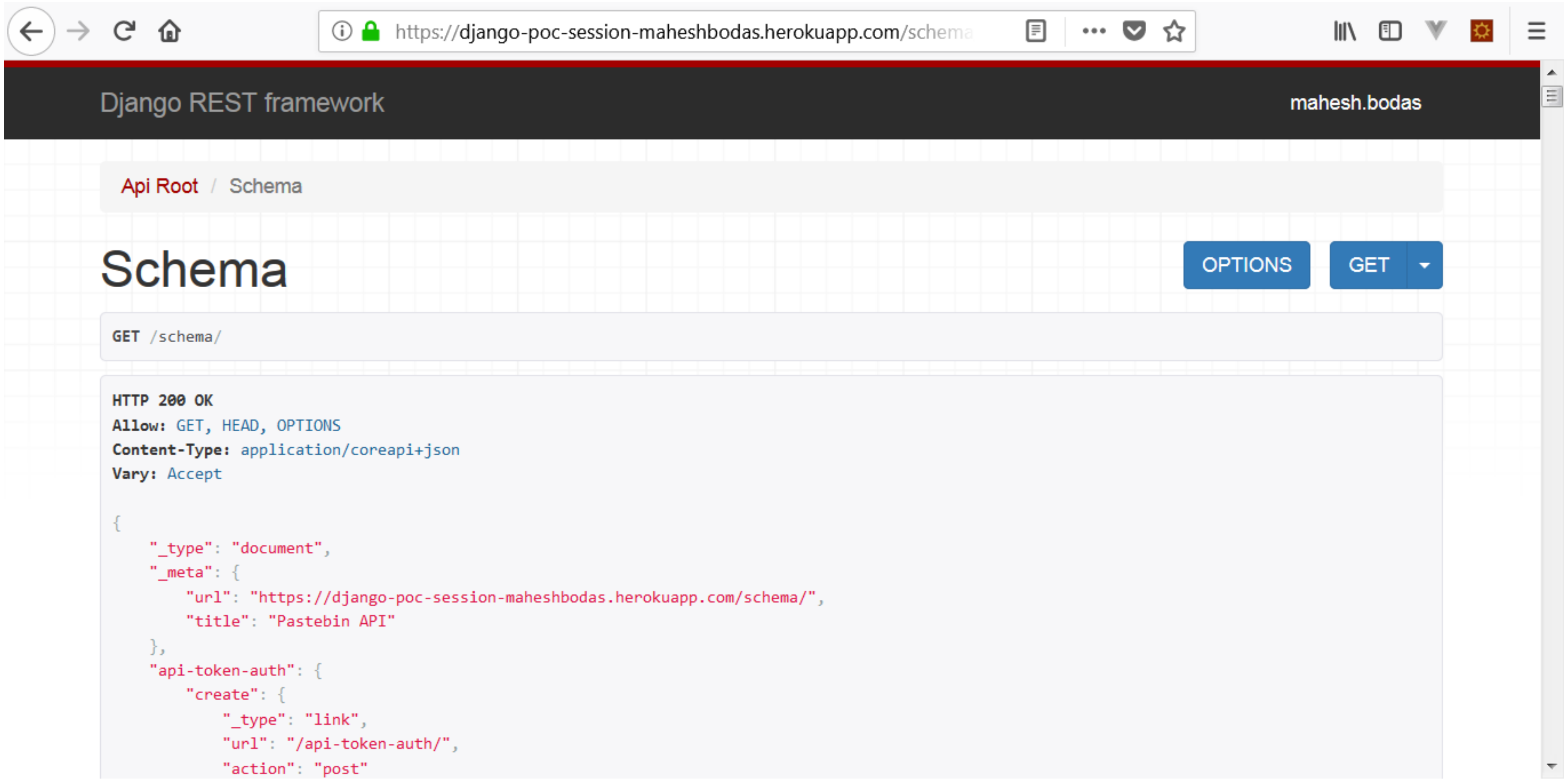
auth

login > create

POST /auth/login/

```
# Load the schema document
$ coreapi get https://django-poc-session-maheshbodas
```

https://django-poc-session-maheshbodas.herokuapp.com/schema



Schema

```
GET /schema/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/coreapi+json
Vary: Accept
```

```
{
    "_type": "document",
    "_meta": {
        "url": "https://django-poc-session-maheshbodas.herokuapp.com/schema/",
        "title": "Pastebin API"
    },
    "api-token-auth": {
        "create": {
            "_type": "link",
            "url": "/api-token-auth/",
            "action": "post"
```