TECHNICAL REPORT: IDB2

Sochi 2014 Winter Olympics

by “4 Is Better Than 6”

Members:

- Austin Hooper
- Ardhimas Kamdani
- Nickson Dalimarta

INTRODUCTION

Description:

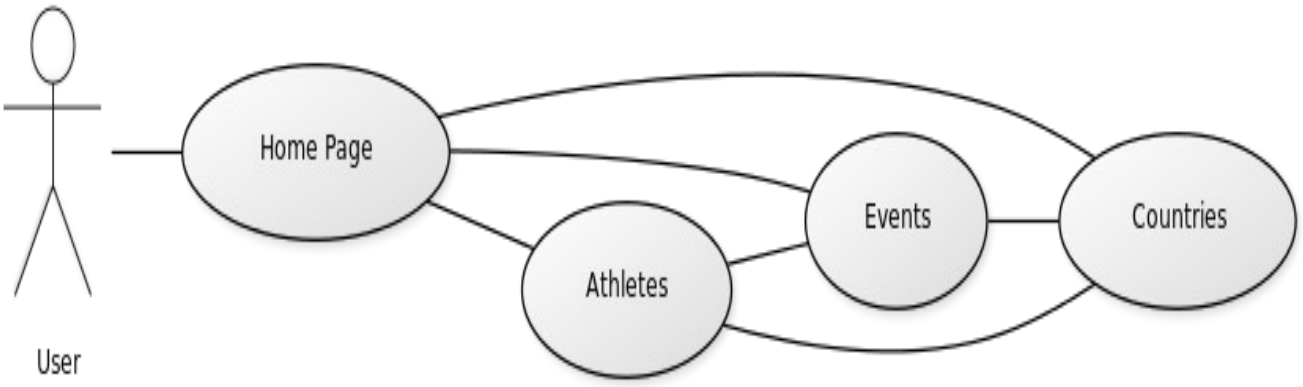
Have you ever found yourself wanting to know the events that went on during the 2014 Sochi Winter Olympics? Or the athletes that took part in said games? How about how well your favorite country did in said events? Then we have a site that is right up your alley!

Use cases (Phase 1) :

* From the Home Page a user can use the navigation bar to be directed to any Athlete, Event, and Country page.

* Navigation Bar exists in all pages. In phase 1, because of limited pages, a user may go to any page by making use of the navigation bar.

Use Case Diagram



DESIGN

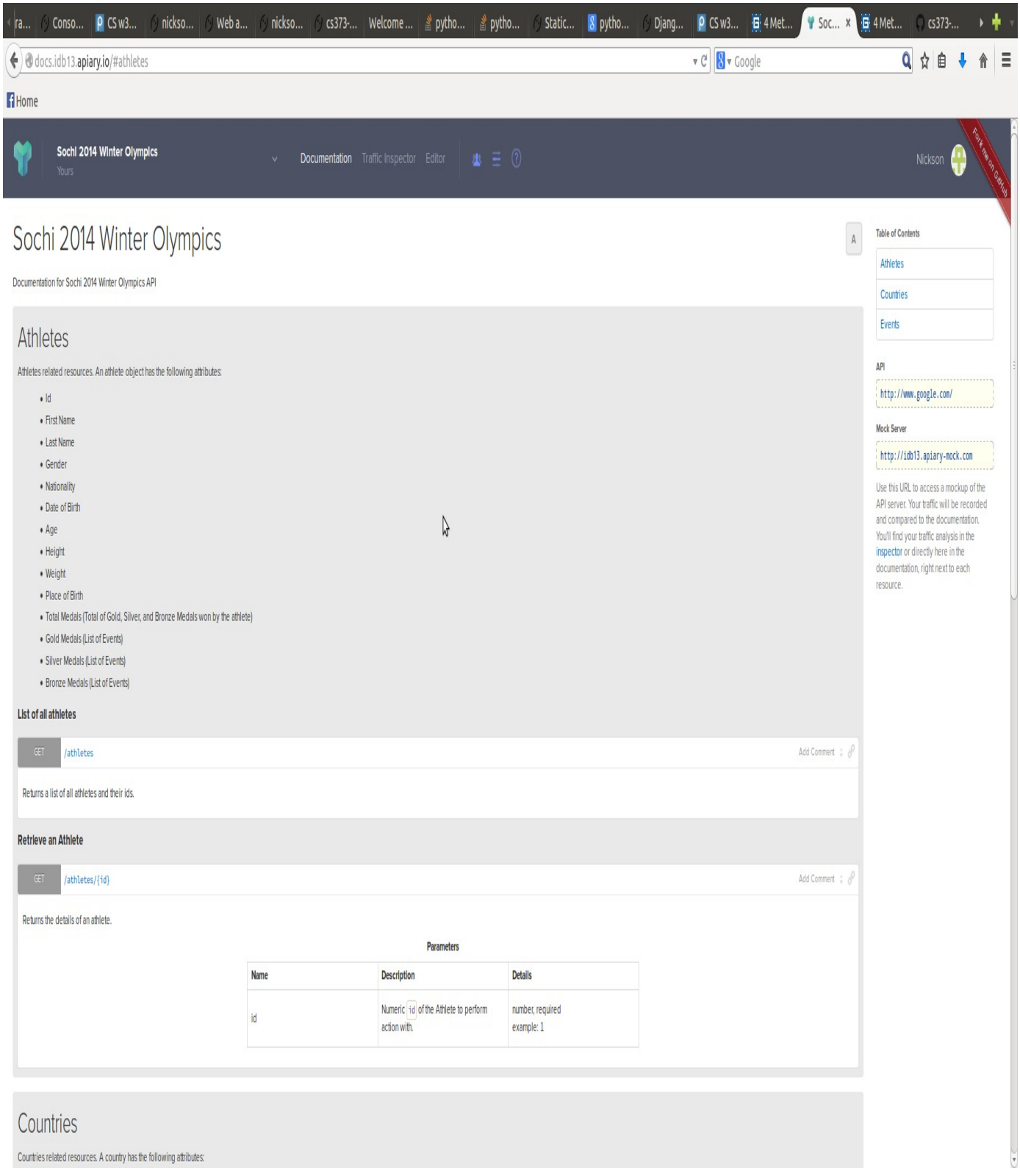
RESTful API:

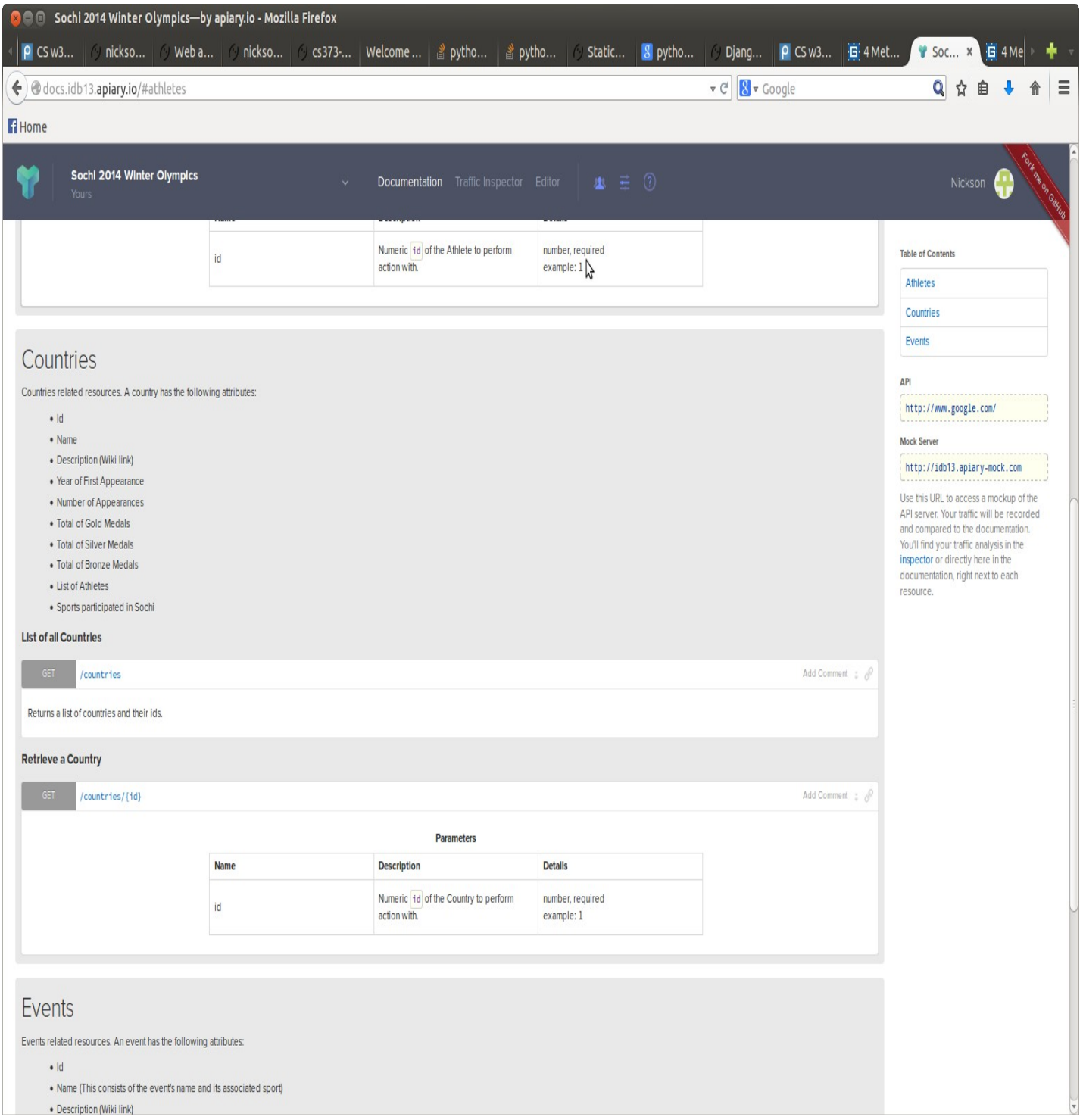
API exists for retrieving information of all available athletes/ events/ countries. Due to the limited functionality, there are only 2 GET methods for each model. One to retrieve a list of all available datas for a model. Each element is a key-value pair of the model's id and name, separated by an equal sign. The other method is used retrieve the details of an object of a specified id (the only parameter of this method). A user who would like to get the information on Ireen Wust, for example, may first use the first method to get this list

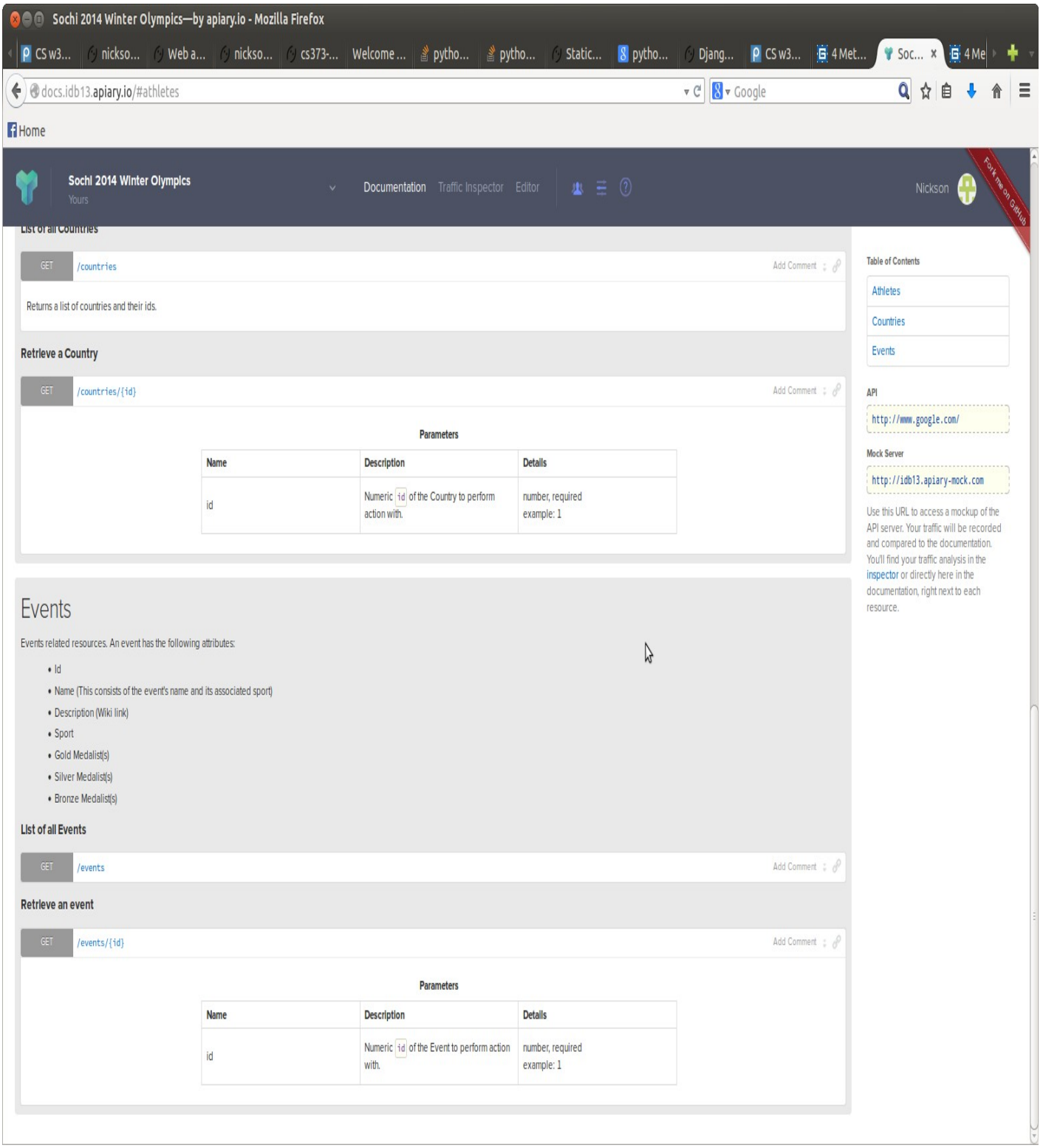
```
"ID=Athletes":["1=Meryl Davis","2=Tobias Arit","3=Ireen Wust"]
```

He can then use the second method to get these details


```
{

    "Id":3,
    "First Name":"Ireen",
    "Last Name":"Wust",
    "Gender":"Female",
    "Nationality":"Netherlands",
    "Date of Birth":"1 April 1986",
    "Age":28,
    "Height":"1.68m (5'6")",
    "Weight":"63 kg (139 lbs)",
    "Place of birth":"Goirle, Netherlands",
    "Total Medals":5,
    "Gold Medals":["Speed Skating:Ladies' 3000m", "Speed Skating:Ladies' Team
      Pursuit"]
    "Silver Medals":["Speed Skating:Ladies' 1000m", "Speed Skating:Ladies' 1500m",
      "Speed Skating:Ladies' 5000m"]
    "Bronze Medals":[]
}
```


docs.idb13.apiary.io/#athletes
Home
Sochi 2014 Winter Olympics
Yours
Documentation
Traffic Inspector
Editor
Nickson
Fork me on GitHub
Sochi 2014 Winter Olympics
Documentation for Sochi 2014 Winter Olympics API
Athletes
Athletes related resources. An athlete object has the following attributes:
• Id
• First Name
• Last Name
• Gender
• Nationality
• Date of Birth
• Age
• Height
• Weight
• Place of Birth
• Total Medals (Total of Gold, Silver, and Bronze Medals won by the athlete)
• Gold Medals (List of Events)
• Silver Medals (List of Events)
• Bronze Medals (List of Events)
List of all athletes
GET
/athletes
Add Comment
Returns a list of all athletes and their ids.
Retrieve an Athlete
GET
/athletes/{id}
Add Comment
Returns the details of an athlete.
Parameters
Name
Description
Details
id
Numeric id of the Athlete to perform action with.
number, required
example: 1
Countries
Countries related resources. A country has the following attributes:
Table of Contents
Athletes
Countries
Events
API
http://www.google.com/
Mock Server
http://idb13.apiary-mock.com
Use this URL to access a mockup of the API server. Your traffic will be recorded and compared to the documentation. You'll find your traffic analysis in the inspector or directly here in the documentation, right next to each resource.


Sochi 2014 Winter Olympics—by apiary.io - Mozilla Firefox
docs.idb13.apiary.io/#athletes
Home
Sochi 2014 Winter Olympics
Yours
Documentation
Traffic Inspector
Editor
Nickson
Fork me on GitHub
id
Numeric id of the Athlete to perform action with.
number, required
example: 1
Countries
Countries related resources. A country has the following attributes:
Id
Name
Description (Wiki link)
Year of First Appearance
Number of Appearances
Total of Gold Medals
Total of Silver Medals
Total of Bronze Medals
List of Athletes
Sports participated in Sochi
List of all Countries
GET
/countries
Add Comment
Returns a list of countries and their ids.
Retrieve a Country
GET
/countries/{id}
Add Comment
Parameters
Name
Description
Details
id
Numeric id of the Country to perform action with.
number, required
example: 1
Events
Events related resources. An event has the following attributes:
Id
Name (This consists of the event's name and its associated sport)
Description (Wiki link)
Table of Contents
Athletes
Countries
Events
API
http://www.google.com/
Mock Server
http://idb13.apiary-mock.com
Use this URL to access a mockup of the API server. Your traffic will be recorded and compared to the documentation. You'll find your traffic analysis in the inspector or directly here in the documentation, right next to each resource.


Sochi 2014 Winter Olympics—by apiary.io - Mozilla Firefox
docs.idb13.apiary.io/#athletes
Home
Sochi 2014 Winter Olympics
Yours
Documentation
Traffic Inspector
Editor
Nickson
Fork me on GitHub
List of all Countries
GET /countries
Add Comment
Returns a list of countries and their ids.
Retrieve a Country
GET /countries/{id}
Add Comment
Parameters
Name
Description
Details
id
Numeric id of the Country to perform action with.
number, required
example: 1
Events
Events related resources. An event has the following attributes:
Id
Name (This consists of the event's name and its associated sport)
Description (Wiki link)
Sport
Gold Medalist(s)
Silver Medalist(s)
Bronze Medalist(s)
List of all Events
GET /events
Add Comment
Retrieve an event
GET /events/{id}
Add Comment
Parameters
Name
Description
Details
id
Numeric id of the Event to perform action with.
number, required
example: 1
Table of Contents
Athletes
Countries
Events
API
http://www.google.com/
Mock Server
http://idb13.apiary-mock.com
Use this URL to access a mockup of the API server. Your traffic will be recorded and compared to the documentation. You'll find your traffic analysis in the inspector or directly here in the documentation, right next to each resource.

Django Models & Attributes:

1. ATHLETES

    * First Name

        - Primary Key

        - Athlete's first name

        - String

    * Last Name

        - Primary Key

        - Athlete's last name

        - String

    * Country

        - Athlete's Country

        - String

    * Gender

        - Athlete's Gender

        - String

    * Birthdate

        - Primary Key

        - Athlete's Date of birth

        - Date

    * Height

- Athlete's Height

- String

* Weight

- Athlete's Weight

- String

* Picture

- Athlete's Picture

- Name of image file with extension

* Video

- YouTube video related to the athlete

- YouTube link in https://www.youtube.com/v/* format

* Gold Medals

- List of events the athlete won the gold medal from

- Foreign Key

* Silver Medals: Events

- List of events the athlete won the silver medal from

- Foreign Key

* Bronze Medals: Events

- List of events the athlete won the bronze medal from

- Foreign Key

FirstName, LastName, and DateofBirth are chosen as the primary keys to prevent

conflicts. It is highly unlikely that participating athletes will have the same name and date of birth.

2. COUNTRIES

* Name
    - Primary Key
    - Name of the coutnry
    - String
* Athlete List
    - List of participating athletes from the country
    - String
* Description
    - Brief description of the country
    - String
* Gold Medals Won
    - Total of gold medals won by the country
    - Dynamically calculated from the medals won by the athletes
    - Integer
* Silver Medals Won
    - Total of silver medals won by the country
    - Dynamically calculated from the medals won by the athletes

* Bronze Medals Won
    - Total of bronze medals won by the country
    - Dynamically calculated from the medals won by the athletes
* Country Code
    - 3 letter International Olympic Committee country code
    - String
* Coordx
    - X-Coordinate of country used in Google Maps
    - Float
* Coordy
    - Y-Coordinate of country used in Google Maps
    - Float
* Coordz
    - Zoom value used in Google Maps
    - Int

3. EVENTS

* Name
    - Primary Key
    - Name of the event
    - String
* Sport

    - Primary Key

    - Name of the associated sport

    - String

* Description

    - A brief description of the event

    - String

* Icon

    - Icon that represents the sport of the event

    - Name of icon file with extension

* Gold Medalist(s)

    - The name of the gold medalist(s). Can be multiple athletes.

    - Foreign Key

* Silver Medalist(s)

    - The name of the silver medalist(s). Can be multiple athletes.

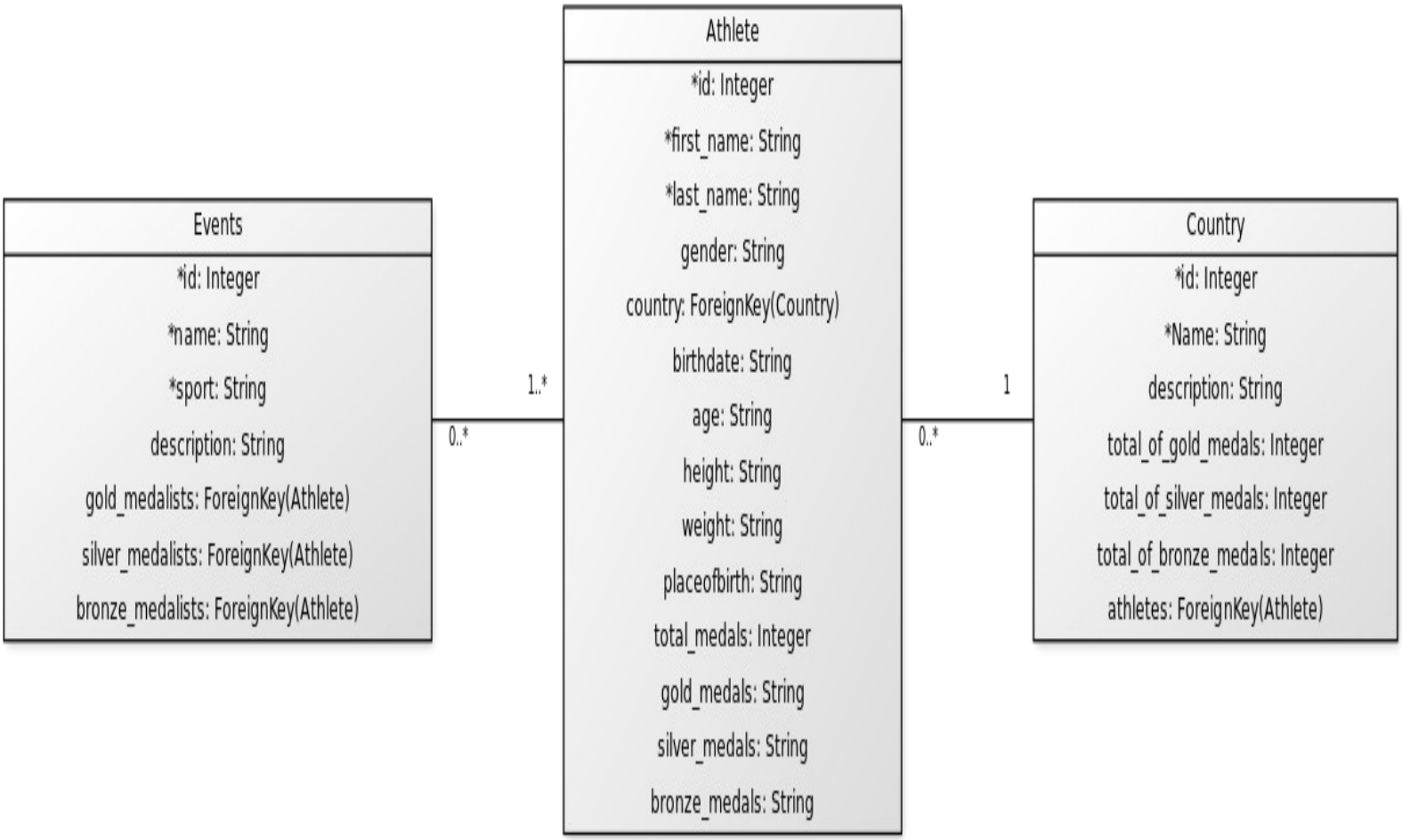
    - Foreign Key

* Bronze Medalist(s)

    - The name of the bronze medalist(s). Can be multiple athletes.

    - Foreign Key

Django Models Diagram



TESTS

Django Models:

tests.py (~/CS373/cs373-idb/cs373_Sochi/home) - gedit

Open Save Undo

tests.py

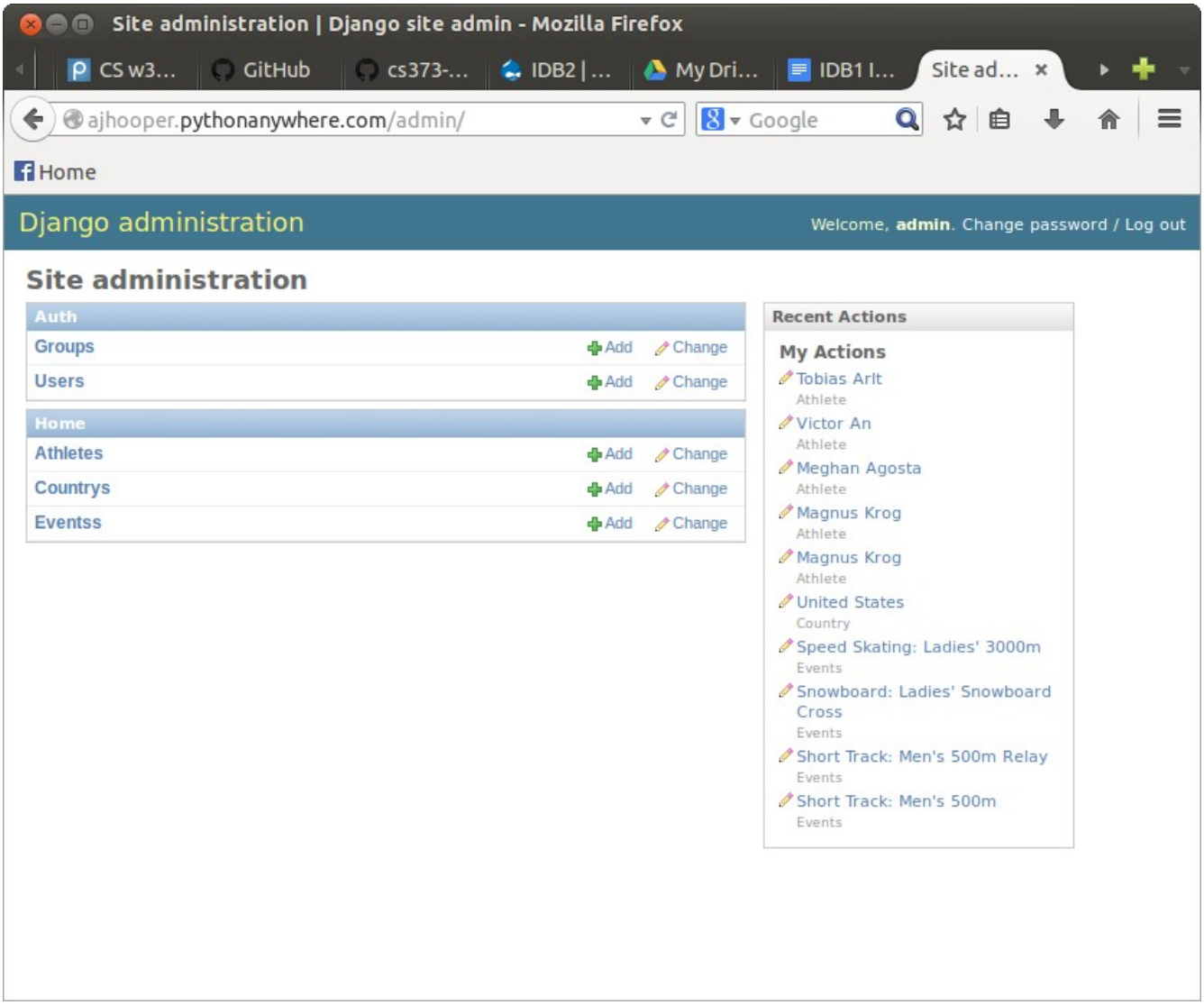
```python
from django.test import TestCase

from home.models import Athlete, Country, Events

class HomeTests(TestCase):

    def test_Athlete1(self):
        athlete = Athlete(self,"John", "Doe", "US")
        self.assertEqual(athlete.first_name, "John")

    def test_Athlete2(self):
        athlete = Athlete(self, "John", "Doe", "US")
        self.assertEqual(athlete.last_name, "Doe")

    def test_Athlete3(self):
        #Have to use get_or_create to make the country object to be passed to athlete
        country, created = Country.objects.get_or_create(name="Egypt", description="Made of sand")
        athlete = Athlete(self, "John", "Doe", country.id)
        self.assertEqual(athlete.country.name, "Egypt")

    def test_Events1(self):
        event = Events(self, "Luge Doubles", "Luge", "Riding on cars")
        self.assertEqual(event.name, "Luge Doubles")

    def test_Events2(self):
        event = Events(self, "Luge Doubles", "Luge", "Riding on cars")
        self.assertEqual(event.sport, "Luge")

    def test_Events3(self):
        event = Events(self, "Luge Doubles", "Luge", "Riding on cars")
        self.assertEqual(event.desc, "Riding on cars")

    def test_Country1(self):
        country = Country(self, "Egypt", "Made of sand")
        self.assertEqual(country.name, "Egypt")

    def test_Country2(self):
        country = Country(self, "Egypt", "Made of sand")
        self.assertEqual(country.description, "Made of sand")

    def test_Country3(self):
        country = Country(self, "Egypt", "Made of sand", 10, 2, 8)
        self.assertEqual(country.total_gold_medals, 10)

    def test_Country4(self):
        country = Country(self, "Egypt", "Made of sand", 10, 2, 8)
        self.assertEqual(country.total_bronze_medals, 8)
```

Python Tab Width: 8 Ln 8, Col 52 INS

Simple tests that test if objects were created successfully.

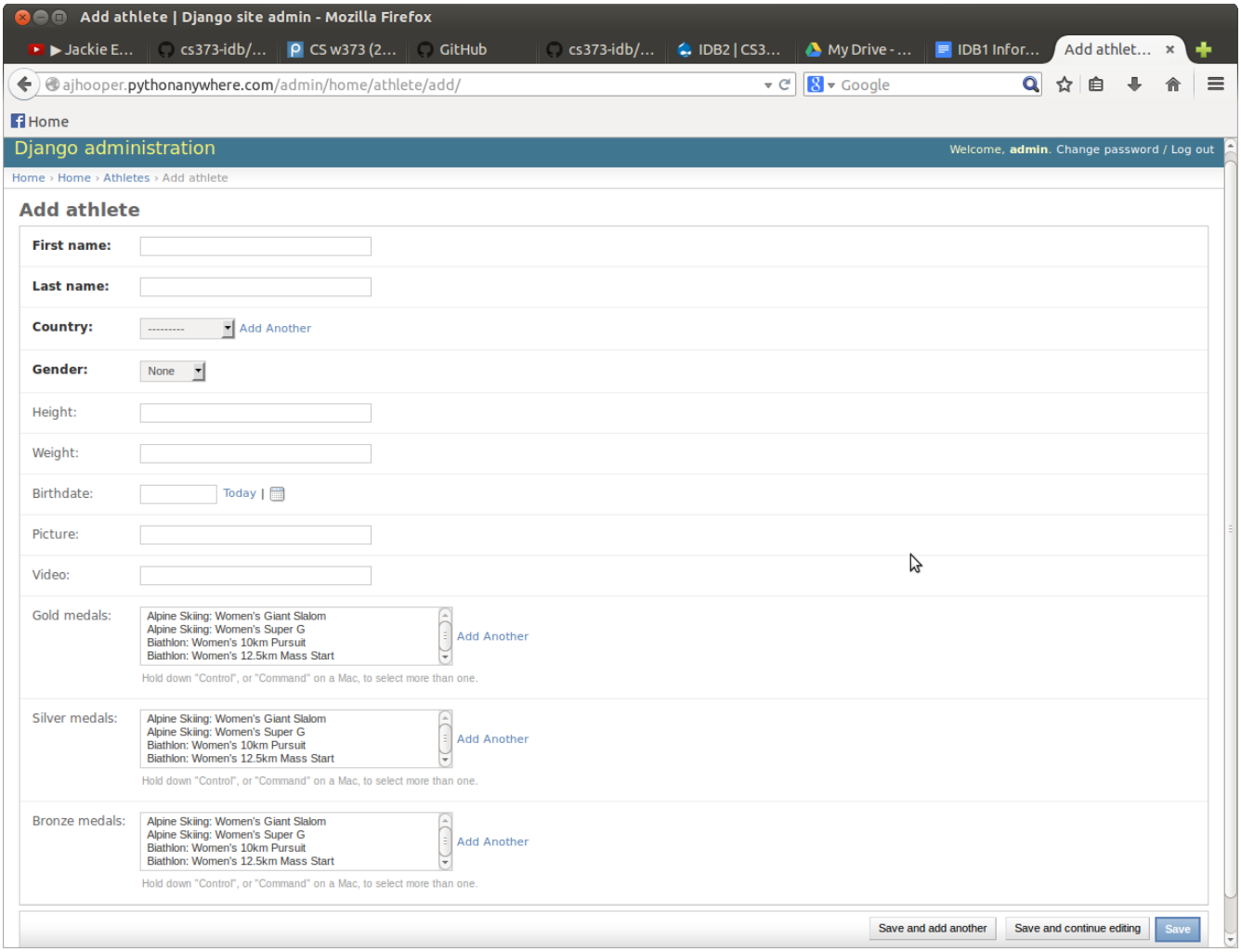
DATABASE

Fortunately Django handles the sql database for us and we do not have to write any sql queries to manage our database. The database can be accessed by going to the admin page. The picture following is the look of the admin page.



Admin page of our website.

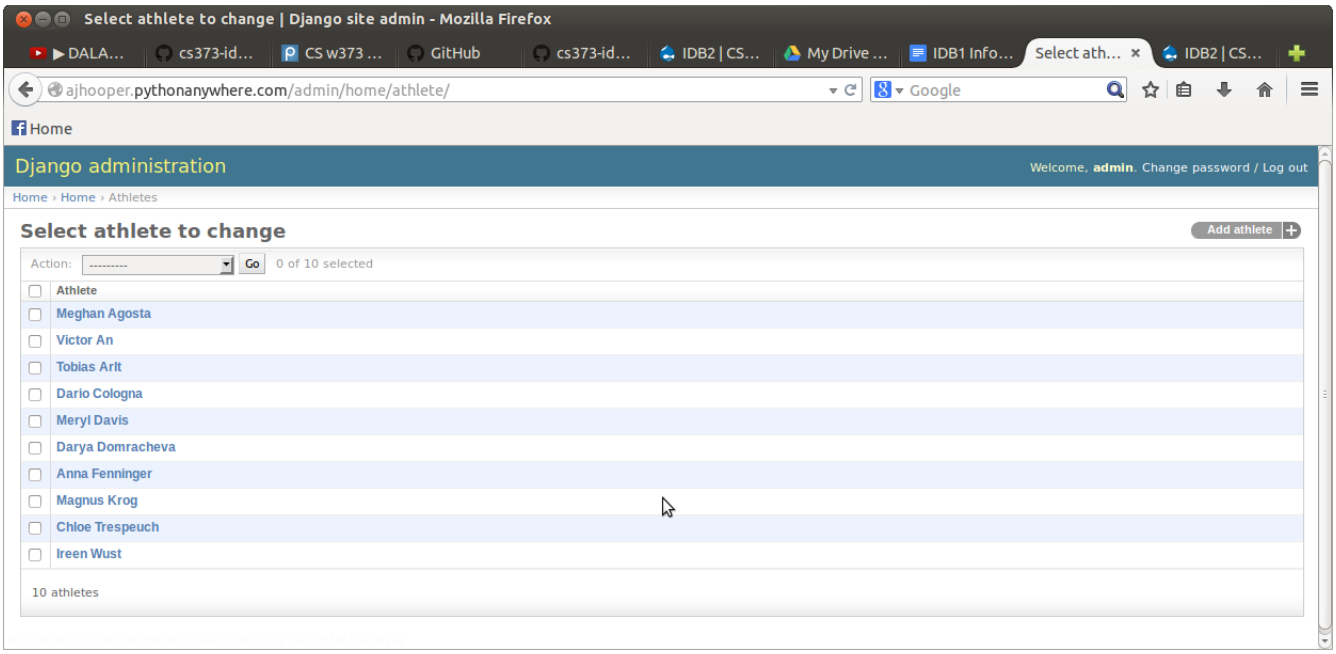
We can easily add a new data to our database by clicking add which will direct us to a form with all the attributes of a particular model. We can just fill in the form to add the desired data.



Add form of athlete.

One important thing to notice is that Django automatically creates a drop-down list for foreign keys. This list is automatically updated whenever there is a change in the relevant databases. In this case, all Events, which are foreign keys for are listed in the drop-down list for Gold, Silver, and Bronze medals.

For updating the existing datas, we can click Change which directs us to the list of existing datas for a particular model.



List of existing athletes

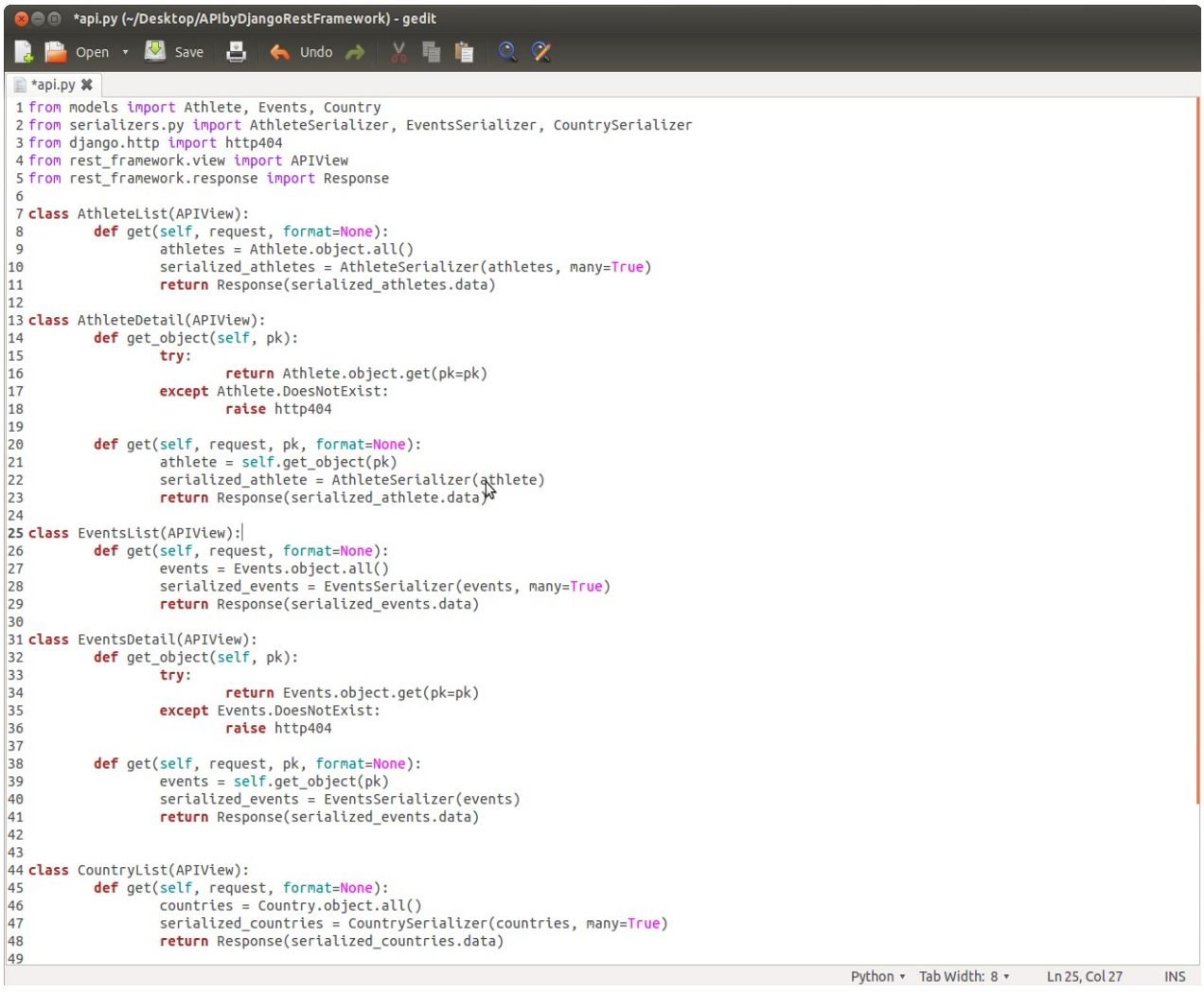
To update an athlete, say Ireen Wust, we just have to click her name which directs us to a filled form. We can then change the attributes and click save to update.

API (Django REST framework or Tastypie?)

There were two options to the software that we can use to build the API. One was the Django Rest Framework, the other one is Tastypie. We decided to create both versions and see which one suited better for our site.

A. Django REST framework

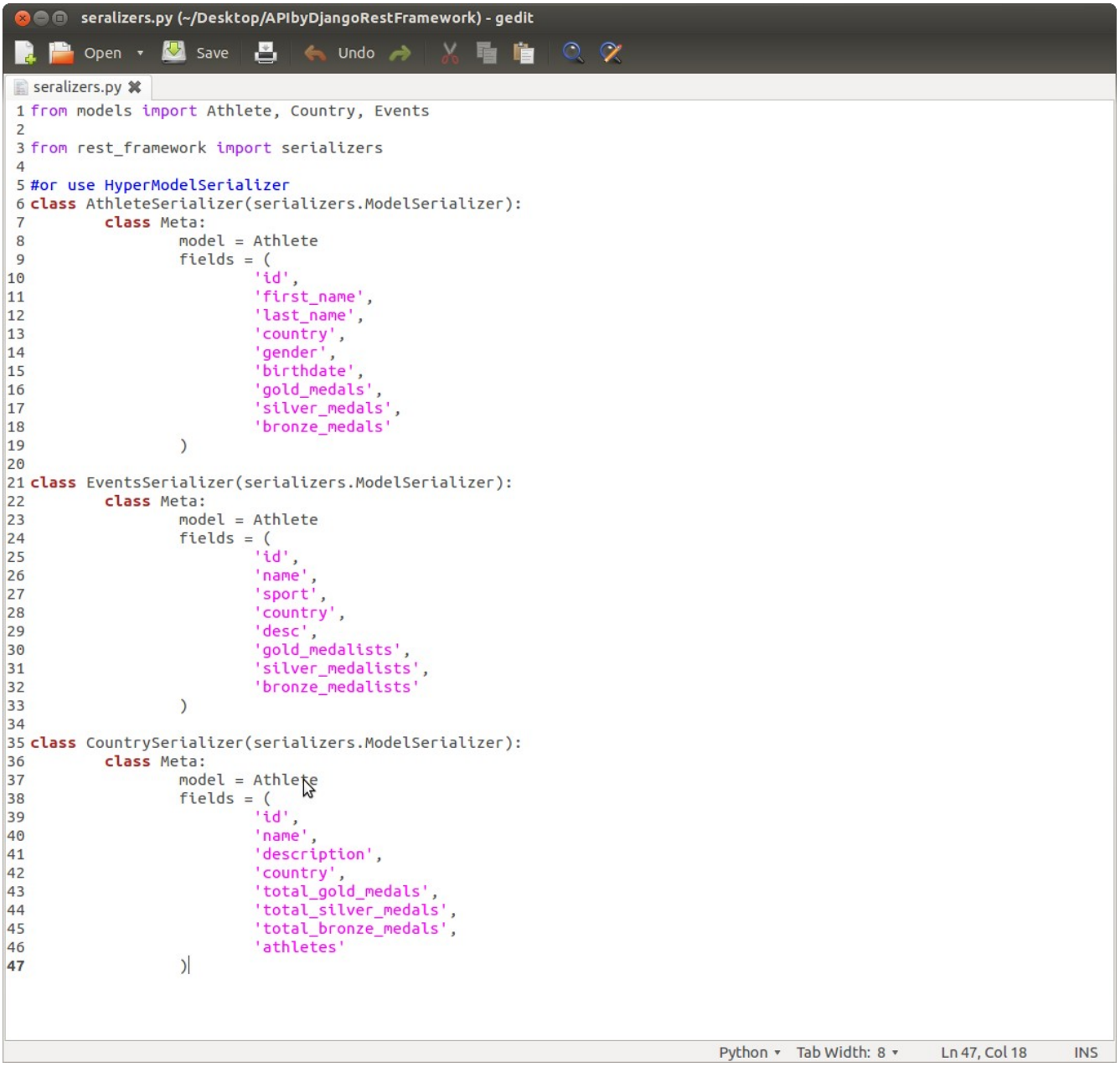
The core file is api.py in which we decribe the kind of APIs that we want as described in DESIGN section for RESTful API (list and details).




```
1 from models import Athlete, Events, Country
2 from serializers.py import AthleteSerializer, EventsSerializer, CountrySerializer
3 from django.http import http404
4 from rest_framework.view import APIView
5 from rest_framework.response import Response
6
7 class AthleteList(APIView):
8         def get(self, request, format=None):
9                 athletes = Athlete.object.all()
10                serialized_athletes = AthleteSerializer(athletes, many=True)
11                return Response(serialized_athletes.data)
12
13 class AthleteDetail(APIView):
14        def get_object(self, pk):
15                try:
16                        return Athlete.object.get(pk=pk)
17                except Athlete.DoesNotExist:
18                        raise http404
19
20        def get(self, request, pk, format=None):
21                athlete = self.get_object(pk)
22                serialized_athlete = AthleteSerializer(athlete)
23                return Response(serialized_athlete.data)
24
25 class EventsList(APIView):
26        def get(self, request, format=None):
27                events = Events.object.all()
28                serialized_events = EventsSerializer(events, many=True)
29                return Response(serialized_events.data)
30
31 class EventsDetail(APIView):
32        def get_object(self, pk):
33                try:
34                        return Events.object.get(pk=pk)
35                except Events.DoesNotExist:
36                        raise http404
37
38        def get(self, request, pk, format=None):
39                events = self.get_object(pk)
40                serialized_events = EventsSerializer(events)
41                return Response(serialized_events.data)
42
43
44 class CountryList(APIView):
45        def get(self, request, format=None):
46                countries = Country.object.all()
47                serialized_countries = CountrySerializer(countries, many=True)
48                return Response(serialized_countries.data)
49
```

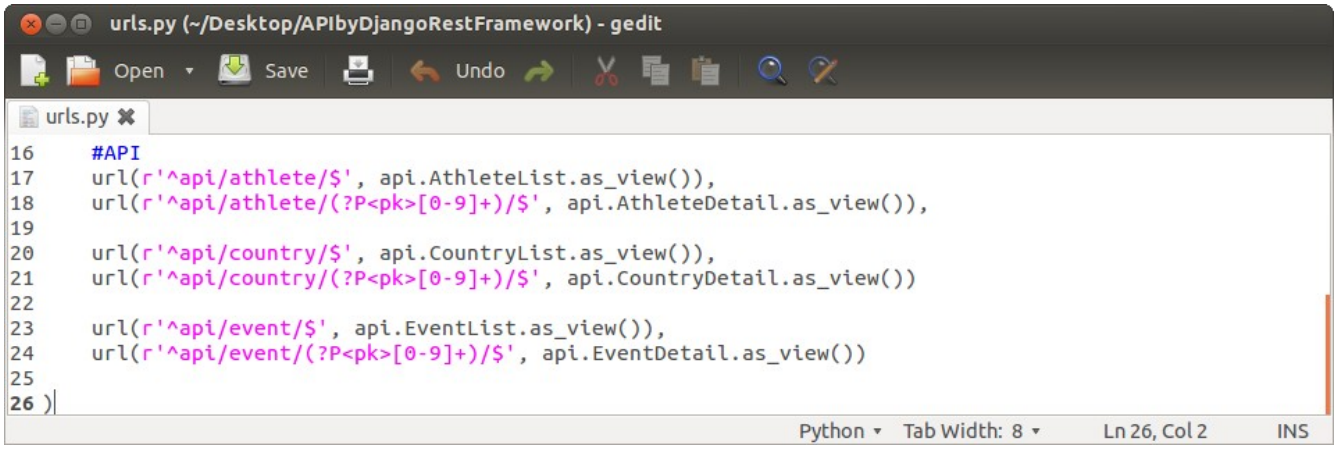
api.py for Django REST framework

For Django Rest Framework, we had to create another file called serializers.py that allows the data to be easily rendered to JSON, XML format.



Django Rest Framework Serializers.py

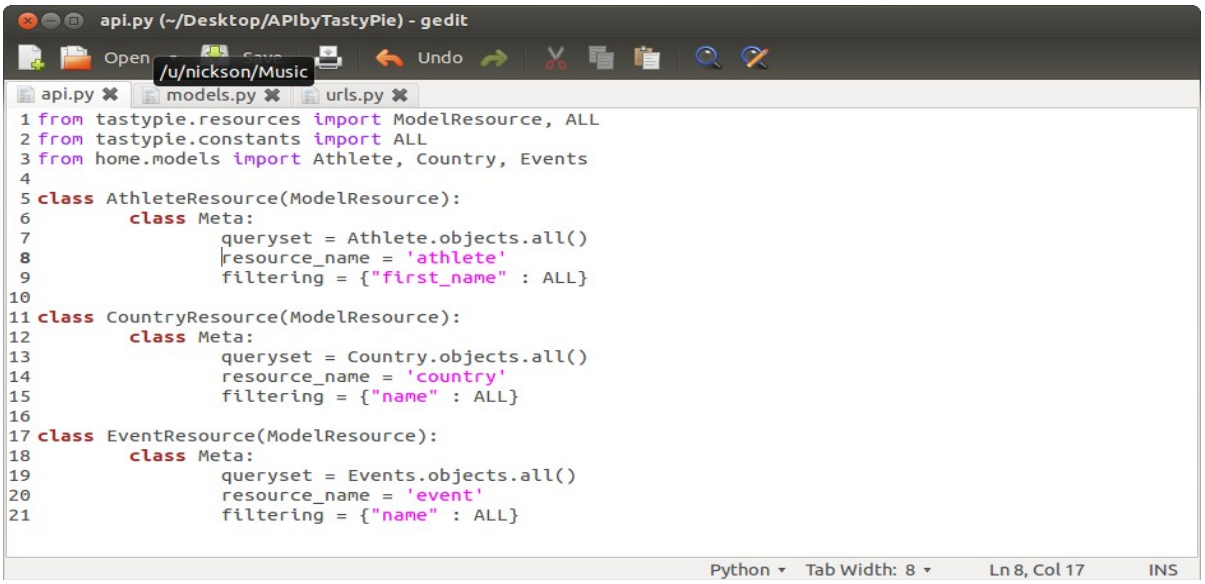
Finally, to make sure that the API pages are running, we update the urls.py to include the new API pages.



Added URLs for API

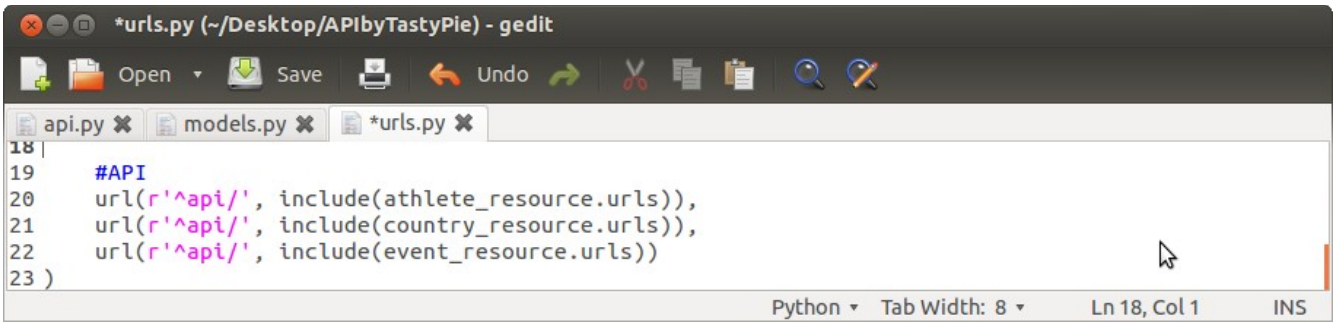
B. Tastypie

Here, we also included the core code for API in a file called api.py.



api.py for Tastypie

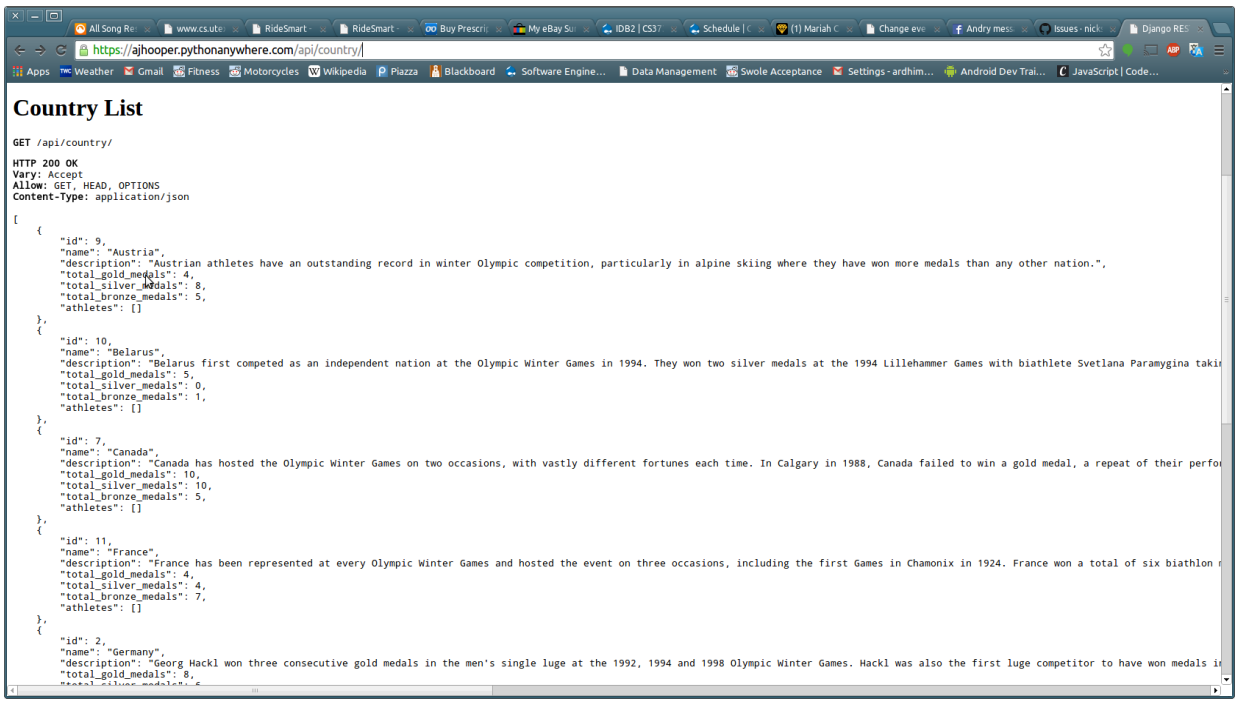
We also had to add the API. However, the ModelResource amazingly handled all the urls and we do not have to create any regular expression.



urls.py for Tastypie

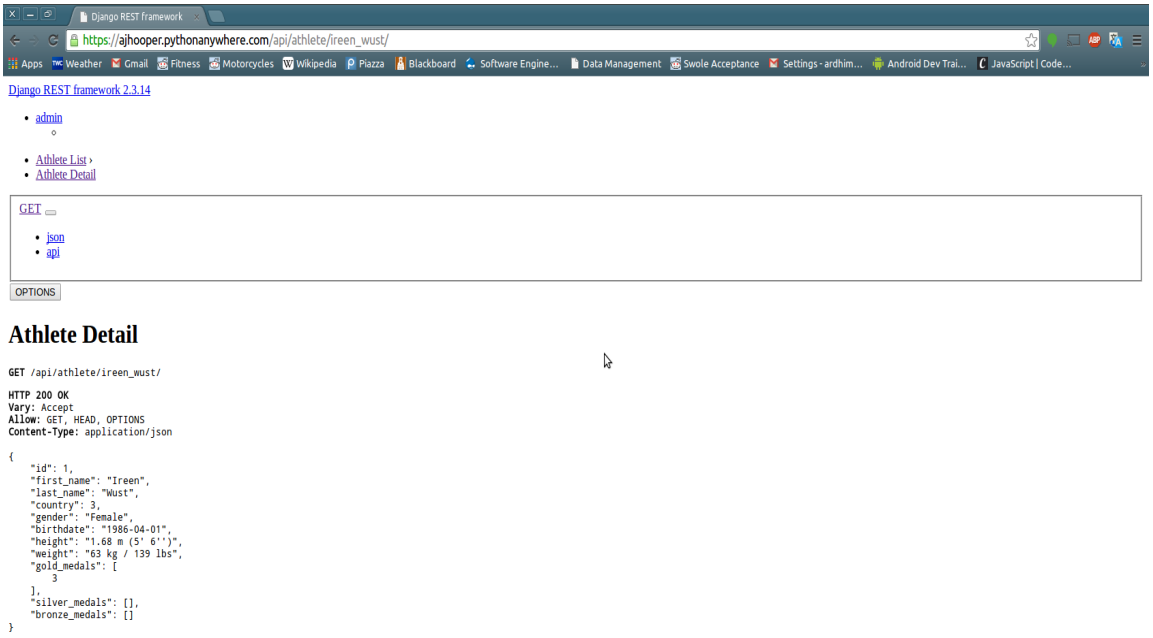
C. Conclusion

Eventually, we decided to use Django Rest Framework to build our API. To access our API, you have to add api in the url path. As an example, for getting the information of all countries, the url will be ajhooper.pythonanywhere.com/api/country/



https://ajhooper.pythonanywhere.com/api/country/

For retrieving information of single athlete/ country/ event, use their full name replacing every space character with underscore ('_') as the last path of the url.



https://ajhooper.pythonanywhere.com/api/athlete/ireen_wust/