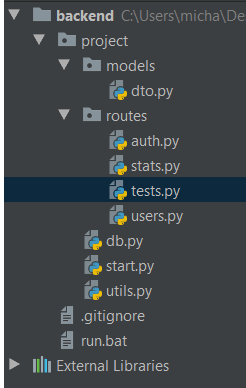
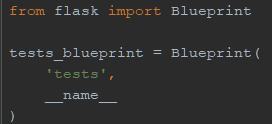
Poradnik jak implementować obsługę żądań http dla aplikacji Chuchnij na przykładzie button testu

# Krok 1 – Implementacja nowej usługi na backendzie

W katalogu project\routes tworzymy plik o nazwie adekwatnej do usługi. Skoro usługa to obsługa testów, plik nazywam „tests.py”.

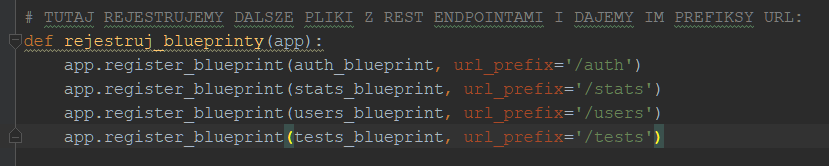


W nowo utworzonym pliku tworzę tzw. Blueprinta:  


Jest on potrzebny, by potem móc zarejestrować usługę w serwerze.

# Krok 2 – Rejestracja usługi w serwerze aplikacji

W pliku project\start.py rejestrujemy blueprint dopisując odpowiednią linijkę kodu w metodzie rejestruj\_blueprinty()

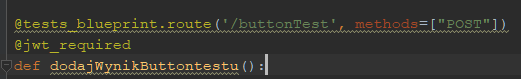


W metodzie tej przypisujemy odpowiedni blueprint oraz określamy tzw. url\_prefix. Jest to ten kawałek adresu uri, który będzie poprzedzał każdy człon adresu powiązanego z każdą funkcją implementowaną w tej usłudze do której odnosi się blueprint.

# Krok 3 – Implementacja funkcji przetwarzającej wybrane żądanie na serwerze

W tym wypadku chcemy przetworzyć wynik button testu, dlatego tworzymy funkcję „dodajWynikButtonTestu()”.

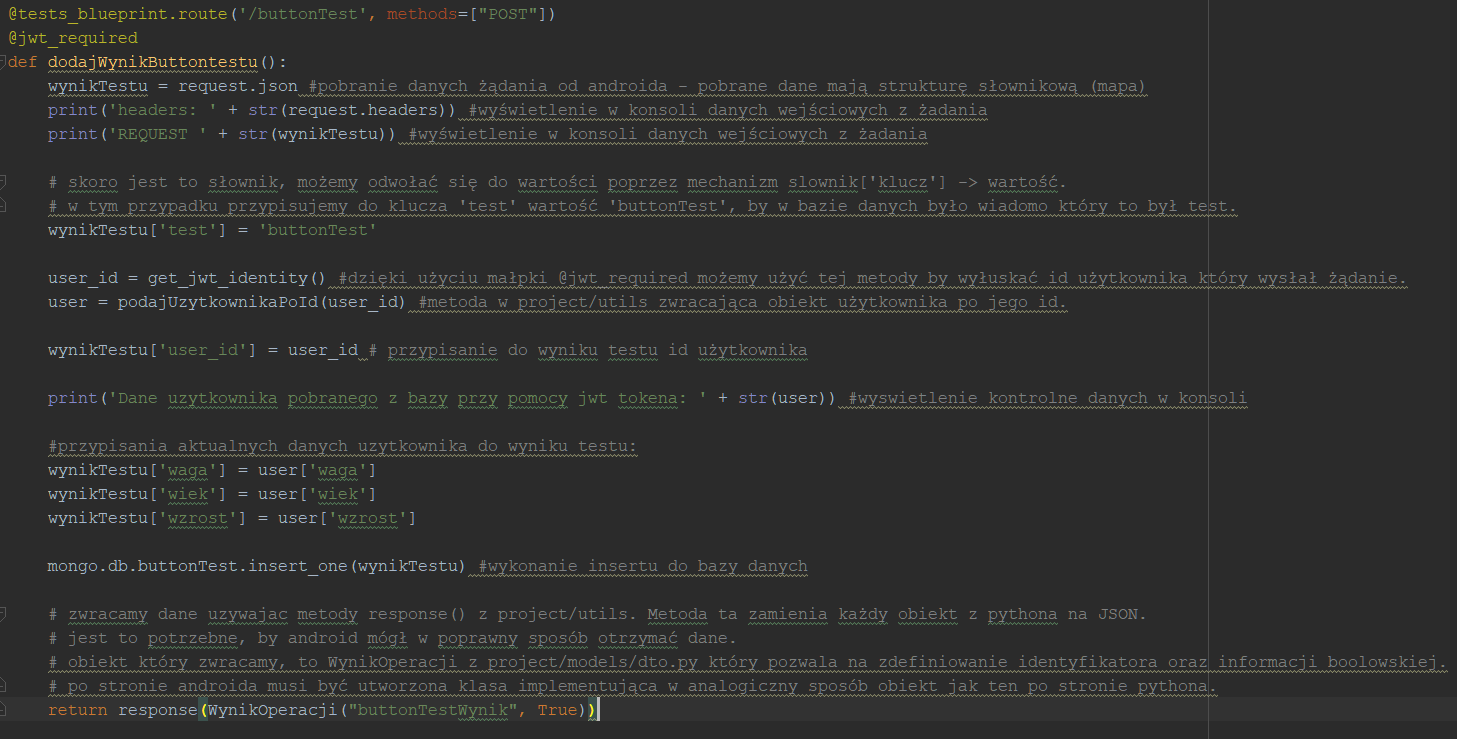
Nagłówek definiujemy w ten sposób:



Należy podać wcześniej utworzony tests\_blueprint po małpce, a następnie określić człon adresu prowadzącego do tej metody oraz metodę/metody http uruchamiające tę funkcję.

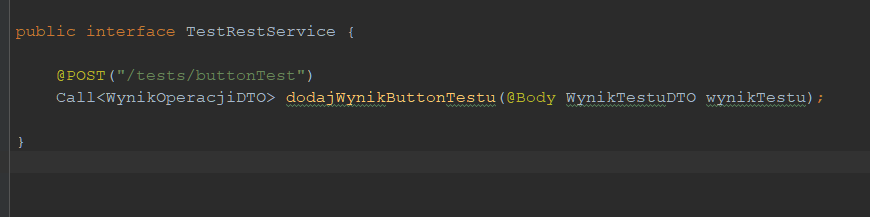
Anotacja @jwt\_required sprawia, że sprawdzana jest poprawność użytkownika, oraz pozwala na określenie go w ciele funkcji. Dzięki tej anotacji, gdy ktoś spróbuje użyć tej funkcji bez zalogowania się, funkcja zwróci błąd http 401.

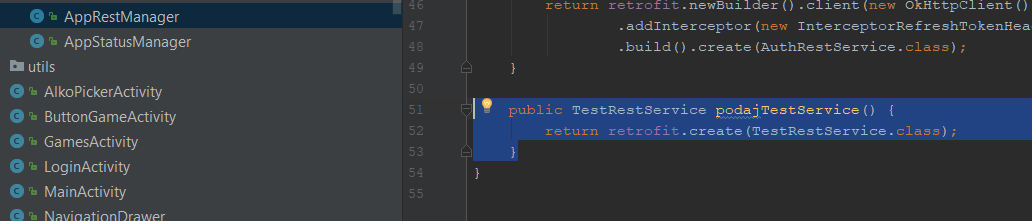
## Cała funkcja:



# Krok 4 – Implementacja serwisu na androidzie

Tworzymy odpowiedni serwis w androidzie poprzez utworzenie interfejsu TestRestService w paczce api.service. Będzie potrzebna metoda do dodania wyniku button testu. Małpka @POST określa metodę http POST oraz odpowiedni człon adresu dostępowego do funkcji na serwerze. Metoda jako ciało żądania wysyła obiekt klasy WynikTestuDTO a spowrotem z serwera otrzymuje WynikOperacjiDTO.

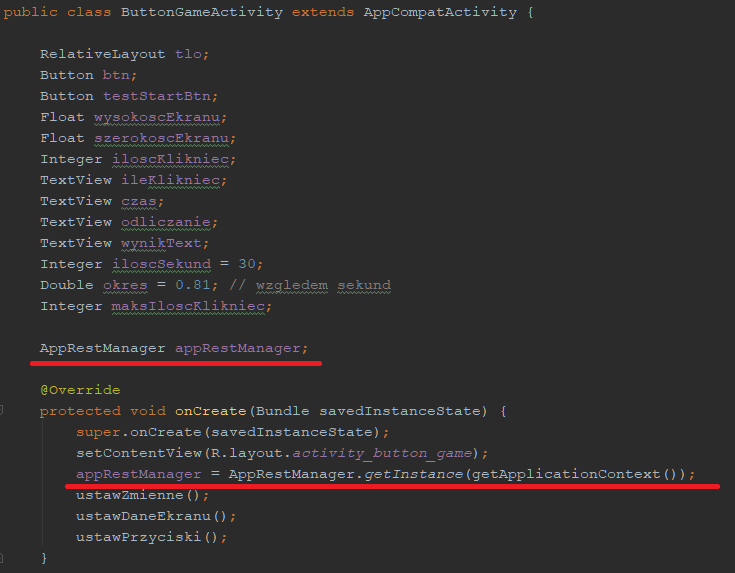


Interfejs jedynie mówi o tym, jakie metody mają być zaimplementowane, jednak biblioteka Retrofit sama generuje te metody. Aby jednak to zrobiła, musimy wywołać metodę tworzącą obiekt serwisu na bazie tego interfejsu. Robimy to w klasie singletonowej służącej do wydobywania serwisów, która znajduje się w singletons\AppRestManager.  


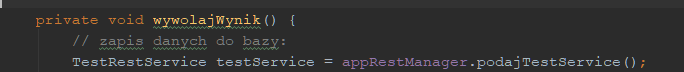
Tyle wystarczy, by wydobyć gotowy serwis pozwalający na wysyłanie danych do serwera.

# Wysłanie danych z androida do serwera

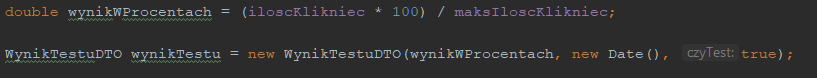
W ButtonGameActivity istnieje metoda wywołująca wynik. Aby ten wynik wysłać do serwera, należy wydobyć za pomocą singletona AppRestManager serwis testów.



Teraz możemy zaimplementować wysyłanie danych do serwera. Najpierw musimy wydobyć TestRestService z naszego appRestMangera.



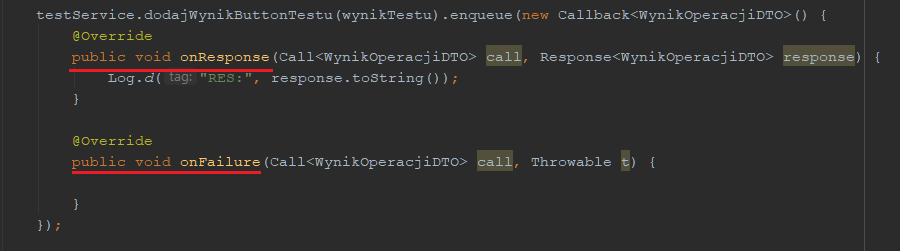
Później tworzymy obiekt WynikTestuDTO – obiekt który będzie wysłany do serwera.



Na koniec wywołujemy metodę serwisu „dodajWynikButtonTestu()”. Używając do tego metody enqueue, żądanie jest asynchroniczne.



Asynchroniczne żądanie oznacza, że kod poniżej wywołania tej funkcji będzie się wykonywał nie czekając na wynik z serwera, dlatego aby móc odebrać dane z serwera, metoda enqueue wywołuje metody onResponse lub onFailure z callbacka który musimy zaimplementować – metody te wywołują się dla wyniku serwera.



W tym przypadku gdy nie było błędu z serwera – logujemy wynik response. Gdy był błąd, nie robimy nic. W produkcyjnej aplikacji jednak trzeba sensownie reagować na błędy.