

# API

## API endpoints

/users/\*

- GET /users/:device\_id

```
{
  "type": "object",
  "properties": {
    "achievements": {
      "type": "object",
      "properties": {
        ":achievement_id": {
          "type": "object",
          "properties": {
            ":achievement_type": {
              "type": ["object", "null"],
              "properties": {
                "badge": { "type": "string" },
                "value": { "type": ["integer", "number"] },
                "next_badge_at": { "type": ["integer", "number", "null"] }
              }
            },
            "required": ["badge"]
          }
        }
      }
    },
    "ranking": {
      "type": "object",
      "properties": {
        ":achievement_id": {
          "type": "object",
          "properties": {
            ":achievement_type": {
              "type": ["object", "null"],
              "properties": {
                "value": { "type": "number" },
                "rank": { "type": "integer" },
                "device_id": { "type": "string" }
              }
            },
            "required": ["value", "rank", "device_id"]
          }
        }
      }
    }
  }
}
```

```

    }
  }
}
},
"required": ["achievements", "ranking"]
}

```

- GET /users/:device\_id/ranking/?filter=:filter&from=X&to=Y

```

{
  "type": "object",
  "properties": {
    ":achievement_id": {
      "type": "object",
      "properties": {
        ":achievement_type": {
          "type": ["object", "null"],
          "properties": {
            "value": { "type": "number" },
            "rank": { "type": "integer" },
            "device_id": { "type": "string" }
          },
          "required": ["value", "rank", "device_id"]
        },
      }
    },
  }
}

```

- GET /users/:device\_id/ranking/:achievement\_id/?filter=:filter&from=X&to=Y

```

{
  "type": "object",
  "properties": {
    ":achievement_type": {
      "type": ["object", "null"],
      "properties": {
        "value": { "type": "number" },
        "rank": { "type": "integer" },
        "device_id": { "type": "string" }
      },
      "required": ["value", "rank", "device_id"]
    },
  }
}

```

- GET /users/:device\_id/achievements/?filter=:filter

```
{
  "type": "object",
  "properties": {
    ":achievement_id": {
      "type": "object",
      "properties": {
        ":achievement_type": {
          "type": ["object", "null"],
          "properties": {
            "badge": { "type": "string" },
            "value": { "type": ["integer", "number"] },
            "next_badge_at": { "type": ["integer", "number", "null"] }
          },
          "required": ["badge"]
        }
      }
    }
  }
}
```

- GET /users/:device\_id/achievements/:achievement\_id/?filter=:filter

```
{
  "type": "object",
  "properties": {
    ":achievement_type": {
      "type": ["object", "null"],
      "properties": {
        "badge": { "type": "string" },
        "value": { "type": ["integer", "number"] },
        "next_badge_at": { "type": ["integer", "number", "null"] }
      },
      "required": ["badge"]
    }
  }
}
```

/ranking/\*

- GET /ranking/?filter=:filter&from=X&to=Y

```
{
  "type": "object",
```

```

"properties": {
  ":achievement_id": {
    "type": "object",
    "properties": {
      ":achievement_type": {
        "type": ["array", "null"],
        "items": {
          "type": "object",
          "properties": {
            "value": { "type": "integer" },
            "device_id": { "type": "string" }
          },
          "required": ["value", "device_id"]
        }
      }
    }
  }
}

```

- GET /ranking/:achievement\_id/?filter=:filter&from=X&to=Y

```

{
  "type": "object",
  "properties": {
    ":achievement_type": {
      "type": ["array", "null"],
      "items": {
        "type": "object",
        "properties": {
          "value": { "type": "integer" },
          "device_id": { "type": "string" }
        },
        "required": ["value", "device_id"]
      }
    }
  }
}

```

## Flowchart API

Poniższy rysunek pokazuje sposób działania wszystkich dostępnych endpointów API wraz z lokalizacją powiązanego kodu w plikach projektu oraz typami zwracanych odpowiedzi.

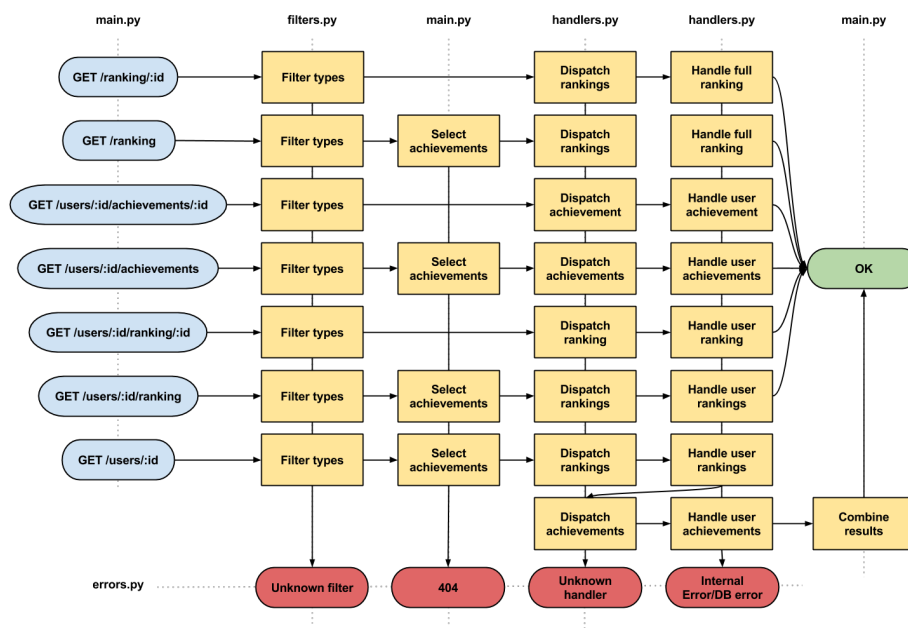


Figure 1: Przepływ obsługi zapytania

Pierwszym krokiem przetwarzania jest opcjonalna filtracja na podstawie podanego w parametrach zapytania filtra. Następnie w zależności od endpointu'u wybierane są odpowiednie identyfikatory i dla każdego z nich uruchamiany jest dispatch. Dispatch wybiera odpowiednią funkcję obsługującą zapytanie na podstawie typu achievementu oraz zapytania. Ostatnim krokiem jest uruchomienie odpowiedniej funkcji obsługującej zapytanie, która to generuje wynik końcowy zapytania.

## Słowniczek używanych pojęć

- :device\_id - ID urządzenia z Androidem.
- :achievement\_id - ID osiągnięcia. Może przyjmować każdą wartość będącą [sensorem AWARE](#), za wyjątkiem 'Aware'. Nazwa osiągnięcia pisana jest małymi literami, a zamiast spacji użyto znaku '\_'. Oprócz nich możliwymi wartościami dla :achievement\_id są:
  - wifi\_special
  - network\_data
  - battery\_usage

Po dokładne informacje dotyczące możliwych filtrów, zdefiniowanych tabel autorzy odsyłają do zawartości domyślnego pliku konfiguracyjnego `config.json`, do sekcji Konfiguracja oraz Lista osiągnięć.

- `:achievement__type` - typ osiągnięcia. Możliwe wartości do przyjęcia to:
  - `count` dla osiągnięć ilościowych
  - `procent` dla osiągnięć
  - `time` dla osiągnięć czasowych
  - `mean` dla osiągnięć przyznawanych na podstawie średniej, np. dla średniego zużycia baterii.
  - `funny` dla różnego rodzaju śmiesznych odznak, np. dla nazw sieci WiFi.
  - `security` dla osiągnięć związanych z bezpieczeństwem różnych sensorów, np. dla WiFi.
- `filter` - umożliwia filtrację dla osiągnięć po ich typie. Może przyjmować jedną z wartości zdefiniowanych w `:achievement__type` oraz:
  - `all` dla wszystkich rodzajów osiągnięć
- `from` - określa początek przedziału dla wyliczania rankingów.
- `to` - określa koniec przedziału dla wyliczania rankingów.

## Przykłady

- `GET /users/:id/achievements?filter=count` - Ze wszystkich dostępnych achievement'ów wybierane są tylko te o typie count, następnie dla każdego ID takiego achievementu uruchamiana jest skonfigurowana w pliku konfiguracyjnym funkcja obsługująca zapytanie. Wyniki są łączone w jeden obiekt JSON, po czym zostają zwrócone jako wynik zapytania.
- `GET /users/:id/achievements/wifi` - Ze wszystkich dostępnych achievement'ów wybierany jest tylko ten o ID równym wifi, następnie uruchamiana jest skonfigurowana dla niego funkcja obsługująca zapytanie, a jej wynik zwracany jest jako wynik zapytania.
- `GET /users/:id/ranking?from=X&to=Y` - Wybierane są ID wszystkich dostępnych rankingów, następnie dla każdego z nich uruchamiana jest skonfigurowana funkcja obsługująca zapytanie, do której przekazywane są dodatkowe parametry from oraz to. Wynikiem zapytania jest kombinacja wyników działania handlerów.

- GET /users/:id - Dla danego użytkownika najpierw obliczane są pozycje w rankingach dla wszystkich achievement'ów, a następnie obliczane są zdobyte przez niego odznaki. Wyniki łączone są w jeden obiekt JSON i zwracane jako wynik zapytania.
- GET /ranking/wifi - Dla achievement'u o ID wifi uruchamiana jest funkcja obliczająca zbiorcze rankingi wszystkich użytkowników.

## Konfiguracja

Konfiguracja jest zapisana w formacie JSON. Dozwolone są jednak komentarze, które są później ignorowane w procesie wczytywania konfiguracji. Poniżej przedstawiono część konfiguracji ze szczególnym uwzględnieniem interesujących części:

```
{
  "achievements": {
    "default": {
      "count": {
        "thresholds": [10000, 100000, 1000000],
        "badges": [
          "Babies first steps.",
          "Keep up the good work!",
          "We're gonna need MapReduce here..."
        ]
      },
      "procent": {
        "thresholds": [0.01, 0.1, 0.5],
        "badges": [
          "Ghost",
          "Active",
          "Dominator"
        ]
      },
      "time": {
        "thresholds": ["72h", "90d", "53w"],
        "badges": [
          "Fish",
          "Initiate",
          "Veteran"
        ]
      },
      "handlers": {
        "count": "count_based",

```

```

        "procent": "procent_based",
        "time": "time_based"
    },
    "wifi": {
        "tables": ["wifi", "sensor_wifi"],
        "count": {
            "badges": [
                "Wave",
                "Flood",
                "Tsunami"
            ]
        }
    },
    "network_data": {
        "no_defaults": true,

        "percent": {
            "table": "network_traffic",
            "thresholds": {
                "receiver": 0.9,
                "sender": 0.5
            },
            "badges": {
                "receiver": "Leecher",
                "sender": "Seeder"
            }
        },
        "handlers": {
            "percent": "network_percent_data"
        }
    }
}

```

Zawartość "defaults" jest kopiowana do każdego osiągnięcia. Zawartość JSONowych obiektów jest kopiowana rekursywnie, wszystkie inne wartości są natomiast kopiowane tylko wtedy jeśli w osiągnięciu nie ma pola o tej nazwie. Przenoszeniu domyślnej części konfiguracji można zapobiec umieszczając pole "no\_defaults": true.

Zawartość "tables" wewnątrz osiągnięć kopiowana jest poziom niżej do sekcji "count", "time", "percent", itd. Zapobiec temu można definiując w każdej z tych sekcji własne pole "tables".

Obiekt "handlers" w konfiguracji ma specjalne znaczenie. Określa jakie typy osiągnięcia są obsługiwane.



Przykładowo odpytując serwer poprzez

GET /users/:id/achievements/wifi?filter=count

szukany jest w konfiguracji dla osiągnięcia wifi typ count. Osiągnięcie wifi powinno zatem zawierać minimalnie następujące informacje:

```
{
  "count": {
    "tables": []
  },
  "handlers": {
    "count": "count_based"
  }
}
```

Powoduje to wyszukanie w rejestrze handlerów dla osiągnięć użytkownika - handlers.user.achievements - handlera dla "count\_based" i przekazania mu konfiguracji "count".