

# **MyMonit**

## **Overview of the solution**

**May, 20th, 2022**



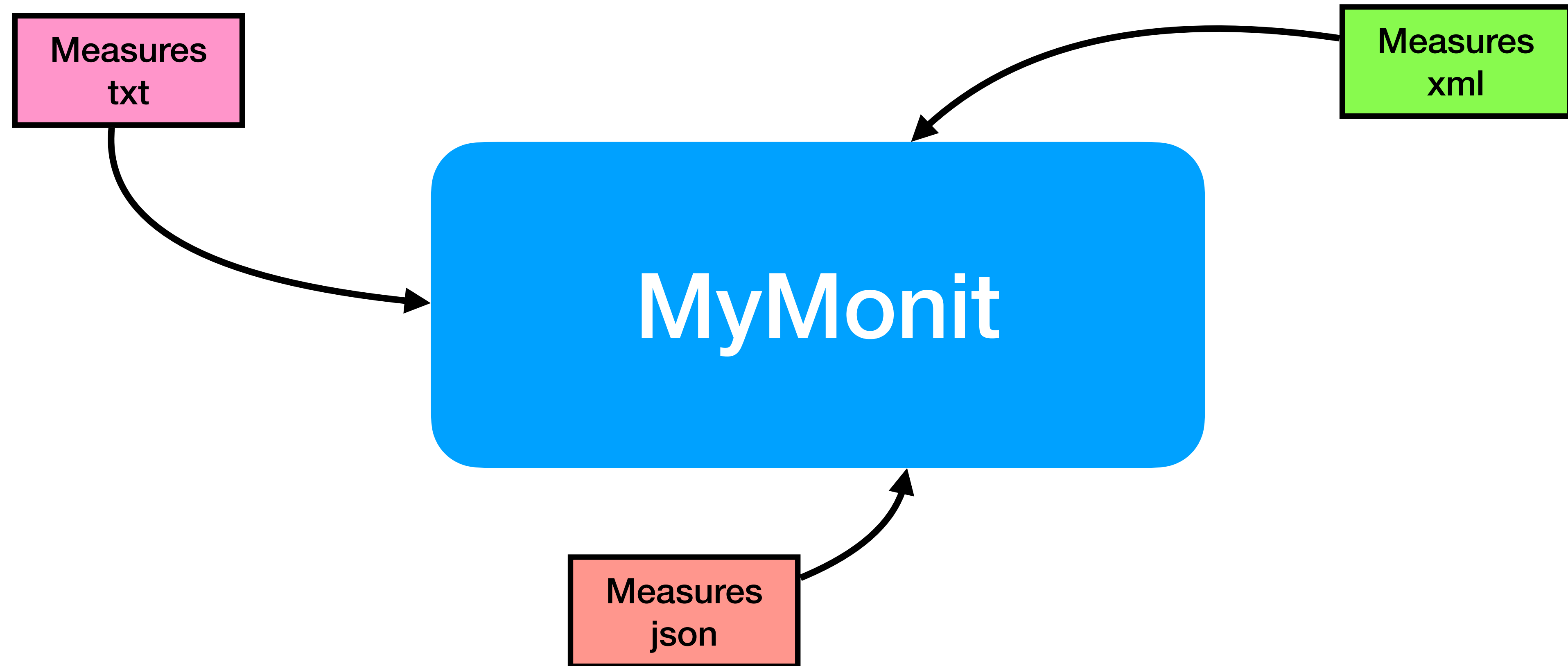
# The problem

Measures  
txt

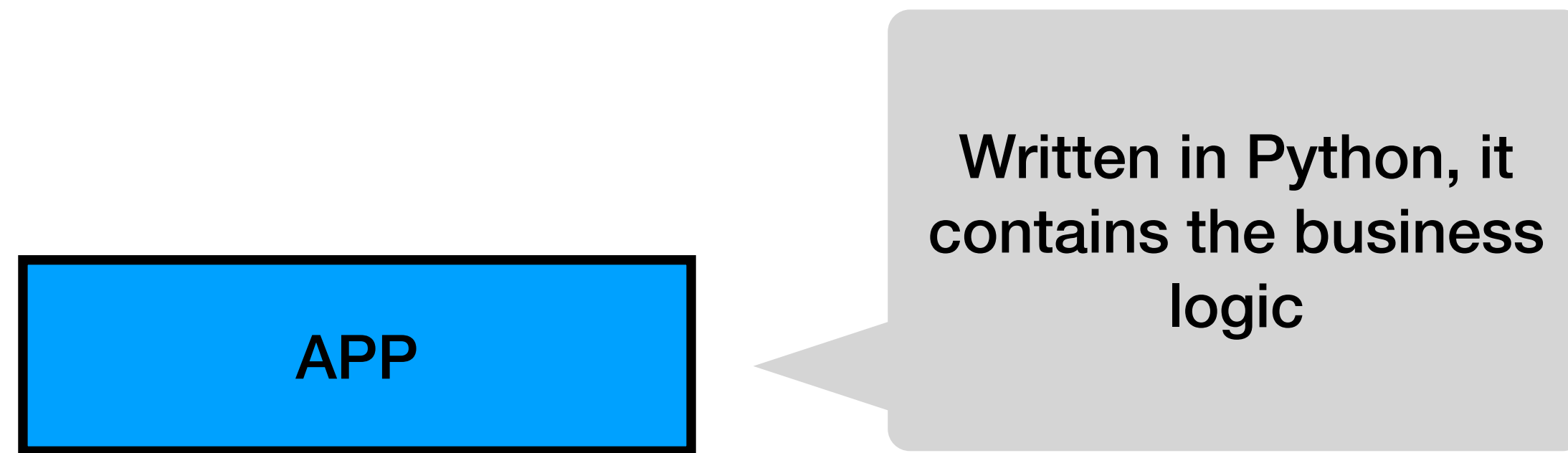
Measures  
xml

Measures  
json

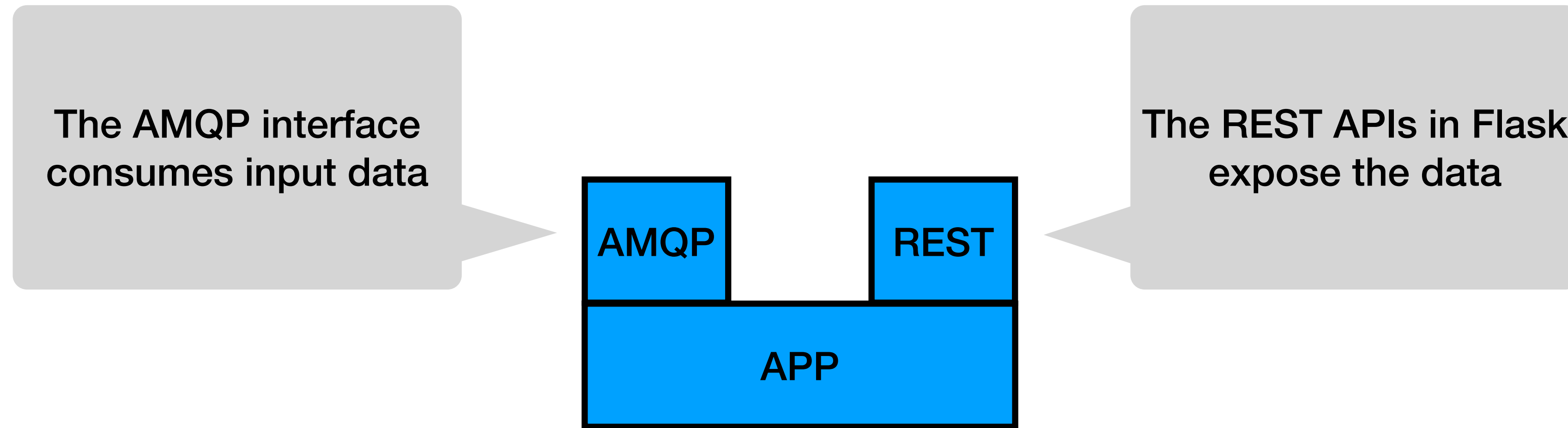
# The solution



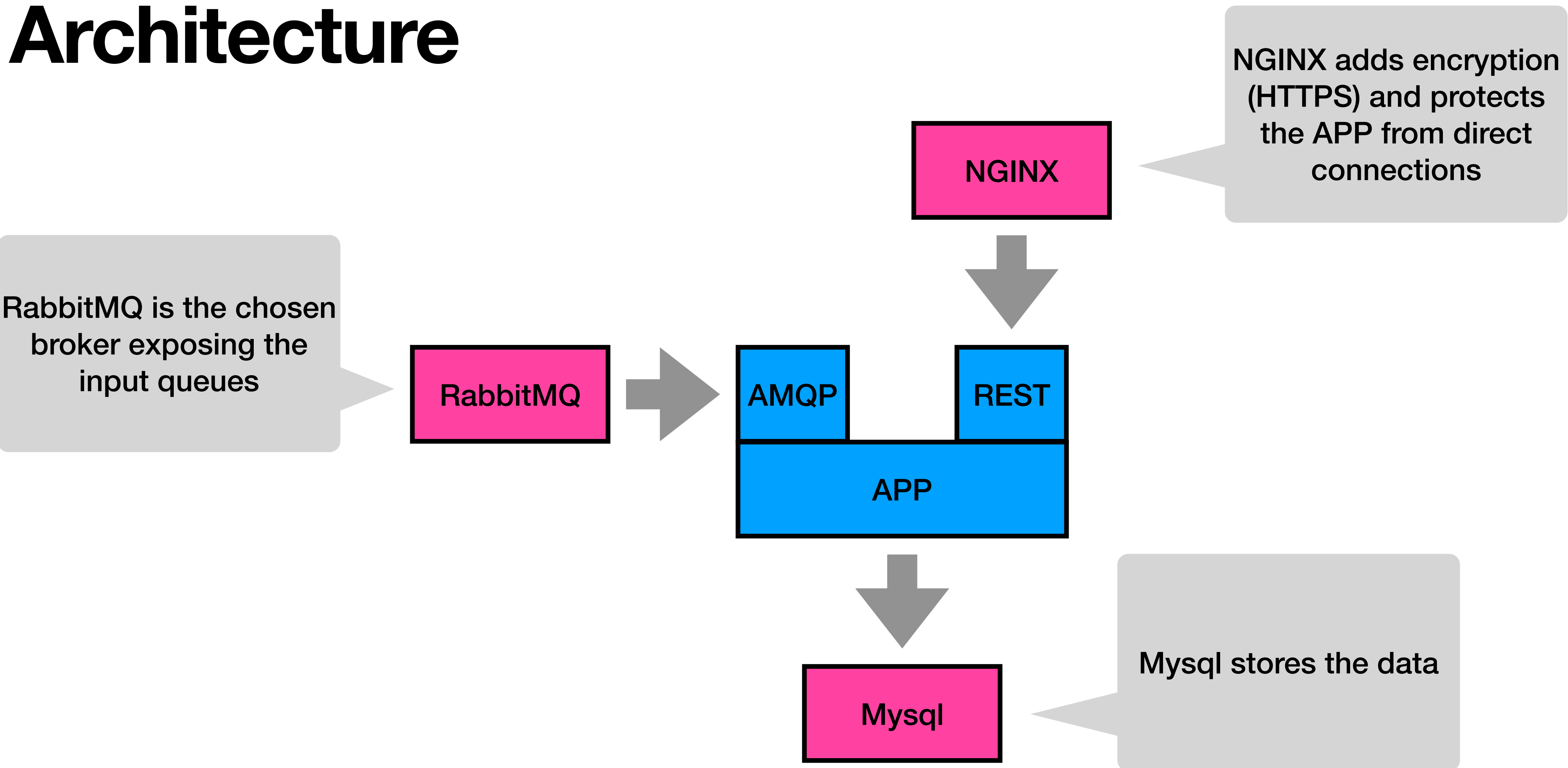
# Architecture



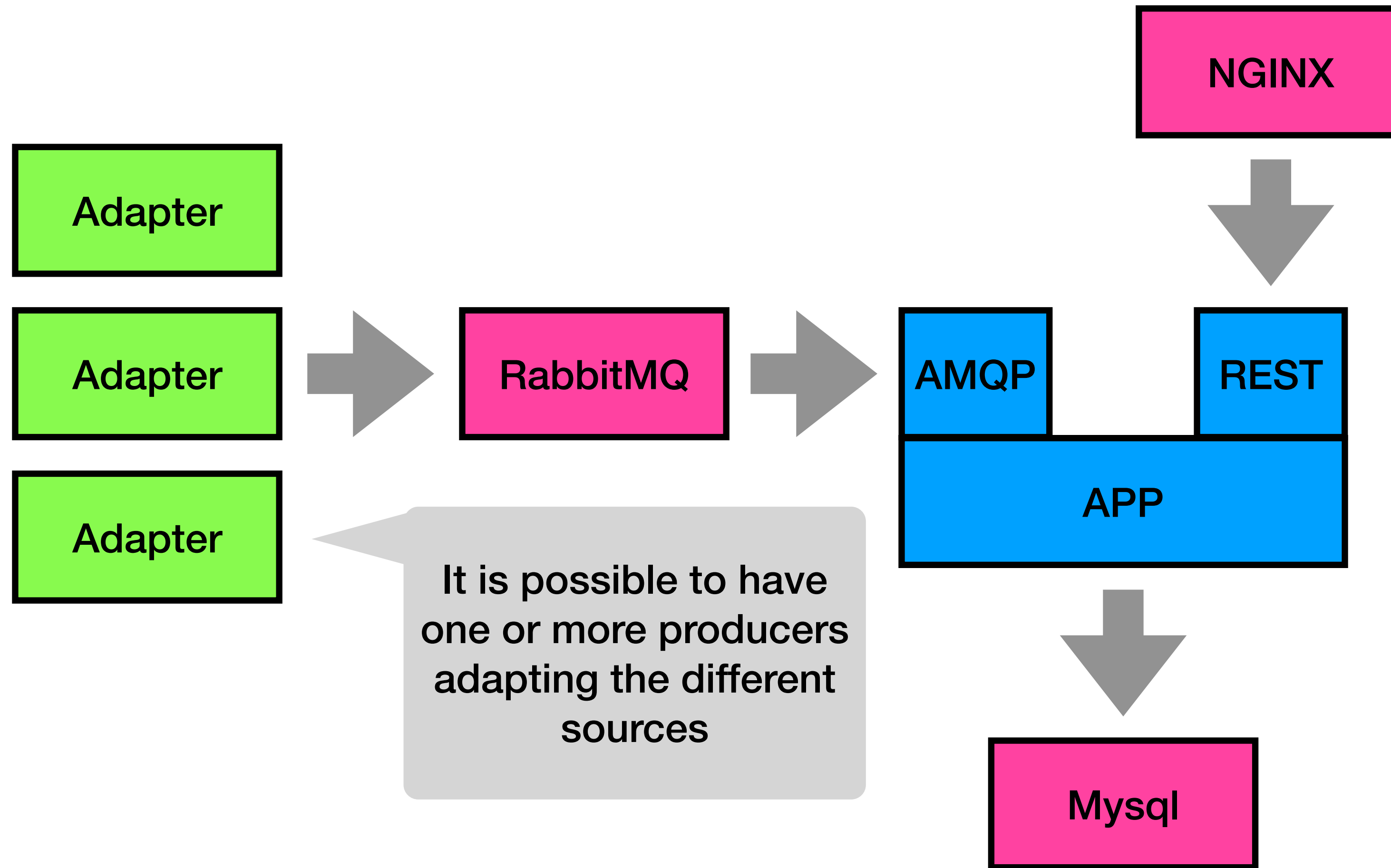
# Architecture



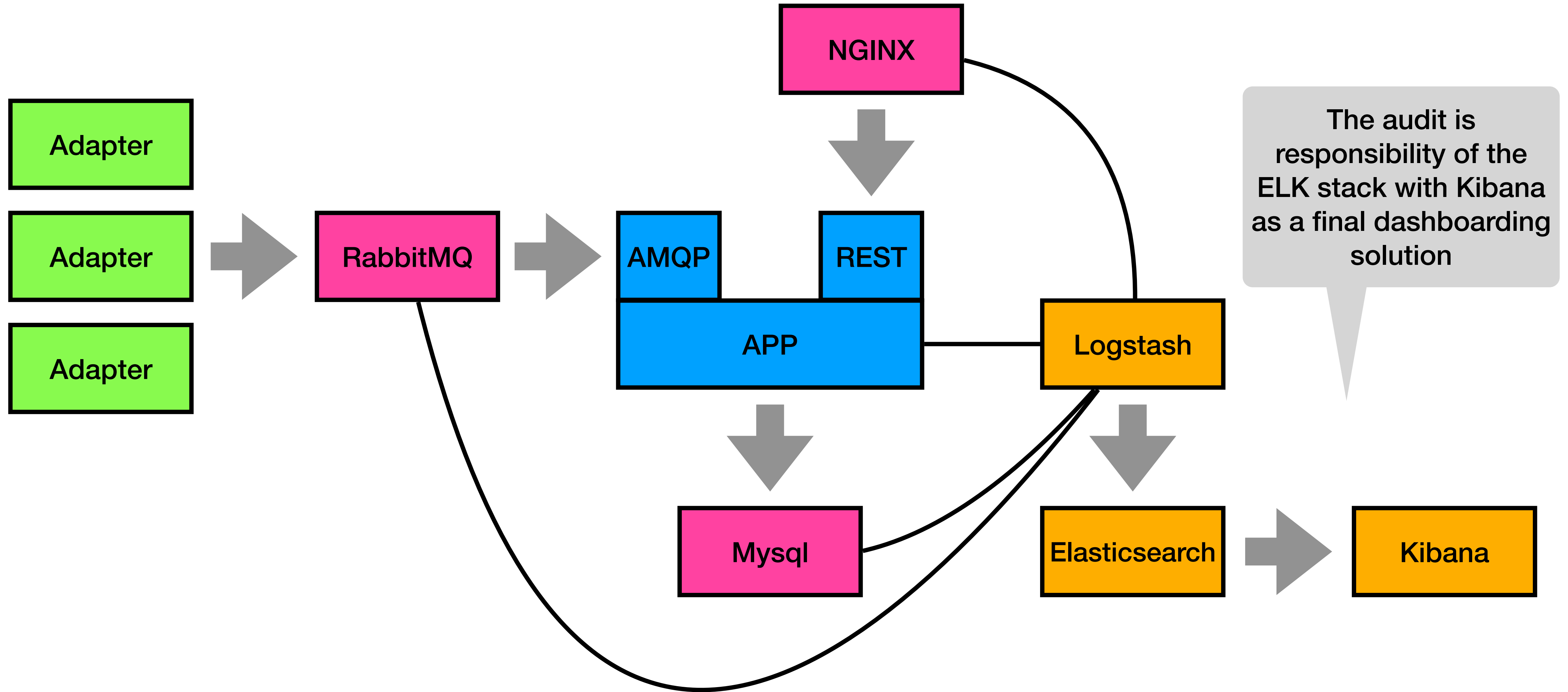
# Architecture



# Architecture

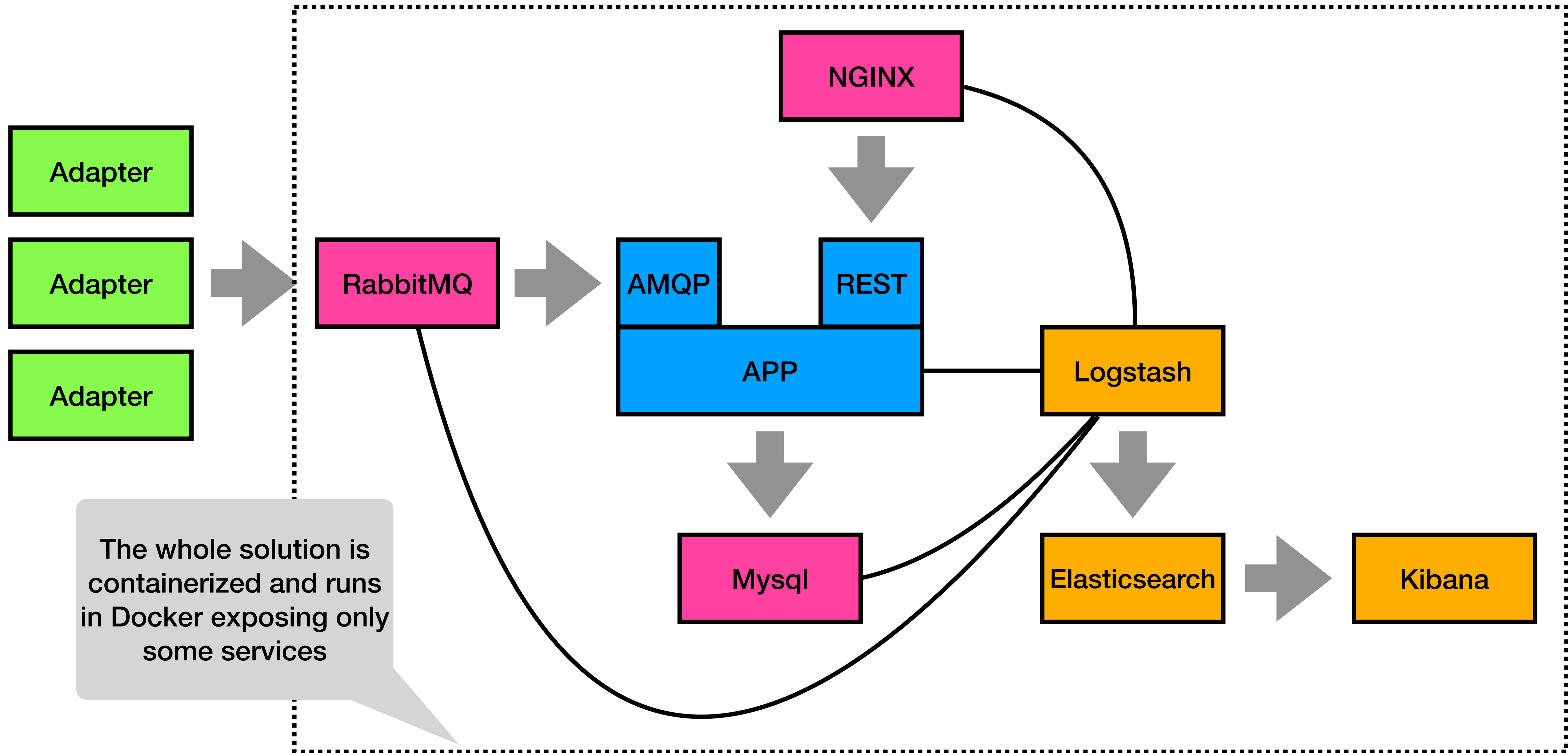


# Architecture

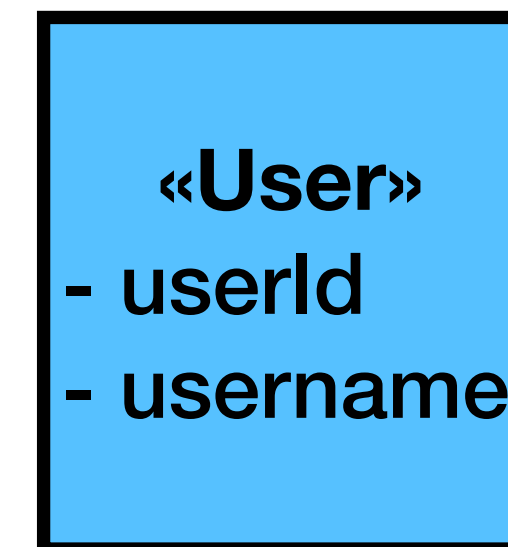
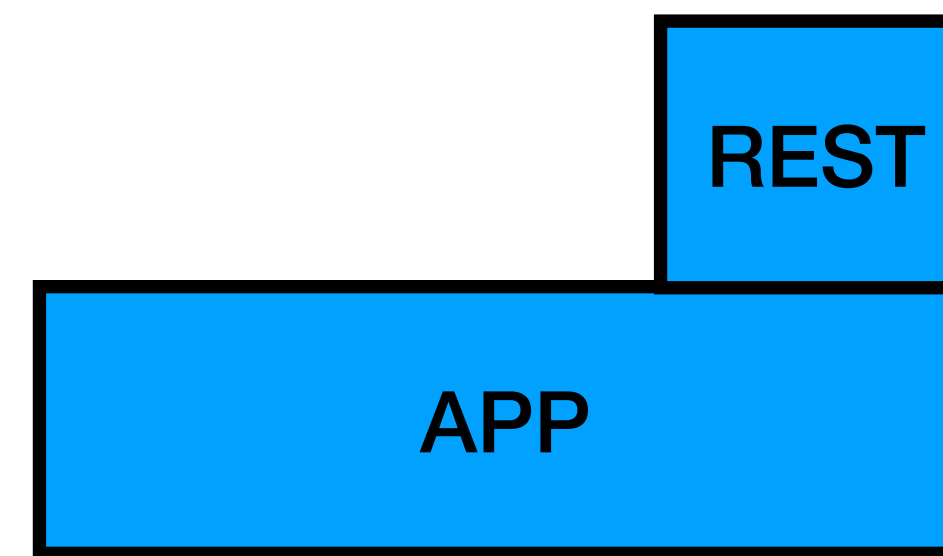
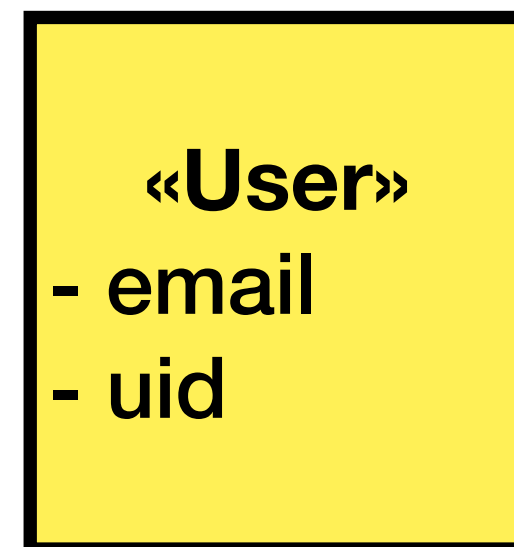




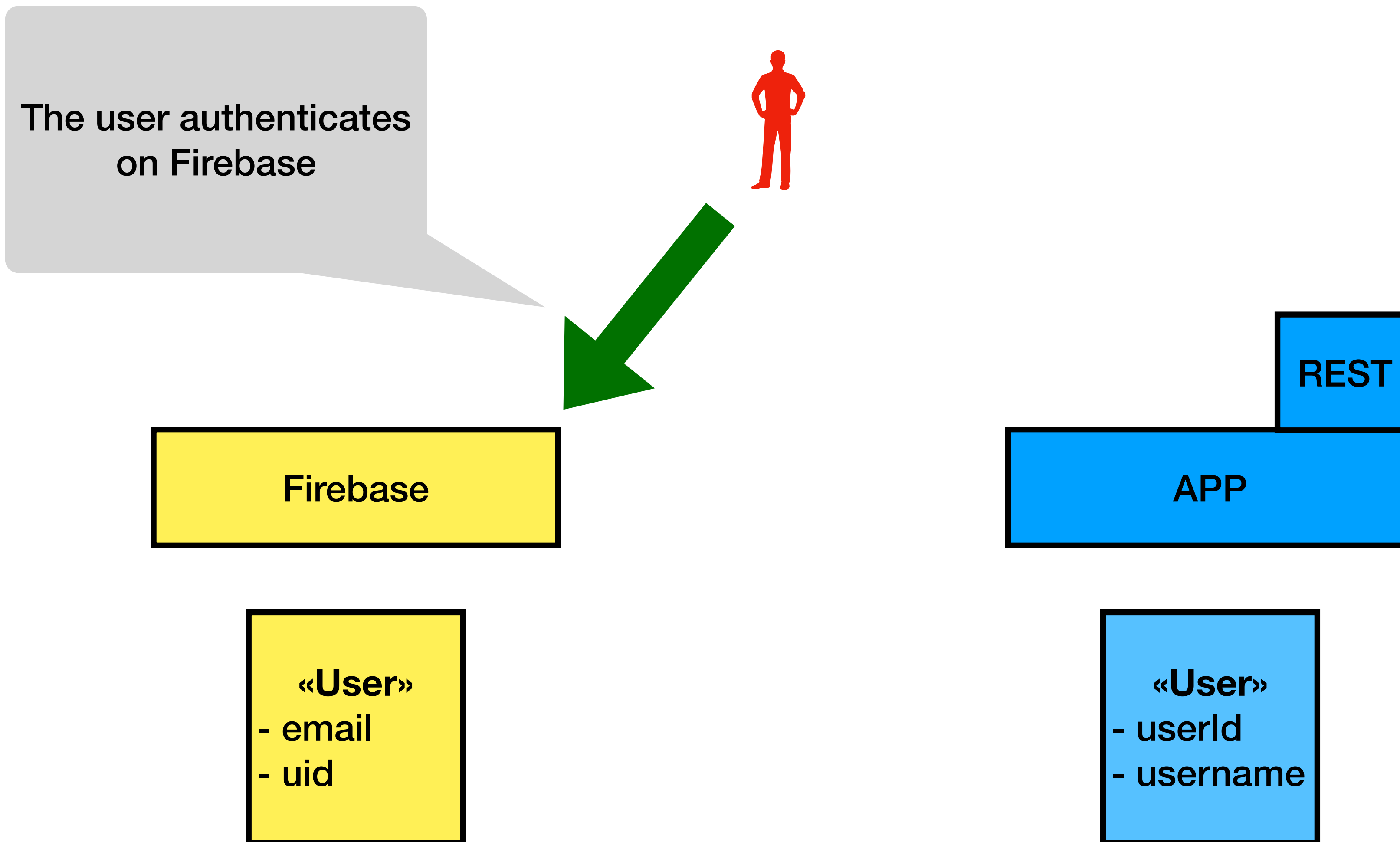
# Architecture



# Authentication

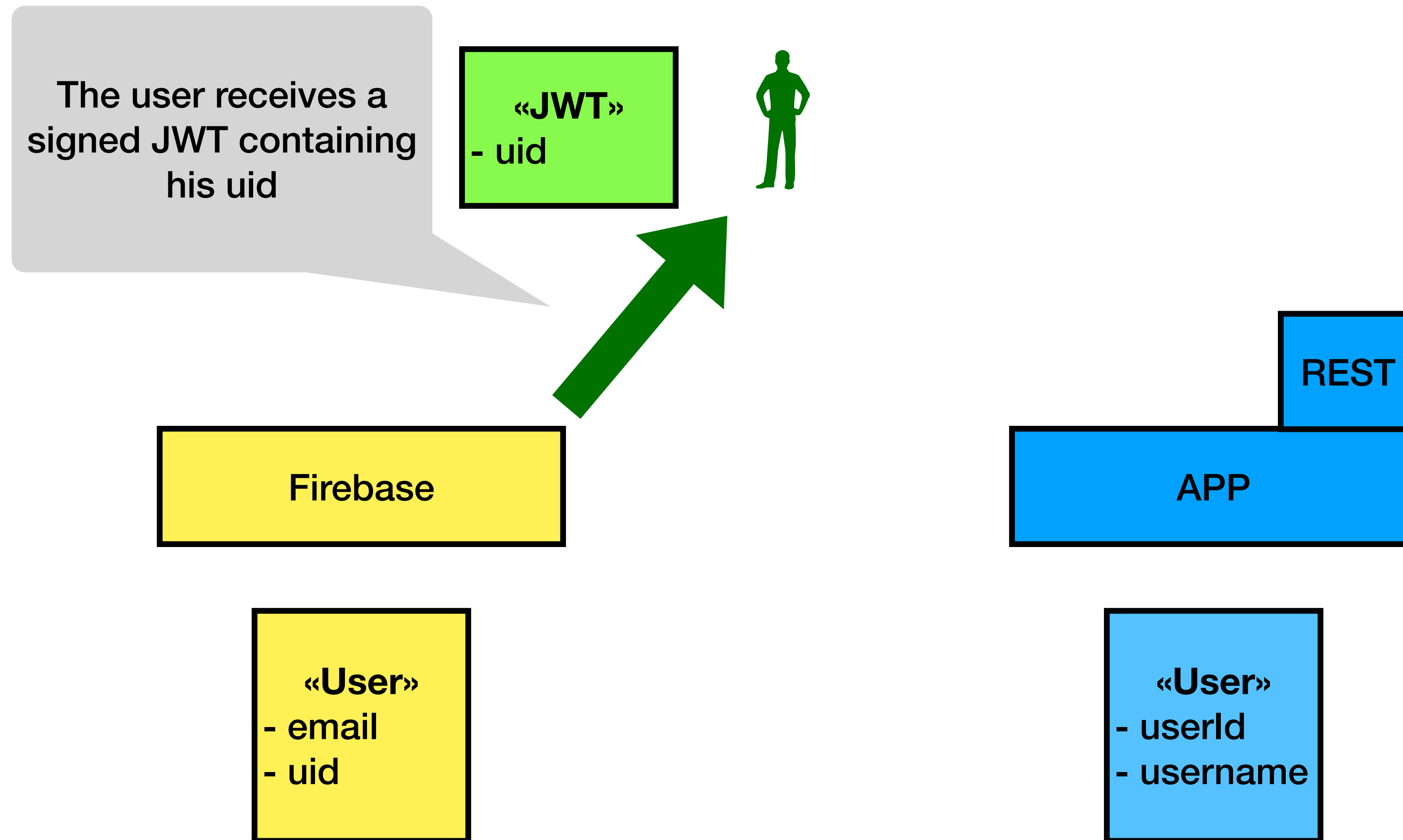


# Authentication

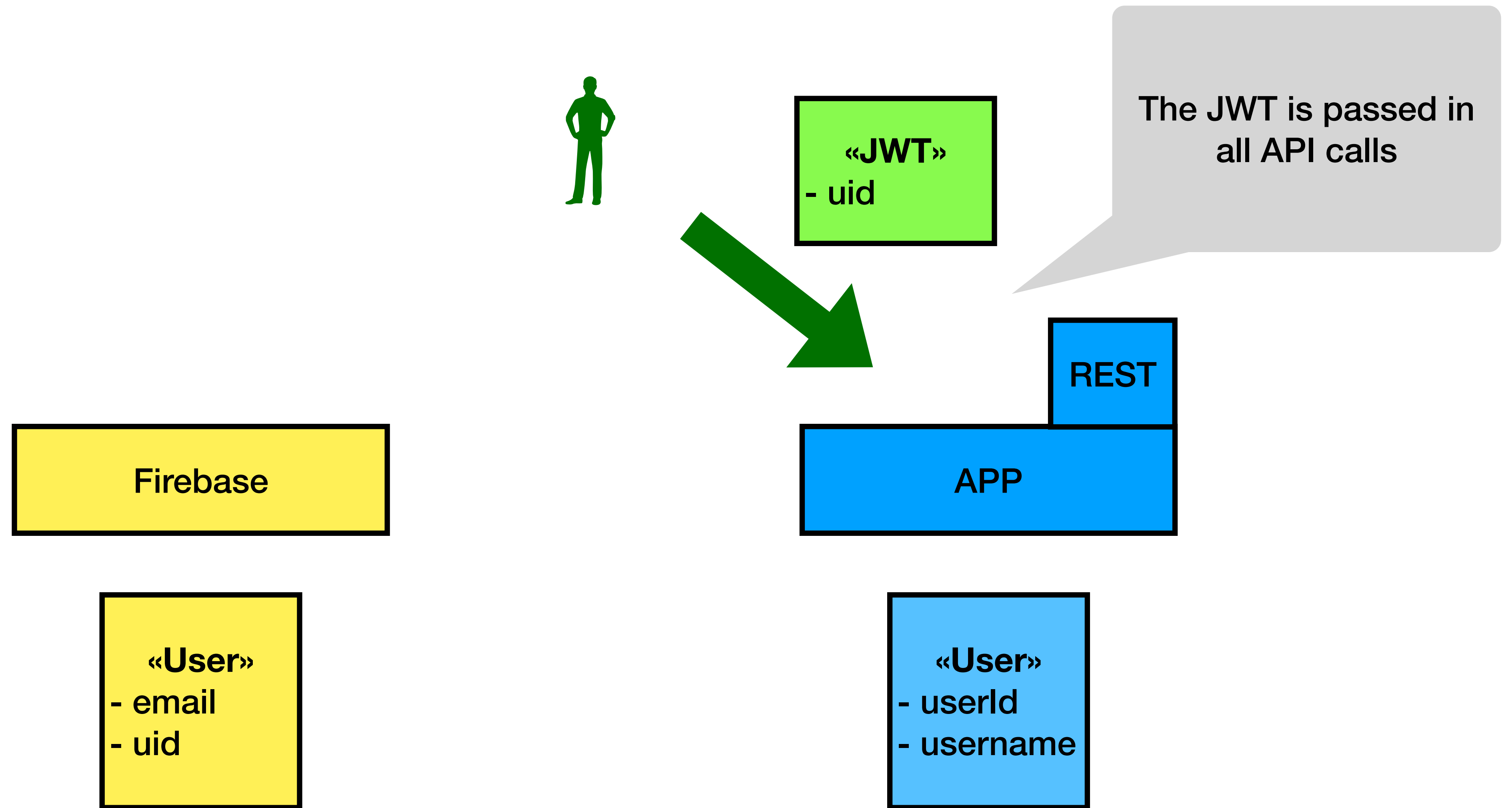




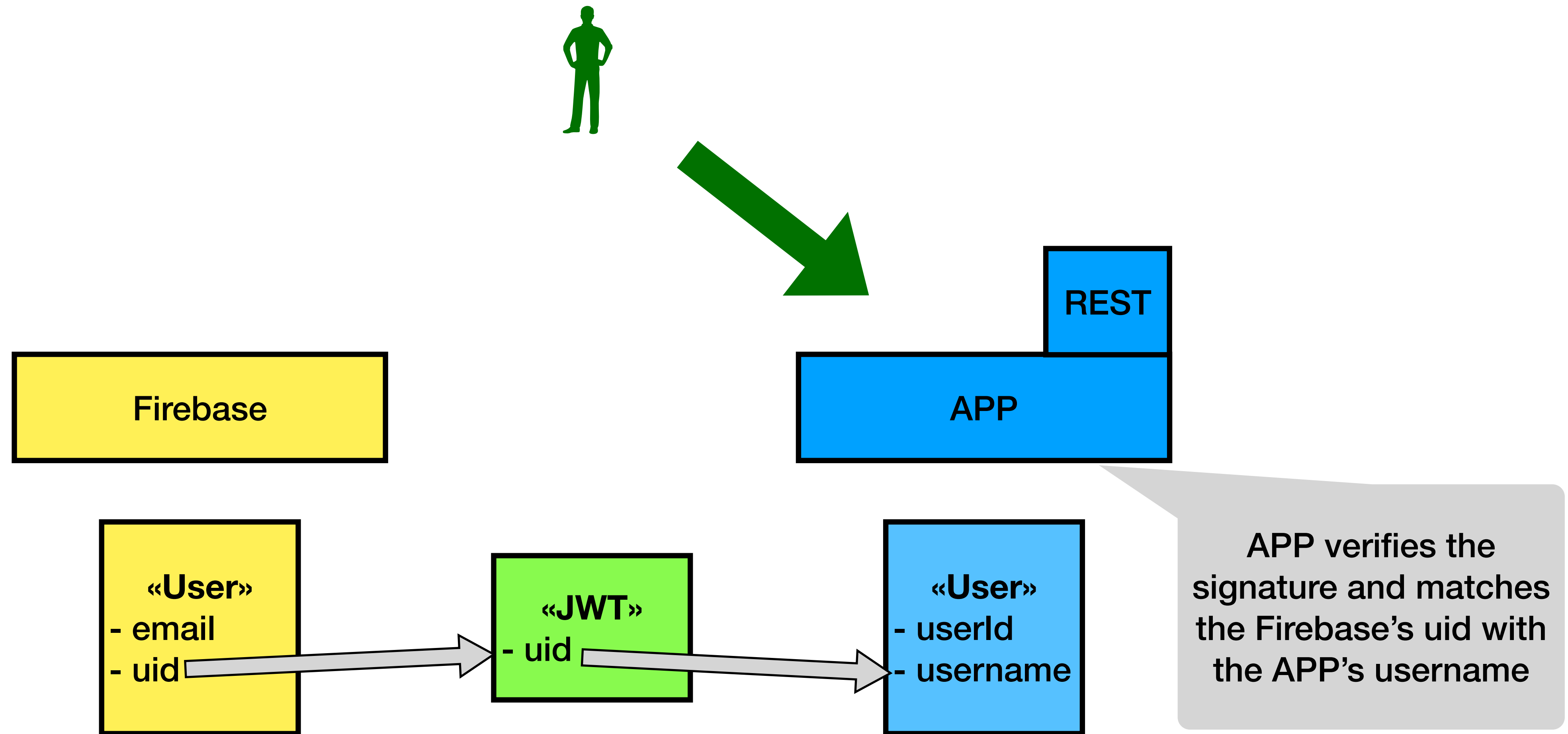
# Authentication



# Authentication

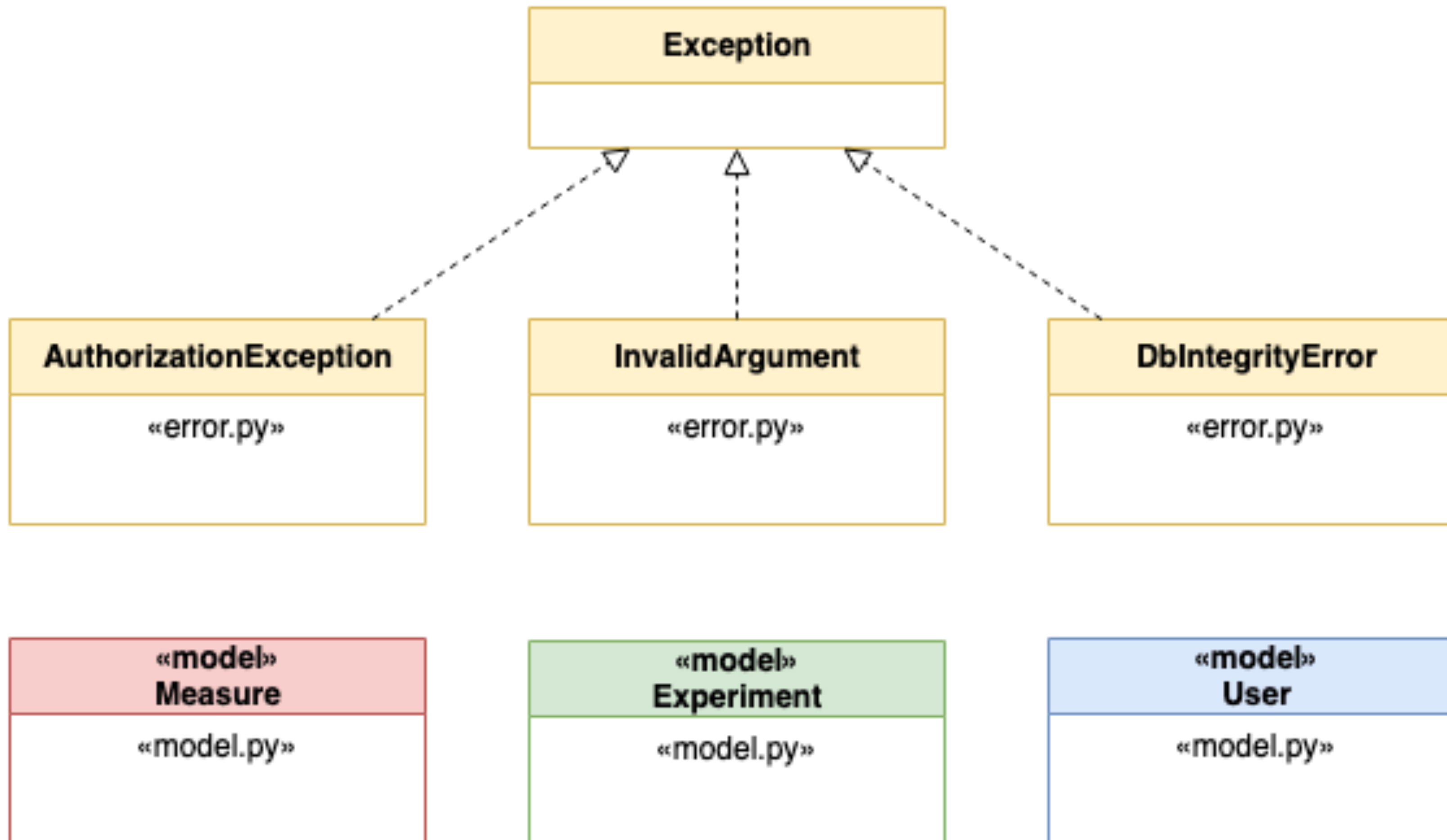


# Authentication

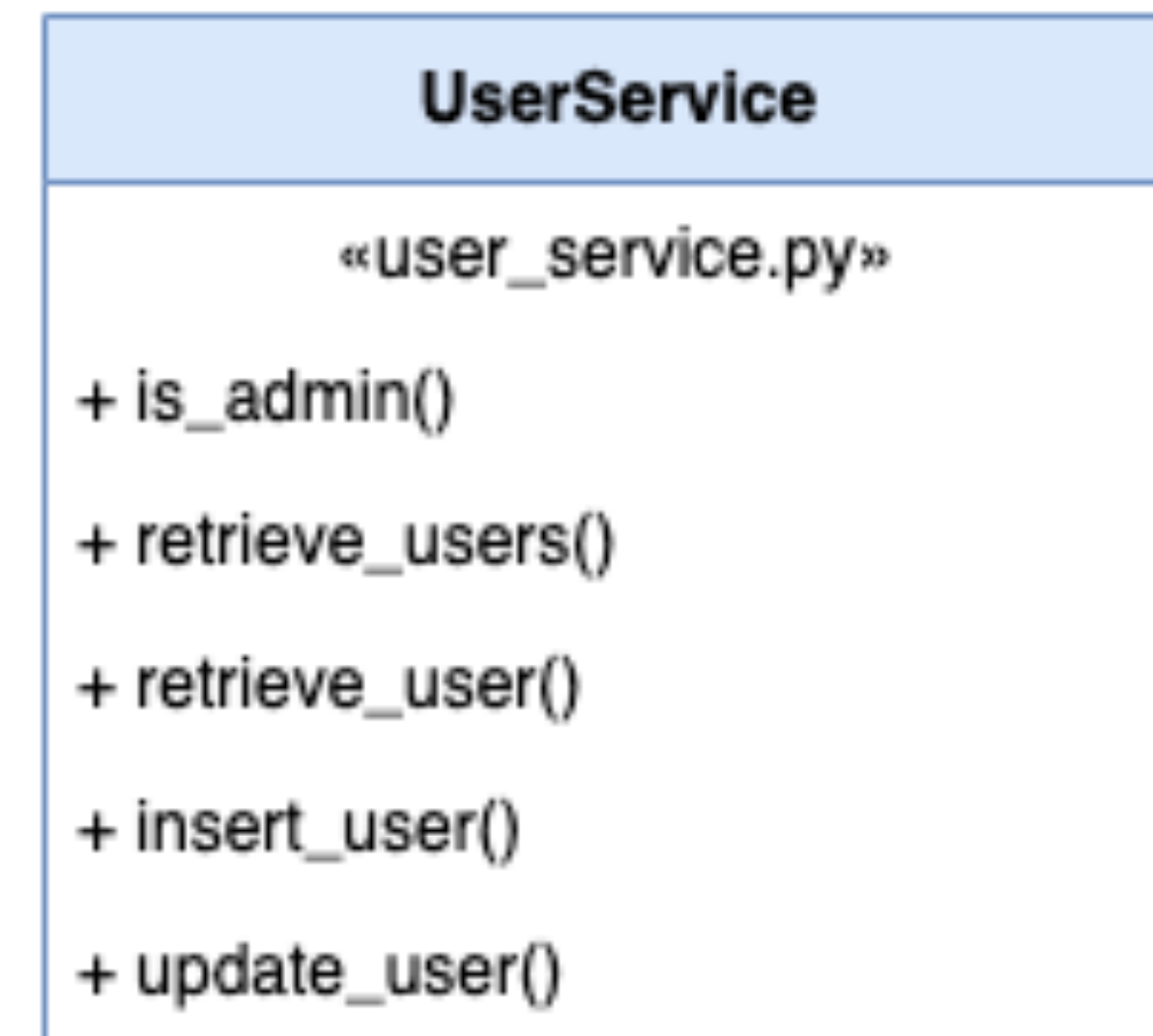
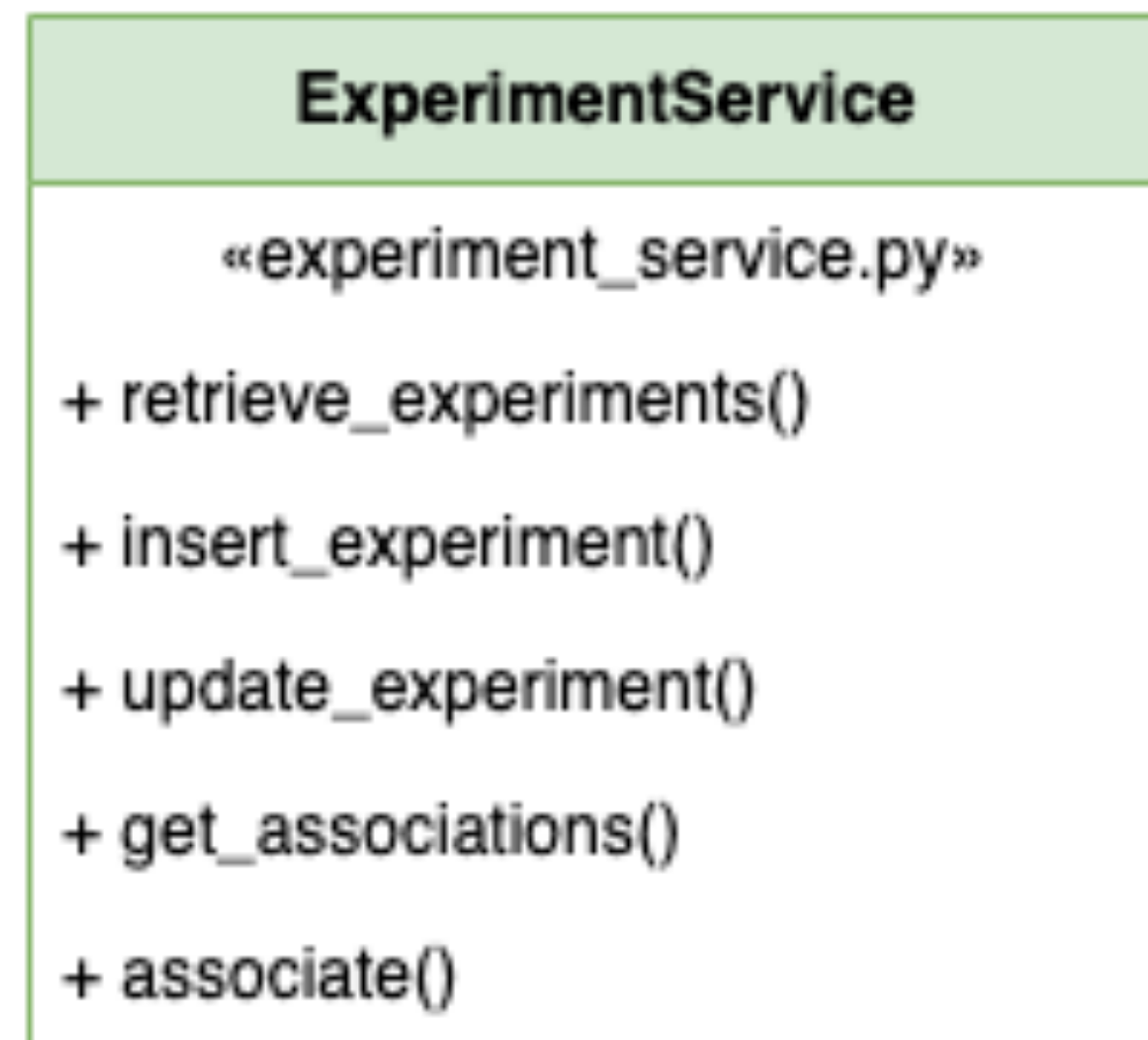
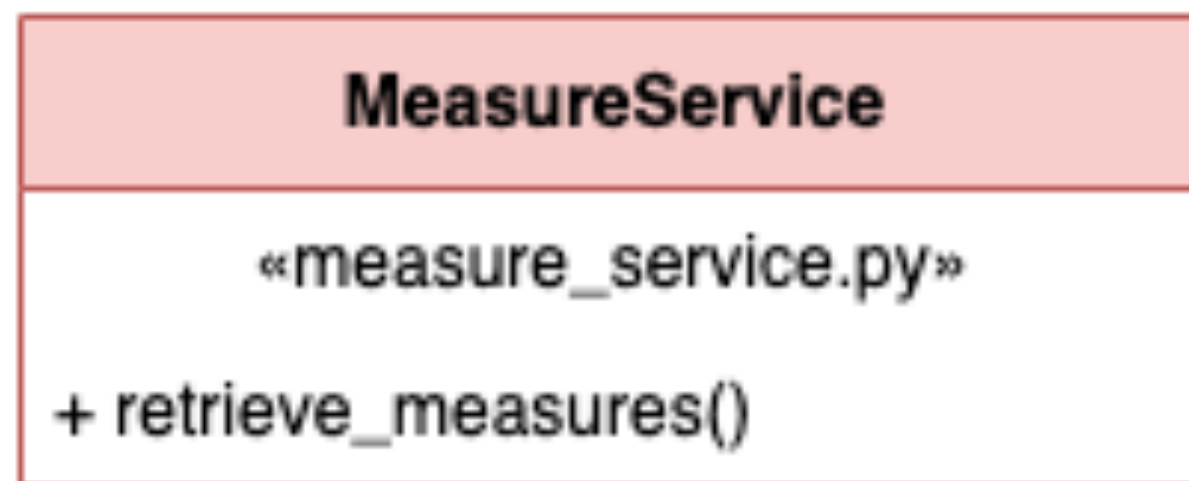




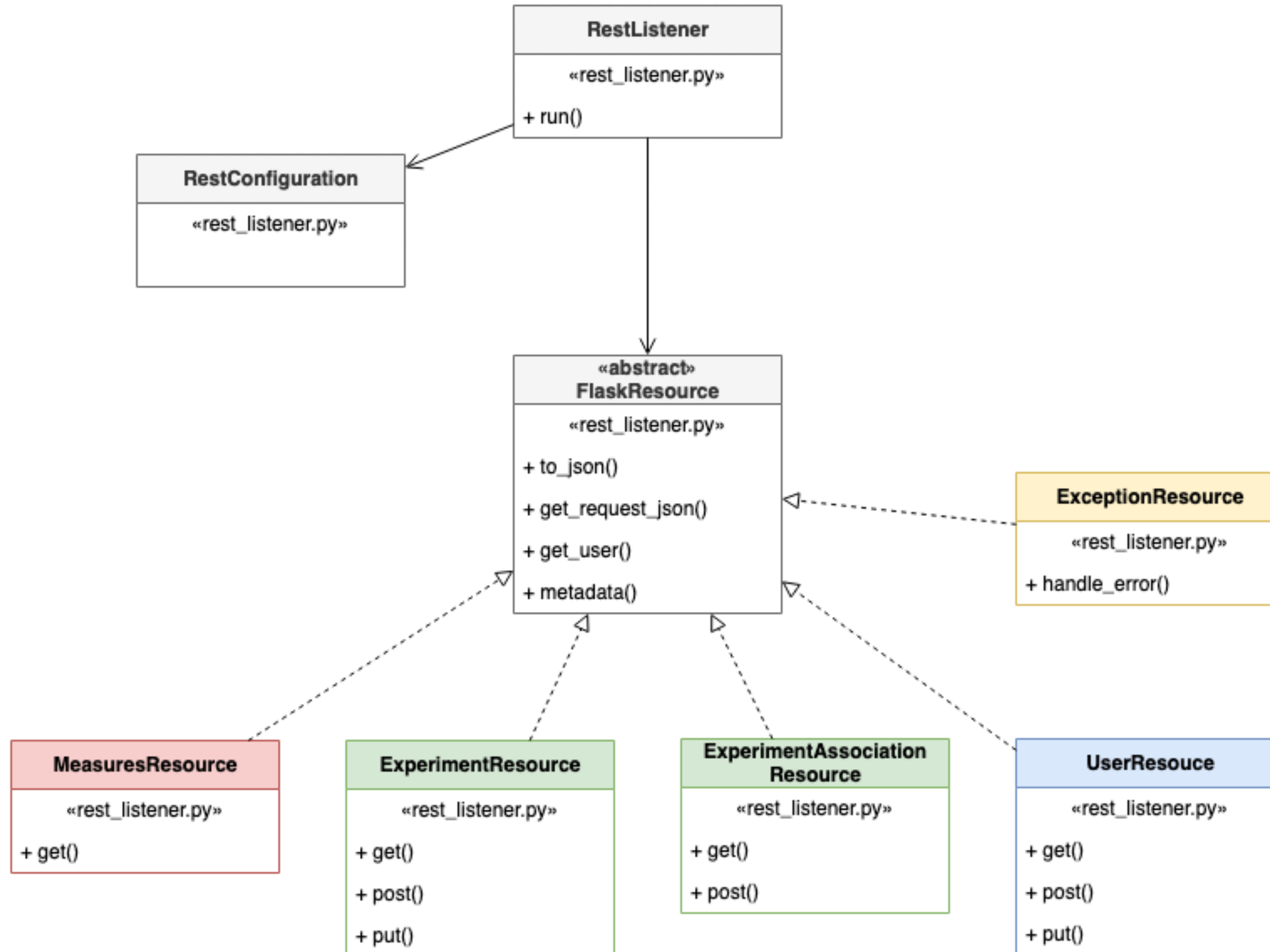
# Architecture



# Architecture

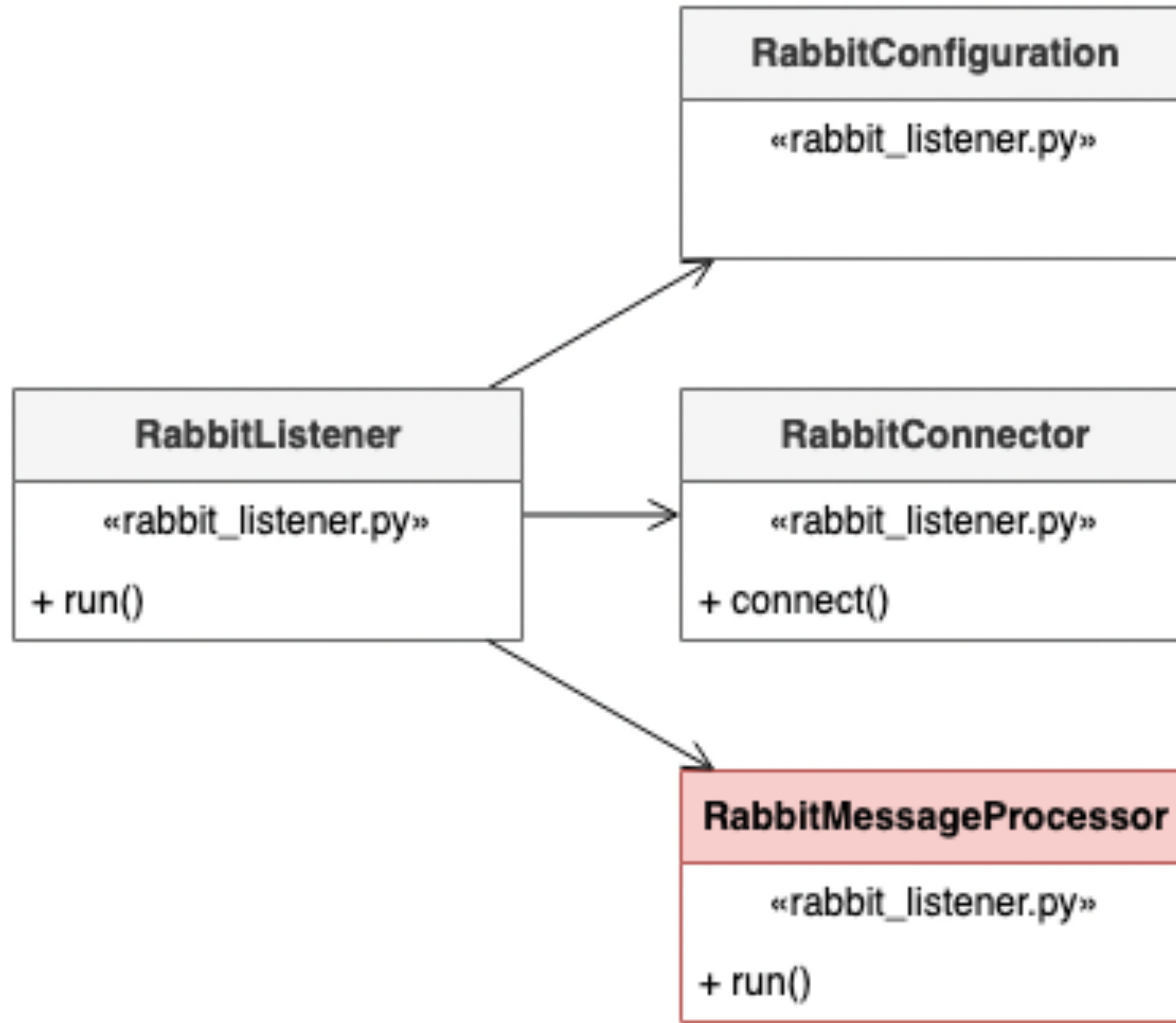


# Architecture

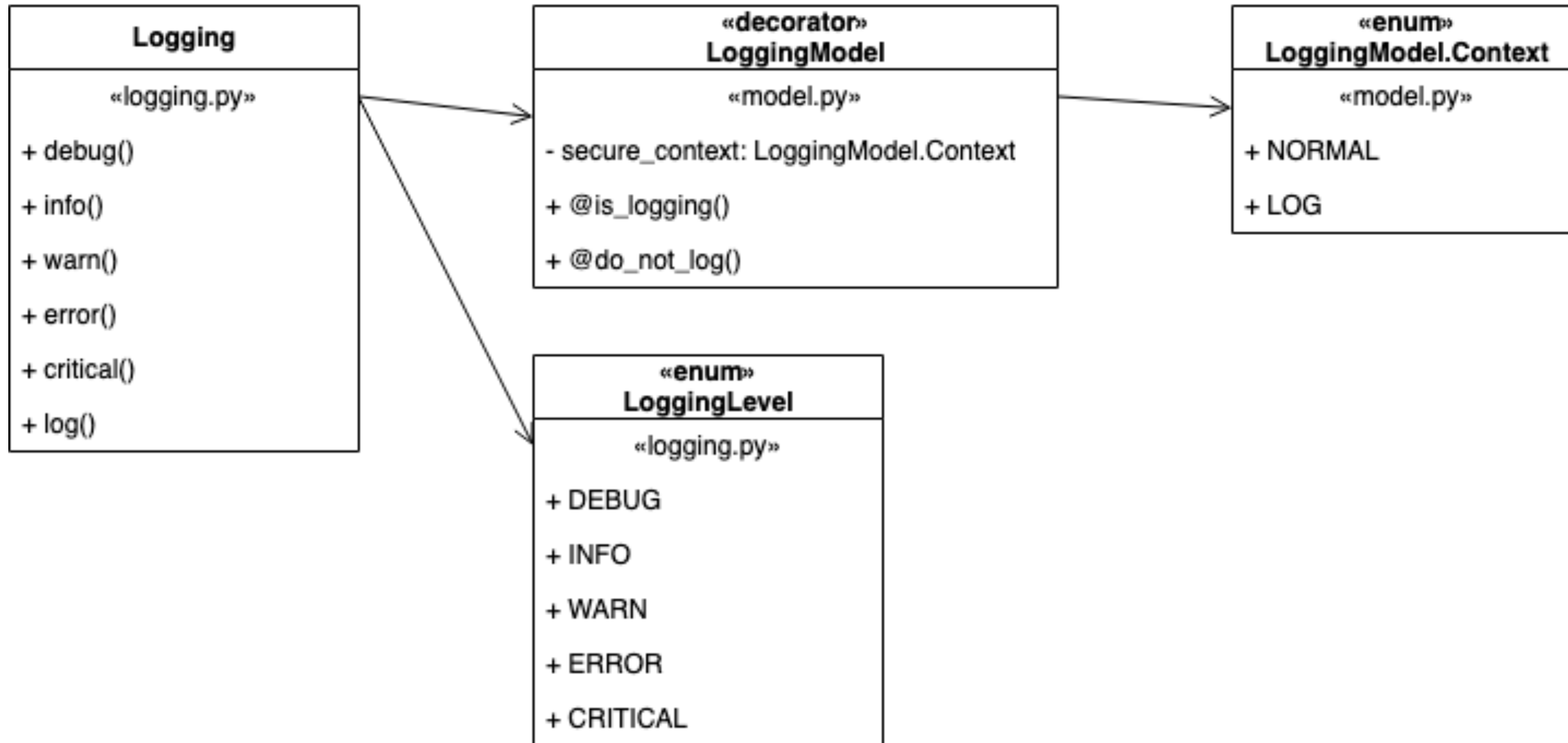




# Architecture



# Architecture



# Architecture

```
@staticmethod
def is_logging(func):
    def inner(*args, **kwargs):
        LoggingModel._set_context_value(LoggingModel.Context.LOG)
        func(*args, **kwargs)
        LoggingModel._set_context_value(LoggingModel.Context.NORMAL)
    return inner
```

```
@staticmethod
def do_not_log(func):
    def inner(*args, **kwargs):
        return func(*args, **kwargs) if LoggingModel._get_context_value() == \
            LoggingModel.Context.NORMAL else '***'
    return inner
```

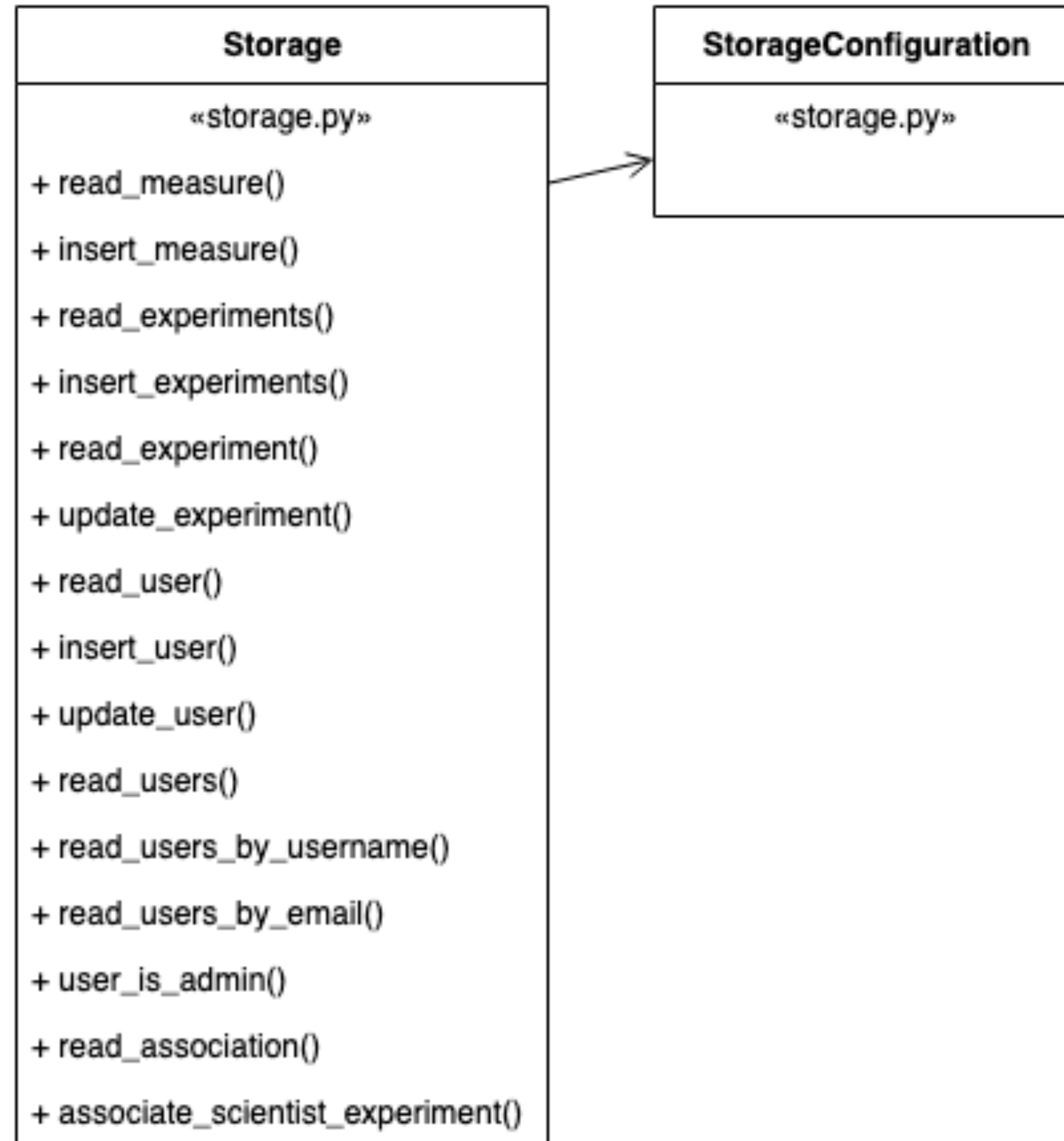
```
@staticmethod
def _get_context():
    return LoggingModel.Context.secure_context

@staticmethod
def _get_context_value():
    thread_id = threading.get_native_id()
    context = LoggingModel._get_context()
    if thread_id not in context:
        context[thread_id] = LoggingModel.Context.NORMAL
    return context[thread_id]

@staticmethod
def _set_context_value(value: Context):
    thread_id = threading.get_native_id()
    context = LoggingModel._get_context()
    context[thread_id] = value
```



# Architecture



# Architecture

Container
«app.py»
- config: dependency_injection.provider.Configuration
+ main()
+ init()

```
def init():
    '''initialise the application'''
    container = Container()

    # db
    container.config.db_user.from_env('DB_USER', default = 'root', as_ = str)
    container.config.db_password.from_env('DB_PASSWORD', default = 'password', as_ = str)
    container.config.db_host.from_env('DB_HOST', default = 'localhost', as_ = str)
    container.config.db_database.from_env('DB_DATABASE', 'my_monit', as_ = str)
    # rabbit
    container.config.rabbit_url.from_env('RABBIT_URL', default = 'localhost', as_ = str)
    container.config.rabbit_user.from_env('RABBIT_USER', default = 'guest', as_ = str)
    container.config.rabbit_password.from_env('RABBIT_PASSWORD', default = 'guest', as_ = str)
    container.config.rabbit_exchange.from_env('RABBIT_EXCHANGE', default = 'mymonit', as_ = str)
    container.config.rabbit_routing.from_env('RABBIT_ROUTING', default = 'measures', as_ = str)
    container.config.rabbit_queue.from_env('RABBIT_QUEUE', default = 'measures', as_ = str)
    # rest
    container.config.rest_host.from_env('REST_HOST', default = '0.0.0.0', as_ = str)
    # logstash
    container.config.logstash_host.from_env('LOGSTASH_HOST', default = 'localhost', as_ = str)
    container.config.logstash_port.from_env('LOGSTASH_PORT', default = 5959, as_ = int)
    # firebase
    if 'GOOGLE_APPLICATION_CREDENTIALS' not in os.environ:
        # os.getcwd() is the whole project's root
        os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = './containers/app/private_key.json'
```

```
logging = providers.Singleton(
    Logging,
    host = config.logstash_host,
    port = config.logstash_port
)

storage_configuration = providers.Singleton(
    StorageConfiguration,
    db_user = config.db_user,
    db_password = config.db_password,
    db_host = config.db_host,
    db_database = config.db_database
)

rabbit_configuration = providers.Singleton(
    RabbitConfiguration,
    rabbit_url = config.rabbit_url,
    rabbit_user = config.rabbit_user,
    rabbit_password = config.rabbit_password,
    rabbit_exchange = config.rabbit_exchange,
    rabbit_routing = config.rabbit_routing,
    rabbit_queue = config.rabbit_queue
)

rest_configuration = providers.Singleton(
    RestConfiguration,
    rest_host = config.rest_host
)

storage_connector = providers.Singleton(
    StorageConnector,
    storage_configuration = storage_configuration,
    logging = logging
)

storage = providers.Singleton(
    Storage,
    storage_connector = storage_connector,
    logging = logging
)
```

# Build process

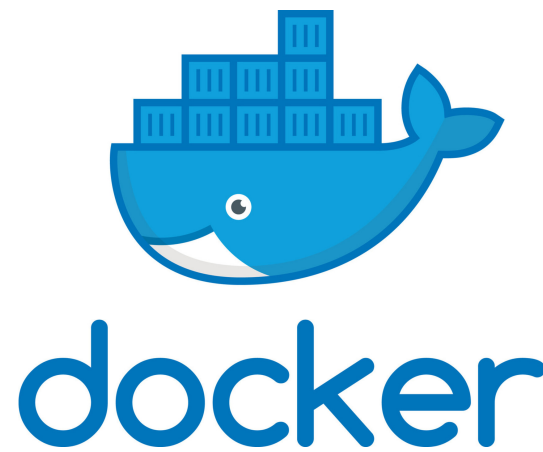
```
1  '''Setup the project'''
2
3  from setuptools import setup, find_packages
4
5  setup(
6      name='my_monit',
7      version='0.1.0',
8      setup_requires=['pytest-runner', 'pytest-pylint'],
9      tests_require=['pytest', 'pylint'],
10     packages=find_packages(include=['my_monit']),
11     test_suite = 'test',
12     install_requires=[
13         "flask >= 2.0.0",
14         "pika >= 1.2.0",
15         "mysql-connector-python == 8.0.28",
16         "python-logstash >= 0.4.6",
17         "dependency_injector >= 4.39.1",
18         "pyjwt >= 2.3.0",
19         "colorama >= 0.4.0",
20         'firebase-admin>=5.2.0'
21     ],
22     python_requires='>=3.9'
23 )
24
```



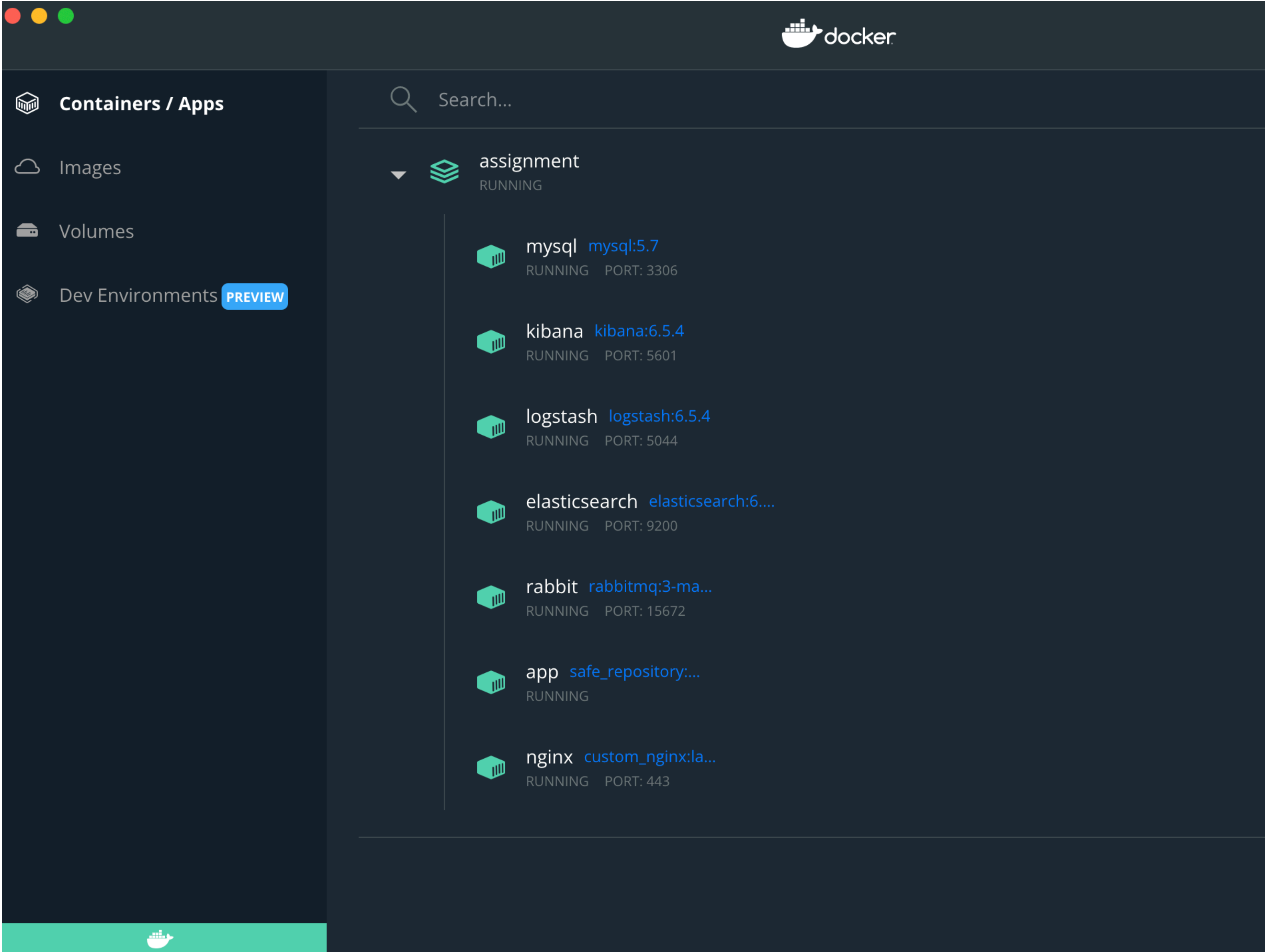
# Build process

```
1 FROM nginx:latest
2
3 COPY nginx.conf /etc/nginx/conf.d/default.conf
4 COPY key.pem /root/ssl/key.pem
5 COPY cert.pem /root/ssl/cert.pem
6 RUN rm /var/log/nginx/access.log && rm /var/log/nginx/error.log
7
8 RUN curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-8.1.0-amd64.deb
9 RUN dpkg -i filebeat-8.1.0-amd64.deb
10 COPY filebeat.yml /etc/filebeat/filebeat.yml
11 RUN filebeat modules enable nginx
12 COPY nginx.yml /etc/filebeat/modules.d/
13 COPY filebeat-restart.sh /
14 COPY docker-entrypoint.sh /
15 RUN chmod a+x /filebeat-restart.sh && chmod a+x /docker-entrypoint.sh
```

```
1 FROM python:3.9
2
3 COPY setup.* /home/
4 COPY main.py /home/
5 COPY .bandit /home/
6 COPY .static-tmp/* /home/static/
7 COPY my_monit/*.py /home/my_monit/
8 WORKDIR /home
9 RUN pip3 install -e .
10
11 ENTRYPOINT python3 main.py
12
```



```
10 container_name: rabbit
11 ports:
12   - "5672:5672"
13   - "15672:15672"
14
15 mysql:
16   # for M1
17   # platform: linux/x86_64
18   image: mysql:5.7
19   container_name: mysql
20   command: mysqld --sql_mode="" --character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci
21   volumes:
22     - ./containers/mysql/dbschema:/docker-entrypoint-initdb.d
23   ports:
24     - "3306:3306"
25   environment:
26     - MYSQL_ROOT_PASSWORD=password
27
28 elasticsearch:
29   image: elasticsearch:6.5.4
30   container_name: elasticsearch
31   environment:
32     - node.name=ws-es-node
33     - discovery.type=single-node
34     - cluster.name=ws-es-data-cluster
35     - bootstrap.memory_lock=true
36     - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
37   ulimits:
38     memlock:
39       soft: -1
39
```



# Quality control - coverage

Coverage report: 81%

<i>Module</i>	<i>statements</i>	<i>missing</i>	<i>excluded</i>	<i>coverage</i>
my_monit/__init__.py	0	0	0	100%
my_monit/app.py	57	0	0	100%
my_monit/errors.py	9	3	0	67%
my_monit/experiment_service.py	45	6	0	87%
my_monit/logging.py	48	2	0	96%
my_monit/measure_service.py	12	0	0	100%
my_monit/model.py	115	7	0	94%
my_monit/rabbit_listener.py	71	12	0	83%
my_monit/rest_listener.py	157	49	0	69%
my_monit/storage.py	107	42	0	61%
my_monit/user_service.py	50	6	0	88%
<b>Total</b>	<b>671</b>	<b>127</b>	<b>0</b>	<b>81%</b>



# Quality control - pylint

```
***** Module my_monit.logging
my_monit/logging.py:78:15: W0703: Catching too general exception Exception (broad-except)

-----
Your code has been rated at 9.98/10 (previous run: 9.16/10, +0.83)
```

# Quality control - bandit

```
Test results:
>> Issue: [B104:hardcoded_bind_all_interfaces] Possible binding to all interfaces.
    Severity: Medium    Confidence: Medium
    CWE: CWE-605 (https://cwe.mitre.org/data/definitions/605.html)
    Location: my_monit/app.py:138:63
    More Info: https://bandit.readthedocs.io/en/1.7.4/plugins/b104\_hardcoded\_bind\_all\_interfaces.html
137         # rest
138         container.config.rest_host.from_env('REST_HOST', default = '0.0.0.0', as_ = str)
139         # logstash


-----

Code scanned:
    Total lines of code: 1189
    Total lines skipped (#nosec): 0
    Total potential issues skipped due to specifically being disabled (e.g., #nosec BXXX): 0

Run metrics:
    Total issues (by severity):
        Undefined: 0
        Low: 0
        Medium: 1
        High: 0
    Total issues (by confidence):
        Undefined: 0
        Low: 0
        Medium: 1
        High: 0
Files skipped (0):
```



# Quality control - swagger

 **Swagger**  
Supported by SMARTBEAR

Explore


# MyMonit

1.0.0

OAS3

Servers

http://{url} ▾


Authorize 

users ▾

POST

/users/


create users



GET

/users/


retrieve users



PUT

/users/{user\_id}

update user




experiments ▾

GET

/experiments/


retrieve experiment



POST

/experiments/


create experiment



PUT

/experiments/{experiment\_id}

update experiment



measures ▾

# Quality control - postman

Scratch Pad

NewImport

Collections

APIs

Environments

Mock Servers

Monitors

History

+

☰

...

MyMonit

users

POST create users

PUT update user

GET retrieve users

experiments

GET retrieve experiment

POST create experiment

PUT update experiment

measures

GET retrieve measures

scientist/experiment

GET retrieve scientists/ experiment

POST associate scientists to experi...

PUT update userlocalhostGET retrieve userGET retrieve experPOST create user+...localhost

MyMonit / users / create users

Save

Send

POST

{{url}}/users/

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Cookies

Beautify

1

2

3

4

5

{

.... "user\_id": "S123",

.... "name": "Foo Bar",

.... "username": "firebaseUid",

.... "email": "foo@bar.com",

Response



Click Send to get a response