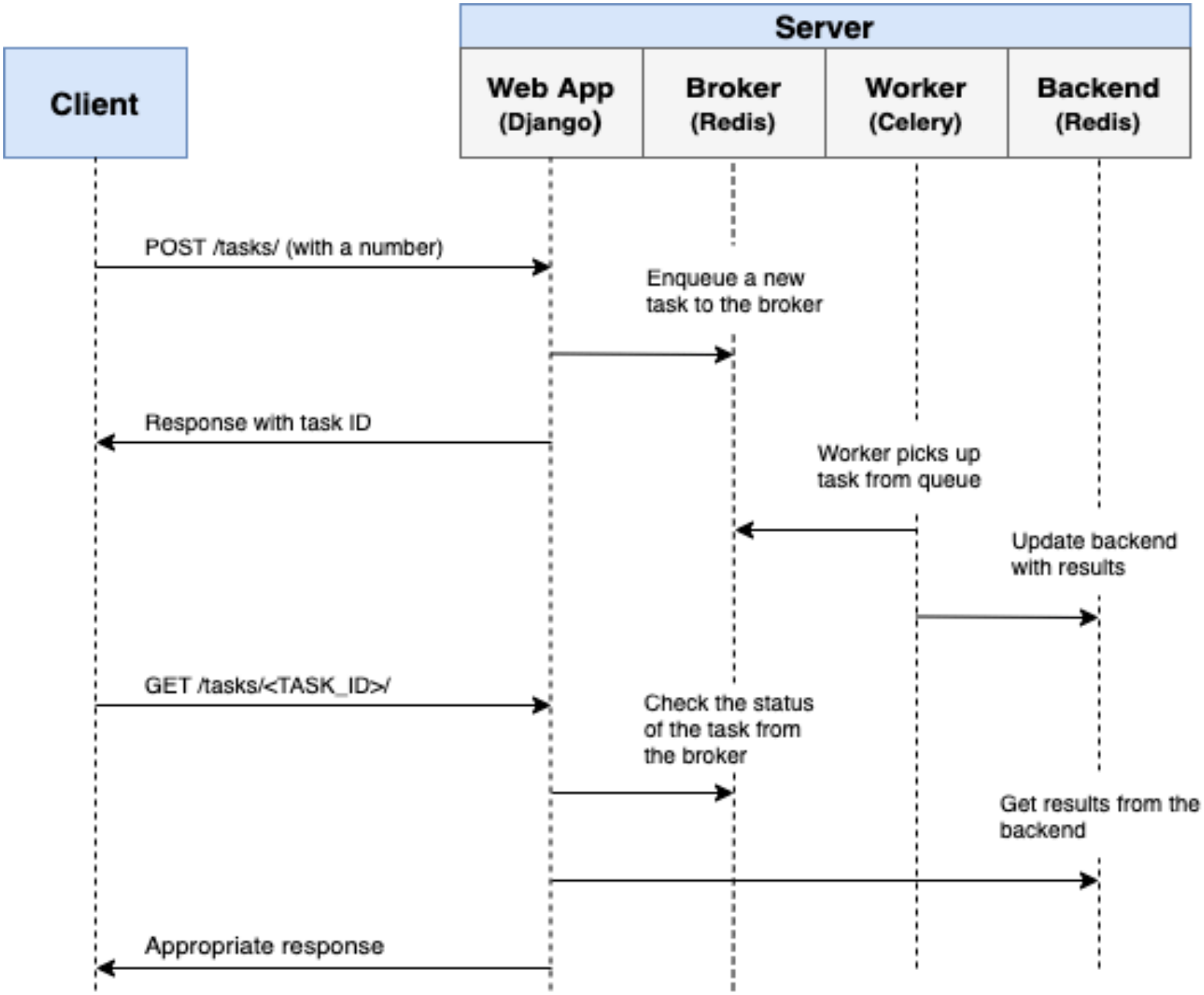
16. What is Celery, Redis, Worker, Flower in Django?



—> Celery, Redis, Worker and Flower are all components that can be used in a Python-based distributed task processing system.

DJANGO WEEK - 5

Celery :

—> Celery is a distributed task queue that enables the execution of asynchronous (occurring at the same time) tasks in a decoupled (separate) manner. It uses message brokers such as RabbitMQ or Redis to distribute tasks to worker nodes.

- Support for multiple message brokers.
- Task Scheduling and Retry Mechanisms.
- Result storage and retrieval (Accessing Data From Storage).
- Decoupled task processing.
- Extensible with custom plugins.

Redis :

—> Redis is an in-memory data structure storage, that can be used as a distributer, database, cache and message broker for Celery.

—>It provides low-latency data storage and retrieval, making it an ideal choice for real-time task p

- In-memory data storage.
- Pub / Sub messaging.
- Data Persistence (Continuing to exit after the termination of the program).
- High Availability and clustering (Grouping of similar things).
- Data Expiration and Eviction Policies.

Worker :

—> A celery worker is responsible for executing tasks that are distributed by the message broker.

- Task prefetching and execution.
- Task routing and scheduling.
- Result storage and retrieval.
- Worker pool management.
- Extensible with custom plugins.

Flower :

—> Flower is a web-based celery monitoring and management tool that provides real-time insights into task processing and worker status.

- Real-time task monitoring
- Worker status and metrics
- Task history and result retrieval
- Alerts and notifications
- REST API and webhooks

—> In summary,

- Celery provides a powerful distributed task queue for Python.
- Redis is a popular message broker for Celery that offers low-latency data storage a retrieval.
- The Celery Worker is responsible for executing tasks.
- Flower is a web-based monitoring and management tool for Celery applications. It provides real-time insights into task processing and worker status.

1. A client sends a task to the message broker (Redis in this case) using the Celery API. The task is serialised and sent to the message broker as a message.

2. A worker node listens for messages on the message broker and retrieves the task when it is available.

3. The worker node deserializes the task and executes it.

4. The result of the task is sent back to the message broker and stored in a result backend (such as Redis or database).

5. The client can retrieve the result of the task using the Celery API.

Benefits :

- Improved application responsiveness by offloading long-running tasks to background workers.
- Enhanced scalability by handling increased workloads without performance bottlenecks.
- Simplified task management with a centralized task queue and a monitoring dashboard.

17. What is Django Rest Framework (DRF)?

—> Toolkit for building APIs with Django
—> It is used to display all the items in our database and an endpoint for adding new items all through this API and returns back data in JSON or Database.

pip install djangorestframework

—> Now, go ahead to settings.py and add INSTALLED_APPS = [‘rest_framework’,]
—> Now, create a new folder in the root directory API and inside of that folder create a new file named __init__.py, also create views.py and urls.py
—> Now, After adding serializers.py
—> Check VS Code.

DJANGO WEEK - 5

18. What is serializers in Django?

—> Serializers are used to convert complex data types, such as Django model instances, into Python data types that can be easily rendered into JSON, XML or other content types.
—> Serializers also provide deserialization, allowing parsed data to be converted back into complex types after first validating the incoming data.
—> Serializers in Django are a part of the Django REST Framework, a powerful and flexible toolkit for Web APIs.

19. What are Decorators in Django?

—> A decorator is a design pattern in Python that allows a user to add new functionality to an existing object without modifying its structure.
—>They are typically applied to functions and they play a crucial role in enhancing or modifying the behavior of functions.
—> They are placed before the definition of a function.
—> We can easily decorate a function using the @decorator syntax.
—> We can pass arguments to decorator by wrapping them inside of another decorator function.
—> We can apply multiple decorators to a single function by stacking them.
—> It can be used with the methods of a class or the whole class.
—> Classes can also be used as decorators.
—> Decorators in Python have several real-world usages like measuring execution time, authentication,logging, etc.

20. What is a Generator?

—> A Generator in Python is a function that returns an iterator using the Yield keyword.
—> A generator function in Python is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return. If the body of a def contains yield, the function automatically becomes a Python generator function.
—> It is a special type of iterator that allows you to control the flow of data and its generation. It is implemented using the yield keyword, which turns a function into a generator. Generators are useful when you want to work on large-data sets, as they generate data on-the-fly, conserving memory.
—> We can create a generator function by simply using the def keyword and the yield keyword.
—> They are powerful tool in Python that allows you to generate values on the fly without having to store them all in the memory at once. You can create a generator using a generator function, which is a function that contains one or more ‘yield’ statements. Generators are useful for iterating over large or infinite sequences without having to store all of the values in memory at once.

```
def simpleGeneraorFun():
    yield 1
    yield 2
    yield 3
for value in simpleGeneratorFun():
    print(value)
```

21. What is Context in Django?

—> The context is a dictionary-like object that contains data that will be used to render a template.

—> It is used to pass data from the view to the template, allowing you to dynamically generate HTML pages.

—> Context is used in Django REST Framework when you want to add extra data to the serializer in addition to the object being serialized.