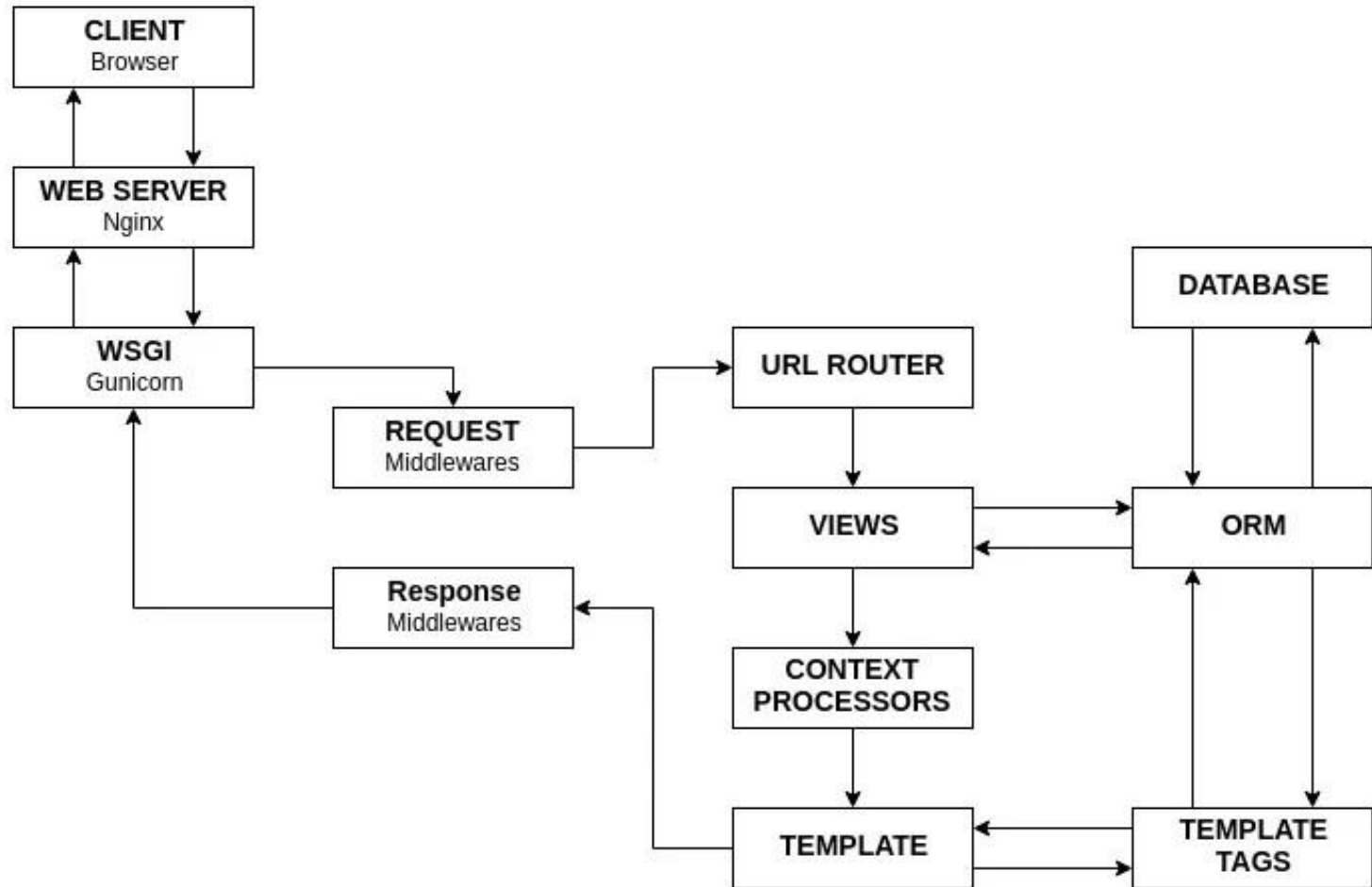


# Request & Response Cycle

# django Request & Response





# Request & Response Cycle

1. The client sends the request (from this browser, for example, an http, https or websocket request).
2. It goes through the internet (many internet pipelines, routers, switches) and arrives at the server. Firewall check.
3. It goes to the web server (Nginx, Apache), which serves static content, like image, javascript file, css file, serve ssl traffic, and provide security.
4. If needed, the request is redirected to the Python server. WSGI (web server gateway interface) or ASGI (asynchronous server gateway interface). Interface between Django application and web server.




# Request & Response Cycle

5. Requests go to the middleware of the Django application. **A Middleware** is like a module that can run before the request goes to the view and after it. It can also alter the request and/or the response.

It does not show on the cycle image, but the middleware can ask the database for some data.

You will be able to retrieve the user's session, is the user logged in?, load the user data, and load the users' permissions (the last one is not done by Django, we will need to built it).



# Request & Response Cycle

6. Now, Django redirects it to the URL router. You have one at the general level (in `movie_theater/urls.py`), and then it should redirect it to the URL router of the app. If the URL does not match the one from the request, Django will return an error.

7. Django redirects the request to the appropriate view. You can have a method view or a class view. We will see later the class view. Class views will offer the possibility to create more complex views or to use predefined classes built by Django to develop app faster

8. On the view, you can ask the database for some data.

9. Once your view is done, it will send the data to the template.



# Request & Response Cycle

10. The template will fill with the data you defined in the view. We can use template functions called template tags inside the template.
11. Once the view and the template are done, the request, which is now called the response, is back to all the middleware in the reverse order than when it first did it in the way in. The wsgi or asgi servers get the response from the Django application. The response can be the response expected, or it can be an error. The wsgi/asgi is going to return an HTML error.
12. Then the Nginx will return the response given by the wgsi/asgi server.



# Let's add a middleware to your project

Where should we save the middleware file?

Should we need to add it to our settings?

