HTTP(s)

HTTP is inherently stateless, meaning each request-response cycle is independent and the server doesn't naturally remember previous interactions. To maintain application state across multiple requests—especially for user authentication—web applications employ several mechanisms.

The most common approach uses session management with cookies: when a user logs in, the server creates a unique session identifier (session ID), stores session data server-side (in memory, database, or cache), and sends the session ID to the client as a cookie. On subsequent requests, the browser automatically includes this cookie, allowing the server to retrieve the associated session data and recognise the authenticated user.

Alternative methods include token-based authentication (such as JSON Web Tokens/JWT), where the server generates a cryptographically signed token containing user information that the client stores and sends with each request, eliminating the need for server-side session storage. Local/session storage in browsers can also persist data client-side, though this is typically used alongside server-side validation.

For APIs, OAuth 2.0 and similar protocols provide delegated authentication using access tokens with defined lifespans. Security considerations are critical: session IDs must be randomly generated and unpredictable, transmitted over HTTPS to prevent interception, and protected against attacks like session hijacking and cross-site request forgery (CSRF).

Modern frameworks like Django, Express, and Spring Boot provide built-in session management systems that handle these complexities, automatically managing session creation, validation, expiration, and secure cookie configuration to maintain authenticated state across the stateless HTTP protocol.

Django Database Migrations

Performing Django database migrations to MariaDB involves several systematic procedures that ensure proper database schema management and data integrity Rip TutorialMedium.

The process begins with installing the necessary Python database adapter, typically mysqlclient, which serves as the driver for both MySQL and MariaDB Medium. In Django's settings.py, the database configuration requires specifying 'ENGINE': 'django.db.backends.mysql' (MariaDB shares the MySQL backend), along with database name, user credentials, host, and port 3306 Rip TutorialDigitalOcean.

Before running migrations, you must create the database and user on the MariaDB server using SQL commands like CREATE DATABASE, CREATE USER, and GRANT ALL PRIVILEGES MediumDigitalOcean.

Django's migration system then uses two primary commands: python manage.py makemigrations to create migration files based on model changes, and python manage.py migrate to apply these migrations and create the necessary database tables according to the INSTALLED_APPS setting Django Documentation.

The migrate command examines migration files and executes the required SQL statements to update the database schema Django Documentation.

For existing projects migrating from another database, developers can use dumpdata --natural-foreign to export data as JSON and loaddata to import it into MariaDB after running initial migrations Libove.

Django supports MariaDB 10.5 and higher, expecting UTF-8 encoding and proper transaction support Django Documentation. The migration system maintains a history of schema changes, allowing developers to roll forward or backward through database versions, making it a robust solution for managing database evolution throughout the application lifecycle.

Sonnet 4.5