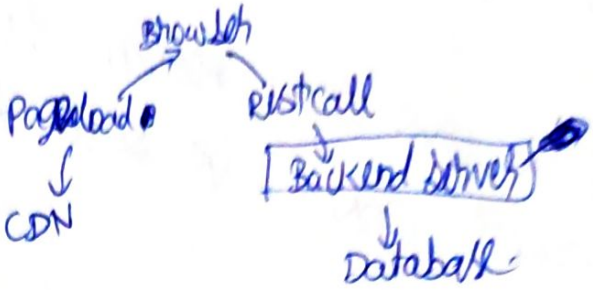
Java Application.

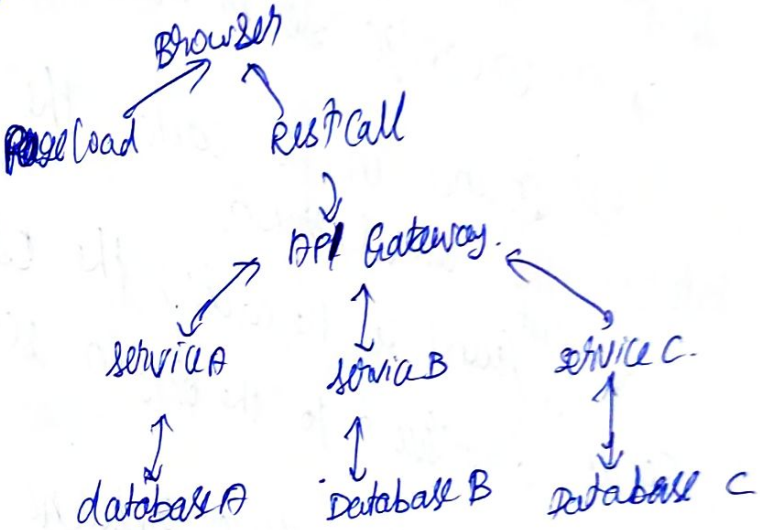
Three tier - architecture

front - mid - database.



API call by javascript

microservices



CDN: Content Delivery Network.

* Distributed servers that deliver web content to users based on their geographic location.
* Stored on multiple servers in various locations
* Routes the request to the server closest to the user.
* Faster than getting this from the origin server.

→ origin server
→ Edge server.

A web request:

Browser sends a request to the DNS (Domain name server) to resolve the domain name of the URL to its corresponding IP address

* The DNS server returns the IP address of the closest CDN edge server to the user's location.
* The browser sends a request to the CDN edge server.
* The CDN edge server checks its cache.
* If found in the cache, the CDN edge server returns the cached version.
* If not found in the cache, the CDN edge server sends a request to the origin server.
* The origin server returns the requested content to the CDN edge server.
* The CDN edge server stores it in cache.
* The CDN edge server returns the content to the browser.

Rest requests: - common protocol for javascript applications for making a call to the servers

theory of how to use that mechanism

* Javascript code constructs an HTTP request
* adds endpoint API URL and HTTP method
* includes necessary parameters or data.
* Server gets request and returns response
* JS receives response and processes it.
* Response manifests as a UI change.

Capable backend:

* Intercept requests
* Extract data from requests
* Process the request.
* Pull necessary data from database.
* process the data and prepare response
* Return the response

Handling requests:

HTTP - - HyperText Transfer Protocol.

↓ protocol for sending request over internet.

* a message sent by a client (such as a browser) to a server to request specific information

mechanism

— HTTP request:

* Request line ("GET /index.html HTTP/1.1")
* Headers: meta info about the request
* Body: Data to be sent with the request

HTTP Response

* Status line: First line Eg: 'HTTP/1.1 200 OK'
* Headers: meta info about the response
* Body: Actual data being sent

HTTP — stateless protocol.

Cookies:

Allows multiple requests to be "tied together".

Cookies:

* sent to the client in the Set-Cookie header of a response
* sent back to the server in the cookie header of subsequent requests.

Date .....

With server sessions

Login flow:

* user enters login credentials into a web form.
* form data is submitted using HTTP POST.
* Server verifies the credentials.
* Server generates a new session for the user.
* Server creates a unique token (session ID).
* Server sends a response with a Set-Cookie header.
* Browser stores the cookie on the user's device.

Cookie exchange flow:

* Browser sends the cookie back to server in Cookie header.
* Server looks up session info based on the session ID.
* Server generates a personalized response.

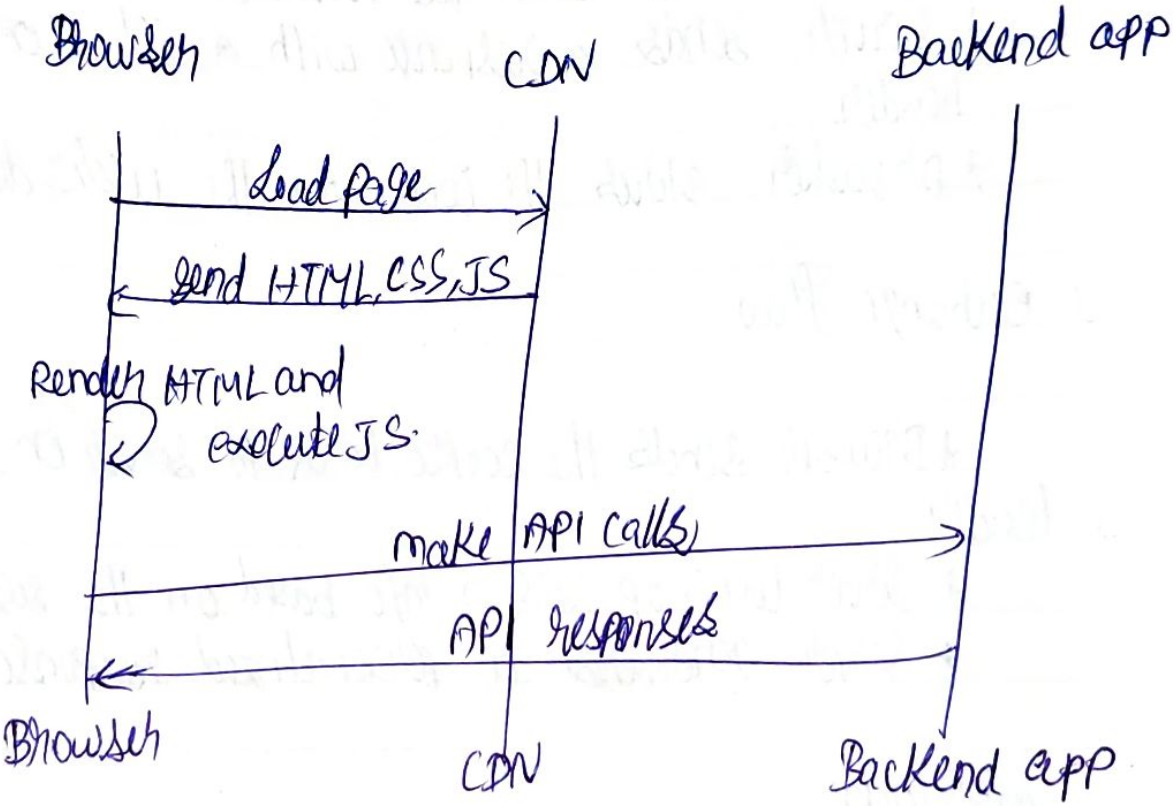
Cookie approach:

Automatic

* Browser memory.
* Local and Session storage.

A capable backend.

* Intercept requests
* Extract data from requests
* Process the request
* Pull neccesary data from databases
* Process the data and prepare response
* Return the response.



What we need:

Java * A programming language

Tomcat, JT. * A server runtime.

spring, spring-boot * A framework to handle common concerns

* A framework to interact with the database

* A framework to handle security

* A framework to manage infrastructure (microservices)

Expt. No.....
Date .....

Page No.....

Github

Pull request workflow:

* Fork the repository.
* clone the repository.
* create a new branch
* make changes
* push the changes
* create a pull request.
* Review and discuss
* merge or decline
* update and delete the branch