

Caching and performance

<https://github.com/heig-vd-dai-course>

[Web](#) • [PDF](#)

L. Delafontaine and H. Louis, with the help of GitHub Copilot.

Based on the original course by O. Liechti and J. Ehrensberger.

This work is licensed under the [CC BY-SA 4.0](#) license.

Objectives

- Understand the concepts of caching
- Understand how caching can improve performance
- Understand how HTTP features can help to cache data
- Implement caching in a web application



Caching

More details for this section in the [course material](#). You can find other resources and alternatives as well.

Caching

Process of **storing a copy of a resource to serve it faster.**

Caching can **improve the performance** of a system and **reduce the load** on the backend.

Caching can be done on the **client-side** or **server-side.**



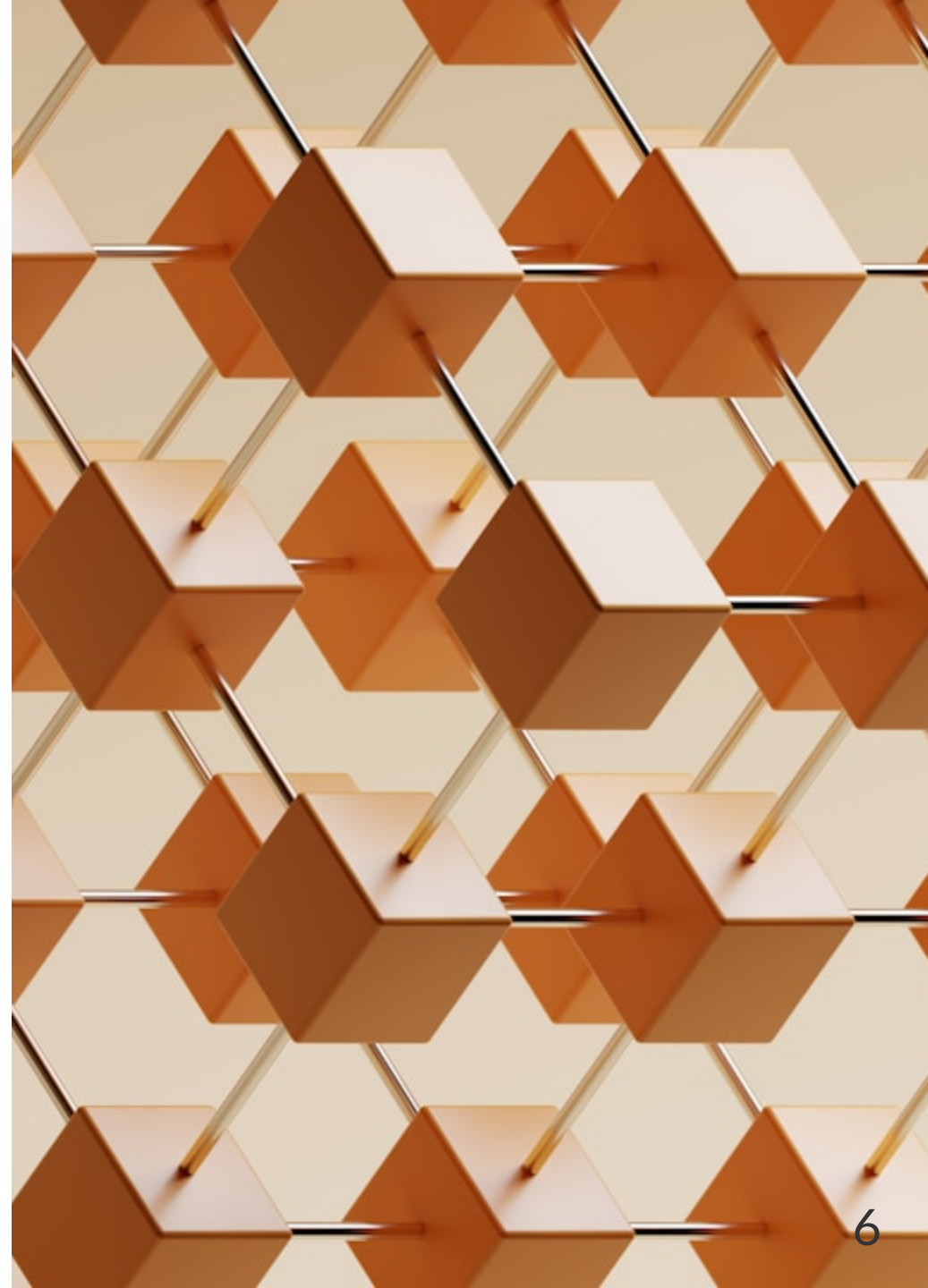
Types of caching

- **Client-side caching** (private caches): once a client has received a response from a server, it can store the response in a cache. The next time the client needs the same resource, it can use the cached response instead of sending a new request to the server.
- **Server-side caching** (shared caches): the server stores data in a cache with the help of a reverse proxy or by the web application. The next time the server needs the same resource, it can use the cached response instead of processing the request again.

CDN

Content Delivery Network (CDN) is a network of servers that are geographically distributed around the world.

Improve performance by serving static content (images, videos, etc.) from the closest server.



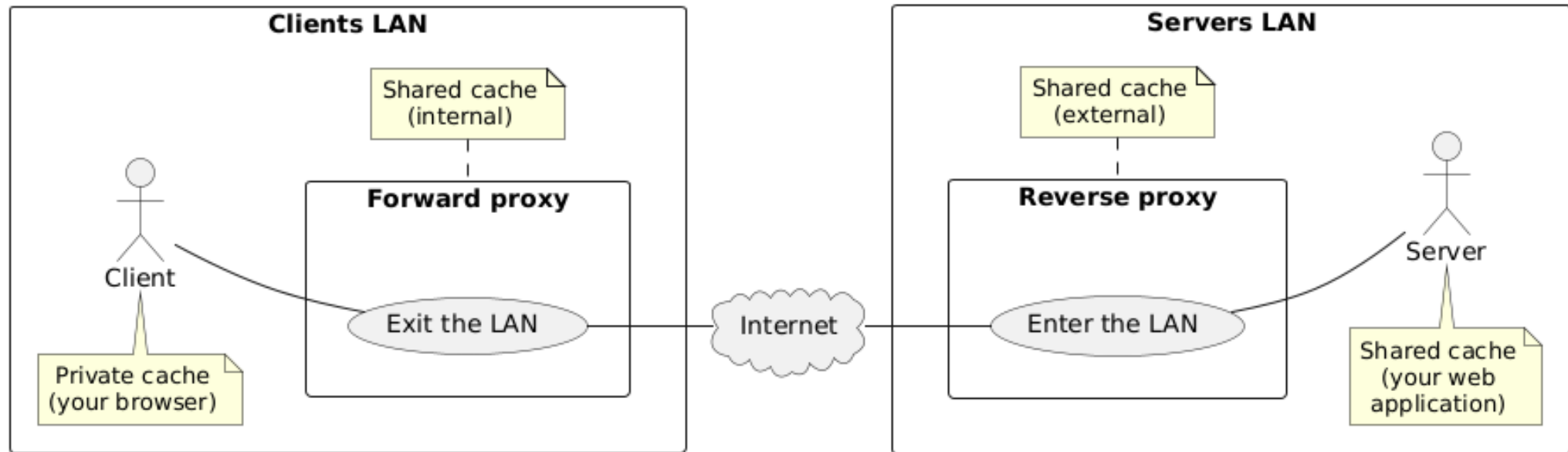
Where to cache?

The best would be to cache at each level of the system to ensure the best performance but it is not always possible or faisable:

- **Client-side:** the cache is stored on the client
- **Server-side:** the cache is stored on the server
- **CDN:** the cache is stored on a CDN

Private caches are caches that are only used by one client. Public caches are caches that are used by multiple clients.

Where to cache?



Managing cache with HTTP

More details for this section in the [course material](#). You can find other resources and alternatives as well.

Managing cache with HTTP

Managing cache is challenging because it is difficult to know when to invalidate the cache (the data can be stale (= outdated)).

Two main caching models:

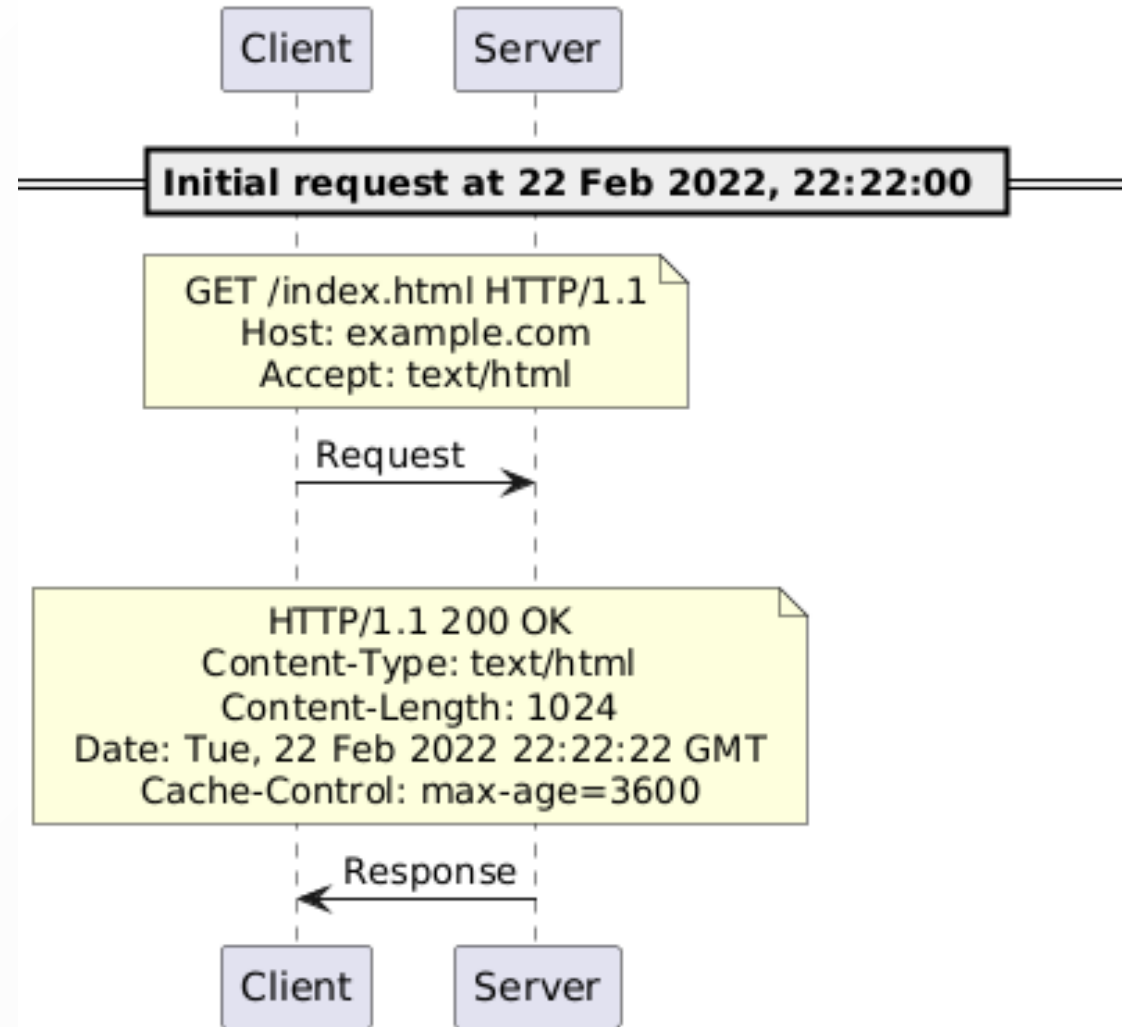
- **Expiration model:** the cache is considered valid for a certain amount of time
- **Validation model:** the cache is considered valid until the data is modified

Expiration model

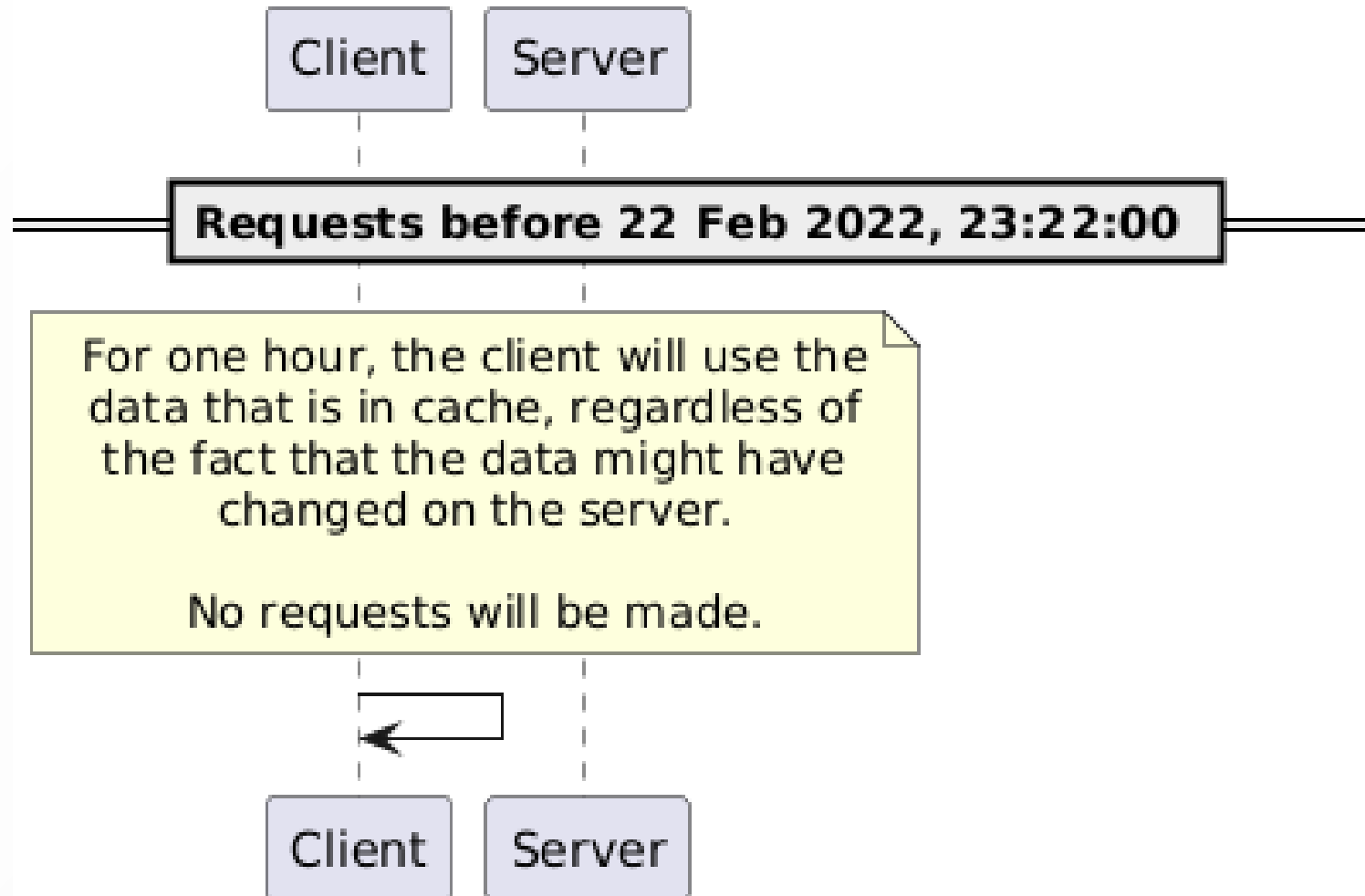
- The cache is valid for a certain amount of time
- If the cache is not expired, the cache is used
- Uses the
`Cache-Control: max-age=<secondes>` header
- The cache is invalidated after the expiration time



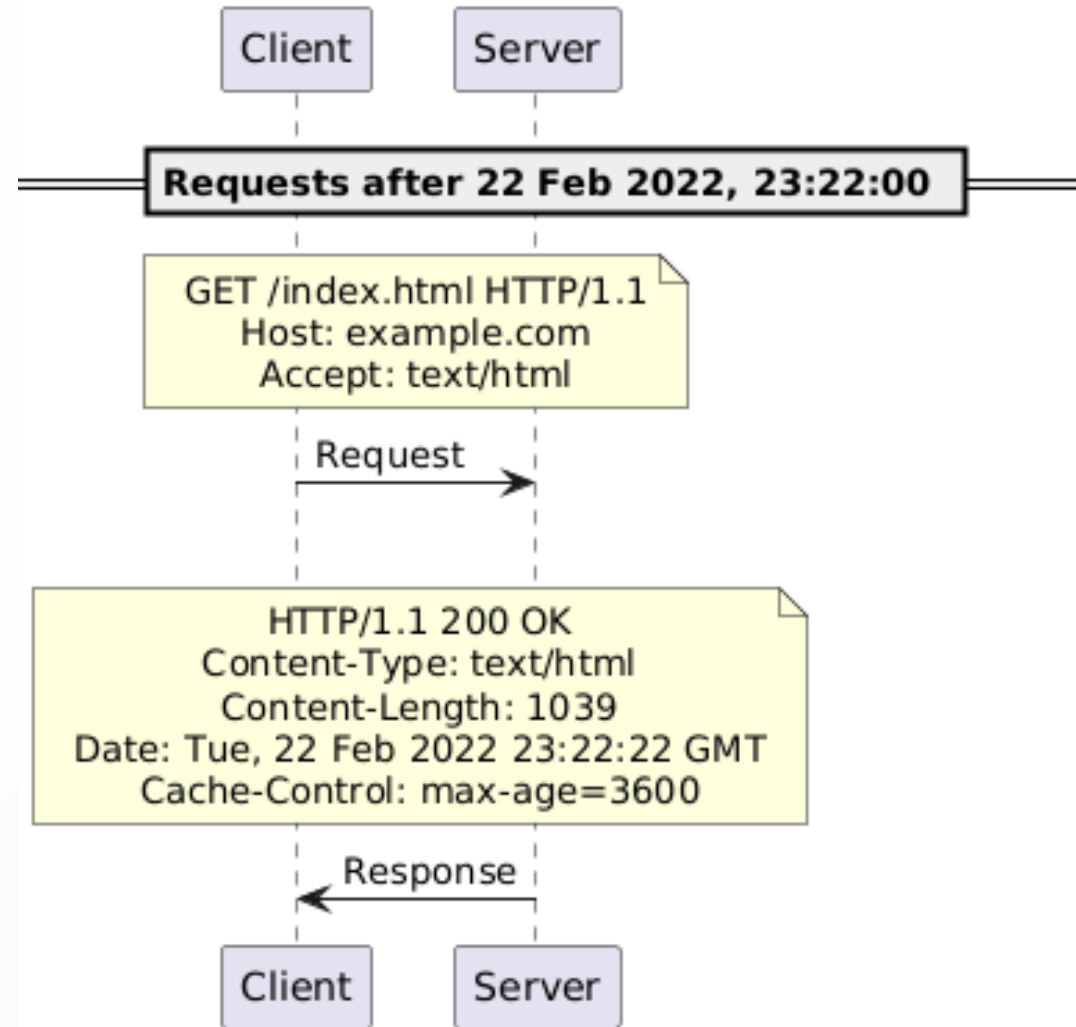
Expiration model - part 1/3



Expiration model - part 2/3



Expiration model - part 3/3



Validation model

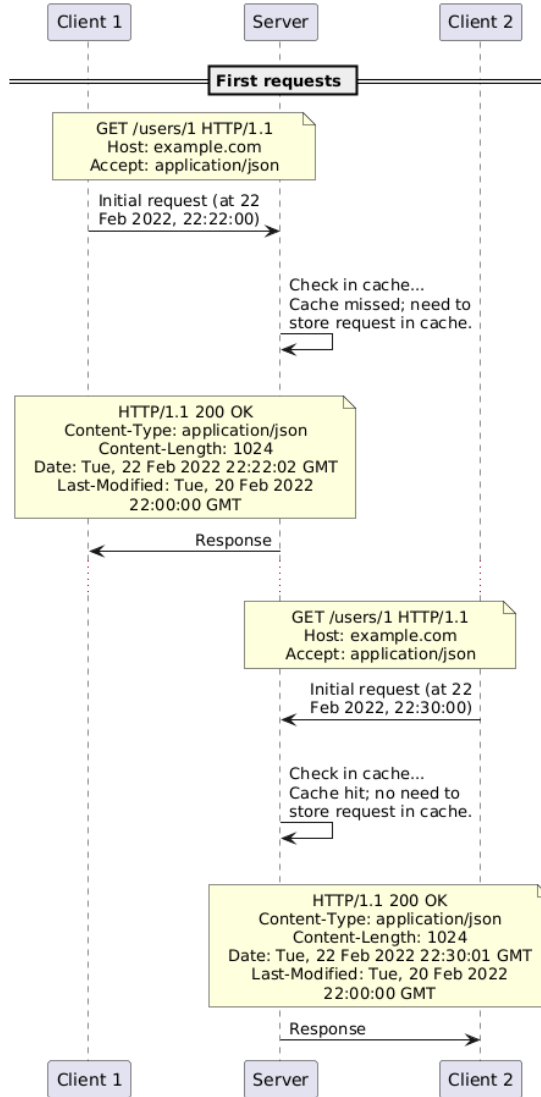
- The cache is valid until the data is modified
- If the cache is not expired, the cache is used
- Two ways to validate the cache:
 - Based on the **Last-Modified** header
 - Based on the **ETag** header



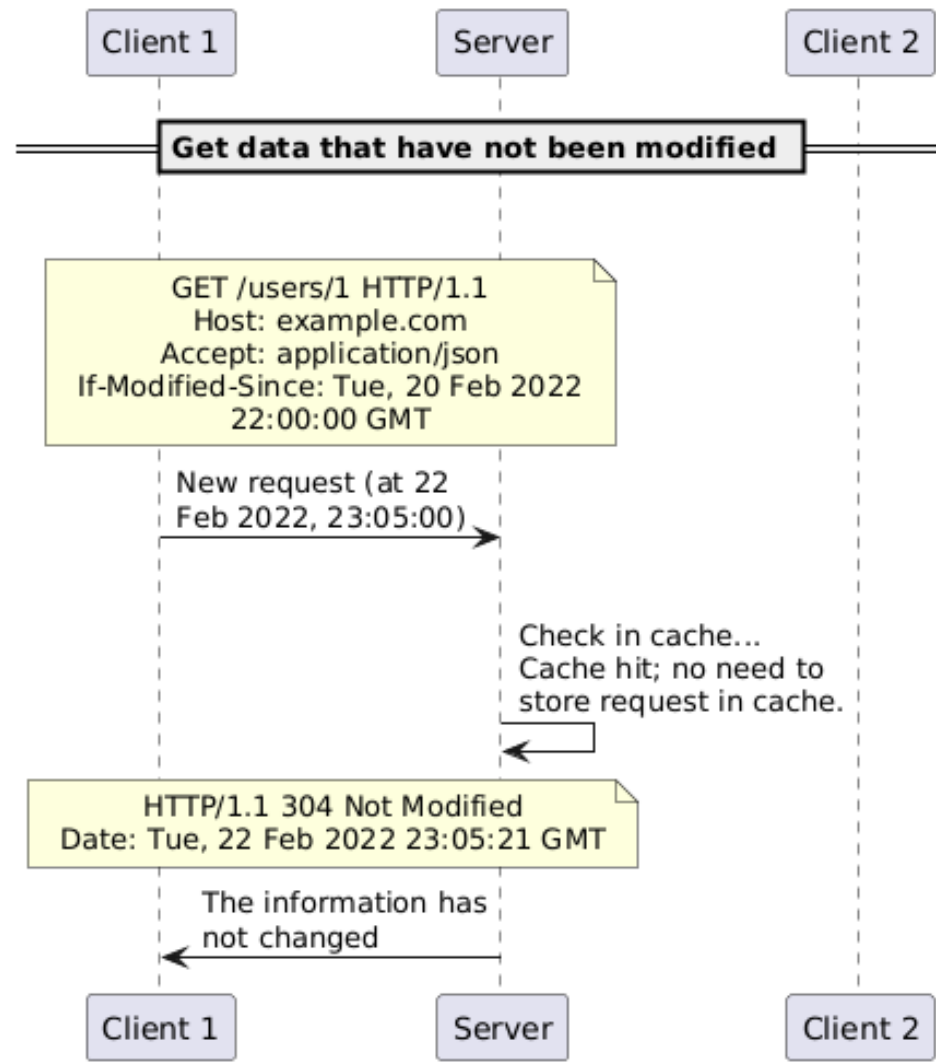
Based on the `Last-Modified` header

- `Last-Modified`: indicates the date and time at which the resource was last updated.
- `If-Modified-Since`: returns a `304 Not Modified` if content is unchanged since the time specified in this field (= the value of the `Last-Modified` header).
- `If-Unmodified-Since`: returns a `412 Precondition Failed` if content has changed since the time specified in this field (= the value of the `Last-Modified` header) **when you try to update/delete the resource.**

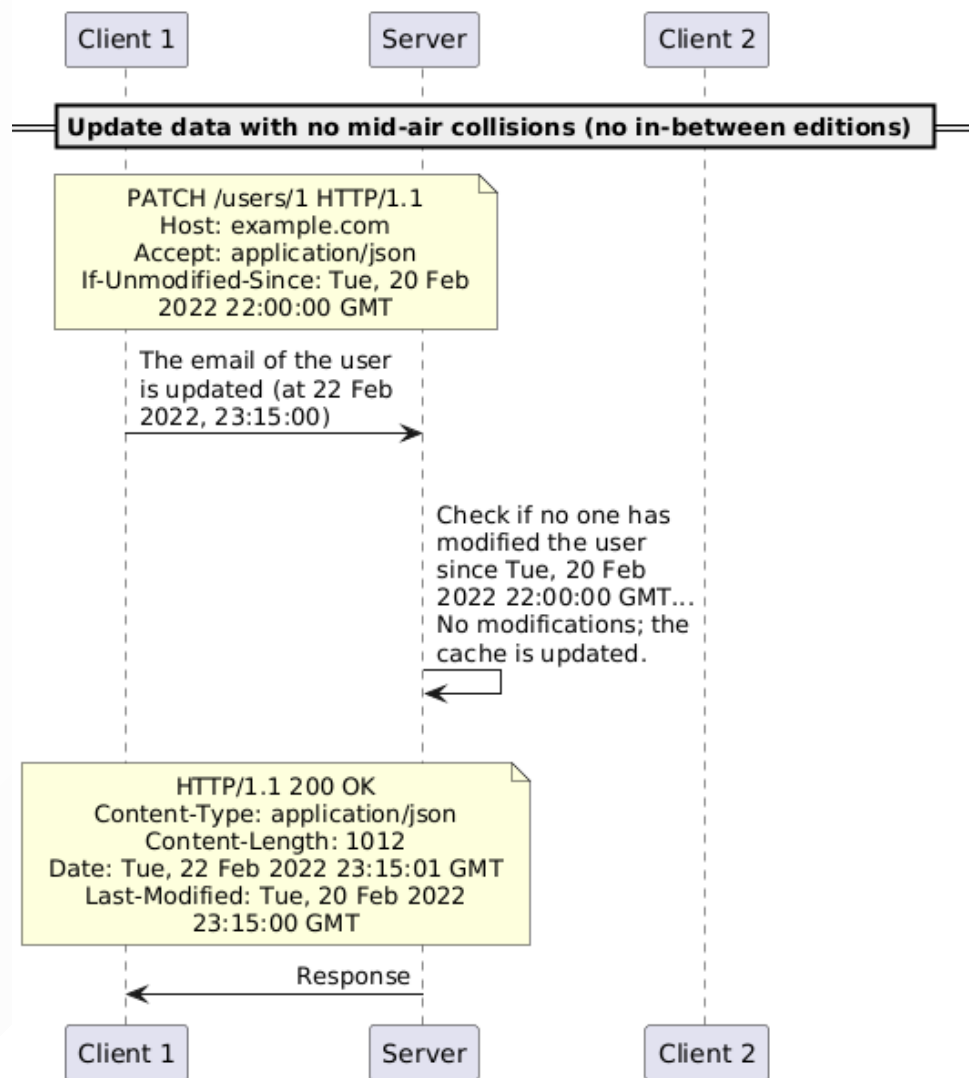
Validation model based on the Last-Modified header - part 1/4



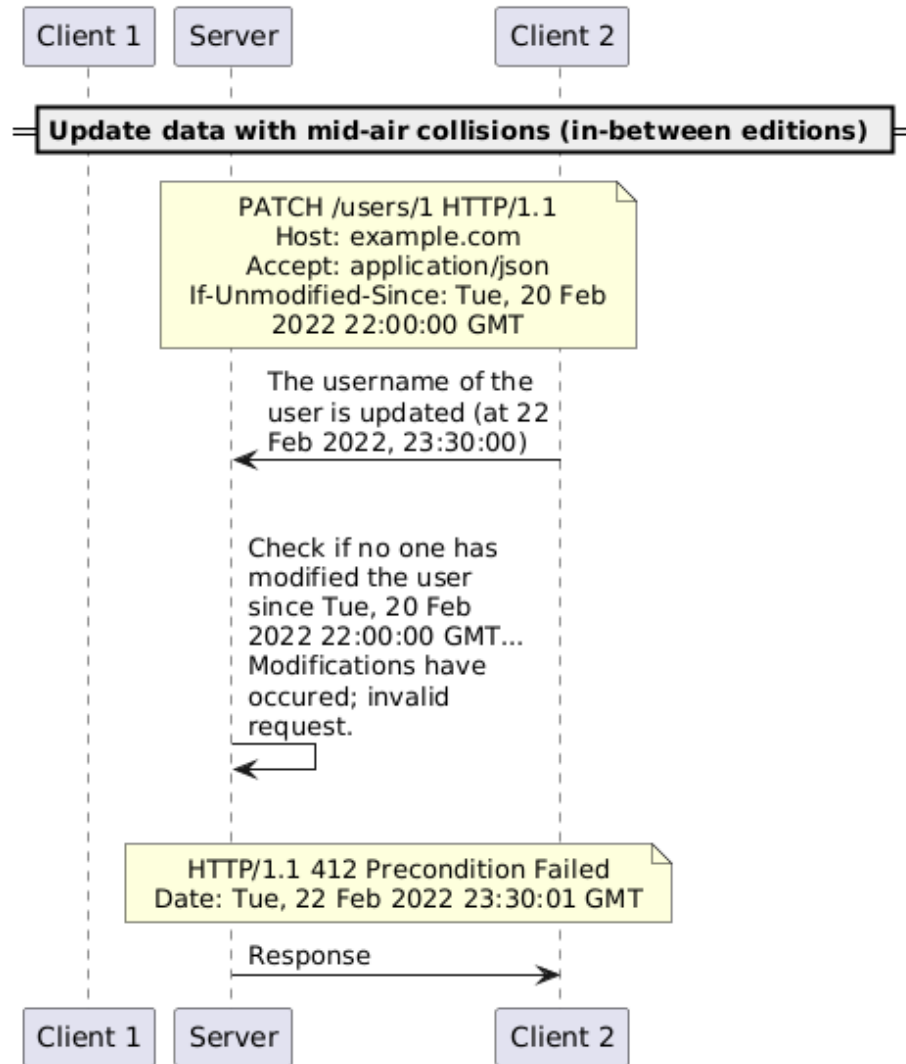
Validation model based on the Last-Modified header - part 2/4



Validation model based on the Last-Modified header - part 3/4



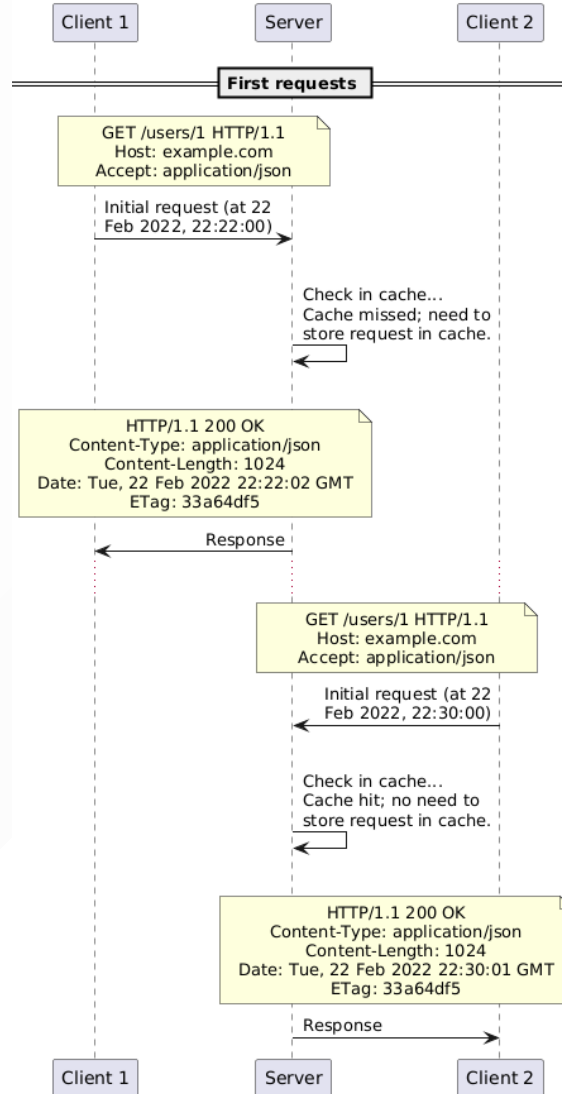
Validation model based on the Last-Modified header - part 4/4



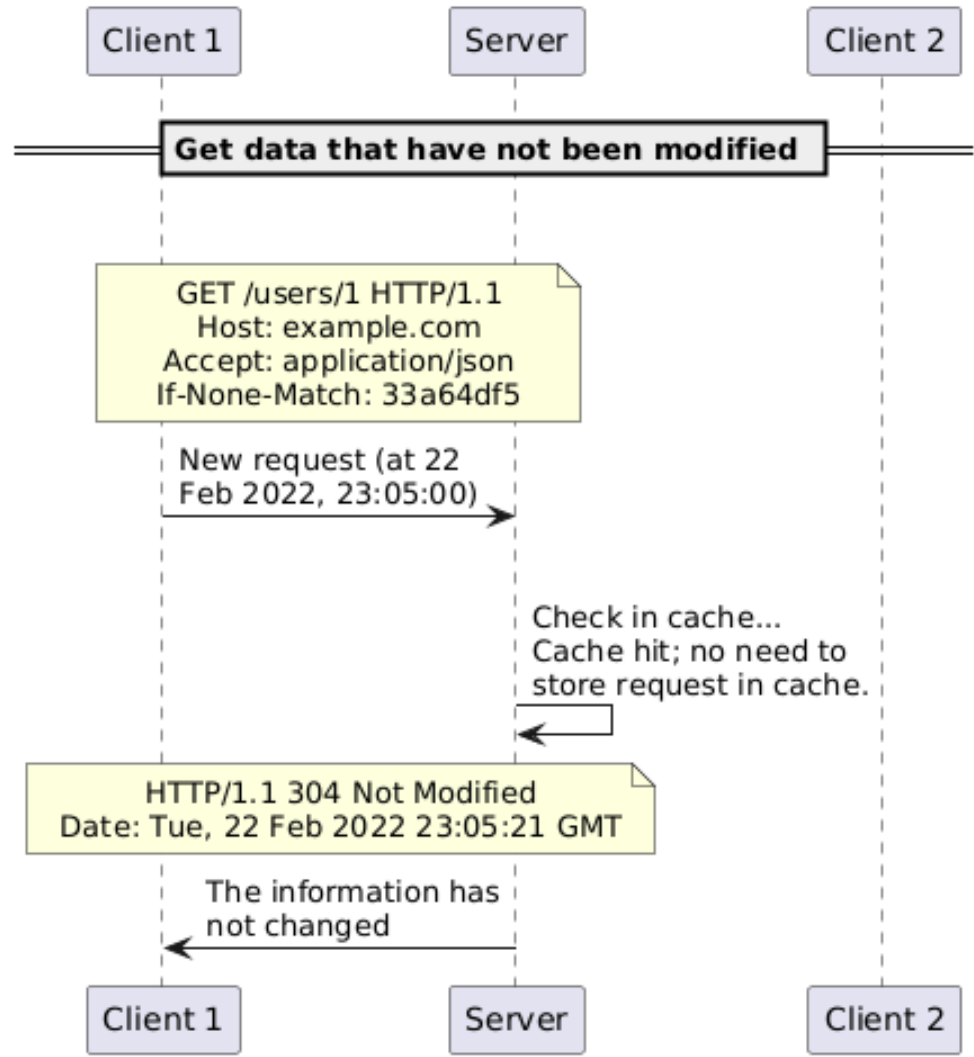
Based on the `ETag` header

- `ETag`: provides the current entity tag for the selected representation. Think of it like a version number or a hash for the given resource.
- `If-None-Match`: returns a `304 Not Modified` if content is unchanged for the entity specified (`ETag`) by this field (= the value of the `ETag` header).
- `If-Match`: returns a `412 Precondition Failed` if content is changed for the entity specified (`ETag`) by this field (= the value of the `ETag` header) **when you try to update/delete the resource.**

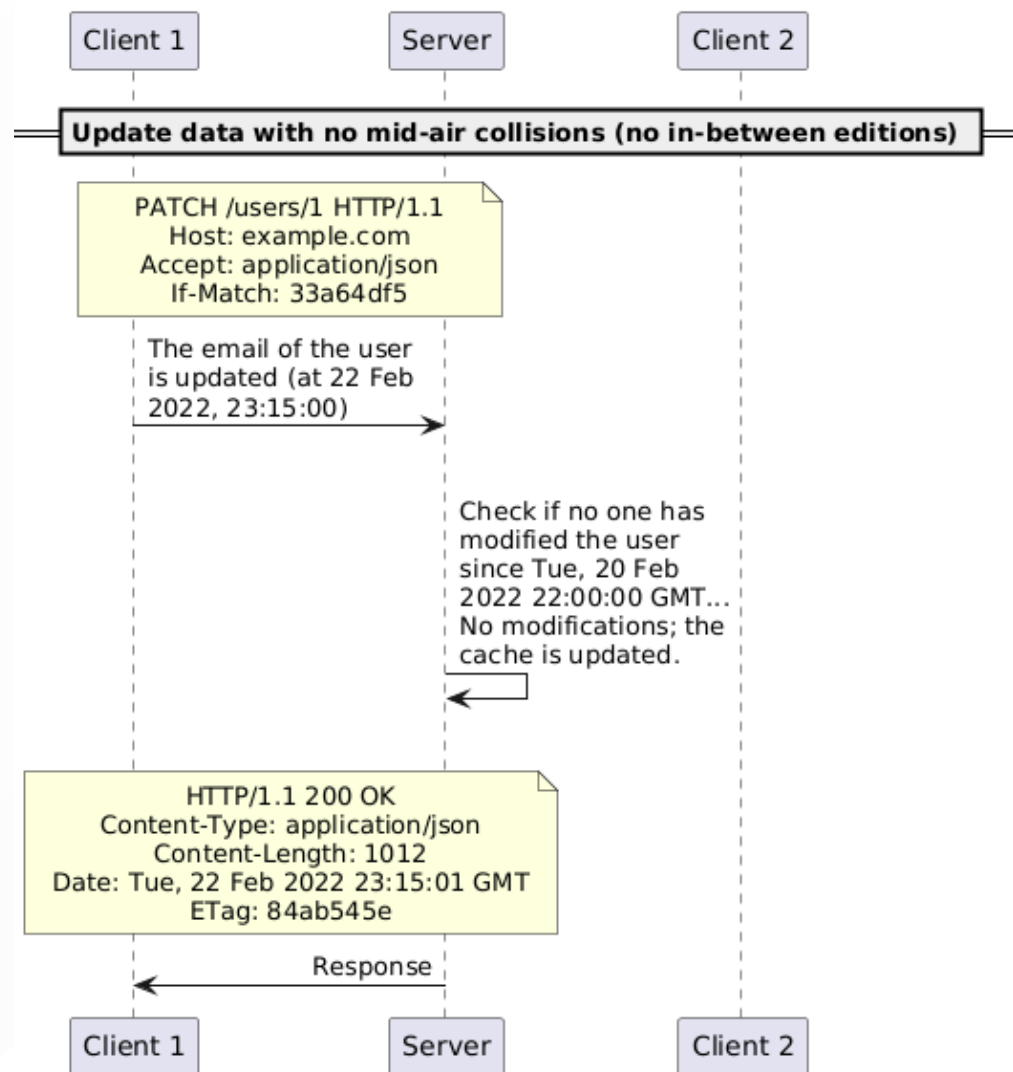
Validation model based on the ETag header - part 1/4



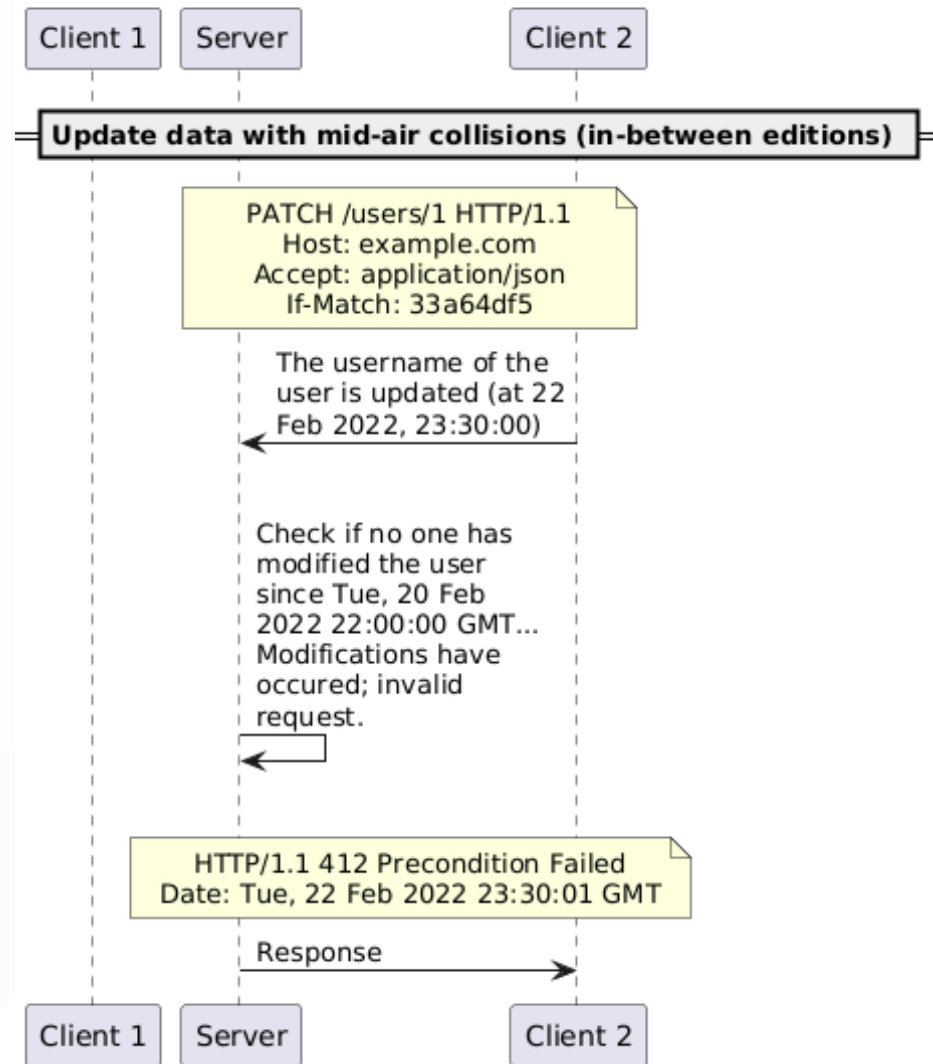
Validation model based on the ETag header - part 2/4



Validation model based on the ETag header - part 3/4



Validation model based on the ETag header - part 4/4



Is it possible to use both models?

Yes, it is possible to use the expiration model and the validation model at the same time:

- No request attempt at all if the cache is not expired
- Validation model when the cache is expired



Managing cache with proxies

More details for this section in the [course material](#). You can find other resources and alternatives as well.

Managing cache with proxies

Proxies can cache responses to reduce the load on the backend and improve performance.

Traefik offers a caching middleware in its Enterprise version. Out of reach for this course.



Managing cache with key-value stores

More details for this section in the [course material](#). You can find other resources and alternatives as well.

Managing cache with key-value stores

Redis is a popular key-value store that can be used to store cache data.

We will implement this manually in Javalin.

The Redis logo is written in a bold, red, cursive script font. The letters are thick and rounded, with a slight slant to the right. The 'R' is particularly large and prominent, followed by 'e', 'd', 'i', 's' in a similar style.

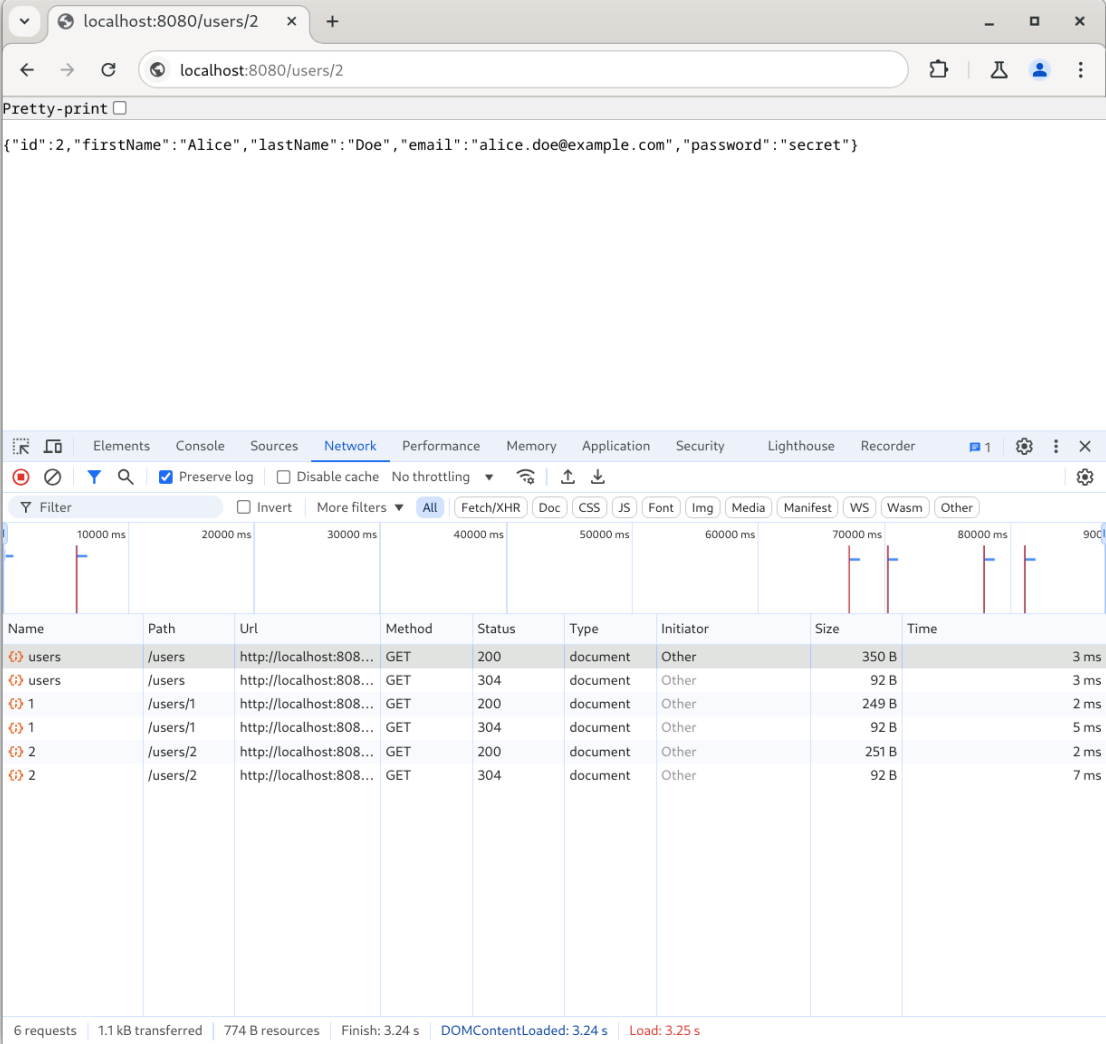
Questions

Do you have any questions?

Practical content

What will you do?

- Implement and validate the validation model based on the `Last-Modified` header in your previous web application using curl and a web browser
- This is your last practical content for this course..!



The screenshot shows a web browser window at `localhost:8080/users/2`. The page content displays a JSON object: `{"id":2,"firstName":"Alice","lastName":"Doe","email":"alice.doe@example.com","password":"secret"}`. Below the page content, the browser's developer tools are open to the Network tab, showing a list of requests. The table below summarizes the requests shown in the log.

Name	Path	Url	Method	Status	Type	Initiator	Size	Time
users	/users	http://localhost:808...	GET	200	document	Other	350 B	3 ms
users	/users	http://localhost:808...	GET	304	document	Other	92 B	3 ms
1	/users/1	http://localhost:808...	GET	200	document	Other	249 B	2 ms
1	/users/1	http://localhost:808...	GET	304	document	Other	92 B	5 ms
2	/users/2	http://localhost:808...	GET	200	document	Other	251 B	2 ms
2	/users/2	http://localhost:808...	GET	304	document	Other	92 B	7 ms

Summary statistics at the bottom of the network log: 6 requests, 1.1 kB transferred, 774 B resources, Finish: 3.24 s, DOMContentLoaded: 3.24 s, Load: 3.25 s.

Find the practical content

You can find the practical content for this chapter on [GitHub](#).



Finished? Was it easy? Was it hard?

Can you let us know what was easy and what was difficult for you during this chapter?

This will help us to improve the course and adapt the content to your needs. If we notice some difficulties, we will come back to you to help you.

 [GitHub Discussions](#)

You can use reactions to express your opinion on a comment!

What will you do next?

We are arriving at the end of the third part of the course.

An evaluation will be done to check your understanding of all the content seen in this third part.

More details will be given in the next chapter.



Sources

- Main illustration by [Richard Horne](#) on [Unsplash](#)
- Illustration by [Aline de Nadai](#) on [Unsplash](#)
- Illustration by [Fermin Rodriguez Penelas](#) on [Unsplash](#)
- Illustration by [Shubham's Web3](#) on [Unsplash](#)
- Illustration by [Andrik Langfield](#) on [Unsplash](#)
- Illustration by [Karen Grigorean](#) on [Unsplash](#)
- Illustration by [MChe Lee](#) on [Unsplash](#)