

## Оптимистическая блокировка.

Эта техника включает в себя добавление номера версии или временной метки к данным, которые проверяются API, когда клиент пытается изменить данные. Если номер версии или временная метка не совпадают, клиенту сообщается, что данные были обновлены другим пользователем, и предлагается обновить данные перед внесением дальнейших изменений.

Давайте рассмотрим некоторые примеры реализации.

Первый вариант - блокировка **по времени** (самый простой способ, но не подходит, если есть большая нагрузка).

Ниже пример реализации оптимистической блокировки с использованием временных меток и HTTP-заголовков (Last-Modified и If-Unmodified-Since). Предположим, есть приложение для редактирования документов, в котором пользователь пытается обновить документ.

- Пользователь запрашивает документ для редактирования:

GET /documents/123 HTTP/1.1

- Сервер возвращает документ вместе с заголовком Last-Modified, указывающим последний раз, когда документ был обновлен:

HTTP/1.1 200 OK

Content-Type: application/json

Last-Modified: Mon, 11 Apr 2022 10:00:00 GMT

```
{
  "title": "Название документа",
  "content": "Это содержание документа".
}
```

- Пользователь вносит изменения в документ и отправляет запрос на обновление с заголовком If-Unmodified-Since, предоставляя метку времени, когда он последний раз получал документ:

PUT /documents/123 HTTP/1.1

Content-Type: application/json

If-Unmodified-Since: Mon, 11 Apr 2022 10:00:00 GMT

```
{
  "title": "Обновленное название документа",
  "content": "Это обновленное содержание документа".
}
```

- Если документ не был изменен другим пользователем с момента указанной метки времени, сервер обновляет документ и возвращает успешный ответ:

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "message": "Документ успешно обновлен".
}
```

- Если документ был изменен другим пользователем после указанной метки времени, сервер возвращает ответ 412 Precondition Failed, указывая, что обновление не может быть выполнено из-за конфликтующих изменений:

HTTP/1.1 412 Precondition Failed

Content-Type: application/json

```
{
  "message": "Документ был изменен другим пользователем. Пожалуйста, обновите документ и примените свои изменения снова."
}
```

В этом сценарии пользователю необходимо обновить документ, просмотреть изменения, внесенные другим пользователем, а затем снова применить свои изменения, прежде чем отправлять очередной запрос на обновление.

Пример использования: вы создаете простую систему управления контентом, в которой несколько редакторов могут редактировать статьи нечасто, и вероятность возникновения конфликтов невелика. Оптимистическая блокировка по времени может быть простым и эффективным решением в таких случаях.

Это самый простой способ реализации, но он не подходит, если у вас много пользователей и много нагрузки — ведь тогда в один момент времени всё же может пройти несколько запросов к ресурсу.