

Рассмотрим второй вариант - технику блокировки **по версиям** (тоже довольно простой способ).

Оптимистическая блокировка с версиями - это техника, которая может быть использована для обработки параллелизма в API с помощью номера версии для обнаружения конфликтов, когда несколько пользователей пытаются обновить один и тот же ресурс одновременно.

Допустим, у нас есть API, который позволяет пользователям обновлять запись о клиенте. Запись клиента имеет номер версии, который увеличивается при каждом обновлении записи. Вот как мы можем реализовать оптимистическую блокировку для обновления записи клиента.

- Сначала клиент получает запись о клиенте и номер ее текущей версии, отправляя GET-запрос к API:

GET /customers/123 HTTP/1.1

- Сервер возвращает запись клиента вместе с номером текущей версии в ответе:

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "id": 123,
  "name": "John Doe",
  "email": "john.doe@example.com",
  "версия": 1
}
```

- Затем клиент отправляет запрос на обновление в API, включая в тело запроса обновленную запись клиента и номер ее версии:

PUT /customers/123 HTTP/1.1

Content-Type: application/json

```
{
```

```
"id": 123,  
"name": "John Doe",  
"email": "j.doe@example.com",  
"версия": 1  
}
```

- Сервер получает запрос на обновление и сверяет номер версии записи клиента с номером версии, указанным в запросе. Если номера версий совпадают, сервер обновляет запись клиента и возвращает ответ 200:

HTTP/1.1 200 OK

- Если номера версий не совпадают, сервер возвращает ответ 409 Conflict, указывающий на то, что запись клиента была обновлена другим пользователем или клиентом, и запрос на обновление не может быть обработан:

HTTP/1.1 409 Conflict

Пример использования: вы создаете приложение для управления задачами, в котором пользователи могут создавать, обновлять и удалять задачи. Приложению требуется более точный способ отслеживания изменений, чем оптимистическая блокировка по времени, но не нужна детализация, обеспечиваемая хэшами ETag.

Это тоже несложный способ, при этом он не позволяет детально разрешать конфликты обновления документа (например, если два клиента изменили текста в независимых местах ресурса, такая блокировка все равно заставит одного из них получить конфликт и обновиться для внесения своих правок). Также тут сложно использовать кеширование.