










How do we design effective and safe APIs?

The diagram below shows typical API designs with a shopping cart example.

Design Effective & Safe APIs		blog.bytebytego.com
 Design a Shopping Cart		
Use resource names (nouns)	 GET /querycarts/123	 GET /carts/123
Use plurals	 GET /cart/123	 GET /carts/123
Idempotency	 POST /carts	 POST /carts {requestId: 4321}
Use versioning	 GET /carts/v1/123	 GET /v1/carts/123
Query after soft deletion	 GET /carts	 GET /carts? includeDeleted=true
Pagination	 GET /carts	 GET /carts? pageSize=xx&pageToken=xx
Sorting	 GET /items	 GET /items? sort_by=time
Filtering	 GET /items	 GET /items? filter=color:red
Secure Access	 X-API-KEY=xxx	 X-API-KEY = xxx X-EXPIRY = xxx X-REQUEST-SIGNATURE = xxx <small>hmac(URL + QueryString + Expiry + Body)</small>
Resource cross reference	 GET /carts/123? item=321	 GET /carts/123/items/321
Add an item to a cart	 POST /carts/123? addItem=321	 POST /carts/123/items:add { itemId: "items/321" }
Rate limit	 No rate limit - DDos	 Design rate limiting rules based on IP, user, action group etc

Note that API design is not just URL path design. Most of the time, we need to choose the proper resource names, identifiers, and path patterns. It is equally important to design proper HTTP header fields or to design effective rate-limiting rules within the API gateway.

Over to you: What are the most interesting APIs you've designed?