1. POST: http://localhost/api/job/getalljobs

   It will get all jobs data in array of objects to freelancer to user it needs (userId).
   It will get all job data + status of post reaction
   Status: 0 ➔ no reaction
   Status: 1 ➔ Like
   Status: 2 ➔ Dislike
   This status does not store in database.
   Response: All data of job {

```
   clientId:{type: String},
    postName:{type:String},
    category:{type:String},
    description:{type:String},
    additionalFiles:{type:Array},
    projectType:{type:String},
    screaningQuestions:{type:Array},
    coverLetter:{type:Boolean},
    skills:{type:Array},
    experienceLevel:{type:String},
    visibility:{type: String},
    freelancersNo:{type: Number},
    estimatedBudget:{type:Number},
    postStatus:{type:Number},
    likers:{type:Array},
    dislikers:{type:Array},
    proposals:{type:Object},
    hiring:{type:Object},
    }
createdAt: {Date},
updatedAt: {Date}

}
```

2. POST: http://localhost/api/job/getonejob

   It will get one job data to freelancer to user it needs {userId, jobId} in object request.
   With Status of reaction like getalljobs

3. GET: http://localhost/api/job/ getclientjobs /:clientId
   It will retrieve all jobs of specific client.

4. POST: http://localhost/api/job/multiple-upload
   It Is using in upload multiple file in the job post input and will return files names in array when submit it.
   In file input write

```
<form action="/multiple-upload" method="POST" enctype="multipart/form-data">
    ...
    <input type="file" multiple>
    ...
</form>
```

See: https://bezkoder.com/node-js-upload-multiple-files/

5. GET: http://localhost/api/job//downloadjobpostfiles/:name
   To Download job post file. In header parameters pass the name of file.
6. GET: http://localhost/api/job/downloadproposalfiles/:name
   To Download proposal file. In header parameters pass the name of file.
7. GET: http://localhost/api/job/downloadjobfiles/:name
   To Download received job file. In header parameters pass the name of file.

---

8. POST: http://localhost/api/job/createjob
   If you want to create the job post and save data in one time.
   {clientId:String,postName: String, category: String, description: String, projectType: String, screaningQuestions: Array, coverLetter: Boolean, skills: Array, experienceLevel: String, visibility: String, freelancersNo: Number }

If you want to Create each part individually:
From (6 => 12) need (userId)

9. POST: http://localhost/api/job/createTitle
{postName: String, category: String}
10. PATCH: http://localhost/api/job/createDescription
{description: String}
11. PATCH: http://localhost/api/job/createDetails
{projectType: String, screaningQuestions: Array, coverLetter: Boolean}
12. PATCH: http://localhost/api/job/createExpertise
{skills: Array, experienceLevel: String}
13. PATCH: http://localhost/api/job/createVisibility
{visibility: String, freelancersNo: Number}
14. PATCH: http://localhost/api/job/createBudget
{estimatedBudget: Number}
15. PATCH: http://localhost/api/job/postJob
Make post status = 1 ➔Posted
0 ➔ draft
2 ➔ Archived

//UPDATE job post:
All updates(13 => 18) need jobId in header

16. PATCH: http://localhost/api/job/editTitle/:jobId
{postName: String, category: String}

17. PATCH: http://localhost/api/job/editDescription/:jobId
{description: String}

18. PATCH: http://localhost/api/job/editDetails/:jobId
{projectType: String, screaningQuestions: Array, coverLetter: Boolean}

19. PATCH: http://localhost/api/job/editExpertise/:jobId
{skills: Array, experienceLevel: String}

20. PATCH: http://localhost/api/job/editVisibility/:jobId
{visibility: String, freelancersNo: Number}

21. PATCH: http://localhost/api/job/editBudget/:jobId

{estimatedBudget: Number}

---

22. DELETE: http://localhost/api/job/deletJob/:jobId
    To delete the job post.

---

23. POST: http://localhost/api/job/like
    When user click in like post
    Need{userId, jobId} in object request.

24. POST: http://localhost/api/job/dislike
    When user click in dislike post
    Need {userId, jobId, reason}

25. POST: http://localhost/api/job/unlike
    When user click in unlike post (غير رأيه)
    Need{userId, jobId} in object request.

26. POST: http://localhost/api/job/undislike
    When user click in undislike post (غير رأيه برضو)
    Need {userId, jobId}

---

27. POST: http://localhost/api/job/createproposal

```
userId:req.body.userId,
jobId:req.body.jobId,
bid: req.body.bid,
upworkFees: req.body.upworkFees,
received: req.body.received,
coverLetter:req.body.coverLetter,
```

المفروض كمان فيه😂(files)

```
proposal.status = 0 → submit proposal
proposal.status = 1 → withdraw proposal
proposal.status = 2 → hiring the proposal
```

proposal.status = 3 → finished

28. POST: http://localhost/api/job/withdrawproposal
    Withdraw the proposal →
    proposals.proposalList.proposal.status →1

29. GET: http://localhost/api/job/getproposals/:jobId
   Get all proposals of specific job
   jobId in header

30. POST: http://localhost/api/job/getoneproposal
   Get all proposals of specific job
   jobId,userId

31. POST: http://localhost/api/job/acceptproposal
   Accept proposal
   Req(jobId,userId)
   Status of proposal → 2
   Proposal will stored in hiring

32. POST: http://localhost/api/job/receivejob
   Req(message, userId, jobId)
   المفروض كمان فيه😊(files)

   Status of proposal → 3
   If(number of finished proposals = no. of freelancers)
   ➔job status ➔ 2 //finished