The workers workflow is very simple. They use manageTasks.php running on csdev to create new tasks and update existing tasks in the Tasks.??? file. Tasks can be moved from status to status or the information can be updated.

The management workflow is where the fun happens…

1. A manager goes to a browser and enters localhost:7546. The default index.html page is displayed in his browser.
2. Manager chooses the “Maintain tasks” link and the file tasks.html requested.
3. When tasks.html is returned, an AJAX request is made to the node server to retrieve the list of all tasks. This can be done by sending a get request to the server. In your node server, check for this file name and react as follows:
4. The node server receives the request and calls getTaskInfo.php on the csdev server to get the task list (probably using the node request module)
5. getTaskInfo.php reads Tasks.??? and returns as a response the list of In Development tasks with a Content-Type header set to the appropriate value. It should likely return them sorted as that is how the client needs them displayed.
6. The node server receives the response and sends the response back to the AJAX request.
7. The node server returns the response to the AJAX request with the list of in development tasks.
8. This response is processed by tasks.html and the list is displayed.
9. The AJAX request is sent to the server every 30 seconds if the page just sits there.
10. If the manager chooses to select a different list, the process is repeated with the call to the node server including the new list request. REMEMBER TO STOP ANY OTHER AJAX REQUEST BEFORE SENDING A NEW ONE.
11. If the manager chooses to display the details of a task, a GET request is sent to the node js server (Once again, this can be done by sending a unique “filename” to the server).
12. The node server receives the request and calls getTaskDetail.php on the csdev server to get the task info (probably using the node request module)
13. getTaskDetail.php reads Task.??? and finds the matching id. It returns as a response the information about the selected task with a Content-Type header set to the appropriate value.
14. The node server receives the response, formats it as HTML and sends the response back to the request. This is not an AJAX request.
15. This response is processed by tasks.html and the list is displayed.
16. The manager can then choose to display a new list or even look at a different defect.

1 The node server can, if it wants, format the information using a module of some sort and return the HTML page.

Remember: Managers uses their phones sometimes.

Remember: YOU are the server when it comes to node particularly, you have the ability to control how you react to specify types of request.

I would suggest only have tasks.html and displaying both list and detail in it. You can display both if on a desktop and one or the other if on mobile.