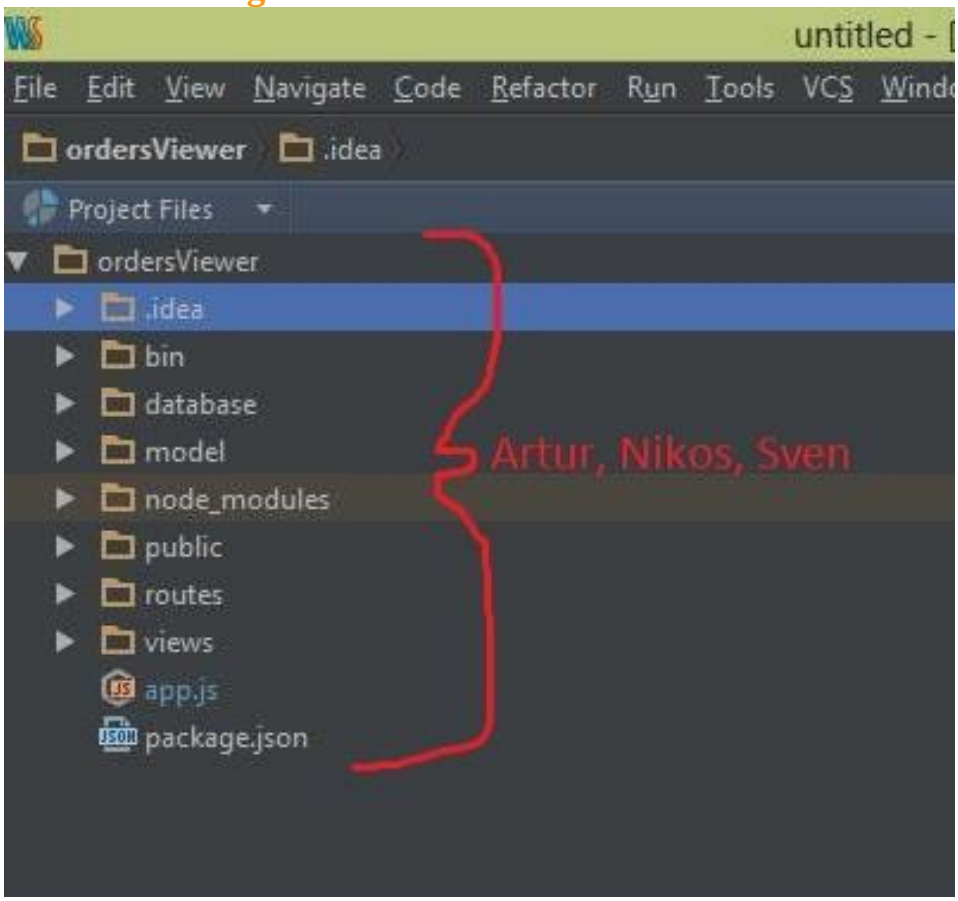
A design description of the chosen design (both on front and backend)

Frontend:
There is only one page, which can load partial content via ajax requests. The contant that is being loaded is accessed via an API. The reasons for choosing to have only one page that loads partial content is:
1. To reduce load on the network by sending only the absolutely neccesary data
2. To make use of ajax, as we should use what we learn here.

A back button helps the user navigate back to the "list" pages every time he went into an internal "details" page.

Backend:
Our express server defines an API via routes. A Router object takes care of these requests and sends back a response, that is the relevant jade-generated partial HTML. These jade files are populated by data that the Router gets from the database. Although these querries to the DB could happen in the routes files, we chose to have dedicated files that take care of that: They are in the model folder, and they expose functions that allow DB querries. The reason for doing that, is that this way we increase flexibility of our code. If in the future we decide to change our DB to a relational DB, we only have to change the code inside the model files.

A section stating who did what



We worked equally on this project. GitHub doesn't say so because we uploaded everything on Friday. We should have followed GitHub since the beginning of the CA.

A description of what you have added (bonus tasks and/or part 3)

✔ ***Bonus I:*** *Add the customer and employee name to the details and make the clickable such that a click will lead to the details of customers and employee, respectively. Add a back link to these pages, too.* ---------- ***Artur, Nikos***

✔ ***Bonus II:*** *Add pages for listing categories, products, employees, and customers. Clicking on elements in the categories page should lead you to a list of products related to this category, i.e. clicking on the Beverages category should give you all products in this category. Clicking on elements in the products, employees, and customers list should lead to a list of orders related to these elements, i.e. clicking on the product Chai will give you all orders with that product.* ------ ***Sven, Nikos***

✔ ***Extra related to that:*** *From the customers and employees lists, the user can not only navigate to the relevant orders for each listing, but also to their relevant details.* ***Nikos***

Part III (optional): *Make the list pages editable.Add links to the list such that you can choose to*

✖ *edit or*

✔ *delete the element (use bootstrap glyphs). A click on the delete button must ask for some kind of confirmation from the user, otherwise a web crawler could delete all you data (As Henrik said, this confirmation is that we jsut use a DELETE method in our ajax call instead of a GET)*

We had problems with database script. We think that there is something wrong with it. By, clicking on the order id we are expected to show order details. But sometimes with some orders this doesn't work because database script is not loaded entirely.

Test

We had a lot problem with Mocha. Finally we stayed with only two tests and we definitely would do more if we had more time.