Tasks:
1. Investigation and questions:

BFF is a microservices oriented pattern which handles the specific tasks required by the front end on the microservices. This means that it handles the orchestration of the calls to all microservices needed for specific tasks removing that responsibility from the Front End. The BFF layer is coupled to the FE, making it easier to be adapted as the UI requires.

REST is the underlying architectural principle of the web. Makes communication easy between layers because there is no knowledge of each others needed. REST is a set of rules used to work on the http protocol. It makes the api development really easy to be understood.

HTTP Transaction oriented protocol which follows a request response schema between a client and a server. Used to enable communications between servers and clients gathering a series of methods to indicate the desired action to be performed on the resource.

Requests. The petition made to the server by the client with the method to use and the protocol version. It contains a header where some fields can be added to indicate some things to the server. Some of the most notable are:

- Authorization: Credentials for HTTP authentication
- Cookie: A small piece of data sent from a website that can be stored on the user's browser to remember state information.
- Referer: The address of the previous web page from which a link to the currently requested page was followed.

Response: The answer sent from the server to the client with the status code and the body of the response. Contains the header where some fields can be added to indicate things to the client. Some of the most notable are:

- Content-type: used to indicate the media type of the resource.
- Content-encoding: used to compress the media-type.
- Content-Length: indicating the size of the entity-body

Json: light text format used to exchange data consisting of attribute–value pairs. It is the most common data format used for asynchronous browser/server communication. Originally designed to be a subset of JavaScript and processed very fast with it.

Event oriented programming: is an architectural pattern which promotes producing, detecting, consuming and reacting to events. When an event is produced, the system reacts, changes it's state if needed and responds. This pattern is formed by creators and consumers.

Non blocking I/O: It is a way to make I/O operations where the processing of other instructions can be made without waiting for them to be finished. It makes the resource consuming more efficient because the CPU is not waiting for this operations that last way more in comparison to data processing.

Heap allocation: Heap allocation on NodeJS is done by v8, which is an engine that executes js script files. It has some tools which can be used to change way node garbage colector behaves like:

- ulimit -m which increases the number of sockets NodeJS is allowed to have. The default is 1024
- –nouse-idle-notification which prevents the garbage collector to run constantly

Vertical vs horizontal scaling: The difference between this two is that to scale horizontally you add more consumers to colect the petitions, and to scale vertically you add more processing power and memory. There is a catch, not every application can scale horizontally and you need to design your application with this in mind. NodeJS can scale horizontally because it can delegate the execution of the requests to separate components and focus on new requests until the delegated component return with the processed result. Also since Node is written on JavaScript it works very fast in passing through and manipulating JSON retrieved from the API, reducing time used per request.

Npm nvm n nrm are package administrators used to handle node version and its components. NPM is used to handle dependencies and its versions. NVM and N are used to switch between node versions and NRM is used to switch between node registries.