Node Challenge

# Project’s files

You can find the project’s files within the GitHub Web App in the link below:  
<https://github.com/Tsalmas-Anastasios/quintessential-node-challenge>

Also, you can find the files of the project inside the folder /final-files.

# File structure

Here described the file structure of the project and the files of it.

File structure (includes the files in list format):

* **tslint.json:** File that to configure the rules for TypeScript writing (e.g. should not use the double equality ‘==’, but should only use the three equals ‘===’)
* **tsconfig.json:** TypeScript configuration
* README.md
* **package.json:** Includes all the dependencies and information for the project and nmp install command
* **gruntfile.js:** File for configuring the Grunt compiler (from TypeScript to JavaScript)
* .npmrc: File that used from web servers and servers in general to configure the npm package manager
* **.gitignore:** File that contains the ignored files from Git Version Control System
* **.env:** File contains the Environment Variables
* **.env.dev:** File contains the Environment Variables for Development mode
* **.env.example:** File contains the Environment Variables as template
* **.env.prod:** File contains the Environment Variables for production mode/state
* **/.vscode:** Folder that contains the configuration file for Visual Studio Code Run & Debug, to be run using the nodemon module
* **/database:** Folder that contains all the files to create the database that is required from the project environment
  + **auth\_tokens.sql:** Create table file for table with name ‘auth\_tokens’. This table used from the system in order to save the access and refresh tokens for authentication/authorization of the user
  + **dump.sql:** Whole database replica file
  + **post\_comments.sql:** Create table file for table with name ‘post\_comments’. This table used from the system in order to save and manage the comments from the posts that are saved in the app
  + **posts.sql:** Create table file for table with name ‘posts’. System uses this table in order to save and manage the post from the Social Media app
  + **users.sql:** Create table file for table with name ‘users’. This table saves the users’ data to create accounts to use the app
* **/src:** This directory has inside the whole source code of the project (written in TypeScript)
  + **/config:** Directory named config includes all the configuration files used by the system
    - **/certs:** Certifications directory. App uses them to start the localhost in https mode and not in http
    - **db.ts:** configures the details for the database connection
    - **index.ts:** includes all the configuration information for the app
  + **/graphql:** Directory graphql contains all the files that are important (or not) for the graphql in order to work properly inside the project. This directory contains configuration files as well as files that defines Graphql types for modeling and interfaces for the query parameters.
    - **/query-models:** This directory contains the interfaces for the query parameters for graphql
      * **index.ts:** This file includes all the interfaces’ exports in order to import only one file in other typescript files
      * **PostCommentQuery.ts:** File that has inside the interface for the arguments (args) to fetch the post comments with specific query
      * **PostQuery.ts:** The same with the file “PostCommentQuery.ts”, with the difference that these parameters concerning for the Posts entities.
    - **/types:** Includes the Graphql types for the entities
      * **Post.ts:** Post entity in Graphql model
      * **PostComment.ts:** Post comment entity in Graphql model
    - **Query.ts:** File that has inside all the Graphql queries
    - **Schema.ts:** Configuration file for the graphql Schema that used by the system (app)
  + **/lib:** Includes all the files that implement connection services like database for the app
    - **/connectors:** directory that contains all the files including the connections with outsource systems, like database
      * mysql-db.ts: Implements the connection with MySql database with some custom code to work properly with project’s environment
    - **/middlewares:** Has inside all the files for the project’s middlewares
      * authenticateToken.mw.ts: Middleware that the project uses to authorize the given Bearer Token in Header named ‘Authorization’
    - **index.ts:** This file includes all the services classes ’ exports in order to import only one file in other typescript files
    - **post-comments.service.ts:** Service that contains all the functions used by system to manage the post’s comments entities
    - **post.service.ts:** Service that contains all the functions used by system to manage the posts entities
    - **stringValidator.service.ts:** Service that has inside all the functions to handle cases of a string, like the function isEmail. This function checks if a string is email or not.
    - **tokenAuthentication.service.ts:** Service that includes the functions to handle (like create new, refresh, validate and more) the authentication/authorization Bearer tokens
    - **user.service.ts:** Service that contains all the functions used by system to manage the users entities
    - **utilities.service.ts:** This service contains the UtilitiesService class that contains various functions used in various parts of the application as well as summarizes within properties various modules and functionality used in various parts of the application in order to avoid constant importing into various parts.
  + **/models:** This directory contains all the TypeScript models (in class format) for the app’s entities
    - **index.ts:** This file includes all the models classes’ exports in order to import only one file in other typescript files
    - **Post.ts:** Post entity
    - **PostComment.ts:** Post comment entity
    - **User:** User entity
  + **/routes:** Contains the routes (URLs) that the app implements
    - **/auth:** Directory contains the files that implement the routes for the AUTHentication and AUTHorization for a user/account
      * **login.ts:** File with the routes for the login process in the app, like login route and refresh token route
      * **logout.ts:** File contains the logout route that deletes the record with tokens (auth\_tokens table in database) from the database
    - **/errors:** Directory contains the files for http errors, like error 404
      * error-404.ts: Implements the route for the 404 error
    - **/users:** Directory contains the files for users manipulation and handling inside the app
      * register.ts: File to register the users
    - **comments.ts:** Implements all the routes to handle the posts’ comments
    - **index.ts:** This file has the index route (‘/’) inside of it
    - **posts.ts:** Implements all the routes to handle the posts
  + **app.ts:** Startup file to run the server, and initialize data for the app

Thank you in advance,

Anastasios Tsalmas