Guides and Structure:

Backend:

Backend Structure:

There are five important folders: build, config, images, prisma and src.

The backend use typescript as the programming language, and because NodeJs doesn't know how to execute typescript, I have to transpile the source code in typescript to javascript, and all this new javascript code is stored in the *build* folder.

The *config folder* is where the files with the necessary information to launch the server are stored.

Images is the folder where the profile pictures of our users are stored.

Prisma is the folder with all the code related to the ORM we are using, it has two files schemas.prisma with the models of the DB and seed.js that is executed each time the command "npm run seed" is executed.

The *src* folder has the source code of the server. The server is launched in the app.ts file. And it has four more folders:

- *Jobs* that have the code of the server cron features.
- *Modules* that have the code of all the endpoints of the API.
- *Plugins* that have the code of functionalities uploaded to the global context of the server. Like middleware or the questionnaire's algorithms.
- *Utils* that have other features needed for the app. Like the emailer or the DB prisma client.

Launch backend development mode:

1. Open a command console in the folder SleepQuality and install the npm modules.
2. Go to the backend folder and create a file .env.development with the same structure as .env.example fulfilling each variable with your respective data.
3. Now with the console in SleepQuality/backend execute the command "npx prisma generate".
4. Run command "npm run dev" and the server should be working.

To run the command "npm run seed" you have to first create the documents for the models in the database.

Create new questionnaire:

There is a POST route in the API (This route can only be called by an administrator), /api/questionnaires, where the administrator has to send in the body of the request the new questionnaire with all the information that he wants to create. The body of the questionnaire must has this structure:

Fields of body:

- name: name of the new questionnaire.
- questions: JSON object, with each key a question of the questionnaires and their values represent the different types each respective answer can be, their possible values are: PRIMARY_TEXT, PRIMARY_NUMBER, PRIMARY_BOOL, SECONDARY_TEXT, SECONDARY_NUMBER and SECONDARY_BOOL. Where PRIMARY means that this answer is mandatory to fill it, and SECONDARY that is not mandatory.
- additionalInformation: Array of JSON objects, where each object can have these fields: questions (array of integers), enum (array of strings), description (string), relation (map of key -> string and values -> number) and default (boolean). The only required field is questions; the others are optional. The field questions relate to the questionnaire's question indexes to which this additional information applies. Enum is used when a question or questions only have a closed number of possible answers, all of them listed in his array. Description is used when a question or questions need a specific description. Relation is used in the algorithm's computation, when some question answers have an associated score. And default is to mark the questions we already can answer with the user's information that we have in our DB.
- instructions: Is a string with all the additional information that questionnaire needs.

```
{
  name:,
  questions: {},
  additionalInformation: [],
  instructions:
}
```

Create new admin:
There is a POST route in the API (This route can only be called by an administrator), api/administrator/:id, where the route param id is the unique identifier of the user that will be a new admin.
To create the first administrator. You have to go directly to the DB data in Mongo Atlas, and upgrade an user manually to admin with their functionality to edit the data of the DB. To upgrade an user to admin you have to change the role of the user to ADMIN.

Create new doctor:
There is a post route in the API (This route can only be called by an administrator), api/administrator/doctors/:email or api/administrator/doctors, with the first route you will create a doctor with the route param email that is an email of an already created and verified user. The second route is to create many doctors at once, where you can send an JSON object in the body of the request, with a field ids that is an array with all the unique identifiers of the users you can convert to doctors.

Add new algorithm:
The code of the backend has to be modified. Specifically the file SleepQuality/backend/src/plugins/algorithm/algorithm.controller.ts. This file exports a Javascript object where its keys are the names of all the questionnaires in our app (The object

must have all the questionnaires) and the values are a function which executes all the code of the questionnaire's algorithm.
Before creating the functionality of the new algorithm, you have to add the new fields to a specific entity in our DB that will store the algorithm results. This can be done in the file SleepQuality/backend/prisma/schema.prisma adding the fields in the model QuestionnaireAlgorithm. Finally you have to execute the command "npx prisma generate" with the console open in the backend folder.
To add a new algorithm, just add a new entry to the object, with the name of the questionnaire and create a new function that calculates the algorithm and stores it in the DB.

Frontend:

The Frontend was built using React Native, an open-source UI software framework used to develop applications for Android , Android TV, iOS , macOS, tvOS, Web, Windows and UWP by enabling developers to use the React framework along with native platform capabilities.

The principal folder is src, divided in 6 folders:

api : It contains all the requests to the api, divided in requests about users and requests about questionnaires.

assets : It contains the images that we need for the application.

components : It contains the React Native components which will be used in the application.

navigation : It contains the stack of the application, created using React Navigation.

screens : It contains all the screens which will be displayed in the application

utils : It contains some functions which are used in some other files

Create new questionnaire:

If the questionnaire wanted to be added to the application is a questionnaire whose questions are Enums and all of them are Mandatory to have an answer, it is only needed to add a

PrimaryEnumQuestionnaire to the screen src/screens/Questionnarie.js if the name coincides with the name of the new Questionnaire.

If the questionnaire wanted has others requirements, a new component has to be created in the folder src/components/typesQuentionnaries. This componet will receive the questions, logo, additional information, instructions… as the other components in this folder do.

Then, this component needs to be added to the screen src/screens/Questionnarie.js if the name coincides with the name of the new Questionnaire.