**ReadMe: Application GUI**

**Application components:**

1. **json folder:** This folder contains JSON files with valid strings of the object data, in order to examine the POST states in cases of: update, add, filter.
2. **Models folder:** Within this folder are the agreements and models of the two collections with which operations were performed during the project:

**studentModel.js:** The students collection receives a unique schema that also includes a sub-schema of courses, which together connect to the model.

**logModel.js:** The log collection receives a unique schema and model that documents the set of actions in the application.

1. **node\_modules folder:** is responsible for all the "packages" installed in the project during the work, which was created by a special installation for this folder.
2. **Public folder:** inside you can find the style.css file that is externally responsible for all project design operations and is called in each pug file separately.
3. **Routes folder:** This folder is responsible for the paths in the application and contains the student "router" file. You can find all the GET / POST operations, the path operations of updating, adding, filtering and deleting. It now contains the studentRuoter.js file responsible for the / student path, which is sent from the main application (server.js).
4. **Views folder:** is responsible for the display and structure of the application display, decided to create a separate pug file for each operation:

**Index.pug:** The person in charge of the page displayed at the beginning of the application, which is displayed immediately at login.

**addStudents.pug:** Responsible for displaying the structure of new students' add-on system.

**updateStudent.pug:** Responsible for displaying the structure of the update system presented to the user.

1. **server.js:** A file responsible for connecting to the primary server in the application and from which you can view the connection of a secondary router which is a studentRouter. In this file you can view the following connections:

**Connection to a database - Academy:** responsible for all actions and updates made in the application directly to the student collection.

**Connection to a database -Academy Log:** Responsible for recording all actions performed in the Student Administrator application. Takes care to record each action, the full date on which the action was performed, the type of action and the manner of sending.

**mylogDB ()** - An asynchronous function associated with the Academy Log database, is responsible for all documentation of its application to the server at any time and its presentation in another database.

**Special function used:**

**window.addEventListener** - This JavaScript function is displayed in the header part of the Pug pages. In order to direct the user according to comments and errors that may arise because the user has to fill out forms that are sent to the database and meet the condition of correctness. The uniqueness of this thing is that whenever a particular error pops up the mongoose along with node.js sends back to the user what the detailed error is and thus makes the user experience more user-friendly and understandable. Thus whenever there is a comment with the help of the reserved word message you can send the comment to the customer and ask him to confirm that he has read.

**Thank you,**

**Natalie Pitts**