**ДОКУМЕНТАЦИЯ BACKEND**

**1 ОСНОВНИ КОМПОНЕНТИ И ФУНКЦИЙ НА BACKEND**

**1.1 ИЗПОЛЗВАНИ ТЕХНОЛОГИИ**

**- javascript**

**- node.js (javascript сървър)**

**- mariaDB (База данни, подобна на mySQL, със разликата че със отворен код)**

**1.2 БАЗА ДАННИ**

**Базата данни е базирана на mariaDB, вариант на mySQL, но със отворен код. Тя съдържа три таблици, и е отговорна за данните за коментарите, потребителите и новинарските статии.**

**1.3 РЕГИСТРАЦИЯ**

**Кода за регистрация е напълно имплементиран, както и вързката със сървъра, и mariaDB база данни, като може един потребител да се регистрира, и ще бъде запазен във базата данни**

**1.4 ЛОГИН, И СЕСИОННО ЗАПАЗВАНЕ**

**Код и функций за сесионно запазване, като се провряват въведените данни за логин със потребителските данни във базата данни, и се запазва сесията, като по този начин не трябва да се влиза отново. Тази функция е имплементирана до там, че проверката за потребителските данни е функционална**

**2.0 ПОДРОБНО ОПИСАНИЕ НА БАЗА ДАННИ**

**2.1 ПРЕГЛЕД НА ТАБЛИЦИ**

**Базата данни съдържа три на брой таблици:**

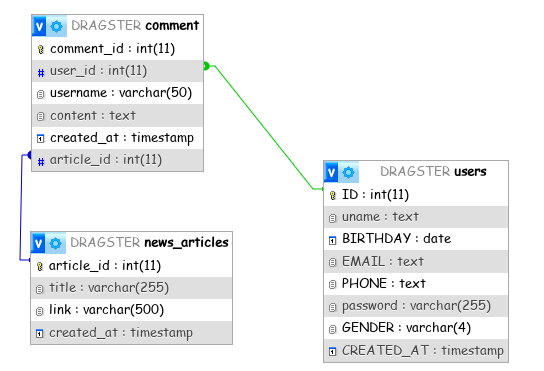
**comment – Отговорна за запазването на информация, отнасяща се до коментарите, публикувани от отделни потребители**

**users – Отговорна за запазването на информация, отнасяща се до самите потребители, включително коментарите, които са публикували**

**news\_articles – Отговаря за новините и блог постовете, включително със**

**връзка към коментарите, публикувани от потребители**

**2.2 ОТНОШЕНИЕ НА ТАБЛИЦИ, И ER ДИАГРАМА**

****

**2.3 ТАБЛИЦА ЗА ПОТРЕБИТЕЛИ: „users“**

**Съдържа следните полета:**

**| ID         | int(11)      | NO   | PRI | NULL auto\_increment |              | uname | text | NO | | NULL |               | BIRTHDAY | date | NO | | NULL |               | EMAIL | text | NO | | NULL |**

**| PHONE | text | YES | | NULL |               | password | varchar(255) | NO | | NULL |            | GENDER | varchar(4) | YES | | NULL |               | CREATED\_AT | timestamp | YES | | current\_timestamp() |**

**ID – цяло число, като служи за номер на потребителя. Пореден идентификационнен номер, които се задава последователно при регистрация. Т.Е 2129 - я потребител ще има потребителско ID от 2103**

**uname – идва от username, потребителско име на потребителя**

**BIRTHDAY – Рожден ден, във формат DATЕ, като се задава от потребителя при регистрация**

**EMAIL – Имейл**

**PHONE – Телефонен номер, по желание**

**password – Парола**

**GENDER – Пол, по желание**

**CREATED\_AT – Дата на създаване, като се зачита от самия момент на подаване на заявката, т.е. натискането на бутона „РЕГИСТРИРАНЕ“**

**2.4 ТАБЛИЦА ЗА КОМЕНТАРИ „comment“**

**съдържа следните полета:**

**| comment\_id | int(11) | NO | PRI | NULL |auto\_increment|   
| user\_id | int(11) | NO | MUL | NULL |                 | username | varchar(50) | NO | | NULL |                | content | text | NO | | NULL |               | created\_at | timestamp | YES | | current\_timestamp() |              | article\_id | int(11) | NO | MUL | NULL |**

**comment\_id – пореден номер на коментара, работи на аналогичен принцип на ID във таближата за потребителите.**

**user\_id – Извежда се от таблицата за потребителите, и представлява идентификационния номер на публикуващия потребител**

**username – Името на потребителя, публикувал коментара, може и да се премахне във някоя бъдеща точка , тъй като тази информация може да се изведе от user\_id чрез съответния код.**

**content – Съдържанието на коментара, буквално текста.**

**created\_at – Дата на публикуване, като се зачита от натискането на бутона „Публикувай“**

**article\_id – Идентификационнен номер на статията, към която е написан коментара**

**2.5 ТАБЛИЦА ЗА СТАТИИ „news\_articles“**

**съдържа следните полета:**

**| article\_id | int(11)      | NO   | PRI | NULL |auto\_increment |   
| title | varchar(255) | NO   |     | NULL | |                | link | varchar(500) | NO   |     | NULL | |                | created\_at | timestamp    | YES  |     | current\_timestamp() |**

**article\_id – пореден идентификационнен номер на статия**

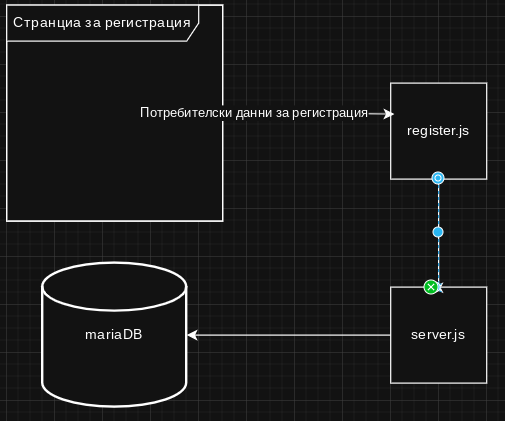
**title – заглавие на статия**

**link – Връзка към статия**

**created\_at – Дата на създаване**

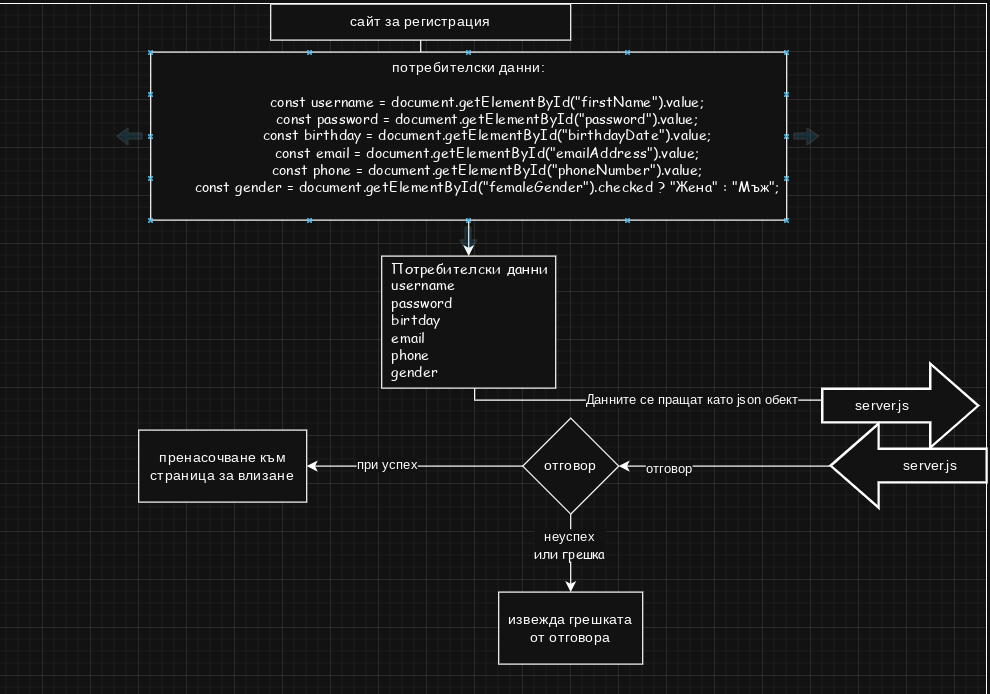
**3.0 РЕГИСТРАЦИЯ**

**3.1 Принцип на работа**

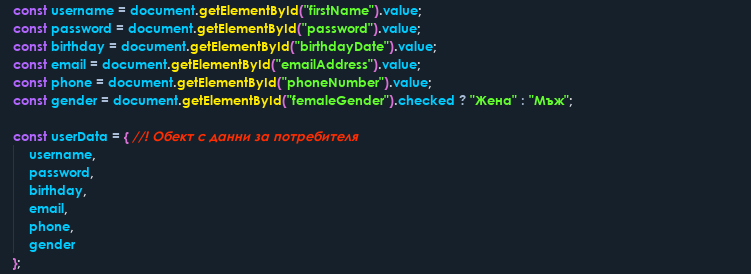
**При системата за регистрация има два javascript файла. Register.js и server.js. Целта на Register.js е да приема въведените данни от полетата във страницата за регистрация, и да ги подава на server.js, където се записват във базата данни. Със UML диаграма системата е следната:** **3.2 register.js, диаграма, и разяснение**

**3.2.1 UML диаграма**

**register.js отговаря за приемането на потребителски данни от сайта за регистрация, като ги превръща във json обект. И следователно ги подава на server.js за запис във базата данни. UMD диаграма на файла представлява:**

**3.2.2 РАЗЯСНЕНИЯ ПО КОДА, И ДОКУМЕНТИРАНЕ СТЪПКА ПО СТЪПКА**

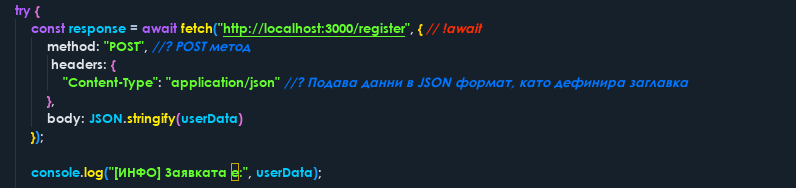
**3.2.2.1 ПЪРВА СТЪПКА: ПРИЕМАНЕ НА ДАННИ:**

**Във първата стъка трябва да приемем данните от това какво е въвел**

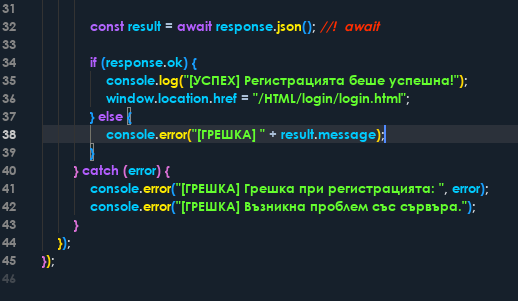
**със getEelementByid получаваме информацията от формата от тояа какво е въвел потребителя, и го запазва като обект.**

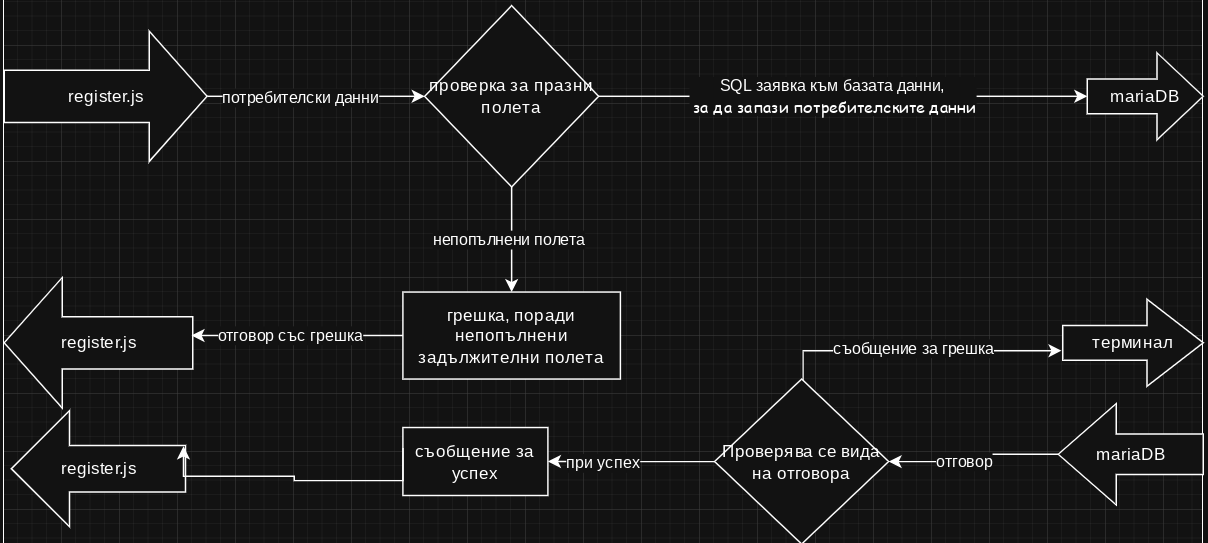
**3.2.2.2 ВТОРА СТЪПКА: ПРЕДВАНЕ НА ДАННИ:**

**Във тази стъпка, данните се превръщат във JSON обект, и се изпращат на server.js, откъдето очаква отговор.**

 **3.2.2.3 ТРЕТА СТЪПКА: ПОЛУЧАВАНЕ НА ОТГОВОР**

**И във третата и последна стъпка, очакваме и получаваме отговор от server.js, във които случай, ако има грешка, извежда съобщение във браузърната конзола, във противен случай изпраща потребителя във страница за регистрация**

****

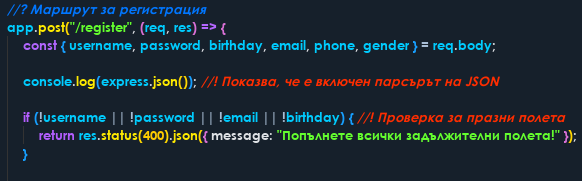
**3.3 SERVER.JS и връзка към базата данни**

**3.3.1 UML диаграма**

**3.3.2 РАЗЯСНЕНИЕ ПО КОДА**

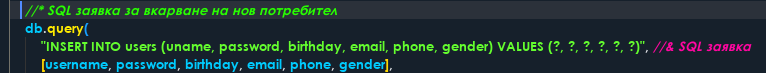
**3.3.2.1 СТЪПКА 1: ПРИЕМ И ПРОВЕРКА**

**Във тази атъпка приемаме обекта със данните, форматиран във вид json, кате там се извършва проверката за празни полета, като след това се подава заявка към базата данни.**

****

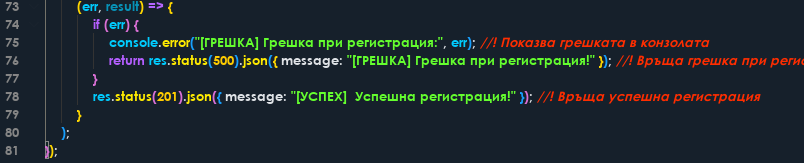
**3.3.2.2 СТЪПКА 2: ЗАЯВКА И ИЗАЧАКВАНЕ**

**Във тази стъпка правим заявка към SQL сървъра като подаваме стоиностите от полетата от json обекта към базата данни, и изчакваме отговор.**

****

**3.3.2.3 СТЪПКА 3: ОТГОВОР И ПОТВЪРЖДЕНИЕ**

**Във тази стъпка подаваме отговор и потвърждение към register.js, а във случай на грешка, подаваме грешката или я отпечатваме във терминала, във зависимост от грешката.**

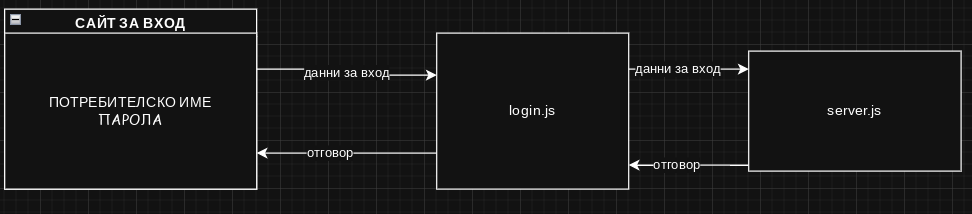


**4.0 ЛОГИН**

**4.1.1 ПРИНЦИП НА РАБОТА**

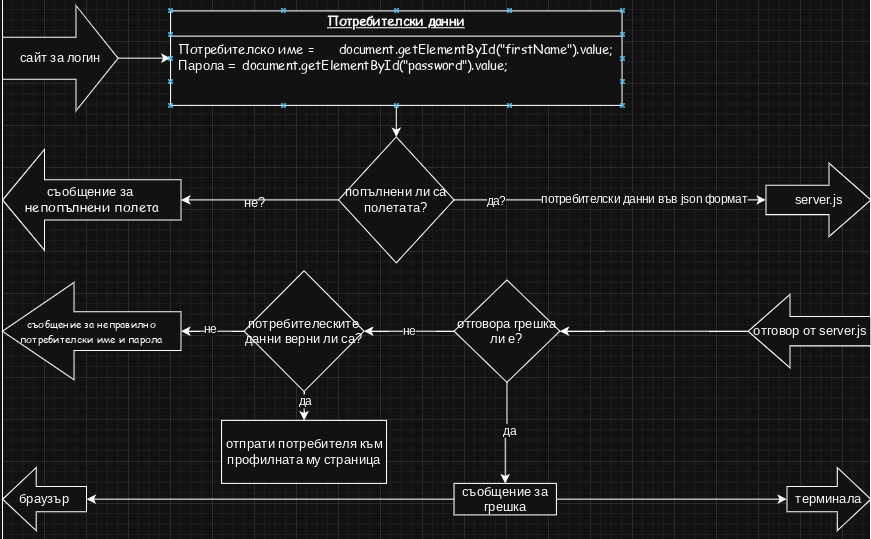
**Принципа на работа на логин е базирана на два файла, login.js и server.js. Като login.js приема въведените данни от страницата за влизане, подава ги на server.js и, последователно, ги проверява спрямо въведената във базата данни информация. Сесионно запазване, за момента, не е имплементирано.**

**4.1.2 UML ДИАГРАМА**

****

**4.2 LOGIN.JS, UML ДИАГРАМА И РАЗЯСНЕНИЯ ПО КОДА**

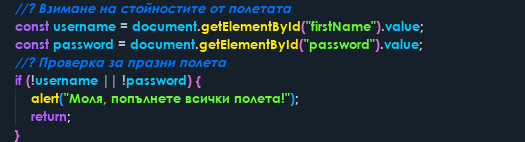
**4.2.1 UML ДИАГРАМА**

****

**4.2.2 МЕТОД НА ОПЕРАЦИЯ НА КОДА**

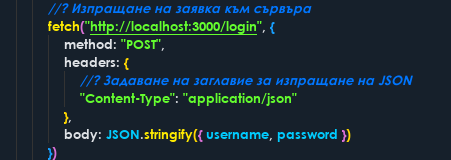
**4.2.2.1 СТЪПКА 1: ПРИЕМ И ПРОВЕРКА**

**Във първата стъпка, данните се приемат от полетата за вход, и се проверя за празни полета, във случай че няма, данните се изпращат към server.js за валидация**

****

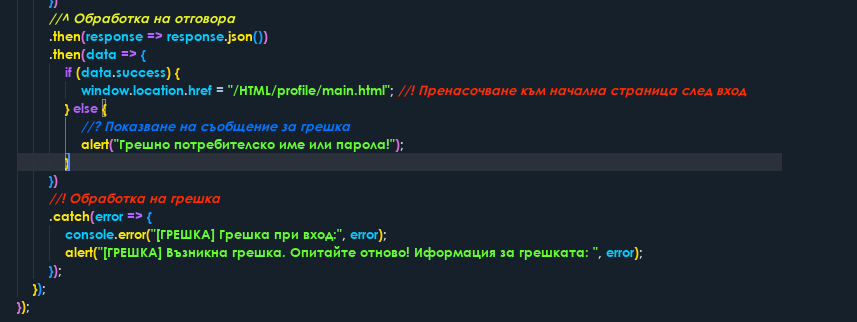
**4.2.2.2 СТЪПКА 2: ПОДАВАНЕ И ИЗЧАКВАНЕ**

**Във втората стъпка, след като данните са подадени, се изчаква отговор от server.js, какъвто и да е, дали грешка, успех или неправилно потребителско име или парола, като след това се предприема действие, спрямо вида на отговора.**

****

**4.2.2.3 СТЪПКА 3: ПРИЕМАНЕ И ОБРАБОТВАНЕ**

**Във втората стъпка, след получен отговор от server.js, във зарисимост от типа на грешката се предприемат две действия: Ако е техническа, тя се подава на терминала и браузърната конзола. Ако не е техническа, и се поради грешно въведени потребителски данни, то се изписва съобщение за невалидни потребителски данни.**

****

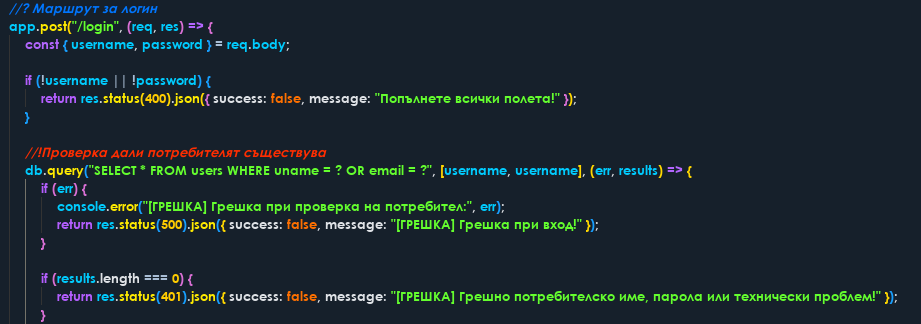
**4.3 SERVER.JS, UML ДИАГРАМА И РАЗЯСНЕНИЯ ПО КОДА**

**4****.3.1 UML ДИАГРАМА**

**4.3.2 МЕТОД НА ОПЕРАЦИЯ НА КОДА**

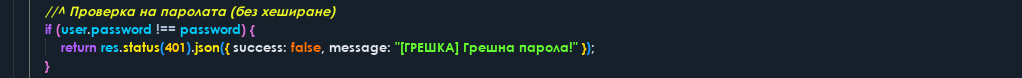
**4.3.2.1 СТЪПКА 1: ПОЛУЧАВАНЕ И ЗАЯВЯВАНЕ**

**Във първата стъпка, server.js получава директно данните от login.js, и ги препраща към базата данни**

****

**4.3.2.2 СТЪПКА 2: ПРОВЕРКА И СВЕРЯВАНЕ**

**Във тази стъпка се сверява подадената парола, срещу тази, записана във базата данни.**

****

**4.3.2.3 СТЪПКА 3: ПОТВЪРЖДАВАНЕ И ИЗПРАЩАНЕ**

**Третата стъпка представлява един ред код, които само подава отговор към login.js**

****