



СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“  
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

## Web технологии (ИС) 2022/23

Екип 15

Курсов проект на тема: Notes in the cloud

Участници:

Валентин Илиев фн: 72079

Десислава Белемезова фн: 72075

Кристина Павлова фн: 72031

Петър Чуклев фн: 72013

## Съдържание:

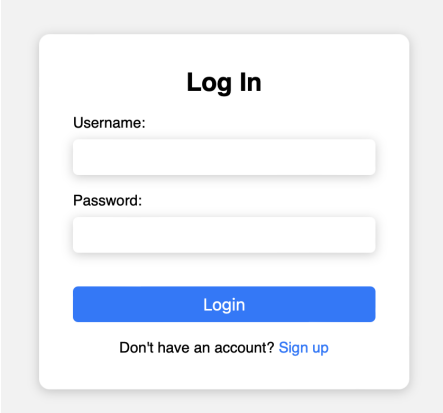
1. Описание на проекта.
2. Front-end.
3. Сървър.
  - Създаване на нов потребител (user)
  - Валидация на потребител (user)
  - Създаване на нова бележка (note)
  - Обновяване на дадена бележка
  - Връщане на информация за бележка
  - Изтриване на бележка
4. База от данни.
5. Изпълнение на кода.
  - линк към проекта в гит: [https://github.com/KristinaPavlova/web\\_team15](https://github.com/KristinaPavlova/web_team15)

## 1. Описание на проекта.

Проектът представлява уеб приложение, което предоставя регистрация на потребители, които могат да създават, записват, редактират и изтриват бележки в техния акаунт. Използвани са framework Angular за изготвянето на Front-end частта и Typescript със SQL за сървъра и базата данни.

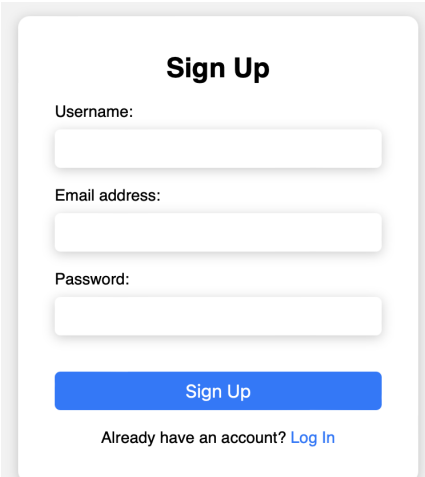
## 2. Front-end.

- Log-in форма

A screenshot of a 'Log In' form. The form is white with a light gray border and is centered on a light gray background. It has a title 'Log In' in bold black text. Below the title are two input fields: 'Username:' and 'Password:'. Below the input fields is a blue button with the text 'Login'. At the bottom of the form, there is a link that says 'Don't have an account? Sign up'.

След попълването на Log-in формата се изпраща заявка към сървъра, съдържаща username и password, с която се проверява в базата данни дали съществува такъв потребител. При успешна проверка, пренасочва потребителят в началната страница с всички негови бележки, като запазва неговия username в local storage. Формата също предоставя достъп за пренасочване на потребителя към Sign-up формата, в случай, че той няма предварително създаден акаунт.

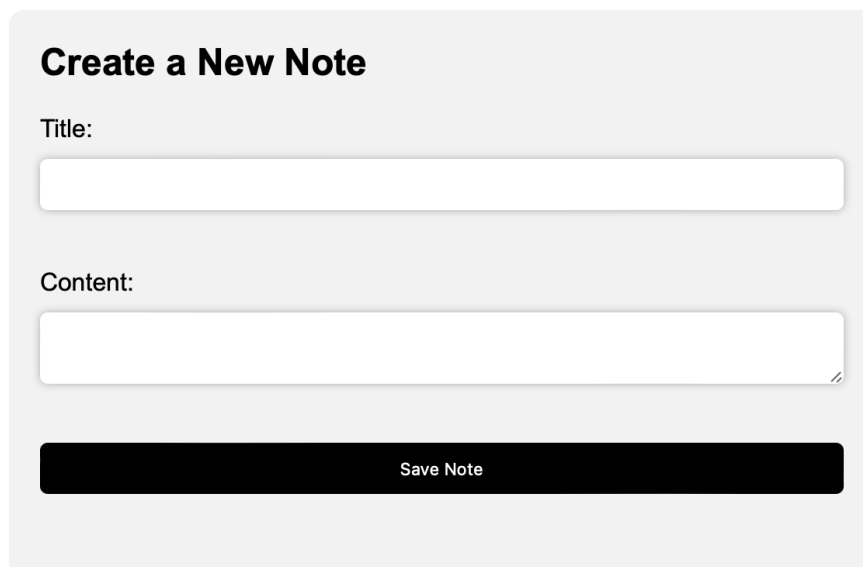
- Sign-up форма

A screenshot of a 'Sign Up' form. The form is white with a light gray border and is centered on a light gray background. It has a title 'Sign Up' in bold black text. Below the title are three input fields: 'Username:', 'Email address:', and 'Password:'. Below the input fields is a blue button with the text 'Sign Up'. At the bottom of the form, there is a link that says 'Already have an account? Log In'.

След попълването на Sign-up формата се изпраща заявка към сървъра. Успешно се създава потребител и се пренасочва в началната страница, където вече може да създава своите бележки. Формата предоставя достъп за пренасочване на потребителя към Log-in формата, в случай, че той

вече има създаден акаунт.

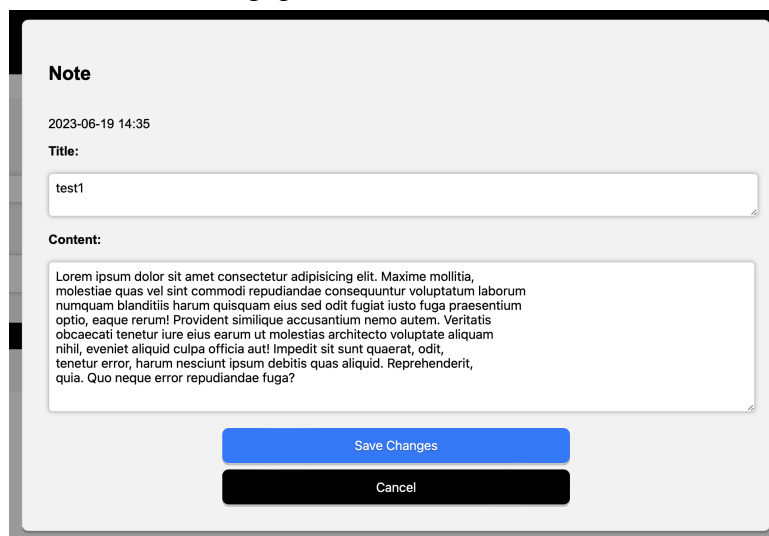
- Create note форма



The image shows a form titled "Create a New Note". It has a "Title:" label followed by a text input field. Below that is a "Content:" label followed by a larger text area. At the bottom of the form is a black button labeled "Save Note".

След попълването на Create note формата се изпраща заявка към сървъра. Успешно се добавя бележката отдясно при останалите бележки, ако има такива.

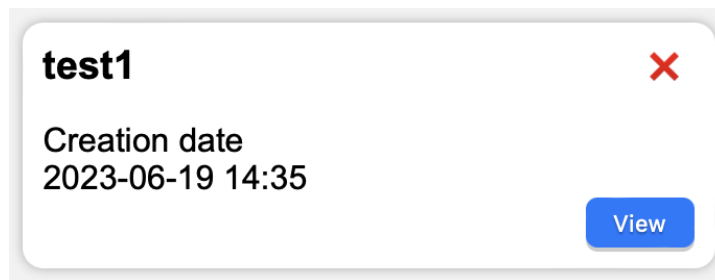
- Preview/Edit note форма



The image shows a form titled "Note". It displays a timestamp "2023-06-19 14:35". Below the timestamp is a "Title:" label followed by a text input field containing "test1". Below that is a "Content:" label followed by a text area containing Lorem ipsum text. At the bottom of the form are two buttons: a blue "Save Changes" button and a black "Cancel" button.

Чрез натискането на бутон "View" се появява бележката с нейното съдържание. Може да се редактира свободно или да се върне потребителят в началната страница чрез бутона "Cancel".

- Note



Всяка бележка се показва отдясно като не се показва съдържанието ѝ, а само заглавието и датата на създаване на бележката.

### 3. Сървър.

Сървърът приема Http заявки от front-end. Всяка заявка трябва да бъде изпратена на определен endpoint. Валидира се метода на заявката (POST, PUT, GET, DELETE) в зависимост от request-a, JSON-a (ако е приложим) и query параметрите (ако са приложими).

Основните файлове в сървърната част са:

- index.ts - началото на програмата , мястото от което се стартира сървъра.
- db.ts - съдържа връзката с базата данни
- routes/noteRoutes.ts - съдържа логиката за заявките които получава сървъра относно бележките
- routes/userRoutes.ts - съдържа логиката за заявките които получава сървъра относно потребителите

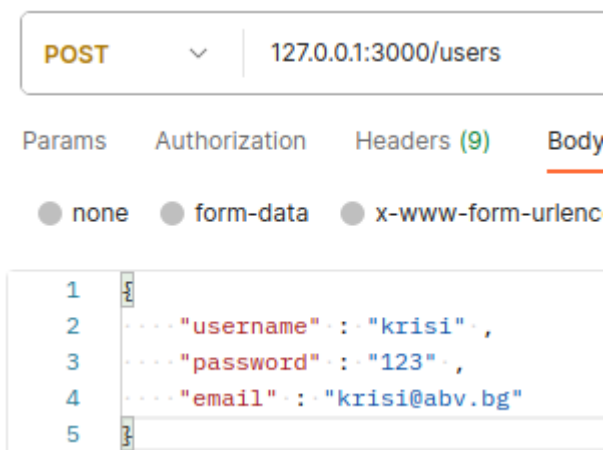
#### ➤ Създаване на нов потребител (user):

Изпраща се POST заявка на указания ip адрес и порт с url: /users

Заявката трябва да съдържа JSON който да съдържа следните компоненти:

- ❖ username (type: string)
- ❖ password (type: string)
- ❖ email (type: string)

примерна заявка в Postman:



Сървъра валидира метода , дали присъства JSON в заявката и прилага JSON схема

при успешна валидация от страна на сървъра се връща response:  
status code - 200 OK

➤ **Валидация на потребител (user)**

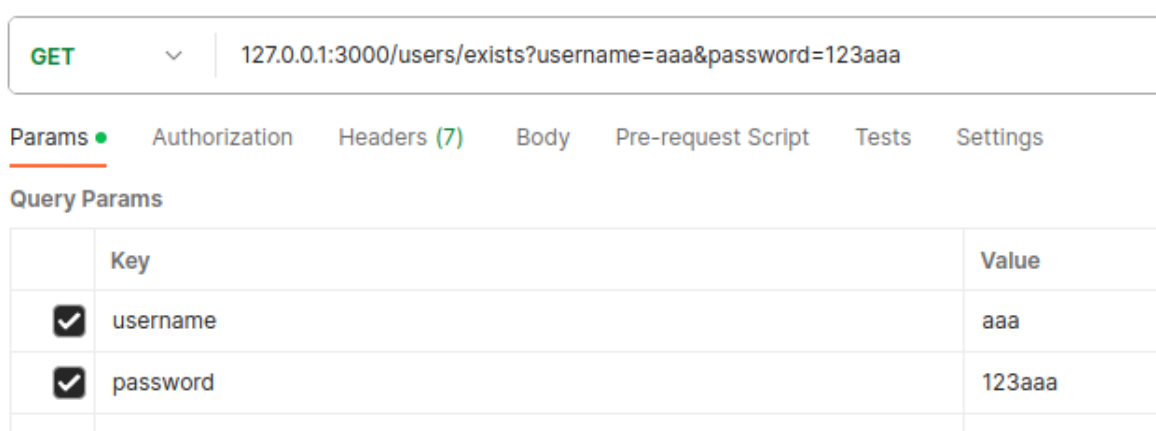
Изпраща се GET заявка на указания ip адрес и порт с url: /users/exists

Заявката трябва да съдържа query параметри :

username (type: string)

❖ password (type: string)

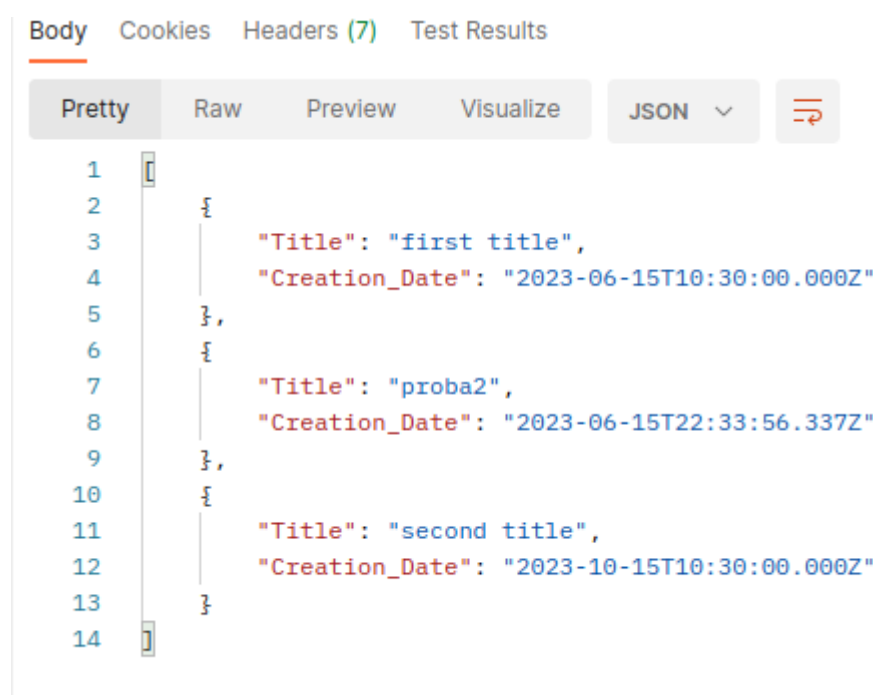
примерна заявка в Postman:



Сървъра валидира метода на заявката и параметрите.

При успешна валидация от страна на сървъра се връща отговор - списък с бележките на потребителя.

примерен response:



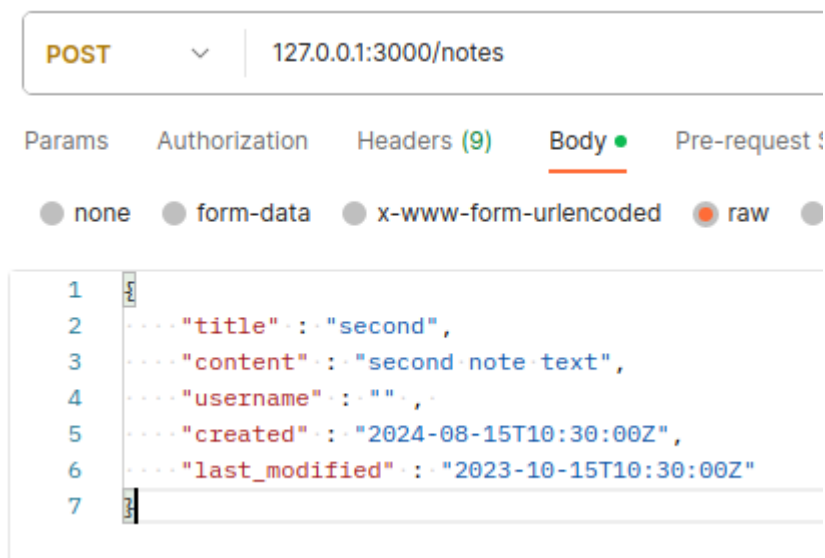
### ➤ Създаване на нова бележка(note)

Изпраща се POST заявка на указания ip адрес и порт с url: /notes

Заявката трябва да съдържа JSON който да съдържа следните компоненти:

- ❖ title (type: string)
- ❖ content (type: string)
- ❖ username (type: string)
- ❖ created (type: string)
- ❖ last\_modified (type: string)

примерна заявка в Postman:



Сървърът валидира метода, дали присъства JSON в заявката и прилага JSON схема

при успешна валидация от страна на сървъра и успешно създадена бележка се връща response:

status code - 200 OK

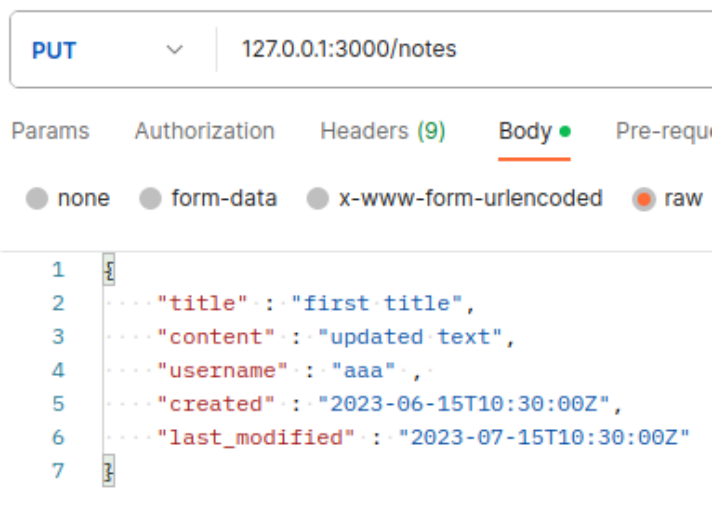
### ➤ Обновяване на дадена бележка

Изпраща се PUT заявка на указания ip адрес и порт с url: /notes

Заявката трябва да съдържа JSON който да съдържа следните компоненти:

- title (type: string)
- content (type: string)
- username (type: string)
- created (type: string)
- last\_modified (type: string)

примерна заявка в Postman:



при успешна валидация и изпълнение на заявката се връща status code 200 OK

### ➤ Връщане на информация за бележка

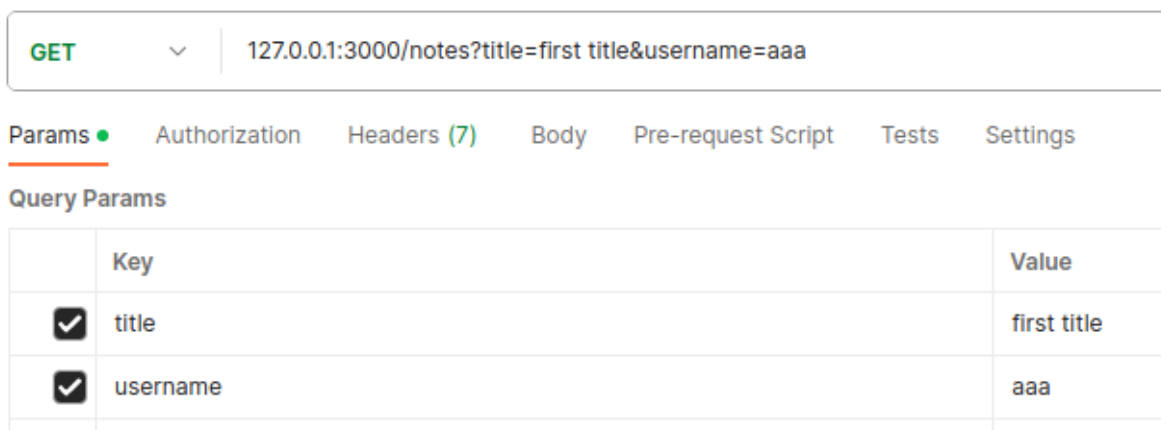
Изпраща се GET заявка на указания ip адрес и порт с url: /users/exists

Заявката трябва да съдържа query параметри :

username (type: string)

■ password (type: string)

примерна заявка в Postman:



примерен response :



```
Body Cookies Headers (7) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "Title": "first title",
3   "Content": "updated text",
4   "Creation_Date": "2023-06-15T10:30:00.000Z",
5   "Last_Modified_Date": "2023-07-15T10:30:00.000Z"
6 }
```

### ➤ Изтриване на бележка

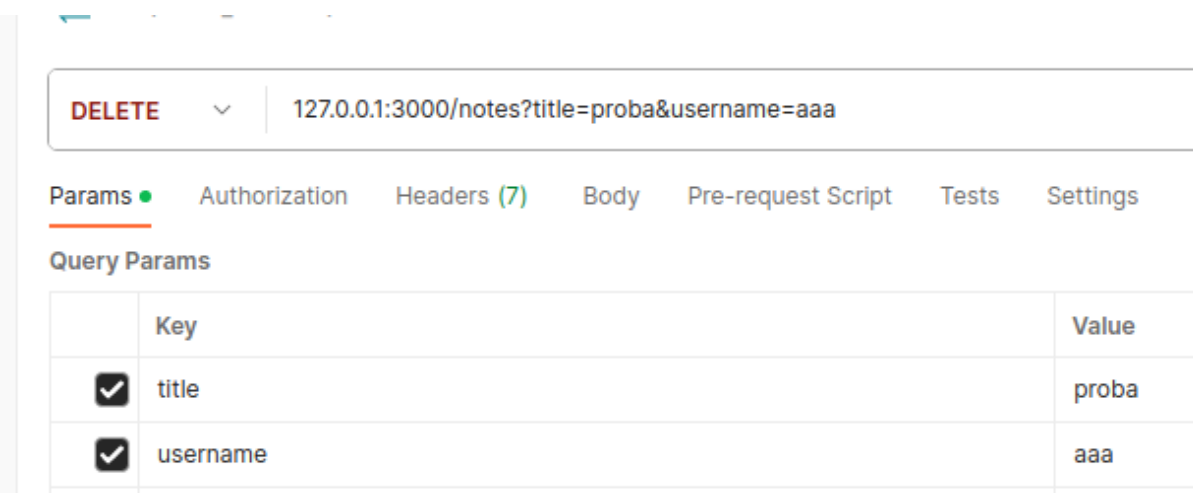
Изпраща се DELETE заявка на указания ip адрес и порт с url: /users/exists

Заявката трябва да съдържа query параметри :

title (type: string)

■ username (type: string)

Примерен request Postman:



При успешна валидация и изпълнение - успешен response:  
status 200 OK

## 4. База от данни.

За базата от данни на приложението сме избрали SQL релационна база от данни. Изборът ни е базиран на пряката релация, която нашите потребители имат със своите бележки. Използваме продуктът на Microsoft SQL Server за host-ване, поддръжка на базата. Към момента свързването от сървъра към базата се осъществява посредством LAN мрежа, на която са и двата компонента посредством SQL Server authentication с username и парола. Базата е изградена от следните таблици:

- **Users**

С колони :

- **Username** - тип VARCHAR(30) ,колоната е зададена като **PRIMARY\_KEY**
- **Email** - тип VARCHAR(50) , уникален е (UNIQUE) NOT NULL
- **Password** - тип VARCHAR(255) отново NOT NULL

- **Notes:**

С колони:

- **Title** - тип VARCHAR(50) , заглавието е уникално (UNIQUE) NOT NULL
- **Content** - тип TEXT
- **Creation\_Date** - тип DATETIME и зададен като NOT NULL
- **Last\_Modified\_Date** - тип DATETIME и зададен като NOT NULL
- **Username\_FK** - тип VARCHAR(30) и е зададен като NOT NULL

**FOREIGN KEY** в таблицата е Username\_FK който служи като референция към Username в таблицата Users.

**PRIMARY\_KEY** в таблицата представлява комбинация от (Username\_FK, Creation\_date))

## 5. Изпълнение на кода.

- **front-end**

- влизане в директорията notes
- ако нямате инсталиран angular
  1. изпълнявате: **npm install -g @angular/cli**
- изпълнява се команда: **ng serve** за стартиране на front-end

- **back-end**

Описани са необходимите инструменти и командите за тяхното инсталиране (при проектирането на проекта е използвана OS Linux за Back-end частта)

Сървърът е писан на Typescript. Необходимо е инсталирането на Typescript компилатор.

(tsc) - използвана версия в проекта: Version 5.1.3 инсталиране на typescript: **npm install -g typescript**

nodejs (node version : v18.16.0) **sudo apt install nodejs**

Node Package Manager (npm : version 9.6.7) **sudo apt install npm**

Package runner (npx : version 9.6.7)

В кода на сървъра са добавени Node модули. Тъй като те са добавени в папката .gitignore не са налични при свалянето на проекта, затова трябва да бъдат инсталирани:

**npm install express**

**npm install body-parser**

```
npm install mssql
```

```
npm install ajv
```