Para começar o projeto, peguei o código utilizado no período passado de Flask e o código disponibilizado pelo senhor, professor e, atualizado por mim para a conclusão do trabalho. Com base no código python, refiz o arquivo, mas logo percebi que somente isso não seria o suficiente para atingir a meta proposta. Pesquisando, descobri que havia uma biblioteca que conectava o flask e o MongoDB. Assistindo algumas aulas sobre o conteúdo, projetei o sistema para cadastro, update e delete nas linhas em python. Uma vez testado e funcionando, retornei ao código JS, agora, precisava fazer o login. Comecei a deletar as operações: “Create”, “Update” e “Delete” do código do trabalho de 2 pontos, em seguida, vi exemplos de códigos que faziam a conexão com o mongo pegando informações e comparando com as inseridas em html, ou seja, um método de login já pronto. Seguindo a ideia e o formato desses códigos atualizei o meu server.js para operar de acordo. Tive alguns problemas porque o MongoDB me fornece o link sem o diretório, então fiquei quebrando a cabeça até entender que eu tinha que editar o link do Mongo para adicionar o diretório também, dessa forma:

Antes:

`mongodb+srv://joaovitorfff:1234@cluster0.3bo2jrl.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0`

Depois:

`mongodb+srv://joaovitorfff:1234@cluster0.3bo2jrl.mongodb.net/faab?retryWrites=true&w=majority&appName=Cluster0`

Note que eu tive que especificar no link o projeto ao qual se referia. Da mesma forma, foi necessário especificar de qual diretório do projeto eu queria pegar as informações, o que no caso foi especificado nessa linha:

```
const user = await mongoose.connection.db.collection('usuarios').findOne({ email: email });
```

Eu defino que a coleção a qual as informações estão para a comparação é a “usuários” com cuidado em relação a pontuação de letras minúsculas e maiúsculas, porque também um tempo aí já que eu tava colocando “Usuários” quando tinha que ser “usuarios” com “u” ao invés de “U” e sem acento… Segui também o exemplo do stackoverflow para procurar primeiro pelo email inserido no html, uma vez que encontrado, ele vê se a senha fornecida é a mesma cadastrada pelo usuário do email encontrado através de um if/else dentro de um try/catch error, refletido nessa parte:

Encontra o email:

```
const user = await mongoose.connection.db.collection('usuarios').findOne({ email: email });
```

Procura a senha cadastrada:

```
app.post('/login', async (req, res) => {
    const { email, senha } = req.body;
    console.log("Email:", email);
    console.log("Senha:", senha);
    try {
            const user = await mongoose.connection.db.collection('usuarios').findOne({ email: email });
        console.log("Usuário encontrado:", user);
        if (user && user.senha === senha) {
            res.send("<h1>Login realizado com sucesso!</h1>");
        } else {
                        res.status(401).send("<script>alert('Email ou senha incorretos'); window.location.href = '/login';</script>");
        }
    } catch (error) {
        console.error('Erro ao buscar o usuário:', error);
        res.status(500).send("Erro interno do servidor");
    }
});
```

Os “console.log” foram adicionados para que fosse possível identificar o problema que eu tive durante o processo de escrita do código. Lembra no começo do arquivo que eu disse que tive um problema com o diretório…? Então, ele procurava o email, mas não encontrava nada e o usuário retornava como “Null”. Aí eu meti log nesse treco até achar em que etapa tava o problema. No final, essa merda tava no link do MongoDB, pika né? Quase me matei. Ai ok, tudo funcionando, fui no site https://css-tricks.com/guides/ e fiz um css pro html, deixar bonitinho né pra você não encrencar. Revisei o código e comecei a escrever isso. Considerações finais? Espero pelo menos um 5 pra fechar com os 2 do trabalho. Amém?