



**Instruções Gerais:**

- Atividade Individual
- Entregar a solução completa em formato ZIP pela tarefa do SIGAA.
- **Horário de Entrega definido na Tarefa do SIGAA**
- Formato de Entrega: Coloque os arquivos referentes ao *front e back* em um diretório. **Apague a pasta node\_modules** em ambos “front” e “back”. Compacte esses diretórios em um único diretório com formato “.zip” e faça a entrega no SIGAA.
- O front já foi fornecido e não deve se fazer nenhuma modificação.
- A estrutura do back também foi fornecida. Sua solução será implementada no arquivo “index.js”.

**Objetivos: Familiarizar com Node e Express:**

- Criar rotas com express
- Tratar requisições e retornar respostas
- Acessar e manipular arquivos com Node simulando base de dados.

**Tarefa Única: Sistema de Cadastro de Usuários para conhecer os cursos de computação da Unifei.**

Nessa tarefa você deverá criar o “back” do sistema de cadastro, implementando as rotas necessárias. A Figura 1 contém a página de login. A Figura 2 a página de cadastro. O cadastro pode ser acessado clicando em “Cadastro”



**Bem vindo ao sistema de cadastro da UNIFEI**



**Entre para acessar os serviços**

Email

Senha

Entrar

[Não possui conta?](#)

[Cadastro](#)

Figura 1



Crie uma nova conta

Usuário

Email

Senha

Criar Usuário

Figura 2

**Funcionamento:**

1 – Login.

Após feito o login, o sistema irá abrir uma página que irá listar os cursos de computação como mostrado na Figura 3. O sistema já possui um usuário cadastrado. (note que se encontra no arquivo “banco-dados-usuario.json”, mas a rota não está implementada.)

Email: [jaum@teste.com.br](mailto:jaum@teste.com.br)

Senha: 123456

**Busque o curso pelo nome ou deixe vazio para retornar todos.**

Curso

Listar

...

Figura 3

2 – Ao clicar no botão “Listar” deixando o campo “Curso” vazio, todos os cursos de computação serão retornados. Esses cursos se encontram no arquivo “cursos.json”. A Figura 4 apresenta o retorno esperado (após o servidor implementar a rota corretamente).

Busque o curso pelo nome ou deixe vazio para retornar todos.

Curso

Listar

**Ciência da Computação**

Duração: 4 anos  
Período: integral

**Sistemas de Informação**

Duração: 4 anos e meio  
Período: noturno

**Engenharia de Computação**

Duração: 5 anos  
Período: integral

**Mestrado em Ciência e Tecnologia da Computação**

Duração: 2 anos  
Período: integral

Figura 4

3 – Ao clicar no botão “Listar” colocando um curso (ou parte do nome) válido, os cursos serão retornados como mostrado nas Figuras 5 e 6. Observe que na Figura 6 três cursos foram retornados.

Busque o curso pelo nome ou deixe vazio para retornar todos.

Curso

Listar

**Sistemas de Informação**

Duração: 4 anos e meio  
Período: noturno

Figura 5

Busque o curso pelo nome ou deixe vazio para retornar todos.

Curso

Listar

**Ciência da Computação**

Duração: 4 anos  
Período: integral

**Engenharia de Computação**

Duração: 5 anos  
Período: integral

**Mestrado em Ciência e Tecnologia da Computação**

Duração: 2 anos  
Período: integral

Figura 6

## Ambiente:

### 1 - Executando o front

Execute o comando “npm i”, na pasta “front” para instalar todas as dependências necessárias. Para executar o “front”, digite o comando “npm run dev”. Esse comando deve ser executado na raiz do “front”, onde se encontra o arquivo “package.json”. Nesse momento o sistema não irá funcionar adequadamente pois não existe um servidor implementado (back).

### 2 – Executando o back

Execute o comando “npm i” na pasta “back” para instalar todas as dependências necessárias. Em seguida execute o comando “nodemon index.js”, no WLS adicione o *flag* -L. Se o comando não funcionar, instale o “nodemon” de forma global com o comando “npm i -g nodemon”.

Observe que nesse cenário você precisará acessar dois terminais diferentes. Um na pasta “front” e outro na pasta “back”.

## Requisito:

Sua tarefa consiste em codificar o arquivo “index.js” que se encontra no diretório “back” e implementar todos os requisitos.

**1 – Rota “/login”.** Codifique a rota “/login” que deverá tratar o verbo POST. A rota deverá realizar as seguintes tarefas.

1.1 – Acessar o arquivo “banco-dados-usuario.json” e verificar se o usuário e senha estão corretos. Se estiver incorreto, deverá retornar a mensagem “**Usuario ou senhas incorretas.**”

1.2 – Acessar o arquivo “banco-dados-usuario.json” e verificar se o usuário existe. Se não existir, deverá retornar a mensagem “**Usuario com email `{email}` não existe. Considere criar uma conta!.**”

1.3 – Acessar o arquivo “banco-dados-usuario.json” e verificar se usuário existe. Se existir e a senha estiver correta, deverá retornar a mensagem “**Autenticado com Sucesso**”

1.4 – Caso tenha sucesso, o front já está preparado para trocar para a página que irá listar as disciplinas. Por isso certifique que a mensagem segue o padrão especificado nos itens 1.1-1.3

1.5 – Dica para iniciar a rota: `app.post('/login', (req,res)=> {})`

1.6 – Os dados vindos de uma requisição POST podem ser extraídos através de “req.body” .

**2 – Rota “/create”.** Codifique a rota “/create” que deverá tratar o verbo POST.

A rota deverá realizar as seguintes tarefas.

1.1 – Acessar o arquivo “banco-dados-usuario.json” e verificar se o email já está cadastrado. Se estiver, retornar a mensagem “**Usuario com email `{email}` já existe.**”

1.2 – Caso o email não esteja cadastrado, o usuário será criado com sucesso e inserido no banco de dados com um “id” incremental. Em seguida deverá retornar a mensagem “**Tudo certo usuario criado com sucesso.**”

1.3 – Caso tenha sucesso, o front já está preparado para mostrar o botão “Fazer Login” logo abaixo do cadastro. Por isso certifique que a mensagem segue o padrão especificado nos itens 1.1-1.2. A Figura 6 apresenta esse botão. Nesse exemplo o usuário com email [jaum@teste2.com.br](mailto:jaum@teste2.com.br) foi criado.

### Bem vindo ao sistema de cadastro da UNIFEI

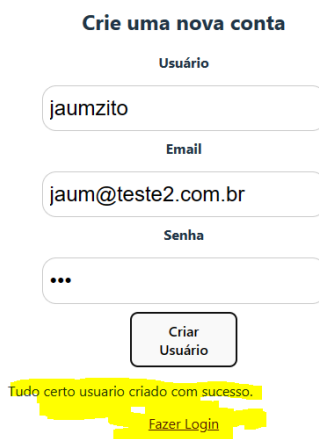


Figura 6

1.4 – Dica para iniciar a rota: `app.post('/create', (req,res)=> {})`

1.5 – Os dados vindos de uma requisição POST podem ser extraídos através de “req.body” .

**3 – Rota “/cursos”.** Codifique a rota de “/cursos” que deverá tratar o verbo GET. A rota deverá realizar as seguintes tarefas.

1.1 – Acessar o arquivo “cursos.json” e o retornar como resposta da requisição todos os cursos. Exemplo: `return res.json(cursos);`

1.2 – Ao utilizar “res.json” já fica implícito um código genérico de sucesso.

1.3 – Dica para criar a rota. `app.get('/cursos', (req,res)=> {})`

**4 – Rota “/cursos/:nome”.** Codifique a rota de “/cursos/:nome” que deverá tratar o verbo GET e possui um “path parameter” (que é diferente de “query parameter” enviado por HTML Form). O front já está preparado para enviar o parâmetro.

A rota deverá realizar as seguintes tarefas.

1.1 – Acessar o arquivo “cursos.json” e o retornar como resposta da requisição os cursos cujo parte do nome está no parâmetro enviado pelo cliente. Para isso será necessário varrer o arquivo “cursos.json” e retornar apenas os cursos que atendem ao colocado no campo de busca.

1.2 – Caso nenhum curso seja encontrado, basta devolver a mensagem: “`Curso Não Encontrado!`”

1.3 – Dicas: Observe que a rota é parametrizada, isto é, “/cursos/:nome” e sua codificação deverá seguir a seguinte assinatura: `app.get('/cursos/:nome')`. Isso significa que haverá um parâmetro que será armazenado na variável “nome” do lado backend. Por isso o símbolo “:” na rota como prefixo da palavra “nome” => “:nome”.

1.4 – Para extrair o parâmetro “:nome”, deve-ser acessar o objeto “req” na propriedade “params”. Isso é um objeto JavaScript que terá uma propriedade chamada “nome”, pois está especificada na rota.

```
const nome = req.params.nome;
```

A constante “nome” contém o parâmetro passado pelo cliente (front) e poderá ser utilizado para buscar no banco de dados qual o curso (ou cursos) foram procurados. Faça o teste com “`console.log(nome)`” passando diferentes parâmetros pelo cliente (front).

Alternativamente é possível também utilizar o operador “desestruturar”

```
const {nome} = req.params;
```