**LISTA DE EXERCÍCIOS DE JAVASCRIPT**

1. Crie um programa em javascript que ao digitar um número apresente a tabuada dele em uma div. Não utilizar o evento click. Utilize o evento onkeyup.
2. Crie um programa que ao digitar um número em uma caixa de texto apresente o mesmo em binário em uma div. Não utilizar o evento click. Utilize o evento onkeyup.
3. No click de um botão leia um nome digitado em uma caixa de texto e o insira em uma caixa de combo.
4. Crie uma caixa de combo com 5 cores. Ao click do botão recupere a cor escolhida pelo usuário e aplique ao corpo da página.
5. Utilizando 9 botões construa o jogo da velha.
6. Crie uma interface que contenha uma caixa de combo com todos os estados do brasil. Observe a seguinte api: <https://brasilapi.com.br/api/ibge/municipios/v1/sp> está é capaz de retornar todas as cidades do estado de são Paulo. Observe que: <https://brasilapi.com.br/api/ibge/municipios/v1/rj> retorna todas as cidades do estado do Rio de Janeiro.
   1. Quando o usuário selecionar um estado em uma caixa de combo, carregue em outra caixa de combo todos as cidades desse estado.
   2. Observe a plataforma <https://brasilapi.com.br/docs>. É possível encontrar diversas apis uteis, gratuitas e em português.
7. Construa uma interface gráfica que permita a entrada de matrícula, nome de um aluno e duas notas.
   1. Apresente em uma div, o nome do aluno, a média e se ele está aprovado ou reprovado.
   2. Insira em uma tabela as colunas nome do aluno, a média e se ele está aprovado ou reprovado.
   3. Insira também os dados da tabela em um array. Tenha como base o código abaixo:

|  |
| --- |
| //push insere elementos em um vetor.  let array =[];  const obj1 = {nome:"helio",email:"helioesperidiao@gmail.com"};  const obj2 = {nome:"maria",email:"maria@teste.com"};  array.push(obj1);  array.push(obj2);  console.log(array);  console.log(array[1]) // é possível acessar os objetos pela posição. |

* 1. Crie um filtro que permita pesquisar alunos pelo nome.
  2. Crie um filtro que permita escolher por meio de uma caixa de combo as seguintes opções: aprovado e reprovado.
  3. Apresente em uma div a quantidade de aprovados e reprovados.
  4. Apresente em uma div a porcentagem de aprovados e reprovados.
  5. Apresente em uma div a médias de novas.
  6. Insira na tabela uma coluna contendo um botão para excluir o aluno.

1. Crie uma interface javascript que seja capaz de consumir uma api python flask Retangulo com as seguintes rotas:
   1. GET: /retangulos/areas/:base/:altura
   2. GET: /retangulos/perimestros/:base/:altura
   3. GET: /retangulos/diagonais/:base/:altura
2. Crie uma API utilizando javascript no back end que atenda os seguintes requisitos:

|  |
| --- |
| Diagrama  Descrição gerada automaticamente |
| CREATE SCHEMA IF NOT EXISTS `PetPAW` DEFAULT CHARACTER SET utf8 ;  USE `PetPAW`;  CREATE TABLE IF NOT EXISTS `PetPAW`.`tipo` (  `idTipo` INT NOT NULL AUTO\_INCREMENT,  `nomeTipo` VARCHAR(45) NULL,  PRIMARY KEY (`idTipo`))  ENGINE = InnoDB;  CREATE TABLE IF NOT EXISTS `PetPAW`.`Raca` (  `idRaca` INT NOT NULL AUTO\_INCREMENT,  `nomeRaca` VARCHAR(45) NULL,  `idTipo` INT NOT NULL,  PRIMARY KEY (`idRaca`, `idTipo`),  INDEX `fk\_raca\_tipo1\_idx` (`idTipo` ASC),  CONSTRAINT `fk\_raca\_tipo1`  FOREIGN KEY (`idTipo`)  REFERENCES `PetPAW`.`tipo` (`idTipo`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  CREATE TABLE IF NOT EXISTS `PetPAW`.`pet` (  `idpet` INT NOT NULL AUTO\_INCREMENT,  `nome` VARCHAR(45) NULL,  `idade` TINYINT NULL,  `obs` VARCHAR(45) NULL,  `idRaca` INT NOT NULL  PRIMARY KEY (`idpet`, `idRaca`),  INDEX `fk\_pet\_raca\_idx` (`idRaca` ASC),  CONSTRAINT `fk\_pet\_raca`  FOREIGN KEY (`idRaca`)  REFERENCES `PetPAW`.`Raca` (`idRaca`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB; |

Construa uma api do tipo rest com as seguintes rotas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Rota | Envio de dados | Funcionalidade | Retorno |
|  |  |  |  |  |
| POST | petshops/tipos/ | {  “nomeTipo”:”Cachorro”  } | Cadastrar um tipo de animal | {  “cod”:”OK”,  “msg”:”Cadastrado com sucesso”,  “Cadastrado”: {  “idTipo”: 1,  “nomeTipo”:”Cachorro”  }  } |
| GET | petshops/tipos/ | “” | Retorna todos os tipos em ordem alfabética | [{  “idTipo”: 1,  “nomeTipo”:”Cachorro”  },  “idTipo”: 2,  “nomeTipo”:”gato”  }  ] |
| GET | petshops/tipos/id | petshops/tipos/1 | Retorna o tipo de animal com o id correspondente. | {  “idTipo”: 1,  “nomeTipo”:”Cachorro”  } |
| PUT | petshops/tipos/id | {  “nomeTipo”:”Cachorro”  } | Altera o tipo | {  “cod”:”OK”,  “msg”:”alterado com sucesso”,  “alterado”: {  “idTipo”: 1,  “nomeTipo”:”Cachorro”  }  } |
| DELETE | petshops/tipos/id | petshops/tipos/1 | Exclui um tipo | {  “cod”:”OK”,  “msg”:”Excluído com sucesso”,  “excluido”: {  “idTipo”: “1”,  “nomeTipo”:””  }  } |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Rota | Envio de dados | Funcionalidade | Retorno |
|  |  |  |  |  |
| POST | petshops/racas/ | {  “nomeRaca”:”vira-lata”,  “idTipo”:1  } | Cadastrar uma raça de animal | {  “cod”:”OK”,  “msg”:”Cadastrado com sucesso”,  “Cadastrado”: {  “idRaca”:1,  “nomeRaca”:”vira-lata”,  “idTipo”:1  }  } |
| GET | petshops/racas/ | “” | Retorna todas as raças em ordem alfabética | [{  “idRaca”:2,  “nomeRaca”:”Labrador”,  “idTipo”:1  “nomeTipo”:”Cachorro”  },  {  “idRaca”:1,  “nomeRaca”:”vira-lata”,  “idTipo”:1  “nomeTipo”:”Cachorro”  }] |
| GET | petshops/racas/id | petshops/racas/2 | Retorna uma raça com o id correspondente. | {  “idRaca”:2,  “nomeRaca”:”Labrador”,  “idTipo”:1  “nomeTipo”:”Cachorro”  } |
| PUT | petshops/racas/ | {  “idRaca”:2,  “nomeRaca”:”Labrador”,  “idTipo”:1  } | Altera uma raça | {  “cod”:”OK”,  “msg”:”alterado com sucesso”,  “alterado”: {  “idRaca”:2,  “nomeRaca”:”Labrador”,  “idTipo”:1  }  } |
| DELETE | petshops/racas/id | petshops/racas/2 | Exclui uma raça | {  “cod”:”OK”,  “msg”:”Excluído com sucesso”,  “excluido”:{  “idRaca”:2,  “nomeRaca”: null,  “idTipo”:null  “nomeTipo”:null  }  } |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Rota | Envio de dados | Funcionalidade | Retorno |
| POST | /pet | {  “nome”:”Bob”,  “idade”: 3,  “obs”: “Bravo”,  “idRaca”:2  } | Cadastra um novo pet | {  “cod”:”OK”,  “msg”:”Cadastrado com sucesso”,  “Cadastrado”: {  “idPet”:3,  “nome”:”Bob”,  “idade”: 3,  “obs”: “Bravo”,  “idRaca”:2  }  } |
| GET | /pet | “” | Retorna todos os pets em ordem alfabética | [  {  “idPet”:3,  “nome”:”Bob”,  “idade”: 3,  “obs”: “Bravo”,  “idRaca”:2,  “nomeRaca”: “Labrador”  }  ] |
| GET | /pet/id | /pet/3 | Retorna o pet com id igual a 1 | {  “idPet”:3,  “nome”:”Bob”,  “idade”: 3,  “obs”: “Bravo”,  “idRaca”:2,,  “nomeRaca”: “Labrador”  } |
| PUT | /pet/id | {  “nome”:”Bob”  “idade”: 3  “obs”: “Bravo”  “idRaca”:2,  “nomeRaca”: “Labrador”  } | Atualiza o pet | {  “cod”:”OK”,  “msg”:”alterado com sucesso”,  “alterado”: {  “idPet”:3,  “nome”:”Bob”,  “idade”: 3,  “obs”: “Bravo”,  “idRaca”:2,  “nomeRaca”: “Labrador”  } } |
| DELETE | /pet/id | /pet/3 | Exclui o pet com id específico. | {  “cod”:”OK”,  “msg”:”excluído com sucesso”,  “excluido”: {  “idPet”:3,  “nome”:”Bob”,  “idade”: 3,  “obs”: “Bravo”,  “idRaca”:2,  “nomeRaca”: “Labrador”  } } |

1. Crie uma interface gráfica para realizar o crud em cada uma das tabelas do exercício 4. Para fazer a interface da tabela pets permita que quando o usuário selecione em uma caixa de combo um tipo e que sejam carregados automaticamente as raças desse tipo em uma outra caixa de combo.