Documentação REST do projecto Easy Parking

Recurso parques ( /api/parques )

|  |
| --- |
| **Listar todos os parques**  Devolve a lista de todos os parques com a sua informação  Permite obter todos os parques ou filtrados pelo nome/parte nome ou apenas de um parque quando dado um parqueID |
| /api/parques?Nome=:text&parqueID=:int **(get)** |
| Parâmetros:  **nome** (opcional): Os parques mostrados têm de conter o texto neste parâmetro no nome do parque;  **parqueID** (opcional): O parque mostrado tem de ter o id igual ao número inteiro texto neste parâmetro. |
| Sucesso (200):  [  { "LugaresPrioritarios": 4, "Tipologia": "Superfície", "ParqueID": 3, "Latitude": 38.706007, "Longitude": -9.175066,Descricao": "Morada: Rua das Fontainhas, 13-45", "Nome": "Alcântara", "LugaresTotal": 202, "precoDiario": 10, "LugaresDisponiveis": 186 },  { “LugaresPrioritarios": 7, "Tipologia": "Estrutura", "ParqueID": 9, "Latitude": 38.74784, "Longitude": -9.177575,"Descricao": "Morada: Rua João Chagas, nº 3 - Urb. A.Moinhos", "Nome": "Alto dos Moinhos", "LugaresTotal": 279, "precoDiario": 19.4, "LugaresDisponiveis": 269 },  … ] |
| Erro:  500: Erro de Servidor |
| Exemplo:  let Nome = document.getElementById("search").value;  let parques = await $.ajax({  url: "/api/parques?Nome=" + Nome,  method: "get",  dataType: "json"  }); |

|  |
| --- |
| **Mudar lugares disponíveis**  Verifica se os lugares disponíveis do parque (com o id nos dados recebidos) somado com o valor (1 ou -1) nos dados recebidos é maior ou igual a 0 e menor ou igual aos lugares totais do parque, de seguida altera os lugares disponíveis para o resultado dessa soma. |
| /api/parques/updateLugares **(put)** |
| Dados:  { "parqueID": 1, "valor" : 1 } |
| Sucesso (200):  {  "fieldCount": 0,  "affectedRows": 1,  "insertId": 0,  "serverStatus": 2,  "warningCount": 0,  "message": "(Rows matched: 1 Changed: 1 Warnings: 0",  "protocol41": true,  "changedRows": 1  } |
| Erros:  500: Erro de servidor |
| Exemplo:  let parqueObj = {  parqueID: sessionStorage.getItem("parqueID"),  valor: valor  }  let result = await $.ajax({  url: "/api/parques/updateLugares/",  method: "put",  dataType: "json",  data: JSON.stringify(parqueObj),  contentType: "application/json"  }); |

Recurso Reservas ( /api/reservas )

|  |
| --- |
| **Obter todos as reservas de um utilizador**  Devolve a lista de reservas do utilizador |
| /api/reservas/reservasUser/:userID **(get)** |
| Parâmetros:  **userID** (obrigatório): id do utilizador a quem pertence as reservas |
| Sucesso (200):  [  { "Estado": "Ativa", "Nome": "Alcântara", "Descricao": "Morada: Rua das Fontainhas, 13-45", "Codigo": "DI1YA0573", "DataHora": "25/01/2021 às 03:41", "DiaReserva": "27/01/2021" },  { "Estado": "Ativa", "Nome": "Alto dos Moinhos", "Descricao": "Morada: Rua João Chagas, nº 3 - Urb. A.Moinhos", "Codigo": "0ZA73EFZT", "DataHora": "25/01/2021 às 04:49", "DiaReserva": "28/01/2021" },  … ] |
| Erros:  500: Erro do servidor |
| Exemplo:  let reservas = await $.ajax({  url: "/api/reservas/reservasUser" + sessionStorage.getItem("userID"),  method: "get",  dataType: "json",  }); |

|  |
| --- |
| **Fazer uma nova reserva**  Recebe o dia da reserva e os id 's do utilizador e do parque num objeto, chama a função getCodigo que gera um código random e verifica se já existe na base de dados |
| /api/reservas/newBooking **(post)** |
| Dados: {  “diaReserva”: “24/02/2020”,  “parqueID”: 1,  “userID”: 5 } |
| Sucesso (200):  {  "CustomerId": 2, "FirstName": "Leonie", "LastName": "Köhler", "Company": **null**,  "Address": "Theodor-Heuss-Straße 34", "City": "Stuttgart", "State": **null**,  "Country": "Germany", "PostalCode": "70174", "Phone": "+49 0711 2842222",  "Fax": **null**, "Email": "[leonekohler@surfeu.de](mailto:leonekohler@surfeu.de)", "SupportRepId": 5  } |
| Erros:  500: Erro de servidor |
| Exemplo:  let login = {  firstname: document.getElementById("fname").value,  lastname: document.getElementById("lname").value,  email: document.getElementById("email").value,  }  let customer = await $.ajax({  url: "/api/customers/login",  method: "post",  data: JSON.stringify(login),  dataType: "json",  contentType: "application/json"  });  sessionStorage.setItem("customerId",customer.CustomerId);  window.location = "profile.html"; |

|  |
| --- |
| **Verificar reservas não utilizadas ou em espera**  Verifica reservas não utilizadas (com diaReserva igual ao dia anterior) ou em espera com diaReserva igual ao dia atual(hoje) e muda estados e os lugares disponíveis do parque a que pertence |
| /api/reservas/newState **(put)** |
| Sucesso (200):  {  "fieldCount": 0, "affectedRows": 1, "insertId": 0, "serverStatus": 2, "warningCount": 0, "message": "(Rows matched: 1 Changed: 1 Warnings: 0", "protocol41": true, "changedRows": 1  } |
| Erros:  500: Erro de servidor |
| Exemplo:  let reservas = await $.ajax({  url: "/api/reservas/newState",  method: "put",  dataType: "json"  }); |

|  |
| --- |
| **Verificar código da reserva para ser utilizada e muda o seu estado**  Verifica o código nos dados, o estado da reserva(ativa) e o dia, de seguida faz update ao estado dessa reserva para utilizada e devolve true, caso os dados não forem válidos devolve false.  Exemplo não implementado pois é chamado pelo parque para validar entrada de um utilizador com reserva no parque |
| /api/reservas/utilizacao/ **(put)** |
| Dados: { "Codigo": "135W69CIL", "Parque\_ID" : 21 } |
| Sucesso (200):  {  msg: "Codigo valido"  } |
| Erros:  500: Erro de servidor  404: Não existe uma reserva que corresponda a esses dados  { "msg": "Código invalido ou utilizador encontra-se no parque errado" } |
| Exemplo: |

|  |
| --- |
| **Listar as notificações do utilizador**  Devolve a lista de notificações (reservas que estão ativas ou em espera) que ainda não foram “apagadas” (reservas com o atributo IsNotificado = 0) |
| /api/reviews/api/reservasNotificacoes/:userID**(put)** |
| Parametros:  **userID** (obrigatório): id do user autenticado para o qual queremos ver as notificações |
| Sucesso (200):  [{"ReservaID":36,"Estado":"Em espera","Nome":"Avenida de Pádua","DiaReserva":"19/02/2021"},{"ReservaID":39,"Estado":"Em espera","Nome":"Alcântara","DiaReserva":"10/03/2021"}] |
| Erros:  500: Erro do servidor |
| Exemplo:  let reservas = await $.ajax({  url: "/api/reservas/reservasNotificacoes/" + sessionStorage.getItem("userID"),  method: "get",  dataType: "json",  });  showNotificacoes(reservas); |

|  |
| --- |
| **“Limpar” notificações das reservas**  Devolve a lista de reviews do parque, calcula a classificação média do parque e introduz no objeto review |
| /api/reviews/api/updateNotificacao/:reservaID **(put)** |
| Parametros:  **reservaID** (obrigatório): id da reserva em que o utilizador clica para “limpar” a notificação |
| Sucesso (200):  {  "fieldCount": 0,  "affectedRows": 1,  "insertId": 0,  "serverStatus": 2,  "warningCount": 0,  "message": "(Rows matched: 1 Changed: 1 Warnings: 0",  "protocol41": **true**,  "changedRows": 1  } |
| Erros:  500: Erro do servidor |
| Exemplo:  let reserva = await $.ajax({  url: "/api/reservas/updateNotificacao/" + reservaID,  method: "put",  dataType: "json"  }); |

Recurso Review ( /api/reviews )

|  |
| --- |
| **Fazer uma nova review**  Recebe os dados da review num objeto e os introduz na base de dados |
| /api/reviews/ **(put)** |
| Dados:  {  “parqueID”: 1, “userID”: 5, “classificacao”: 4, “comentario”: “Otimo parque”  } |
| Sucesso (200):  {  affectedRows: 1  changedRows: 0  fieldCount: 0  insertId: 34  message: ""  protocol41: true  serverStatus: 2  warningCount: 0  } |
| Erros:  500: Erro de servidor |
| Exemplo:  let review = {  parqueID: 1,  userID: 5,  classificacao: 4,  comentario: “Otimo parque  }  let result = await $.ajax({  url: "/api/reviews",  method: "post",  dataType: "json",  data: JSON.stringify(review),  contentType: "application/json"  });  window.location = "infoParque.html"; |

|  |
| --- |
| **Obter todas as reviews de um parque**  Devolve a lista de reviews do parque, calcula a classificação média do parque e introduz no objeto review |
| /api/reviews/reviewsParque/:parqueID **(get)** |
| Parametros:  **parqueID** (obrigatório): id do parque identificado nas reviews |
| Sucesso (200):  {  "reviews":[{"Nome":"Ruben Viana","Classificacao":4,"Comentario":"Barato"},{"Nome":"Ruben Viana","Classificacao":4,"Comentario":"Otimo parque"}],  "classificacaoMedia":"4.0"  } |
| Erros:  500: Erro do servidor |
| Exemplo:  let reservas = await $.ajax({  url: "/api/reservas/reservasUser/" + sessionStorage.getItem("userID"),  method: "get",  dataType: "json",  }); |

Recurso User ( /api/users )

|  |
| --- |
| **Fazer a autenticação de um utilizador**  Recebe o username e password do utilizador e caso exista um utilizador com essa informação devolve o nome e id do utilizador. |
| /api/users/LoginDados/ **(get)** |
| Dados: {  "username": "RViana",  "lastname": "Rv123",  } |
| Sucesso (200):  [  {"userID":5,"nome":"Ruben Viana"}  ] |
| Erros:  500: Erro de servidor |
| Exemplo:  let user = {  username: "RViana",  pass: "Rv123"  }  let dados = await $.ajax({  url: "/api/users/LoginDados/",  method: "get",  dataType: "json",  data: user,  contentType: "application/json"  });  if (dados[0] != null) {  await swal("Sessão Iniciada com sucesso!", "");  sessionStorage.setItem("nome", dados[0].nome);  window.location = "index.html";  } |

|  |
| --- |
| **Fazer um novo utilizador**  Recebe os dados do utilizador num objeto e os introduz na base de dados |
| /api/users/newUSer **(put)** |
| Dados:  { "nome": "Ruben Viana", "username": "RViana", "password": "Rv123", } |
| Sucesso (200):  { affectedRows: 1  changedRows: 0  fieldCount: 0  insertId: 34  message: ""  protocol41: true  serverStatus: 2  warningCount: 0 } |
| Erros:  500: Erro de servidor |
| Exemplo:  let user = {  username: newUsername,  pass: password,  nome: nome,}  let result = await $.ajax({  url: "/api/users/newUSer",  method: "post",  dataType: "json",  data: JSON.stringify(user),  contentType: "application/json"  });  if (result == null) {  swal("Username ja se encontra utilizado, por favor tente novamente com um username diferente!", "");}  swal("User registado") |