* **Somente o proprietário do conteúdo:** restringe o acesso ao conteúdo por usuário.
* **Acesso misto:** restringe o acesso de gravação por usuário, mas permite acesso de leitura público.
* **Acesso com base em atributos:** restringe o acesso a um grupo ou tipo de usuário.

allow < get, list, create, update e delete>

* **Solicitação**: o método ou os métodos invocados na instrução allow que você permite serem executados. Os métodos padrão são: get, list, create, update e delete. Os métodos de conveniência read e write permitem amplo acesso de leitura e gravação no banco de dados ou caminho de armazenamento especificado.
* **Caminho**: o banco de dados ou o local de armazenamento, representado como um caminho de URI.
* **Regra**: a instrução allow, que inclui uma condição que permite uma solicitação se ela for avaliada como verdadeira.

### Correspondência

Um bloco match declara um padrão path que corresponde ao caminho da operação solicitada (o request.path recebido). O corpo do match precisa ter um ou mais blocos match aninhados, instruções allow ou declarações function. O caminho nos blocos match aninhados é relativo ao caminho no bloco match pai.

* **Correspondências parciais**: o padrão path é uma correspondência de prefixo do request.path.
* **Correspondências completas**: o padrão path corresponde a todo o request.path.
* **Caractere curinga de segmento único**: uma variável de caractere curinga é declarada em um caminho ao colocar uma variável entre chaves: {variable}. Essa variável pode ser acessada na instrução match como um string. /example/{singleSegment}
* **Caractere curinga recursivo**: o caractere curinga recursivo, ou de vários segmentos, corresponde a vários segmentos em um caminho ou abaixo dele. Ele corresponde a todos os caminhos abaixo do local para o qual você o definiu. É possível declará-la adicionando a string =\*\* ao final da variável de segmento: {variable=\*\*}. Essa variável pode ser acessada na instrução match como um objeto path. /example/{multiSegment=\*\*}

|  |  |
| --- | --- |
| **Método** | **Tipo de solicitação** |
| **Métodos de conveniência** | |
| read | Qualquer tipo de solicitação de leitura |
| write | Qualquer tipo de solicitação de gravação |
| **Métodos padrão** | |
| get | Ler solicitações de documentos ou arquivos únicos |
| list | Ler solicitações de consultas e conjuntos |
| create | Gravar novos documentos ou arquivos |
| update | Gravar em documentos existentes do banco de dados ou atualizar metadados de arquivos; |
| delete | Excluir dados |

**FUNCAO**

* As funções podem conter apenas uma única instrução return. Elas não podem conter nenhuma lógica adicional. Por exemplo, elas não podem executar loops ou chamar serviços externos.
* As funções podem acessar automaticamente funções e variáveis do escopo em que são definidas. Por exemplo, uma função definida no escopo service cloud.firestore tem acesso à variável resource e às funções integradas como get() e exists().
* Elas podem chamar outras funções, mas não são executadas novamente. A profundidade total da pilha de chamadas é limitada a 20.
* Na versão v2 das regras, as funções podem definir variáveis usando a palavra-chave let. As funções podem ter até 10 vinculações de permissão, mas precisam terminar com uma instrução de retorno.

**A variável**request

#### request.auth

#### request.method

#### request.params

#### request.path

### A variável resource

| **Operador** | **Descrição** | **Associatividade** |
| --- | --- | --- |
| a[i] a() a.f | Índice, chamada, acesso ao campo | da esquerda para a direita |
| !a -a | Negação unária | da direita para a esquerda |
| a/b a%b a\*b | Operadores multiplicativos | da esquerda para a direita |
| a+b a-b | Operadores aditivos | da esquerda para a direita |
| a>b a>=b a<=b | Operadores relacionais | da esquerda para a direita |
| a in b | Existência na lista ou no mapa | da esquerda para a direita |
| a is type | Comparação de tipos, em que type pode ser bool, int, float, number, string, list, map, timestamp, duration, path ou latlng | da esquerda para a direita |
| a==b a!=b | Operadores de comparação | da esquerda para a direita |
| a && b | Condicional E | da esquerda para a direita |
| a || b | Condicional OU | da esquerda para a direita |
| a ? true\_value : false\_value | Expressão ternária | da esquerda para a direita |

# Como funcionam as regras de segurança

 As regras de segurança do Firebase removem a camada do meio (servidor). Assim, é possível especificar permissões baseadas em caminho para clientes que se conectam diretamente aos seus dados.

 match /cities/{city}. ->> /cities/SF ou /cities/NYC.

### Subconjuntos correspondentes

Ao aninhar instruções match, o caminho da instrução match interna é sempre relativo ao caminho da instrução match externa. Os seguintes conjuntos de regras são equivalentes:

service cloud.firestore {  
  match /databases/{database}/documents {  
    match /cities/{city} {  
      allow read, write: if <condition>;  
  
      // Explicitly define rules for the 'landmarks' subcollection  
      match /landmarks/{landmark} {  
        allow read, write: if <condition>;  
      }  
    }  
  }  
}

# Regras de segurança e o Firebase Authentication

## Identificar usuários

auth

**uid**

**token**

| **Campo** | **Descrição** |
| --- | --- |
| email | O endereço de e-mail associado à conta, se essa informação existir. |
| email\_verified | true se o usuário tiver verificado que tem acesso ao endereço email. Alguns provedores verificam automaticamente esses endereços de e-mail. |
| phone\_number | O número de telefone associado à conta, se essa informação existir. |
| name | O nome de exibição do usuário, se ele tiver sido definido. |
| sub | O UID do Firebase do usuário. Ele é exclusivo dentro de um projeto. |
| firebase.identities | O dicionário de todas as identidades associadas à conta desse usuário. As chaves do dicionário podem ser qualquer uma das seguintes: email, phone, google.com, facebook.com, github.com, twitter.com. Os valores do dicionário são matrizes de identificadores exclusivos de cada provedor de identidade associado à conta. Por exemplo, auth.token.firebase.identities["google.com"][0] contém o primeiro ID de usuário do Google associado à conta. |
| firebase.sign\_in\_provider | O provedor de entrada usado para receber esse token. Pode ser uma das seguintes strings: custom, password, phone, anonymous, google.com, facebook.com, github.com, twitter.com. |

## Definir informações personalizadas do usuário

get(/databases/(database)/documents/users/$(request.auth.uid)).data.admin) == true

**match /{document} {**

**allow read: if**

**request.time < timestamp.date(2022, 2, 2);**

**match /usuarios/{userId}/{doc=\*\*} {**

**allow write, read: if**

**request.time < timestamp.date(2022, 2, 2);**

**// request.auth.uid == userId;**

**}**

**}**

**match /usuarios/{userId} {**

**allow write, read: if**

**request.auth.uid != null && request.auth.uid == userId;**

**}**