



## **Persistência e Pesquisa de Dados**

### **Capítulo 5. Banco de Dados em Tempo Real (Real-Time Database)**

**Prof. Gustavo Aguilar**







## **Aula 5.1. Introdução a Banco de Dados em Tempo Real**

- ☐ Surgimento dos bancos de dados em tempo real.
- ☐ Players de mercado.

# ■ Surgimento dos Bancos de Dados em Tempo Real **IGTi**

- Para atender à necessidade crescente do novo paradigma de programação ➔ **Aplicações Reativas**
  - Peculiaridades: elasticidade, resiliência e responsividade.
- Nova classe de sistemas de banco de dados ➔ orientada a push
  - Mantém os dados, no lado do cliente, em sincronia com o estado atual do banco de dados “em tempo real” ➔ imediatamente após a alteração;
  - Facilitar o desenvolvimento de aplicativos reativos (apps e web).
- Na prática ➔ “monitorando” as alterações no banco (dado novo, atualizado ou excluído) ➔ envia para a camada cliente.

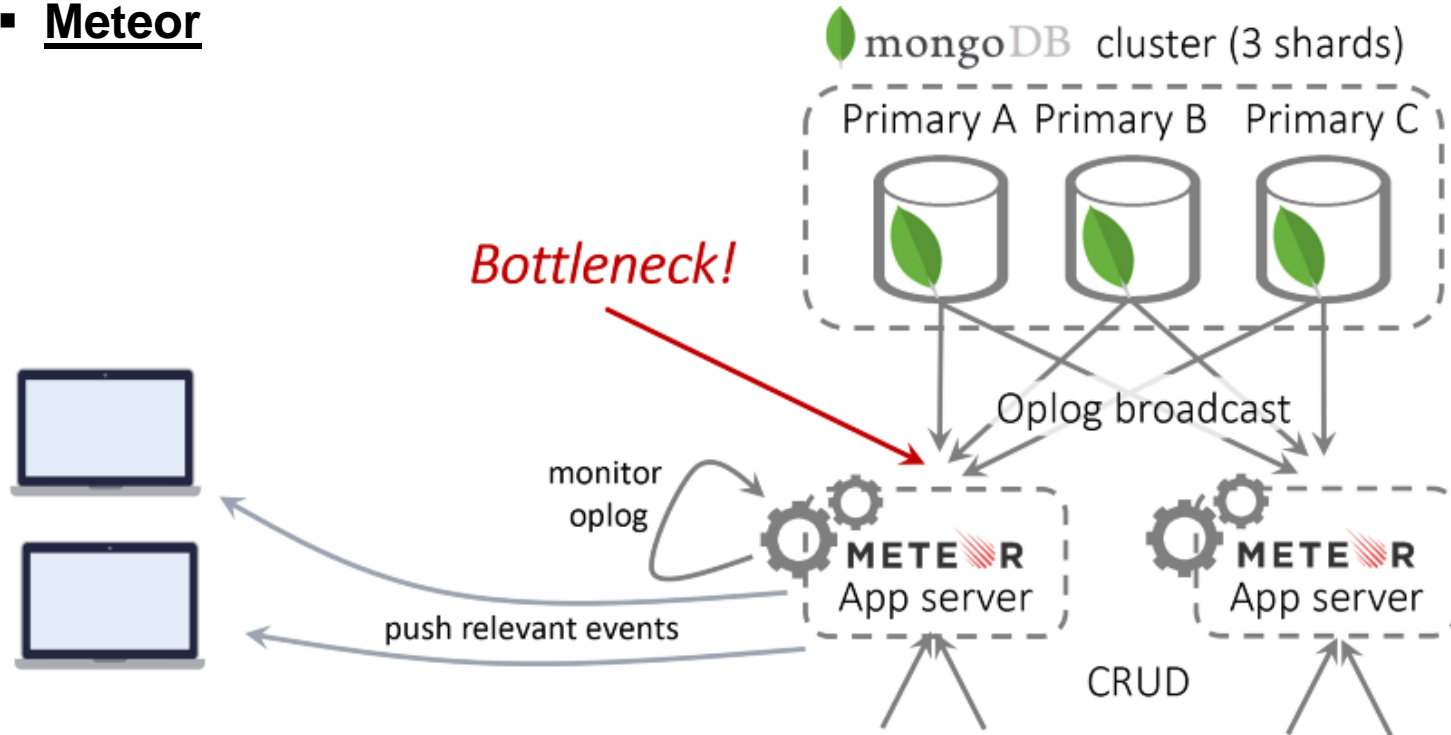
- Atualmente, não há uma grande quantidade de players de mercado investindo no desenvolvimento de bancos de dados em tempo real;
- Existem quatro grandes nomes de peso:

- *Google Firebase*  Firebase
- *Meteor*  METEOR
- *RethinkDB*  RethinkDB
- *Parse*  Parse

## ▪ Meteor

- Framework de desenvolvimento JavaScript;
- Voltado para aplicativos e sites reativos;
- Usa o MongoDB como meio de armazenamento;
- Implementações para detectar mudanças relevantes no banco de dados:
  - Change monitoring + poll-e-diff: combina o monitoramento das operações de escrita e reavaliação periódica de consultas;
  - Monitoração do log de transações (oplog) do MongoDB (mais atual).
- Considerado mais uma solução de banco de dados em tempo real, do que um sistema de banco de dados real-time propriamente dito.

- Meteor

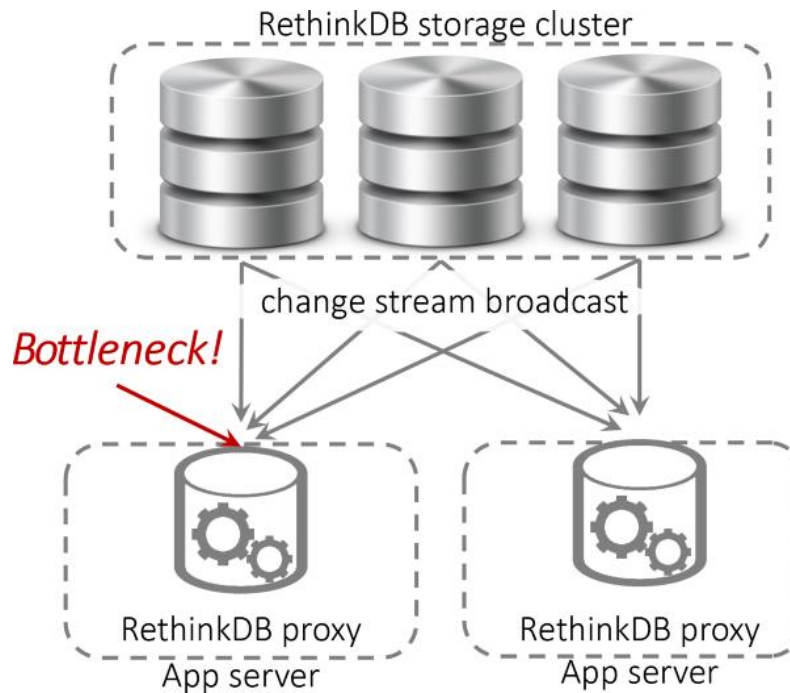


- **RethinkDB:**

- Repositório de documentos JSON;
- Expressividade em consultas comparável ao MongoDB;
- Trabalha com o conceito similar ao Meteor, de monitoração do log de operações, para prover um banco de dados em tempo real;
- Componente principal: **Proxy** recebe todas as alterações feitas nos dados, e a orquestra em tempo real para a camada cliente.



- RethinkDB



- **Parse**

- Também utiliza o MongoDB para armazenamento dos dados;
- Se propõe a ser um framework, assim como o Meteor.

- **Google Firebase**

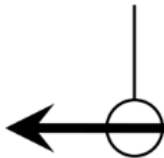
- O mais famoso e que mais evoluiu nos últimos anos;
- Um dos poucos que se apresenta, na íntegra, como um banco de dados em tempo real.

# Players de Mercado

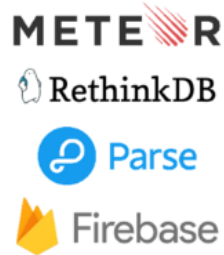


**Database  
Management**

static collections



pull-based

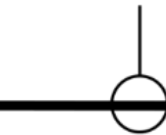


**Real-Time  
Databases**

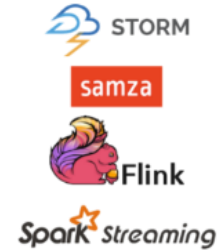
evolving collections



persistent/  
ephemeral streams



**Data Stream  
Management**



**Stream  
Processing**

ephemeral  
streams



push-based

- ☑ Surgimento dos bancos de dados realtime para atender à necessidade crescente do novo paradigma de programação de aplicações reativas;
- ☑ Poucos players de mercado;
- ☑ Apenas alguns se apresentando como um banco de dados em tempo real → destaque para o Google Firebase Realtime Database;
- ☑ Maioria como framework ou solução de banco de dados em tempo real.

- ❑ Introdução ao Firebase Realtime Database.



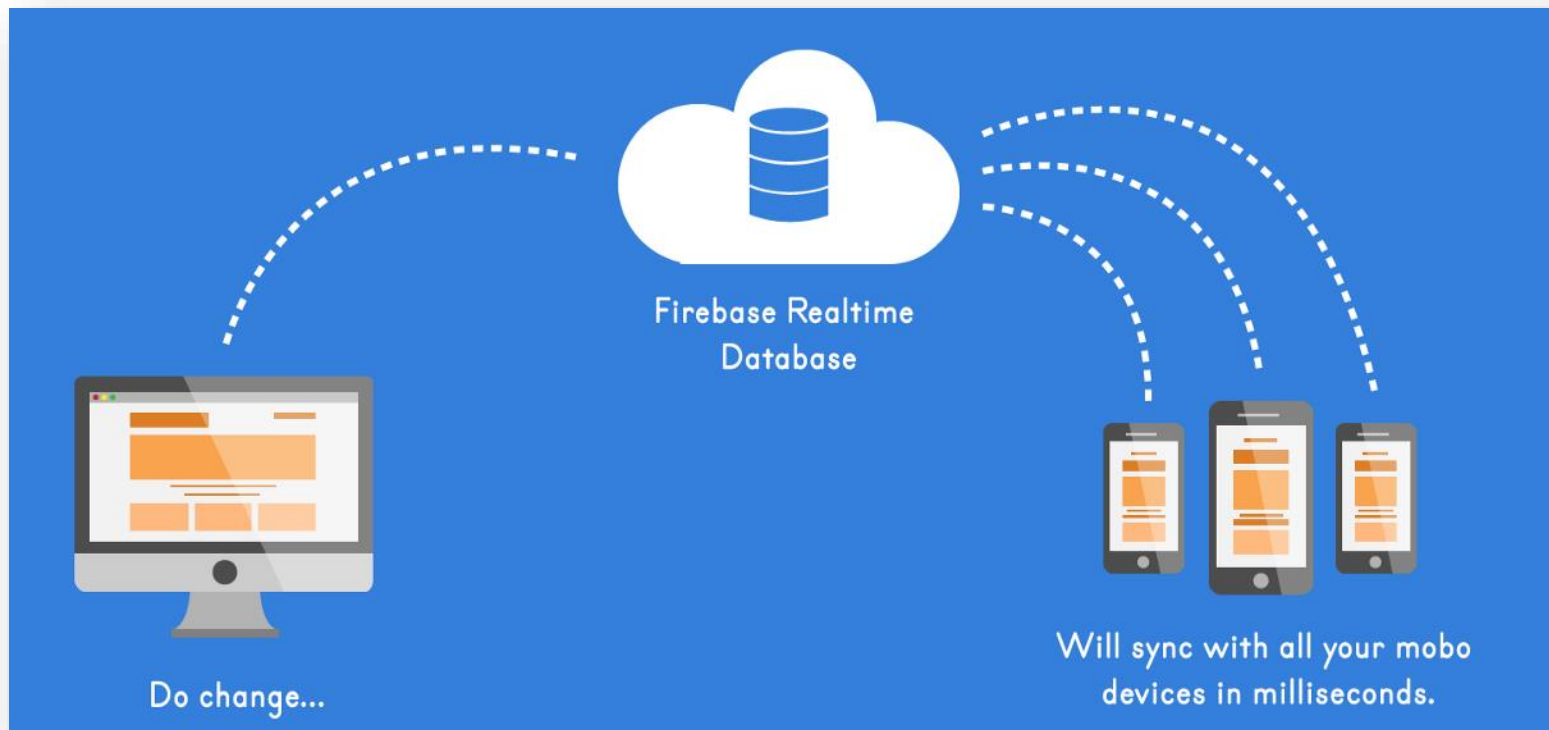
## **Aula 5.2. Introdução ao Firebase Realtime Database**

- ☐ Introdução ao Firebase Realtime Database.
- ☐ Realtime Database ou Cloud Firestore?
- ☐ SDKs disponíveis.

- Componente da plataforma ***Firebase*** de desenvolvimento de aplicativos para dispositivos móveis e Web:
  - Lançada inicialmente pela *Firebase Inc.* em 2011;
  - Adquirida pelo *Google* em 2014.
- Serviço de banco de dados NOSQL hospedado na nuvem:
  - Dados armazenados como uma grande árvore JSON;
  - Sincronizados em tempo real com todos os clientes conectados:
    - Processo viabilizado pelos SDKs de desenvolvimento disponibilizados pela plataforma;
    - Garantem que todos os clientes, ao compartilharem uma mesma instância do Firebase Realtime Database, receberão automaticamente as atualizações com os dados mais recentes.

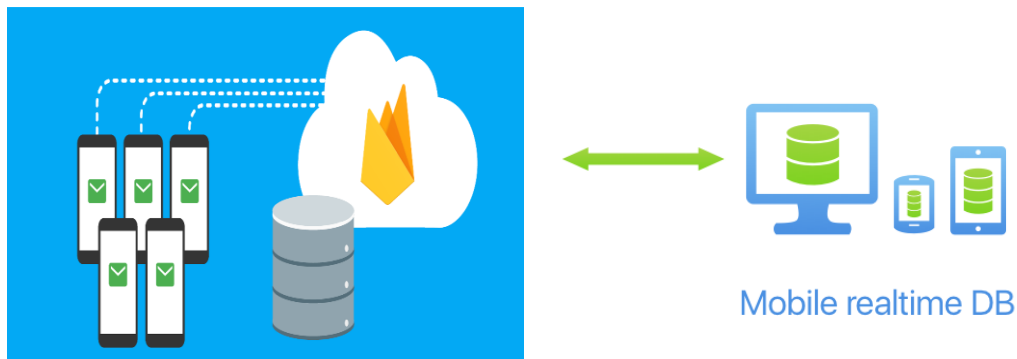


# Introdução ao Firebase Realtime Database



## ▪ Modo off-line:

- Aplicativos permanecem responsivos, mesmo sem conexão;
- O SDK do Firebase Realtime Database mantém os dados em disco;
- Quando a conectividade é restabelecida, o dispositivo cliente recebe as alterações não recebidas e faz a sincronização com o servidor.



## ▪ **Firestore Cloud Firestore**

- Serviço de banco de dados também hospedado nas nuvens;
- Google coloca o Firebase Realtime Database como o banco de dados original do Firebase e o Firebase Cloud Store como o novo banco de dados principal do Firebase;
- Melhor performance e escalabilidade;
- Modelo de dados mais intuitivo: armazenamento dos documentos JSON organizados em coleções;
- Suporte off-line para clientes Web também (no Realtime o suporte é apenas no iOS e no Android).

- **API Admin (Admin SDK):** linguagens Java, Node.js, Python e Go;
- **iOS:** linguagens Swift e Objective-C;
- **Android:** linguagens Java Android e Kotlin Android;
- **Web:** linguagem JavaScript;
- **REST API:** linguagens Clojure, Dart, Go, Java, Perl, PHP, Python, Ruby.

- ☑ Firebase Realtime Database componente da plataforma de desenvolvimento lançada pela Firebase e adquirida pela Google.
- ☑ Serviço de banco de dados real-time armazenado na nuvem que funciona no modo off-line também.
- ☑ Firebase Cloud Firestore como nova opção de banco de dados em tempo real, mais performático, escalável e intuitivo.
- ☑ SDKs disponíveis para praticamente todas as plataformas de dispositivos móveis e linguagens de programação da atualidade.

- ❑ Instalação e configuração do Firebase Realtime Database.



## **Aula 5.3. Instalação e configuração do Firebase Realtime Database**

- ☐ Configuração para SDK Web JavaScript.
- ☐ Teste de conexão e utilização.



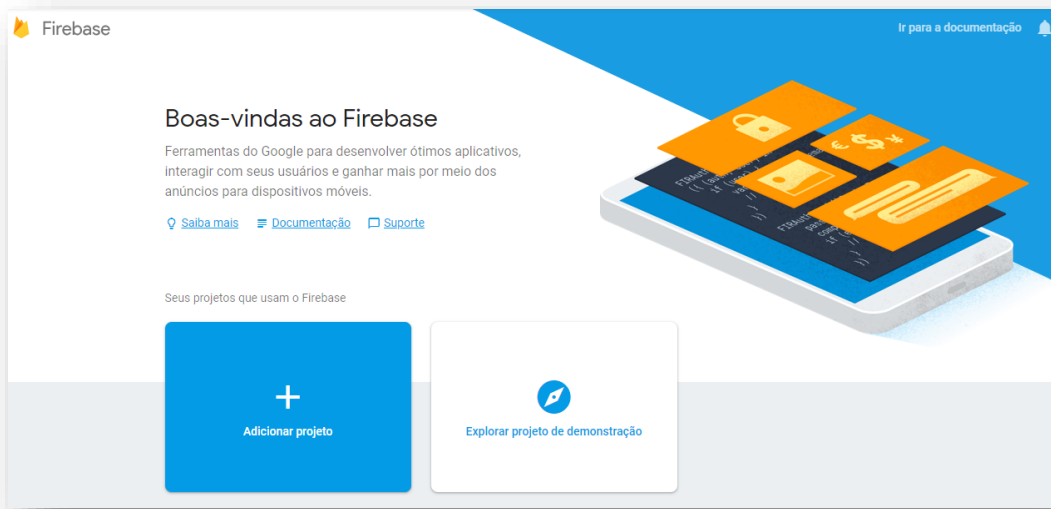
## ▪ Pré-Requisitos:

- App JavaScript, ao qual o Firebase será adicionado;
  - No link <https://firebase.google.com/docs/samples/?hl=pt-br#web>, existem vários exemplos de início rápido, dentre eles um app para Realtime Database (<https://github.com/firebase/quickstart-js?hl=pt-br>);
- Projeto do Firebase;
- Código de inicialização ➔ configurações do projeto;
- Regras de segurança de acesso ao Banco de Dados.

## ▪ Projeto do Firebase

– Criado na **Console do Firebase**

- <https://console.firebase.google.com/?hl=pt-br>
- Opção ***Adicionar projeto***.



# Configuração para SDK Web JavaScript

- Projeto do Firebase:



- **Código de Inicialização:**

## Adicionar o Firebase ao seu aplicativo da Web



Copie e cole o snippet abaixo na parte inferior do seu HTML, antes de outras tags de `script`.

```
<script src="https://www.gstatic.com/firebasejs/5.10.1/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyB-tcS-8tzQ-Vu9iby2ElSrxOVSAEHYz1A",
    authDomain: "demoppd-50a47.firebaseio.com",
    databaseURL: "https://demoppd-50a47.firebaseio.com",
    projectId: "demoppd-50a47",
    storageBucket: "demoppd-50a47.appspot.com",
    messagingSenderId: "774204272250"
  };
  firebase.initializeApp(config);
</script>
```

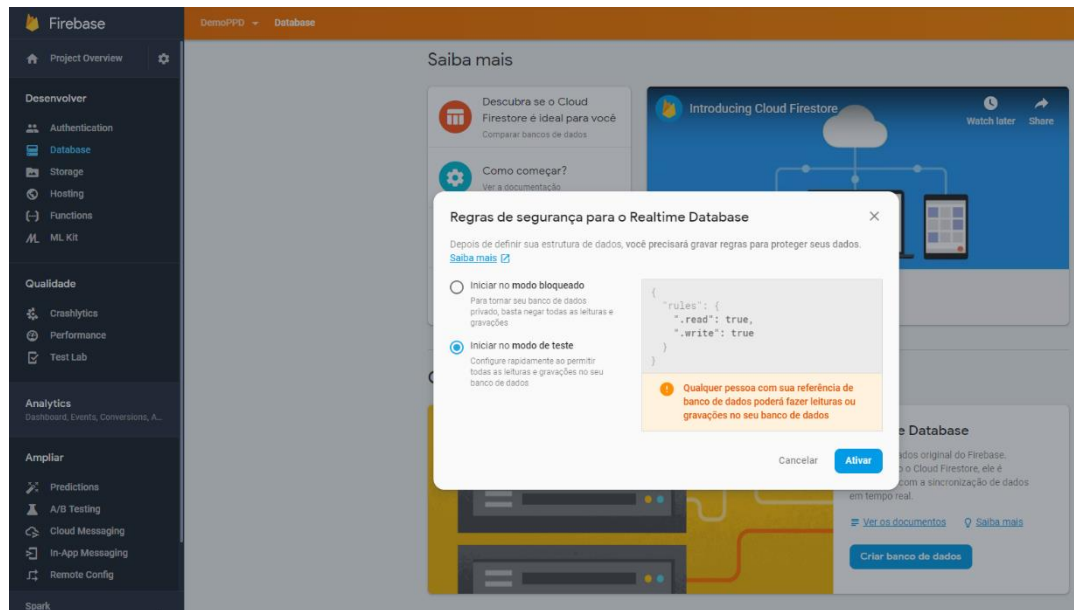
Copiar

# Configuração para SDK Web JavaScript

- Regras de segurança de acesso ao Banco de Dados

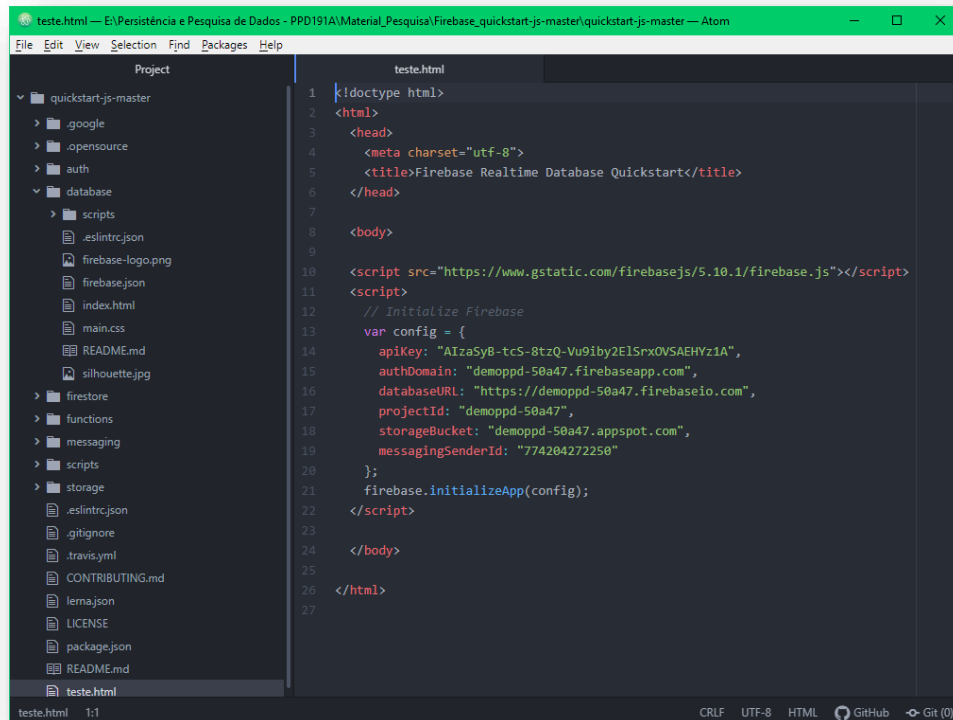
- Console do projeto:

- Database
- Realtime Database
- Criar banco de dados



# Teste de conexão e utilização

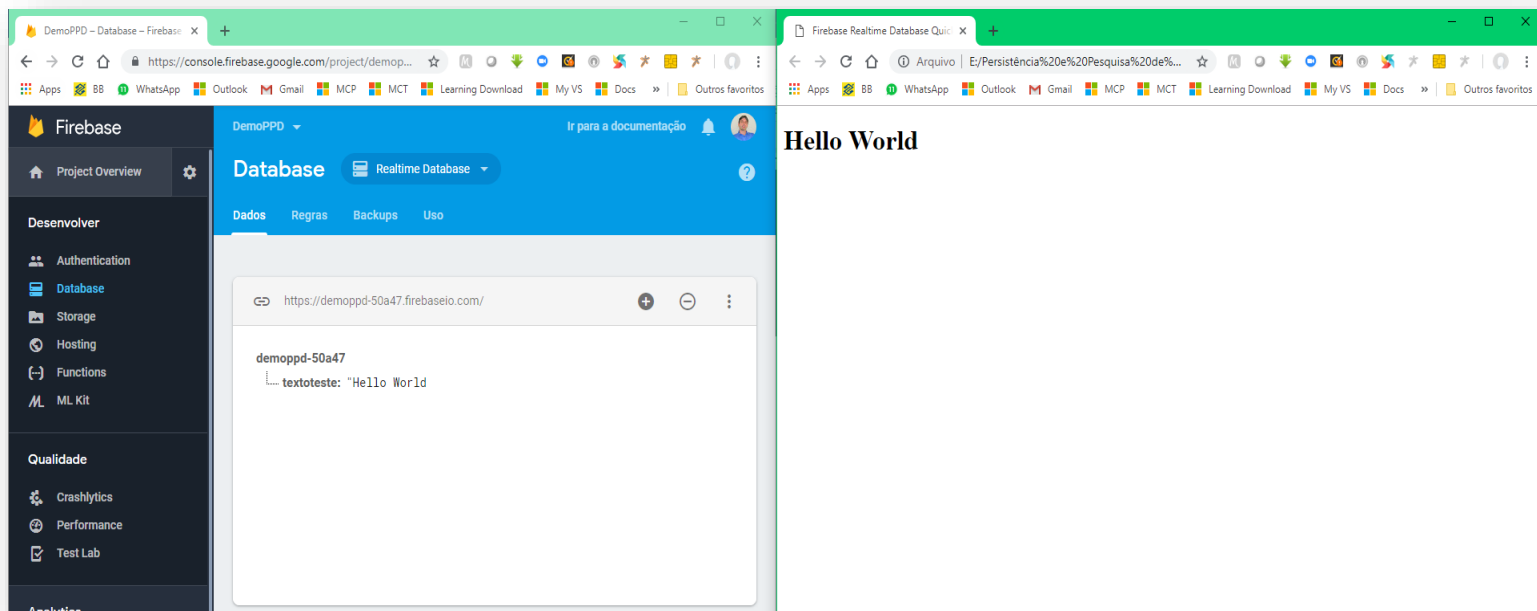
- Arquivo HTML básico;
- Inserir código de inicialização antes das tags de script;



```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Firebase Realtime Database Quickstart</title>
6   </head>
7
8   <body>
9
10    <script src="https://www.gstatic.com/firebasejs/5.10.1/firebase.js"></script>
11    <script>
12      // Initialize Firebase
13      var config = {
14        apiKey: "AIzaSyB-tcS-8tzQ-Vu9iby2ElSrxQVSAEHYz1A",
15        authDomain: "demoppd-50a47.firebaseio.com",
16        databaseURL: "https://demoppd-50a47.firebaseio.com",
17        projectId: "demoppd-50a47",
18        storageBucket: "demoppd-50a47.appspot.com",
19        messagingSenderId: "774204272250"
20      };
21      firebase.initializeApp(config);
22    </script>
23
24  </body>
25
26 </html>
27
```

- Criar um cabeçalho nível 1 (H1) no HTML e atribuir um ID
  - `<h1 id="CabTeste"></h1>`
  - Será sincronizado toda vez que um valor for alterado no banco de dados;
- Instanciar o cabeçalho pelo ID
  - `var CabTeste = document.getElementById('CabTeste');`
- Criar uma referência do banco de dados inicializado previamente e uma localização filha à entrada 'textoteste'
  - `var dbRef = firebase.database().ref().child('textoteste');`
- Sincronizar: `dbRef.on('value', snap => CabTeste.innerText = snap.val());`

# Teste de conexão e utilização





- ☑ Antes de usar o Firebase no SDK Web JavaScript, é preciso criar um projeto no Firebase, configurar as regras de segurança de acesso ao banco de dados e inserir o código de inicialização no script do ap/página.
- ☑ Teste de conexão e utilização podem ser feitos usando a própria console do Firebase.

- ☐ Trabalhando com os dados no Firebase Realtime Database.



## **Aula 5.4. Trabalhando com os Dados no Firebase Realtime Database**

- ☐ Definição da estrutura dos Dados.
- ☐ Gravação de Dados.
- ☐ Leitura de Dados.

- Dados são armazenados como objetos JSON:
  - Uma grande árvore JSON;
  - Hospedada na nuvem;
  - Não há collections como nos bancos NOSQL orientados a documentos;
  - Não há tabelas como nos bancos de dados relacionais
  - Dados adicionados ao Firebase Realtime Database se **tornam um nó na estrutura JSON, com uma chave associada;**
  - Chave pode ser fornecida via código ou gerada automaticamente usando push().

# Definição da Estrutura dos Dados

```
{ "usuarios": {  
  "gustavaagl":  
  {  
    "nome": "Gustavo Aguilar de Araújo Gonzaga Lopes",  
    "conexoes": [ "julianaglaa", "giovanaglaa", "daviglaa"]  
  },  
  "julianaglaa":  
  {  
    "nome": "Juliana Gonzaga Lopes Aguilar de Araújo",  
    "conexoes": [ "gustavaagl", "giovanaglaa", "daviglaa"]  
  },  
  "giovanaglaa":  
  {  
    "nome": "Giovana Gonzaga Lopes Aguilar de Araújo",  
    "conexoes": [ "gustavaagl", "julianaglaa", "daviglaa"]  
  },  
  "daviglaa": { ... }  
}
```

- Para **gravar dados (e também ler)**, no Firebase Realtime Database, primeiramente é necessário instanciar o banco de dados:
  - Feito através do método ***firebase.database()***;
  - Usando variável: *var database = firebase.database()*;
  - Instanciando direto: *firebase.database(). .....*;
- Métodos disponibilizados de acordo com o SDK selecionado;

- No SDK Web:
- **Método `set()`**
  - Substitui os dados no local especificado da árvore JSON, incluindo também qualquer node filho necessário.
  - *function EscreveUsuario (IDUsuario, nome, email*

```
{  
    firebase.database().ref('usuarios/' + IDUsuario).set({  
        nomeusuario: nome,  
        email: email    });  
}
```



- Método *update()*
  - Atualizar campos específicos.

```
function writeNewPost(uid, username, picture, title, body) {  
  // A post entry.  
  var postData = {  
    author: username,  
    uid: uid,  
    body: body,  
    title: title,  
    starCount: 0,  
    authorPic: picture  
  };  
  
  // Get a key for a new Post.  
  var newPostKey = firebase.database().ref().child('posts').push().key;  
  
  // Write the new post's data simultaneously in the posts list and the user's post  
  var updates = {};  
  updates['/posts/' + newPostKey] = postData;  
  updates['/user-posts/' + uid + '/' + newPostKey] = postData;  
  
  return firebase.database().ref().update(updates);  
}
```

- No SDK Web:
- **Método *on()***
  - Detectar automaticamente as alterações em um determinado caminho da árvore JSON.

```
var ContaLikes = firebase.database().ref('posts/' + postId + '/QtdeLikes');  
ContaLikes.on('value', function(snapshot) {  
  AtualizaLikes(postElement, snapshot.val());  
});
```

- **Método *once()***

- Faz uma leitura dos dados, assim que acionado.

```
var IDUsuario = firebase.auth().currentUser.uid;
return firebase.database().ref('/usuarios/' +
IDUsuario).once('value').then(function(snapshot)
{
  var nomeusuario = (snapshot.val() && snapshot.val().nomeusuario) ||
  'Anonymous';
  // ...
});
```

- Fornece inúmeras outras possibilidades, como por exemplo, a utilização de lista de dados e os respectivos métodos para leitura e gravação.
- <https://firebase.google.com/docs/database/web/lists-of-data?hl=pt-br>

- ☑ Definição da estrutura dos dados no formato JSON, em uma grande árvore armazenada na nuvem (Firebase Realtime Database).
- ☑ Para ler e gravar dados é preciso o método `firebase.database()`; para instanciar o banco de dados.
- ☑ Gravação e leitura de dados feitas através de métodos, de acordo com a SDK selecionada.
- ☑ Método de leitura `on()` como cerne da engine do Firebase como um banco de dados em tempo real.

- ☐ Overview das regras do Firebase Realtime Database.



## **Aula 5.5. Overview das Regras do Firebase Realtime Database**

## Nesta aula

☐ Autenticação.

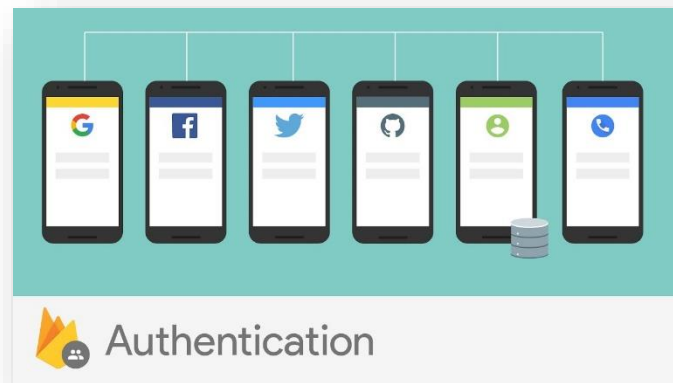
☐ Autorização.

☐ Validação.

☐ Indexação.



- Primeira etapa no desenvolvimento da camada de segurança;
- Processo de identificação dos usuários;
- Firebase Authentication → método ***firebase.auth()***;
- Suporte tradicional com e-mail e senha ou login anônimo;
- Suporte ao drop-in em métodos comuns de autenticação como o Google, Facebook e Twitter;



- Processo de controle do acesso ao banco de dados;
- Utilização de **regras** do Firebase Database:
  - Controlam o acesso de cada usuário;
  - **.read**: descreve se e quando os dados podem ser lidos pelos usuários;
  - **.write**: descreve se e quando os dados podem ser gravados;
  - Ficam armazenadas nos servidores do Firebase;
  - São sempre aplicadas automaticamente, depois de configuradas;
  - Operações só são efetivadas se as regras permitirem;
  - São aplicadas em cascata → vale para os níveis inferiores também.

- Exemplo: permitir que qualquer pessoa leia o caminho /usuarios/, mas não faça gravações.

```
{  
  "rules": {  
    "usuarios": {  
  
    }  
  }  
}
```

*Vale também para  
/usuarios/admin/gustavoagl*

***".read": true,  
".write": false***

- Firebase Realtime Database não usa esquemas de bancos de dados:
  - Facilita fazer alterações durante o processo de utilização.
- Importante uma **validação de dados** mínima:
  - Manter a consistência dos valores dos dados.
- Feita através da regra **.validate** → método **val()**
  - Não são aplicadas em cascata → todas as regras de validação precisam ser avaliadas como verdadeiras para que a gravação seja efetivada.

- Exemplo: dados gravados em /usuarios/ precisam ser uma string com menos de 100 caracteres.

```
{  
  "rules": {  
    "usuarios": {  
      ".validate": "newData.isString() && newData.val().length < 100"  
    }  
  }  
}
```

- Possível ordenar e consultar os dados de várias maneiras;
- Ambientes produtivos ou com grande volume de dados → essencial especificar **índices** para as consultas existentes;
- Especificada com a regra *.indexOn*;
- Exemplo: indexar os campos nome e CPF de uma lista de usuários:

```
{ "rules": {   "usuarios": {  
                                ".indexOn": ["nome", "cpf"]  
                                }  
  }  
}
```

- ☑ As regras (***rules***) são os recursos utilizados no Firebase para especificar e controlar quem tem permissão de leitura e gravação no banco de dados, como os dados são validados e quais os índices existentes.

- ❑ Escalonamento no Firebase Realtime Database.





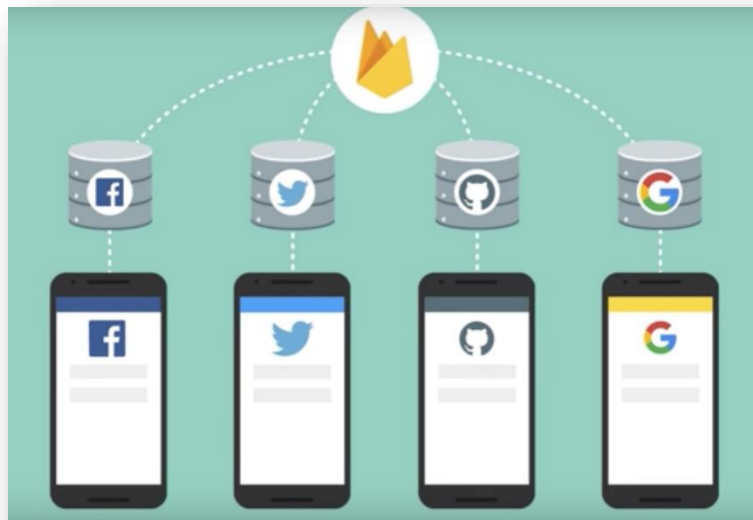
## **Aula 5.6. Escalonamento no Firebase Realtime Database**

- ☐ Restrições de Armazenamento e Processamento.
- ☐ Escalonamento no Firebase Realtime Database.

- Limites e restrições acerca do armazenamento de dados e das operações no banco de dados **para uma instância do Firebase.**
- Exemplos:
  - Limite de 100.000 conexões simultâneas em cada instância;
  - 1.000 operações de gravação/segundo para cada instância.
- Limitadores em aplicações de grande porte;
- Contornados com a técnica de **escalonamento.**

- Criação de várias instâncias do Firebase Realtime Database:
  - Cada uma delas continua com os limites conhecidos;
  - Limite Total = Soma dos limites de cada instância.
- Implementado através da técnica de fragmentação de dados:
  - **Particionamento Horizontal;**
  - Melhor maneira de otimizar o desempenho no Firebase.

- 1) Mapeamento dos dados para vários bancos de dados de acordo com as necessidades específicas do app:
  - Criação do fragmento mestre: mapa dos dados.



- 2) Criação das instâncias de banco de dados necessárias;
- 3) Configuração do app para que ele se conecte à instância do Realtime Database necessária para cada conjunto de dados.

```
// init
const app1 = firebase.initializeApp({
  databaseURL: "https://testapp-1234-1.firebaseio.com"
});

const app2 = firebase.initializeApp({
  databaseURL: "https://testapp-1234-2.firebaseio.com"
}, 'app2');

// Get the default database instance for an app1
var database1 = firebase.database();

// Get a database instance for app2
var database1 = firebase.database(app2);
```

- ☑ Para contornar os limites de armazenamento e processamento de cada instância do Firebase, pode-se escalonar a solução, usando a técnica de fragmentação horizontal de dados.

- ❑ Capítulo 6 - Pesquisa indexada em bases não estruturadas.