03

Comunicando-se com o banco usando o IDBDatabase

Transcrição

Vamos gerar uma conexão que será uma instância de `IDBDatabase`. Primeiramente, criaremos a variável `connection` e depois, vamos adicioná-la em `onsuccess`:

```
<body>
  <script>
      var connection;

      var openRequest = window.indexedDB.open('aluraframe', 1);

      openRequest.onupgradeneeded = e =>  {

            console.log('Criando ou atualizando o banco');
      };

      openRequest.onsuccess = e => {

          console.log('Conexão realizada com sucesso');

          // e.target.result é uma instância de IDBDatabase

          connection = e.target.result;
      };

      openRequest.onerror = e => {

          console.log(e.target.error);
      };

  </script>
</body>
```

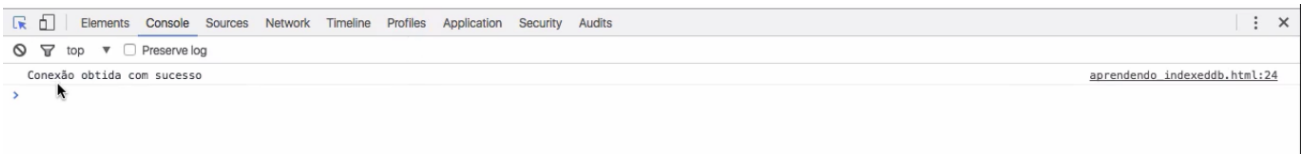
O `result` será o `IDBDatabase`, ou seja, uma conexão que estamos guardando dentro da variável `connection` que está no escopo maior. Mas antes de começarmos a interagir com o banco, adicionaremos uma *Object Store*, que é bastante semelhantes às tabelas dos bancos de dados relacionais. Dentro da `aluraframe` teremos várias Object Stores, no entanto, não é correto chamá-las de tabelas, porque elas não possuem esquemas. Diferente de um banco de dados relacional, em que existem colunas destinadas a texto, números e dados, numa Object Store, podemos gravar objetos de diferentes formatos, contanto, que sejam válidos no JavaScript. E como fazemos para criar a Object Store? Assim que criarmos o banco, criaremos no `onupgradeneeded` uma Object Store chamada `Negociacoes`. Porém, nós só teremos acesso à variável 'connection' no `onsuccess` - que só é executado depois do `onupgradeneeded`. Não há problema, também temos acesso à conexão dentro de 'onupgradeneeded', através de 'e.target.result', que atribuiremos à variável `minhaConnection`:

```
openRequest.onupgradeneeded = e =>  {

       console.log('Criando ou atualizando o banco');
```

```
        var minhaConnection = e.target.result;
        minhaConnection.createObjectStore('negociacoes');
};
```

Do `minhaConnection`, chamamos a Object Store que receberá o nome `negociacoes`. Agora, temos acesso tanto a `onupgradeneeded` como `onsuccess`. Vamos recarregar a página e ver com ficou:

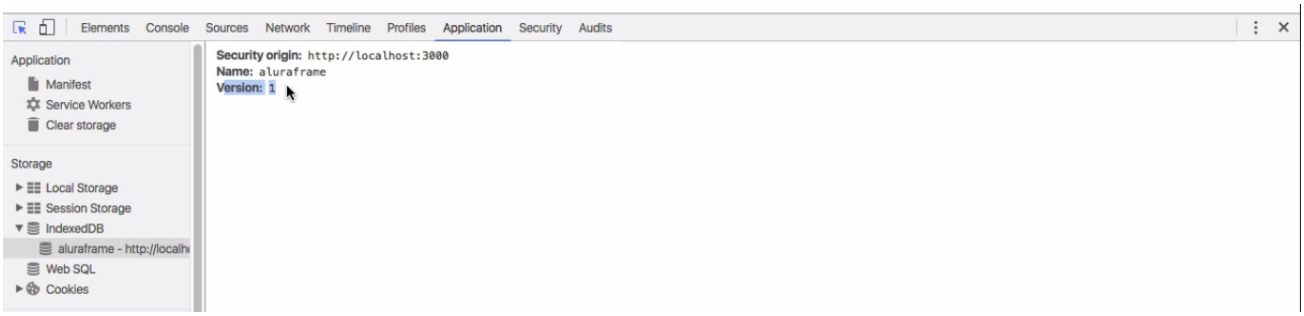


Vemos a mensagem de que a conexão foi bem-sucedida, porém, ele não chamou novamente o `onupgradeneeded`, no entanto, esta ação precisa ser realizada, porque além de criarmos o banco, uma Object Store será gerada. Se observarmos a aba "Application" no navegador, confirmaremos que nenhuma delas foi criada. Como resolveremos isto?

Vamos analisar a linha da variável `openRequest`:

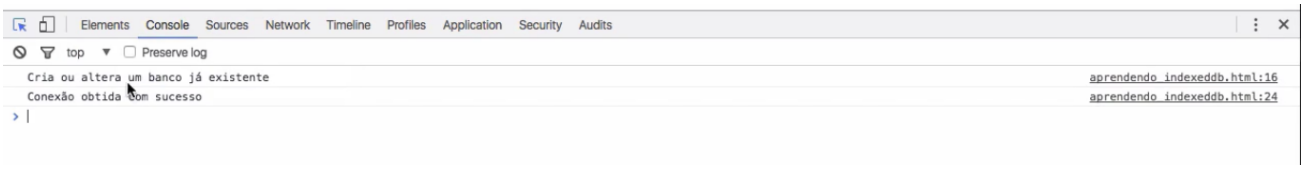
```
var openRequest = window.indexedDB.open('aluraframe', 1);
```

O valor `1` passado como parâmetro é referente à versão do banco, ou seja, no caso, utilizamos a **versão 1**. Isto significa, que quando executarmos o código novamente o `onupgradeneeded` só será invocado caso a versão passada no parênteses seja maior do que a exibida no banco criado:

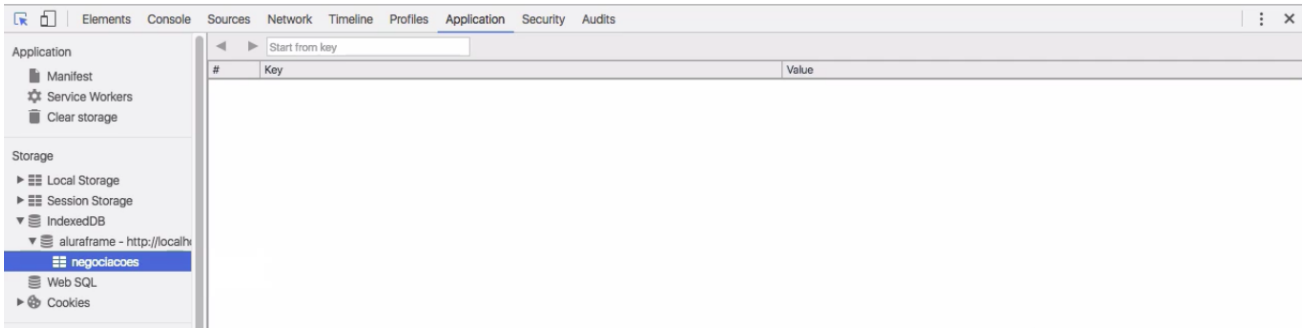


Se a versão for a mesma, ele não entende que a atualização é necessária. Então, vamos alterar o valor para `2`:

```
var openRequest = window.indexedDB.open('aluraframe', 2);
```



Agora, veremos as duas mensagens exibidas no Console. Se acessarmos a aba "Application", veremos que a Object Store `negociacoes` foi criada, mas ainda está vazia.



Ao recarregarmos a página novamente, o `onupgradeneeded` não será chamado novamente, porque a versão do banco continuará sendo a mesma.