08

O método getConnection

Aurélia, com base no que aprendeu neste capítulo, criou o método `getConnection`, que por enquanto lida com a criação da *store* `negociacoes`. Vejamos seu código:

```
class ConnectionFactory {

    constructor() {

        throw new Error("ConnectionFactory não pode ser instanciada");
    }

    static getConnection() {

        return new Promise((resolve, reject) => {

            let openRequest = window.indexedDB.open('aluraframe',4);

            openRequest.onupgradeneeded = e => {

                    if(e.target.result.objectStoreNames.contains('negociacoes')) e.tar
                    e.target.result.createObjectStore('negociacoes', { autoIncrement:

            };

        });
    }
}
```

Contudo, ela lembrou que durante o vídeo foi criado um método para isolar a criação de uma ou mais *stores*, para melhorar a legibilidade do seu código.

Qual das opções abaixo possui a classe `ConnectionFactory` alterada para separar a responsabilidade de criação de *stores* em um método em separado?

Selecione uma alternativa

A

```
class ConnectionFactory {

    constructor() {

        throw new Error("ConnectionFactory não pode ser instanciada");
    }

    static getConnection() {

        return new Promise((resolve, reject) => {

            let openRequest = window.indexedDB.open('aluraframe',4);

            openRequest.onupgradeneeded = e => {
```

```
                    ConnectionFactory._createStores(e.target.result);


            };

        });
    }

    static _createStores() {

        if(connection.objectStoreNames.contains('negociacoes')) connection.delete
        connection.createObjectStore('negociacoes', { autoIncrement: true });


    }
}
```

B

```
class ConnectionFactory {

    constructor() {

        throw new Error("ConnectionFactory não pode ser instanciada");
    }

    static getConnection() {

        return new Promise((resolve, reject) => {

            let openRequest = window.indexedDB.open('aluraframe',4);

            openRequest.onupgradeneeded = e => {

                    ConnectionFactory.createStores(e.target.result);

            };

        });
    }

    static _createStores(connection) {

        if(connection.objectStoreNames.contains('negociacoes')) connection.delete
        connection.createObjectStore('negociacoes', { autoIncrement: true });


    }
}
```

C

```
class ConnectionFactory {

    constructor() {

        throw new Error("ConnectionFactory não pode ser instanciada");
    }

    static getConnection() {
```

```
            return new Promise((resolve, reject) => {

                let openRequest = window.indexedDB.open('aluraframe',4);

                openRequest.onupgradeneeded = e => {

                        ConnectionFactory._createStores(e.target.result);

                };

            });
        }

        static  _createStores(connection) {

            if(connection.objectStoreNames.contains('negociacoes')) connection.delete
            connection.createObjectStore('negociacoes', { autoIncrement: true });


        }
    }
```