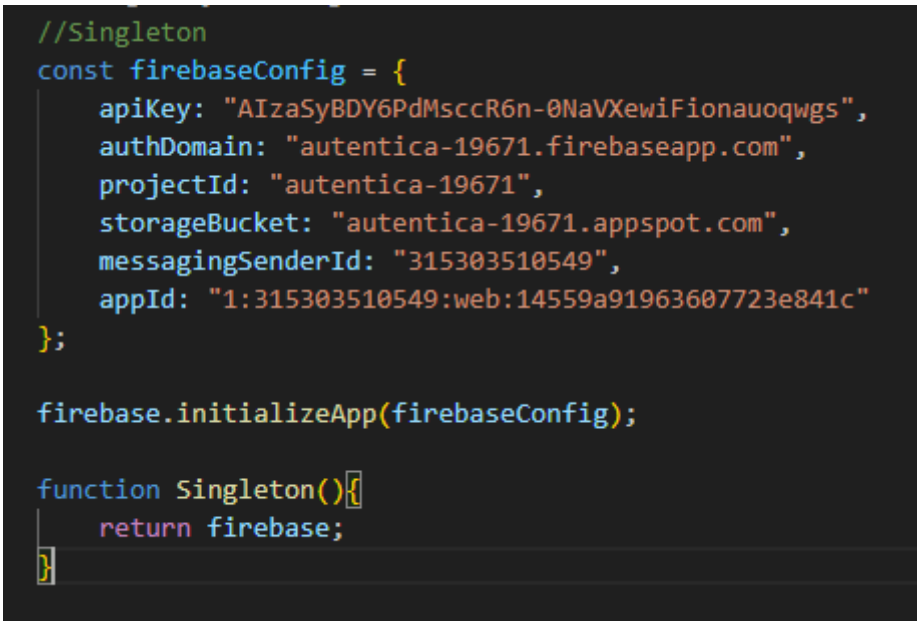
Aluno: Raphael Santos Oliveira

Link para o original: https://github.com/joaoqys/projeto-final-DS.W.git

Padrão Singleton: Além da centralização do acesso ao banco de dados e simplificar o código, o padrão faz com que a aplicação web fique mais leve e dinâmica, permitindo uma melhor optimização do código. Fiz uma única instancia de firebase inicializado que será usado para o cadastro tanto do usuário quanto do cliente.

(src>js>Singleton.js)



```
//Singleton
const firebaseConfig = {
    apiKey: "AIzaSyBDY6PdMsccR6n-0NaVXewiFionauoqwgs",
    authDomain: "autentica-19671.firebaseapp.com",
    projectId: "autentica-19671",
    storageBucket: "autentica-19671.appspot.com",
    messagingSenderId: "315303510549",
    appId: "1:315303510549:web:14559a91963607723e841c"
};

firebase.initializeApp(firebaseConfig);

function Singleton(){
    return firebase;
}
```

Padrão Factory: Além de um código mais conciso, o padrão permite maior escalabilidade no uso das classes e maior segurança, já que há uma camada separando os detalhes da criação da classe, junto de detalhes sigilosos, do usuário. Foi usado para criar as classes do usuário e cliente de maneira mais simples e facilmente escalável

(src>js>Factory.js)

```
//factory

function factory(classe,nome,email,senha,confirmaSenha,cep,telefone,permissao,ramo){
    switch(classe){
        case "Usuario":
            return new Usuario(nome, email,permissao,senha,confirmaSenha);
        break;

        case "Cliente":
            return new clientes(nome,senha,email,cep,telefone,ramo);
        break;
    }
}
```

Padrão Command: Além de permitir parametrizar diferentes classes e realizar comandos, o padrão dá ao código maior flexibilidade, extensibilidade e a simplificação de transações complexas. Usei-o para a checagem das condições de cadastro como a existência e tamanho da senha.

(src>js>Factory.js)

```
//Command

class Checagem{
    ConfirmarDados(){
        console.log("Checando dados");
    }
}

class ChecagemUsuario extends Checagem{
    constructor(Usuario){
        super();
        this.Usuario = Usuario
    }
    ConfirmarDados(){
        if (this.Usuario.getSenha() !== this.Usuario.getConfirmaSenha()) {
            document.getElementById("mensagem").innerHTML = "As senhas não coincidem.";
            return;
        }
    }
}
```

```
    ConfirmarDados(){
        if (this.cliente.getNome().length < 3 || !/^[a-zA-ZÀ-ú\s]+$/.test(this.cliente.getNome())) {···
        } else {···
        }
        if (!this.cliente.getCep().match(/^\d{5}-?\d{3}$/)) {···
        } else {···
        }
        if (!this.cliente.getTelefone().match(/^\d{10,11}$/)) {···
        } else {···
        }
        if (this.cliente.getRamo().length === 0 || /^\s*$/.test(this.cliente.getRamo())) {···
        } else {···
        }
        if (!this.cliente.getEmail().match(/^[\w-]+(\.[\w-]+)*@([\w-]+\.)+[a-zA-Z]{2,7}$/)) {···
        } else {···
        }
        if (this.cliente.getSenha() < 8 || !this.cliente.getSenha().match(/[a-z]/) || !this.cliente.getSenha().match(/[A-Z]/) || !this.cliente.ge
        } else {···
        }
    }
}
```

```
class Controlador{
    constructor(Checagem){
        this.Checagem = Checagem;
    }

    Checar(){
        this.Checagem.ConfirmarDados();
    }
}

function gerarControlador(tipo,classe){
    switch(tipo){
        case "cliente":
            const ControladorC = new Controlador(new ChecagemCliente(classe));
            ControladorC.Checar();
            console.log("Funcionou Cliente");
        break;
        case "usuario":
            const ControladorU = new Controlador(new ChecagemUsuario(classe));
            ControladorU.Checar();
            console.log("Funcionou Usuario");
        break;
    }
}
```