1. Estructura del Proyecto

```
calculator-app/
├── client/                # Frontend
│   ├── src/
│   │   ├── components/    # Componentes React
│   │   ├── hooks/         # Custom hooks
│   │   └── services/      # Llamadas API
├── server/                # Backend
│   ├── controllers/       # Lógica de negocio
│   ├── routes/            # Endpoints API
│   └── utils/             # Helpers
└── shared/                # Código compartido
    └── types/             # Tipos TypeScript
```

2. Código Clave

```
// components/Calculator.tsx
import { useCalculator } from '../hooks/useCalculator';

export const Calculator = () => {
  const { input, result, handleInput, calculate } = useCalculator();

  return (
    <div className="calculator">
      <Display value={input || result} />
      <Keypad onPress={handleInput} onCalculate={calculate} />
      <HistoryPanel />
```

```
    </div>
  );
};

// hooks/useCalculator.ts
export const useCalculator = () => {
  const [input, setInput] = useState('');
  const [result, setResult] = useState('');

  const handleInput = (key: string) => {
    setInput(prev => prev + key);
  };

  const calculate = async () => {
    try {
      const res = await apiService.calculate(input);
      setResult(res.toString());
    } catch (error) {
      alert('Error en cálculo');
    }
  };

  return { input, result, handleInput, calculate };
};
```

Backend (Node.js/Express)

```
// server/controllers/calculator.controller.ts
```

```
import math from 'mathjs';
import { Request, Response } from 'express';

export const calculate = async (req: Request, res: Response) => {
  const { expression } = req.body;

  try {
    // Validación de entrada
    if (!/^[0-9+\-*/().^ ]+$/.test(expression)) {
      throw new Error('Expresión inválida');
    }

    const result = math.evaluate(expression);

    // Guardar en Firestore
    await firestore.collection('calculations').add({
      expression,
      result,
      timestamp: new Date()
    });

    res.json({ success: true, result });
  } catch (error) {
    res.status(400).json({
      success: false,
      error: error.message
```

```
    });
  }
};
```

Servicios Compartidos

```
// shared/services/api.service.ts
import axios from 'axios';

const API_URL = process.env.REACT_APP_API_URL;

export const apiService = {
  calculate: async (expression: string): Promise<number> => {
    const response = await axios.post(`${API_URL}/calculate`, { expression });
    return response.data.result;
  },
  getHistory: async (userId: string) => {
    // Implementación Firebase
  }
};
```

Configuración Firebase

```
// shared/config/firebase.ts
import { initializeApp } from 'firebase/app';
import { getFirestore } from 'firebase/firestore';

const firebaseConfig = {
  apiKey: process.env.FIREBASE_API_KEY,
  projectId: 'calculator-app-123'
```

```
};

const app = initializeApp(firebaseConfig);
export const firestore = getFirestore(app);
```

3. Patrones Implementados

Factory Pattern (Operaciones)

```
// server/utils/operation.factory.ts
interface IOperation {
  calculate(a: number, b?: number): number;
}

class SumOperation implements IOperation {
  calculate(a: number, b: number): number {
    return a + b;
  }
}

export class OperationFactory {
  static createOperation(type: string): IOperation {
    switch(type) {
      case 'sum': return new SumOperation();
      // Otras operaciones...
      default: throw new Error('Operación no soportada');
    }
  }
```

```
}
```

Singleton (Historial)

```
// server/services/history.service.ts
export class HistoryService {
  private static instance: HistoryService;

  private constructor() {}

  public static getInstance(): HistoryService {
    if (!HistoryService.instance) {
      HistoryService.instance = new HistoryService();
    }
    return HistoryService.instance;
  }

  public async save(operation: string) {
    // Implementación Firestore
  }
}
```

4. Seguridad

```
// server/middleware/auth.ts
import { Request, Response, NextFunction } from 'express';
import jwt from 'jsonwebtoken';

export const authenticate = (
```

```
  req: Request,
  res: Response,
  next: NextFunction
) => {
  const token = req.headers.authorization?.split(' ')[1];

  if (!token) {
    return res.status(401).json({ error: 'Acceso no autorizado' });
  }

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET!);
    req.user = decoded;
    next();
  } catch (error) {
    res.status(403).json({ error: 'Token inválido' });
  }
};
```

Validación de Entrada

```
// server/utils/validator.ts
export const validateInput = (input: string): boolean => {
  const safePattern = /^[0-9+\-*/().^ ]+$/;
  return safePattern.test(input) && input.length <= 100;
};
```

5. Despliegue

```
FROM node:18

WORKDIR /app

COPY package*.json ./

RUN npm ci

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

Ejemplo .env

```
# Frontend

REACT_APP_API_URL=http://localhost:3000


# Backend

JWT_SECRET=tu_super_secreto

FIREBASE_API_KEY=tu_key_de_firebase
```

6. Pruebas Unitarias

```
// server/__tests__/calculator.test.ts

describe('Calculator Controller', () => {

  it('should sum two numbers correctly', async () => {

    const mockReq = { body: { expression: '2+2' } };

    const mockRes = { json: jest.fn() };


    await calculate(mockReq, mockRes);
```

```
    expect(mockRes.json).toHaveBeenCalledWith(
      expect.objectContaining({
        success: true,
        result: 4
      })
    );
  });
});
```