

cZr Catering - Arquitectura Técnica y Modelo de Datos (SDD)

ID del Proyecto: LCS-2025

Versión: 1.0.0

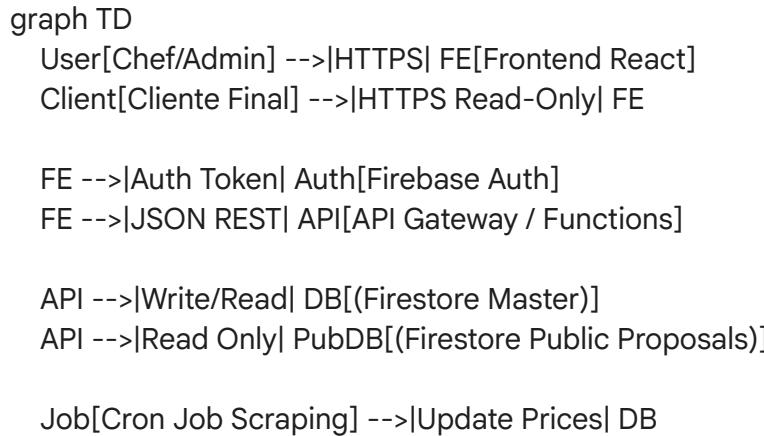
Tipo: Diseño de Sistemas

1. Arquitectura de Software

1.1 Stack Tecnológico

- **Frontend:** React 18 (Vite) + TailwindCSS.
- **Backend / API:** Node.js (Express) corriendo en Cloud Functions (Serverless) para escalabilidad automática.
- **Base de Datos:** Firestore (NoSQL). Elegido por la naturaleza jerárquica y flexible de las recetas y documentos JSON.
- **Autenticación:** Firebase Auth (Manejo de sesiones y tokens).
- **Storage:** Cloud Storage (Imágenes de platos y avatares).

1.2 Diagrama de Flujo de Datos (Data Flow)



2. Modelo de Datos (Schema Definition)

A continuación se detalla la estructura de colecciones de Firestore. Se utiliza NoSQL, por lo que la desnormalización es intencional para optimizar lecturas.

Colección: ingredients

Documento maestro de materia prima.

{

```

"_id": "string (UUID)",
"name": "string (Indexed)",
"slug": "string (unique)",
"unit": "string (kg|lt|uni)",
"category": "string",
"current_cost": {
  "value": "number (float)",
  "currency": "ARS",
  "last_updated": "timestamp",
  "provider_id": "string (ref)"
},
"yield_factors": {
  "cleaning": "number (0-1)", // Merma limpieza
  "cooking": "number (0-1)" // Merma cocción
},
"nutrition": {
  "calories_100g": "number",
  "sodium_mg": "number",
  "fat_g": "number"
},
"tags": ["array<string>"], // Ej: "Estacional:Verano"
"is_active": "boolean"
}

```

Colección: recipes

Documento de ingeniería de menú.

```

{
  "_id": "string (UUID)",
  "name": "string",
  "status": "string (DRAFT|ACTIVE|ARCHIVED)",
  "version": "number",
  "components": [
    {
      "type": "INGREDIENT | SUB_RECIPE",
      "ref_id": "string",
      "name_snapshot": "string", // Para visualización rápida sin join
      "qty": "number",
      "unit_cost_snapshot": "number" // Costo al momento de agregar (referencial)
    }
  ],
}
```

```

"financials": {
    "total_cost": "number",
    "target_margin_percent": "number",
    "suggested_price": "number",
    "final_price": "number"
},
"created_at": "timestamp",
"updated_at": "timestamp"
}

```

Colección: proposals (Pública)

CRÍTICO: Esta colección está separada lógicamente. Contiene datos estáticos.

```

{
    "_id": "string (UUID Public)",
    "client_name": "string",
    "expiration_date": "timestamp",
    "items": [
        {
            "recipe_name": "string",
            "description": "string",
            "image_url": "string",
            "price": "number",
            "nutrition_badges": ["LOW_SODIUM", "VEGAN"]
        }
    ],
    "total_amount": "number",
    "view_count": "number" // Analítica básica
}

```

3. Estrategia de Índices (Firestore)

Para garantizar el requisito no funcional de latencia (<200ms):

1. **Composite Index:** ingredients -> name (ASC) + category (ASC).
2. **Composite Index:** recipes -> status (EQ) + updated_at (DESC).
3. **Single Field Index:** ingredients -> tags (Array-contains).

4. Seguridad (Firestore Rules)

Pseudo-código de reglas de seguridad:

```
match /ingredients/{docId} {  
    allow read: if request.auth.token.role in ['ADMIN', 'CHEF', 'COCINA'];  
    allow write: if request.auth.token.role in ['ADMIN', 'CHEF'];  
}  
  
match /recipes/{docId} {  
    allow read: if request.auth.token.role in ['ADMIN', 'CHEF', 'COCINA'];  
    allow write: if request.auth.token.role in ['ADMIN', 'CHEF'];  
}  
  
match /proposals/{docId} {  
    // Lectura pública permitida para el cliente  
    allow read: if true;  
    // Escritura solo sistema  
    allow write: if request.auth.token.role in ['ADMIN', 'CHEF'];  
}
```