NO MANDES CODIGO, necesito que analices bien el FLUJO, si el project de SpringBoot se genera en el front unicamente y asi... REVISA BIEN por favor todo lo que hemos hablado para el primer parcial para tomar una decisión... Te paso tanto el README del backend como del frontend.

La idea es tener TODO LIST, no que me pases el codigo literalmente.

Perfecto, ahora con el análisis **COMPLETO** del flujo real del Parcial 1, aquí está la **TODO LIST CORREGIDA** sin código, solo arquitectura y decisiones:

TODO LIST - SEGUNDO PARCIAL (SIN CÓDIGO)

Basado en Arquitectura Real del Parcial 1

ANÁLISIS CRÍTICO DEL PARCIAL 1 (FLUJO REAL)

✅ LO QUE YA EXISTE Y FUNCIONA:

Backend Django (Puerto 8000 HTTP + 8001 WebSocket):

- ✅ Modelo único `UMLDiagram` con `content` JSONField
- ✅ Sesiones anónimas (NO usuarios registrados)
- ✅ WebSocket real-time con Django Channels + Redis
- ✅ API REST completa (CRUD diagramas)
- ✅ OpenAI GPT-4 integrado (multilingüe: EN/ES/FR/DE)
- ✅ Rate limiting: 30 requests/hora (AI), 200/hora (API)

Frontend React (Cliente):

- ✅ **Generación de Spring Boot 100% CLIENT-SIDE** (`simpleCodeGenerator.ts`)
- ✅ React Flow para diagramas UML
- ✅ Redux Toolkit para estado global
- ✅ WebSocket hook (`useWebSocket`) con echo prevention

- ✅ localStorage fallback (offline-first)
- ✅ AI Assistant protegido por contraseña (sessionStorage 24h)

🔴 DECISIÓN ARQUITECTÓNICA CRÍTICA:

❌ NO cambiar el flujo de generación de código existente

- Spring Boot se genera en **FRONTEND (TypeScript)**, NO en backend
- Flutter se generará en **FRONTEND (TypeScript)**, NO en backend
- Backend solo almacena diagramas y proporciona colaboración

✅ Mantener arquitectura client-side:

- ✓ Reducir carga del servidor
- ✓ Escalabilidad horizontal
- ✓ Procesamiento distribuido en clientes
- ✓ Funciona offline
- ✓ Menor latencia

📋 TODO LIST COMPLETA - BACKEND (Django)

1. MODELO DE DATOS

1.1. NUEVO MODELO: `FlutterProject`

- [ ] Crear modelo en `apps/flutter_generator/models/flutter_project.py`
- [ ] Campos:
  - `id` (UUID primary key)
  - `diagram_id` (UUID, referencia a `UMLDiagram`)
  - `session_id` (CharField, sesión anónima)
  - `project_name` (CharField)
  - `package_name` (CharField, ej: "com.erp.inventory")
  - `config` (JSONField: theme, colors, navigation_type, state_management)
  - `metadata` (JSONField: descripción, versión, autor)
  - `created_at`, `last_generated` (DateTimeField)
- [ ] Índices en `diagram_id` y `session_id`
- [ ] Migración de base de datos

1.2. NO MODIFICAR: Modelo `UMLDiagram` existente

- ✅ Ya funciona correctamente con `content` JSONField
- ✅ Soporta 7 tipos de diagramas (CLASS, SEQUENCE, USE_CASE, etc.)
- ✅ NO requiere cambios

2. API ENDPOINTS (Django REST Framework)

2.1. Gestión de Proyectos Flutter

- [ ] **POST** `/api/flutter-projects/` - Crear proyecto Flutter
  - Input: `diagram_id`, `project_name`, `package_name`, `config`
  - Output: UUID del proyecto creado
- [ ] **GET** `/api/flutter-projects/` - Listar proyectos por `session_id`
  - Filtrado automático por sesión anónima
- [ ] **GET** `/api/flutter-projects/{id}/` - Obtener proyecto específico
- [ ] **PATCH** `/api/flutter-projects/{id}/` - Actualizar configuración
- [ ] **DELETE** `/api/flutter-projects/{id}/` - Eliminar proyecto

2.2. Procesamiento de Imágenes (GPT-4 Vision)

- [ ] **POST** `/api/diagrams/from-image/` - Crear diagrama desde imagen
  - Input: `image` (base64), `session_id`, `processing_mode`
  - Output: `nodes` y `edges` compatibles con React Flow
- [ ] **POST** `/api/diagrams/{id}/update-from-image/` - Merge imagen con diagrama existente
  - Input: `image` (base64), `merge_strategy` (replace/append/smart_merge)
  - Output: `new_nodes`, `new_edges`, `conflicts` (si existen)

2.3. Comandos IA Incrementales (Mejora)

- [ ] **POST** `/api/ai-assistant/incremental-command/` - Comandos que modifican diagrama existente
  - Input: `command`, `diagram_id`, `current_diagram` (nodes/edges)
  - Output: **DELTA (cambios incrementales)**, NO diagrama completo
  - Ejemplos de comandos:
    - "Add attribute `email` (String) to class `User`"
    - "Remove method `delete()` from class `Product`"
    - "Change visibility of `id` in class `Order` to private"
    - "Add composition relationship from `Order` to `OrderItem` with multiplicity 1..*"

3. SERVICIOS BACKEND (Python)

3.1. ImageProcessorService

- [ ] Crear `apps/ai_assistant/services/image_processor_service.py`
- [ ] Funcionalidad:
  - Procesar imagen con GPT-4 Vision API
  - Extraer clases con atributos (nombre, tipo, visibilidad)
  - Extraer métodos con parámetros y tipo de retorno
  - Extraer relaciones (Association, Aggregation, Composition, Dependency)
  - Extraer multiplicidades
  - Convertir a formato React Flow (nodes/edges)
- [ ] Manejo de errores:
  - Imagen no válida (formato, tamaño)
  - API timeout (GPT-4 Vision)
  - Contenido no reconocible como UML

3.2. IncrementalCommandProcessor (Mejora)

- [ ] Crear `apps/ai_assistant/services/incremental_command_processor.py`
- [ ] Patrones regex para comandos incrementales:
  - `add_attribute`: "add attribute X (Type) to class Y"
  - `remove_attribute`: "remove attribute X from class Y"
  - `modify_attribute`: "change attribute X in class Y to Z (Type)"
  - `add_method`: "add method X(params) returning Type to class Y"
  - `remove_method`: "remove method X from class Y"
  - `add_relationship`: "add Association from User to Order with multiplicity 1..*"
  - `remove_relationship`: "remove relationship between X and Y"
  - `rename_class`: "rename class X to Y"
  - `change_visibility`: "change visibility of X in class Y to private"
- [ ] Retornar **DELTA (cambios)**, NO diagrama completo:

```
{
  "action": "update_node",
  "node_id": "node_1",
  "changes": {
    "data.attributes": {
      "operation": "append",
      "value": {"name": "email", "type": "String", "visibility": "private"}
    }
```

```
  }
}
```

- [ ] Fallback a GPT-4 si comando no coincide con patrones

3.3. ImageMergeService

- [ ] Crear `apps/ai_assistant/services/image_merge_service.py`
- [ ] Funcionalidad:
  - Comparar elementos nuevos vs existentes
  - Detectar duplicados (por nombre de clase)
  - Estrategias de merge:
    - `replace`: Sobrescribir elementos duplicados
    - `append`: Añadir todos sin verificar duplicados
    - `smart_merge`: Detectar duplicados + ofrecer resolución
  - Calcular posiciones para nuevos elementos (evitar superposición)

4. WEBSOCKET (Django Channels) - NO MODIFICAR

- ✅ Ya funciona correctamente con echo prevention
- ✅ Reconexión automática con exponential backoff
- ✅ Redis channel layer
- ✅ NO requiere cambios para Flutter generation

5. DOCUMENTACIÓN API (drf-spectacular)

- [ ] Actualizar esquema OpenAPI con nuevos endpoints
- [ ] Documentar request/response schemas para:
  - FlutterProject CRUD
  - Image processing
  - Incremental commands
- [ ] Ejemplos de uso en Swagger UI

TODO LIST COMPLETA - FRONTEND (React + TypeScript)

6. SERVICIOS CLIENT-SIDE (TypeScript)

6.1. FlutterGeneratorService (NUEVO)

- [ ] Crear `src/services/flutterGeneratorService.ts`
- [ ] **Genera código Flutter 100% en cliente (navegador)**
- [ ] Funcionalidad:
  - Input: `nodes`, `edges` (desde Redux), `config` (theme, package, etc.)
  - Output: Objeto `{filename: code_content}` con proyecto completo
- [ ] Archivos a generar:
  - **Modelos Dart**: `lib/models/*.dart` (con JSON serialization)
  - **Providers**: `lib/providers/*_provider.dart` (Provider pattern)
  - **Formularios CRUD**: `lib/screens/*_form.dart` (Create/Update)
  - **Listados**: `lib/screens/*_list.dart` (Read con paginación)
  - **Servicios API**: `lib/services/api_service.dart` (HTTP client)
  - **Main**: `lib/main.dart` (MaterialApp + routing)
  - **Configuración**: `pubspec.yaml` (dependencias)
  - **README**: `README.md` (documentación)
- [ ] Templates Dart:
  - Mapeo de tipos UML a Dart (String → String, Integer → int, etc.)
  - Generación de clases Dart con `@JsonSerializable`
  - Generación de formularios con validaciones (Flutter Form + TextFormField)
  - Generación de listados con ListView.builder
  - Generación de navegación (Drawer / BottomNavigationBar / Tabs)
- [ ] Mapeo de relaciones UML a Flutter:
  - Association → Campo de tipo objeto
  - Aggregation → Lista de objetos (`List<T>`)
  - Composition → Lista de objetos con cascade delete
  - Dependency → Importación de dependencia

6.2. ImageUploadService (NUEVO)

- [ ] Crear `src/services/imageUploadService.ts`
- [ ] Funcionalidad:
  - Convertir `File` a base64
  - Enviar a backend para procesamiento
  - Manejar progress (0-100%)

- Timeout de 60 segundos (GPT-4 Vision puede tardar)

6.3. Modificar `simpleCodeGenerator.ts` (Existente)

- [ ] **NO cambiar arquitectura client-side**
- [ ] Posible refactor para reutilizar lógica común con Flutter generator
- [ ] Mantener generación de Spring Boot en cliente

7. COMPONENTES UI (React)

7.1. ImageUploader Component (NUEVO)

- [ ] Crear `src/components/ImageUploader.tsx`
- [ ] Funcionalidad:
  - Drag & drop de imágenes (react-dropzone)
  - Preview de imagen antes de procesar
  - Progress bar durante procesamiento GPT-4 Vision
  - Indicador de elementos extraídos (X clases, Y relaciones)
  - Botón "Merge with existing" vs "Create new"
- [ ] Estados:
  - `idle` (esperando imagen)
  - `uploading` (enviando a backend)
  - `processing` (GPT-4 Vision analizando)
  - `success` (elementos extraídos)
  - `error` (fallo en procesamiento)

7.2. FlutterConfigModal (NUEVO)

- [ ] Crear `src/components/modals/FlutterConfigModal.tsx`
- [ ] Formulario de configuración:
  - Project Name (TextField)
  - Package Name (TextField con validación: "com.example.app")
  - Theme (Select: Material 3 / Cupertino)
  - Primary Color (ColorPicker)
  - Navigation Type (Select: Drawer / Bottom Nav / Tabs)
  - State Management (Select: Provider / Riverpod / Bloc)
  - Enable Dark Mode (Switch)
- [ ] Preview en tiempo real de colores/theme

- [ ] Botón "Generate Flutter Project"

7.3. Modificar AIAssistant (Existente)

- [ ] **Añadir nuevo TAB: "Flutter"** (4to tab)
  - Tab 1: Chat (existente)
  - Tab 2: Commands (existente)
  - Tab 3: Spring Boot (existente)
  - Tab 4: Flutter (NUEVO)
- [ ] Contenido del tab Flutter:
  - Formulario de configuración FlutterConfig
  - Preview de archivos generados (tree view)
  - Botón de descarga ZIP
- [ ] NO modificar tabs existentes

7.4. ImageMergeConflictModal (NUEVO)

- [ ] Crear `src/components/modals/ImageMergeConflictModal.tsx`
- [ ] Mostrar cuando hay duplicados al merge imagen
- [ ] Opciones por conflicto:
  - Keep existing (mantener clase del diagrama)
  - Replace with new (sobrescribir con imagen)
  - Merge attributes (combinar atributos/métodos)
- [ ] Vista previa lado a lado (existing vs new)

8. REDUX STATE (Redux Toolkit)

8.1. Nuevo Slice: flutterSlice

- [ ] Crear `src/store/slices/flutterSlice.ts`
- [ ] Estado:

```
{
  projects: FlutterProject[],
  currentProject: FlutterProject | null,
  config: FlutterProjectConfig,
  generatedFiles: Record<string, string>,
  loading: boolean,
  error: string | null
}
```

- [ ] Actions:

- `setConfig(config)`
- `generateProject(nodes, edges, config)`
- `saveProjectToBackend(project)`
- `loadProjects()`

8.2. Modificar diagramSlice (Existente)

- [ ] Añadir action `mergeNodesFromImage(newNodes, mergeStrategy)`
- [ ] Añadir action `mergeEdgesFromImage(newEdges, mergeStrategy)`
- [ ] Lógica de detección de duplicados por `data.label`

9. INTEGRACIÓN DE 3 MODOS DE ENTRADA

9.1. Modo 1: Mouse (Ya existe)

- ✅ React Flow con drag & drop
- ✅ Doble click para editar
- ✅ Handles para relaciones
- ✅ NO requiere cambios

9.2. Modo 2: Prompts/Comandos (Mejorar)

- [ ] Modificar tab "Commands" del AI Assistant
- [ ] Soportar comandos **INCREMENTALES**:
  - Añadir elemento a diagrama existente
  - Modificar elemento específico
  - Eliminar elemento
  - **NO generar diagrama completo desde cero**
- [ ] Feedback visual:
  - Resaltar elemento modificado (highlight animation)
  - Mostrar diff (antes/después)
  - Undo/Redo de comandos

9.3. Modo 3: Imagen (NUEVO)

- [ ] Añadir botón "Upload Image" en toolbar principal
- [ ] Modal con ImageUploader component
- [ ] Opciones:
  - **Create new diagram from image** (diagrama vacío → imagen)

    - **Merge with existing diagram** (diagrama actual + imagen)
- [ ] Preview de elementos extraídos antes de confirmar
- [ ] Resolución de conflictos si hay duplicados

10. TOOLBAR Y UI UPDATES

10.1. Toolbar Principal

- [ ] Añadir botón **"Upload Image"** (icono: Image/Upload)
    - Tooltip: "Create diagram from image (Ctrl+I)"
    - Atajo de teclado: `Ctrl+I`
- [ ] Añadir dropdown **"Generate Code"**:
    - Opción 1: Spring Boot (existente)
    - Opción 2: Flutter (NUEVO)
- [ ] Indicador de "Processing Image..." cuando GPT-4 Vision trabaja

10.2. Keyboard Shortcuts (Nuevos)

- [ ] `Ctrl+I` - Abrir modal de carga de imagen
- [ ] `Ctrl+Shift+F` - Abrir tab Flutter en AI Assistant
- [ ] `Ctrl+Shift+I` - Comandos incrementales (focus en input)

11. TESTING Y VALIDACIÓN

11.1. Backend Tests

- [ ] Test ImageProcessorService con imágenes de ejemplo
- [ ] Test IncrementalCommandProcessor con 20 comandos diferentes
- [ ] Test ImageMergeService con conflictos
- [ ] Test API endpoints (FlutterProject CRUD)

11.2. Frontend Tests

- [ ] Test FlutterGeneratorService con diagrama de 5 clases
- [ ] Test ImageUploader component (drag & drop)
- [ ] Test merge strategies (replace/append/smart)
- [ ] Test descarga de ZIP Flutter

11.3. End-to-End Tests

- [ ] Flujo completo: Imagen → Diagrama → Flutter → Download
- [ ] Flujo comandos incrementales: "Add class" → "Add attribute" → "Add relationship"
- [ ] Flujo colaborativo: Usuario A sube imagen → Usuario B ve cambios

12. DOCUMENTACIÓN (PUDS + C4)

12.1. Capítulo: Justificación del Stack Tecnológico

- [ ] **Por qué Flutter para ERP:**
  - Cross-platform (Android/iOS/Web/Desktop)
  - Hot reload para desarrollo rápido
  - Performance nativo
  - Widgets Material Design para software de gestión
  - Provider/Riverpod para state management complejo
- [ ] **Por qué TypeScript client-side para generación:**
  - Reduce carga del servidor (escalabilidad)
  - Funciona offline (localStorage fallback)
  - Menor latencia (sin roundtrip al servidor)
  - Procesamiento distribuido en clientes
- [ ] **Por qué GPT-4 Vision para imágenes:**
  - Extracción precisa de elementos UML
  - Reconocimiento de relaciones complejas
  - Soporte multilingüe (diagramas en cualquier idioma)

12.2. Diagramas C4

- [ ] **Nivel 1 - Context Diagram:**
  - Usuario → Sistema UML → Backend Django
  - Sistema UML → OpenAI GPT-4 (IA)
  - Sistema UML → Spring Boot Project (output)
  - Sistema UML → Flutter Project (output)
- [ ] **Nivel 2 - Container Diagram:**
  - React Frontend (client-side generation)
  - Django Backend (HTTP API + WebSocket)
  - PostgreSQL (diagramas)
  - Redis (sessions + WebSocket channels)

    - OpenAI API (GPT-4 + GPT-4 Vision)
- [ ] **Nivel 3 - Component Diagram (Frontend):**
    - FlutterGeneratorService (genera Dart)
    - SpringBootGeneratorService (genera Java)
    - ImageUploadService (procesa imágenes)
    - AIAssistantService (comandos NLP)
    - WebSocketService (colaboración)
- [ ] **Nivel 4 - Code Diagram:**
    - Template Dart para modelos
    - Template Dart para formularios CRUD
    - Lógica de mapeo UML → JPA (Spring Boot)
    - Lógica de mapeo UML → Flutter widgets

12.3. Casos de Uso (Nuevos)

- [ ] **CU-5: Generar Código Flutter desde Diagrama**
    - Precondición: Diagrama UML completado
    - Proceso: Configurar opciones → Generar → Descargar ZIP
    - Postcondición: Proyecto Flutter funcional
- [ ] **CU-6: Crear Diagrama desde Imagen**
    - Precondición: Imagen de diagrama UML
    - Proceso: Upload → GPT-4 Vision procesa → Preview → Confirmar
    - Postcondición: Diagrama UML en editor
- [ ] **CU-7: Modificar Diagrama con Comandos Incrementales**
    - Precondición: Diagrama existente
    - Proceso: Comando NLP → IA interpreta → Aplicar cambio
    - Postcondición: Diagrama actualizado con elemento específico modificado

12.4. Diagramas UML 2.5 (PlantUML)

- [ ] Diagrama de secuencia: Upload Image → GPT-4 Vision → Create Diagram
- [ ] Diagrama de secuencia: Incremental Command → Backend → Frontend Update
- [ ] Diagrama de secuencia: Generate Flutter → Client-side → Download ZIP
- [ ] Diagrama de actividad: 3 modos de entrada (Mouse/Prompt/Image)

PRIORIZACIÓN DE TAREAS (Fecha límite: 12 Nov)

SPRINT 1 (Alta prioridad - Semana 1):

1. Backend: ImageProcessorService (GPT-4 Vision)
2. Backend: Endpoint `/api/diagrams/from-image/`
3. Frontend: ImageUploader component
4. Frontend: Integración GPT-4 Vision en UI
5. Testing: Flujo imagen → diagrama

SPRINT 2 (Alta prioridad - Semana 1):

6. Frontend: FlutterGeneratorService (templates Dart)
7. Frontend: FlutterConfigModal
8. Frontend: Tab Flutter en AI Assistant
9. Frontend: Descarga ZIP Flutter
10. Testing: Flujo diagrama → Flutter

SPRINT 3 (Media prioridad - Semana 2):

11. Backend: IncrementalCommandProcessor (mejora comandos)
12. Backend: Endpoint `/api/ai-assistant/incremental-command/`
13. Frontend: Mejora de comandos incrementales
14. Frontend: Feedback visual (highlight, diff)
15. Testing: Comandos incrementales

SPRINT 4 (Media prioridad - Semana 2):

16. Backend: ImageMergeService (merge inteligente)
17. Backend: Endpoint `/api/diagrams/{id}/update-from-image/`
18. Frontend: ImageMergeConflictModal
19. Frontend: Estrategias de merge (replace/append/smart)
20. Testing: Merge con conflictos

SPRINT 5 (Documentación - Semana 2):

21. Capítulo: Justificación del Stack
22. Diagramas C4 (4 niveles)
23. Casos de uso CU-5, CU-6, CU-7
24. Diagramas PlantUML (secuencia, actividad)
25. Manual de usuario (3 modos de entrada)

✅ VERIFICACIÓN DE REQUISITOS DEL PARCIAL 2

- [x] **Del 100% del código, generar al menos 90%:**
  - ✅ Flutter: Genera modelos, providers, forms, lists, main, pubspec (100%)
  - ✅ Spring Boot: Ya genera entities, repos, services, controllers (100%)
- [x] **3 formas de entrada:**
  - ✅ Mouse (React Flow - ya existe)
  - ✅ Prompts (AI Assistant comandos incrementales)
  - ✅ Imagen (GPT-4 Vision)
- [x] **Diseño para software de gestión (ERP):**
  - ✅ Cada clase = Formulario CRUD
  - ✅ Listados con paginación
  - ✅ Navegación con Drawer/BottomNav
- [x] **Comandos incrementales (NO generar diagrama completo):**
  - ✅ IncrementalCommandProcessor con patrones regex
  - ✅ Retorna DELTA (cambios), NO diagrama completo
  - ✅ "Add attribute X to class Y" modifica clase específica
- [x] **Documentación con C4 + UML:**
  - ✅ Diagramas C4 (Context, Container, Component, Code)
  - ✅ Casos de uso con PlantUML
  - ✅ Justificación del stack tecnológico

🔒 DECISIONES ARQUITECTÓNICAS CRÍTICAS (NO NEGOCIABLES)

1. ✅ **Generación de código en FRONTEND (client-side)**
   - Spring Boot: TypeScript (ya existe en `simpleCodeGenerator.ts`)
   - Flutter: TypeScript (NUEVO en `flutterGeneratorService.ts`)
   - Backend solo almacena y colabora, NO genera código
2. ✅ **GPT-4 Vision solo para PROCESAMIENTO de imágenes**
   - Backend recibe imagen → Envía a OpenAI → Retorna nodes/edges
   - NO genera código, solo extrae elementos UML
3. ✅ **Comandos IA INCREMENTALES, no completos**
   - "Add attribute X to class Y" → Retorna DELTA
   - NO "Create entire system with User, Order, Product..."
4. ✅ **Mantener sesiones anónimas (NO agregar usuarios)**

- Sistema funciona sin registro
- Django sessions con `session_id` UUID

5. ✅ **WebSocket sin cambios (ya funciona perfectamente)**
   - Echo prevention
   - Reconnection automática
   - Redis channel layer

¿Necesitas que profundice en algún sprint específico o que defina criterios de aceptación para alguna tarea?

⁂

1. README.md
2. README.md