🎯 Prompt TestSprite per Analisi Completa App "Calendario Vendite"

📱 ANALISI DELL'APPLICAZIONE ANDROID

TestSprite, analizza completamente la mia applicazione Android "Calendario Vendite" per refactoring, ottimizzazione e testing automatico. L'app ha questa logica specifica:

📐 ARCHITETTURA E LOGICA BUSINESS

📊 Struttura Dati Centralizzata

- **CLIENTE = ENTITÀ CENTRALE**: Tutto ruota attorno al cliente selezionato
- **Firebase Backend**: Tutti i dati (celle, tag, note, foto) persistono su Firebase
- **Filtri Progressivi**: Sistema di filtri che include/esclude clienti e agenti
- **Multi-utente**: Diversi utenti possono visualizzare/modificare dati dello stesso cliente
- **Permessi**: Utenti vedono solo clienti di competenza, admin vede tutto

🔄 Flusso Operativo

1. **Selezione Filtri** → Carica dati cliente specifico da Firebase
2. **Click su Cella** → Apre form per inserimento dati
3. **Inserimento**: Ordinati (V), Venduti (S), Tag Persone, Tag Azioni
4. **Calcolo Progressivo Live** → Ricalcola automaticamente based on chronological order
5. **Salvataggio Firebase** → Persist immediato di modifiche
6. **Visualizzazione Condizionale** → Mostra dati solo se presenti

⚡ Logica Calcolo Progressivo CRITICA

- **Base Cronologica**: Il calcolo inizia sempre dalla data più antica con dati
- **Esempio**:
  - 3 Gen: +100 ord, +20 vend → Base: V:20, S:100
  - 5 Gen: +100 ord, +20 vend → Progressivo: V:40, S:160
  - 8 Gen: +100 ord, +20 vend → Progressivo: V:60, S:200
- **Modifica Retroattiva**: Se modifico il 3 Gen, TUTTO il progressivo si ricalcola
- **Real-time Updates**: Ogni modifica/cancellazione innesca ricalcolo completo

---

🔍 AREE DI ANALISI PRIORITARIE

1. REFACTORING ARCHITETTURALE

- **Firebase Operations**: Identifica operazioni ridondanti/inefficienti

- **State Management**: Ottimizza gestione stato cliente selezionato
- **Filter Logic**: Consolida logica filtri duplicata
- **Calculation Engine**: Ottimizza algoritmo calcolo progressivo per performance

2. PERFORMANCE OPTIMIZATION

- **Firebase Queries**: Ottimizza query per caricamento dati cliente
- **Real-time Calculations**: Identifica bottlenecks nel ricalcolo progressivo
- **UI Responsiveness**: Analizza lag durante inserimento/modifica dati
- **Memory Management**: Gestione memoria per dati multiple clienti

3. CODE QUALITY & PATTERNS

- **DRY Principle**: Elimina duplicazioni nella logica filtri/calcoli
- **SOLID Principles**: Migliora separazione responsabilità
- **Observer Pattern**: Ottimizza propagazione cambiamenti calcolo progressivo
- **Repository Pattern**: Standardizza accesso Firebase

4. DATA CONSISTENCY & VALIDATION

- **Firebase Sync**: Garantisci consistenza dati multi-utente
- **Calculation Accuracy**: Valida correttezza calcoli progressivi
- **Error Handling**: Gestione errori Firebase/connessione
- **Data Integrity**: Validazione inserimenti (V/S values, date logic)

🚀 TESTING STRATEGY

📱 UI Testing (su dispositivo Samsung connesso)

- **Filter Workflows**: Test selezione clienti/agenti
- **Cell Interaction**: Test apertura form e inserimento dati
- **Progressive Calculation Display**: Verifica visualizzazione calcoli live
- **Multi-scenario Navigation**: Test cambio cliente e persistenza dati

🔧 Unit Testing

- **Calculation Logic**: Test algoritmo calcolo progressivo con vari scenari
- **Firebase Operations**: Mock Firebase per test CRUD operations
- **Filter Logic**: Test inclusioni/esclusioni filtri
- **Date Handling**: Test ordinamento chronological per calcoli

🔄 Integration Testing

- **Firebase Sync**: Test sincronizzazione dati real-time
- **Client-Server Flow**: Test completo inserimento → Firebase → ricalcolo
- **Multi-user Scenarios**: Simula accessi concurrent (anche se teoricamente non dovrebbero accadere)

- **Offline/Online Transitions**: Test behavior con/senza connessione

⚡ Performance Testing

- **Large Dataset**: Test con molti clienti e dati storici
- **Calculation Speed**: Benchmark velocità ricalcolo progressivo
- **Firebase Response Times**: Test latenza operazioni database
- **Memory Usage**: Monitor consumo memoria durante uso prolungato

---

📋 DELIVERABLES RICHIESTI

1. **Analisi Architetturale Completa** con problemi identificati
2. **Piano Refactoring Prioritizzato** con impact analysis
3. **Codice Ottimizzato** per areas critiche (calcolo progressivo, Firebase ops)
4. **Test Suite Completa** (Unit + Integration + UI tests)
5. **Performance Benchmark Report** con miglioramenti suggeriti
6. **Best Practices Implementation** per Firebase Android apps

---

⚠️ FOCUS AREAS CRITICHE

- **CALCOLO PROGRESSIVO**: È il cuore dell'app - deve essere perfetto e veloce
- **FIREBASE EFFICIENCY**: Ottimizza queries per ridurre costs e latency
- **STATE MANAGEMENT**: Gestione stato cliente selezionato deve essere bullet-proof
- **UI RESPONSIVENESS**: L'app deve essere fluida durante calcoli/inserimenti

Inizia l'analisi completa del progetto Android e genera il piano di ottimizzazione dettagliato.