🧠 Systemarchitektur von tOrder 2 – Modernes GUI für Industrie-Terminals

🎯 Vision

tOrder 2 ist ein umfassendes, robustes und zugleich skalierbares System für Touchscreen-Anwendungen in der industriellen Fertigung. Die Architektur basiert auf den Anforderungen:

- einfacher Anpassbarkeit an verschiedene Werke (einschließlich internationaler Standorte und Sprachvarianten),
- hoher Wartbarkeit, Testbarkeit und Betrieb ohne IT-Unterstützung vor Ort,
- vollständiger Trennung vom Datenbanksystem durch eine zentrale GRPC-API.

🏗️ Systemarchitektur

⚙️ Technologische Grundpfeiler

- **WinUI 3** mit einem eigenen Komponenten-Framework
- **MVVM + Dependency Injection (DI)** zur Trennung von Logik und Darstellung
- **GRPC-API** als Middleware für alle Datenzugriffe (kein direkter DB-Zugriff nötig)
- **Serilog + MQTT** für zentrales Logging, Telemetrie und Alerts
- **Klare XAML-Struktur** mit sauberer Trennung zwischen View, ViewModel und Model
- **Metadaten-System**, Layout-Registry, Element-Registrierung

🪝 Vorteile der GRPC-Schicht

- Keine Anwendung (Touchpanel, Desktop, TV) benötigt direkten DB-Zugang
- GRPC kümmert sich um:
    - Authentifizierung,
    - Zugriffskontrolle,
    - Caching & Datensynchronisation,
    - Datenvalidierung und Integritätsprüfungen
- Änderungen am System oder Produkt lassen sich sofort global ausrollen

---

🔷 Entwicklungsschleife

1. **GUI-Entwurf**: Komponentenzeichnungen, Layouts nach Rollen/Anwendungen
2. **Demo-Erstellung**: XAML-Struktur (Grids, Borders, Controls)
3. **Logikentwicklung**: ViewModels, zentrale Einstellungen, Redesign
4. **Dateneinbindung**: GRPC-Verbindung, Tests, Validierungen
5. **Kundenfeedback**: Designänderungen, Freigabeprozess
6. **Testing**: Übergabe an Tester, Stabilisierung
7. **Rollout**: Merge in ALFA-Version, Feedback und Korrekturen

---

🧪 Testbarkeit & Debugging

- **GRPC-Backend** ist unabhängig testbar – auch ohne GUI
- Da **WinUI Packaged Apps** keine NUnit-Tests erlauben:

    - Nutzung einer **Unpackaged Testumgebung**
    - Debug-Modul mit **Mock-Daten**
- Werkzeuge für Tester:
    - Debug-Konsole
    - Log Viewer
    - Metadaten-Inspektor

🪵 Logging, Telemetrie, Monitoring

- **Serilog MQTT-Appender** für zentrales Log-Streaming
- Hintergrund-Worker zum periodischen Löschen alter Logs
- Automatisierte **zeitgesteuerte Reports**:
    - Fehlerübersicht
    - Produktionszahlen je Schicht
    - Status der Terminals
- **E-Mail-Versand** an Entwicklerteam
- Eigene Fehler-Tabelle mit Analytik-Tools

🖥️ UI-Design & Betriebsmodi

- **Grid-basiertes Layout** für 2 Hauptauflösungen (FullHD, Industrie-Tablet)
- Nicht unterstützt:

    - dynamische Größenänderung zur Laufzeit
    - UltraWide-Gamingformate (z. B. 32:9)
- Betriebsmodi:
    - **Management-Modus**: größere Schrift, hoher Kontrast
    - **Panel-Modus**: Touch-optimiert
    - **TV-Modus**: passive Anzeige (z. B. Produktionsübersicht)

🌐 Internationalisierung & Anpassung

- Echtzeit-Sprachumschaltung (Multilanguage-Support)
- Dynamisches Styling (Farben, Größen, Kontrast)
- Tooltips, Hilfetexte, Schulungsmodus
- Konfiguration je Gerät über `.json` / `.yaml`

🧠 Datenarchitektur & Datenflusssteuerung

- **Pages** = zentrale ViewModels mit eigener Logik
- **SubViews** = angebunden an übergeordnete VM, teilen den Zustand
- **Controls** = CodeBehind, stets gebunden an das DataContext-VM
- Validierungen, Statuswechsel, Events → gesteuert über **Services + VMs**
- **Vollständige DI-Struktur** für Datensteuerung und Initialisierung

---

🟪 ALFA-Version – Produktphase

- **ALFA** = stabile Version für erste Tests in der Produktion
- Enthält:
    - Schulungsmodus
    - Management-Freigabewerkzeuge
    - Projektdokumente (Layout-Registrierung, GUI-Designs)
- Ziel: Erfassung von Nutzerfeedback & Optimierungen

📄 Dokumentation

- GUI-Entwürfe & Zeichnungen
- UI-Architektur & Anwendungsschicht
- Codekonventionen & Architekturprinzipien
- Testtools, Utilities, Debugging
- Ticket-System & Projektsteuerung

📋 Praxiserprobung & Referenz

- **2 Jahre Erfahrung** mit über **40 Kiosk-Typen** bei **DGS**
    - Eigenverantwortlicher Entwurf, Entwicklung, Wartung
    - Backend (GRPC, DB), Frontend (WPF, XAML)
- Erprobt:

- stabile Synchronisation
- fehlerfreie Buchungen (Aufträge, Schichten)
- Validierung NOK/OK
- Kontrolle von Material, Chargen, QS-Proben

💡 Zusammenfassung der Vorteile

- **Komplett entkoppeltes Ökosystem** – kein direkter DB-Zugriff nötig
- **Zentrale GRPC-Schicht** – Sicherheit, Kontrolle, Skalierbarkeit
- **Effizienter Datenfluss & Logging** – kein großer IT-Support nötig
- **Konsistentes UI auf allen Geräten** – einfache Wartung
- **Bereit für globalen Rollout** – dokumentiert, testbar, modular erweiterbar

Stand des Projekts tOrder und fehlende Komponenten

1) Layout & Bildschirm-Anpassung

Erledigt:

- Dynamische Anpassung der Elementgrößen an die aktuellen Fensterabmessungen (Fullscreen und Fensterbetrieb) **ohne** klassisches Element-Scaling, stattdessen über **Element-Registrierung**.
- Sekundäres Editierfenster ermöglicht Test- und Feinabstimmungs-Anpassungen **vor** dem Rollout.

- Gedacht ist auch an **Online-Konfiguration pro Benutzer-Login** (Erstsetup während der Schulung).
- **Tooltips** dienen der schnellen Identifikation von Elementen bei der Feinjustierung.

Fehlt:

- **Interaktives Tutorial** und **Popup-Hilfen** mit Erklärungen zu einzelnen UI-Elementen.

---

2) Datenschicht

Aktueller Stand:

- Daten werden aus einer **beigefügten XML-Datei** geladen (ungeeignet für den Zielbetrieb).

Erforderliche Änderungen:

- Umstellung auf **gRPC-Kommunikation**.
- **Testdaten** und deren Kombinationen **direkt über den gRPC-Service** einspielen.
- Grundlage für **Automatisierungstests** vorbereiten.

---

3) Grafische Komponenten

Erledigt:

- WinUI-Komponenten um **projekttaugliche Eigenschaften** erweitert.
- UI-Bausteine für **vollständig dynamische Konfiguration** entworfen (Visualisierung nach Wunsch und pro Benutzer).

Fehlt / Nachzuarbeiten:

- **Mehrsprachigkeit (Multilanguage)** – vollständige Implementierung, idealerweise **nach** Abschluss der Haupt-GUI des Demos.

- **Farbschemata und Stilistik** – teilweise zentralisiert, hohe Kombinationszahl erfordert inkrementelles Vorgehen.

- **Effekte und Animationen** – derzeit nur das Nötigste; vollständige Ausgestaltung **nach Definition** der Animationspunkte/Transitions.

---

4) Sicherheit

Aktueller Stand:

- App-Sicherheit **rahmenhaft** über das Framework, **ohne** aktive Schutzmechanismen gegen Angriffe oder **Anomalie-Detektion**.

- WinUI wird zeitnah **Open-Source** → erhöhtes Erfordernis für Schutzmaßnahmen.

Erforderliche Änderungen:

- Absicherung von **gRPC** und **MQTT** mittels **TLS-Verschlüsselung** (lokal und standortübergreifend).

- Für Remote-Transfers **VPN** und **nicht-Default-Ports** verwenden.

- **Aktives Monitoring** von Sicherheitsereignissen implementieren.

---

5) Architektur & Server-Anbindung

Zielzustand bis Ende 2025:

- **Funktionsfähiges Demo** mit mind. **80 % GUI** und notwendiger Logik für einen **realen Test in 01/2026**.

- **gRPC-Server** als **zentrale Datendrehscheibe**.

- **Telemetrie-Server** mit Logs & Monitoring jeder tOrder-Instanz über **MQTT**, verarbeitet als **C# Worker Service**.
- **Web-Frontend (Node-RED)** mit Grafiken & Statistiken.
- **MS SQL-Datenbank** für:
    - Logs (mit priorisiertem Aufräumen),
    - Telemetrie,
    - Kiosk-Konfigurationen,
    - Schulungs-/Simulations-/Testdaten.

---

6) Entwicklungsprozess & Dokumentation

Plan:

- Entwicklungsfortschritt auf einer **internen Firmen-Social-Plattform** (Alternative zu MS Teams) dokumentieren → **transparente Übersicht** für Management & Leitung.
- **Zentralisierte Dokumentation** an einem Ort, zugänglich für alle Teammitglieder.
- **Verbindliche Methodiken** definieren, um die Entwicklung zu vereinheitlichen → **Inkonsistenzen** im Projekt vermeiden.
- Nach Abschluss jedes Blocks → **Übergabe an Test** und **Freigabe** durch das Management.
- **Anforderungs- & Prioritätensteuerung** via **Jira**.

Prioritäten (Kurzüberblick)

1. **Migration** der Datenschicht von **XML** → **gRPC**.

2. **Sicherheitsmaßnahmen** einführen (TLS, VPN, Security-Monitoring).
3. **GUI abschließen** (Mehrsprachigkeit, zentrale Farbverwaltung, Animationslogik).
4. **Telemetrie- & Logging-Ökosystem** aufbauen.
5. **Dokumentation** und **vereinheitlichte Entwicklungsprozesse** etablieren.