Ablaufplan — Dokument-Erstellung von Grund auf (Generator)

Farb-Legende (Status):

✓	◑	○	★	✗
fertig	teilweise	offen	neu / Baustelle	entfaellt / nie

Ziel: PDFs von Null erzeugen — Text, Bilder, Tabellen, Vektorgrafik — mit wählbaren Seitengrößen (A-Reihe, US, frei) und Doppelseiten (Spread-Ansicht + physische Doppelseiten). Baut auf der vorhandenen Lese-/Editier-/Content-Infrastruktur auf und sichert jeden Schritt per Round-Trip über den eigenen Reader.
Prinzip: unten anfangen (Datei schreiben) -> Seiten/Größen -> zeichnen -> Layout -> Tabellen -> Fassade. Jeder Meilenstein ist eigenständig nützlich und testbar; die harten Constraints bleiben (nur BCL, null NuGet, wirft-nie-Prinzip, TDD, Review).

M33 — Voll-PDF-Writer (Fundament)

Warum: Heute gibt es nur IncrementalWriter.AppendUpdate (anhängen an ein bestehendes PDF). Für ein neues Dokument fehlt ein Schreiber, der eine komplette Datei erzeugt.
Liefert:
- Writer/PdfDocumentWriter: aus einem Objektgraph (Katalog, Seitenbaum, alle indirekten Objekte) eine vollständige Datei schreiben — Header %PDF-1.7 + Binär-Markerkommentar, Body (alle Objekte via PdfObjectWriter mit korrekten Byte-Offsets), klassische xref-Tabelle, Trailer (/Size /Root [/Info]), startxref, %%EOF.
- Ein schlankes In-Memory-Dokumentmodell (PdfObjectStore): Objekte halten, Nummern vergeben, Referenzen bilden.
- Round-Trip-Test: schreiben -> mit PdfDocument.Load wieder laden -> Objektgleichheit/Struktur prüfen.

Nutzt/erweitert: PdfObjectWriter, Objects, Structure (nur Reader zum Verifizieren).

M34 — Dokument-/Seiten-Builder + Seitengrößen + Doppelseiten

Liefert:

- PageSize: A-Reihe A0–A6, Letter/Legal/Tabloid/Executive, plus frei (Breite/Höhe in Punkten oder mm); Orientation Hoch-/Querformat (tauscht B/H). Z.B. PageSize.A4, PageSize.Custom(210, 297, Unit.Millimeters), .Landscape().
- PdfDocumentBuilder: Create(), AddPage(PageSize) -> PageRef, baut den /Pages-Baum (mit /MediaBox je Seite), Build() -> byte[] über den M33-Writer. Optionale Metadaten (/Info: Titel/Autor/Ersteller, Erstelldatum via PdfDate).
- Doppelseiten:
- Spread-Ansicht (Buch/Magazin): Katalog-/PageLayout = TwoPageLeft/TwoPageRight (nebeneinander), plus „Deckblatt zählt einzeln"-Option (OneColumn/Cover). /PageMode optional.
- Physische Doppelseite: Helfer AddSpread(PageSize) legt eine doppelt-breite Seite an (MediaBox = 2×Breite) für echte Druck-Spreads — bzw. zwei zusammengehörige Einzelseiten.
- Round-Trip-Test: A4/Letter/Custom/Landscape/Spread erzeugen -> laden -> MediaBox/Seitenzahl/PageLayout prüfen.

Nutzt/erweitert: M33-Writer, PageEditing-Muster (Seitenbaum).

M35 — Seiten-Zeichen-API (Text · Bild · Vektor als echter Content)

Warum: ContentStreamBuilder erzeugt Content, aber es gibt keine öffentliche „auf Seite X zeichnen"-API; Bild-/Font-Einbetter sind da, aber nicht mit einer neuen Seite verdrahtet.

Liefert:

- PageCanvas (je Seite eine Zeichenfläche): verwaltet den /Contents-Stream und /Resources (Fonts/XObjects) der Seite. Methoden:

- Text: DrawText(text, x, y, font, size, color) (Baseline-Positionierung, WinAnsi/eingebettet), MeasureText(...).
- Bild: DrawImage(jpegOrPngBytes, x, y, width, height) — bettet einmal ein (RasterImage.Parse -> Embed), referenziert per Do.
- Vektor: DrawLine/DrawRectangle/DrawEllipse/DrawPolygon/DrawPath, SetFill/SetStroke/LineWidth, Deckkraft (ExtGState).
- Zustand: Save/Restore, Translate/Scale/Rotate (Transform).
- Font-Verwaltung: Standard-14 sofort; TrueType/OTF einbetten (via Type0FontBuilder), Ressource einmal pro Dokument, mehrfach nutzbar (baut auf M20 FontCatalog-Ideen).
- Round-Trip-Test: Seite mit Text+Bild+Rechteck zeichnen -> laden -> mit PageTextExtractor/PageGraphicsExtractor (M24/M25!) verifizieren, dass Text/Position/Bild/Pfad korrekt herauskommen. (Schöne Symmetrie: die Extraktions-Schicht validiert die Generierungs-Schicht.)
Nutzt/erweitert: ContentStreamBuilder, JpegImage/PngImage/RasterImage, Standard14Fonts, Type0FontBuilder, TrueTypeFont, WinAnsiEncoding; verifiziert über M24/M25.

M36 — Text-Layout (Absätze, Umbruch, Ausrichtung, Seitenfluss)

Liefert:
- DrawTextBlock(text, rect, style): Wortumbruch in ein Rechteck, harte \n (aus M32), Zeilenabstand, Ausrichtung links/zentriert/rechts/Blocksatz, mehrere Absätze, Absatzabstand.
- Seitenfluss: Überlauf -> nächste Seite (Rückgabe „was nicht mehr passte" bzw. automatische Folgeseite über den Builder). Ränder (Margins) je Seite.
- Inline-Grundlagen: fett/kursiv über Font-Varianten (Standard-14-Stile aus M20), Größe/Farbe je Absatz.
- Test: langer Text in schmale Spalte -> korrekte Zeilen/Umbrüche/Ausrichtung; Überlauf auf Folgeseite.
Nutzt/erweitert: M35-Canvas, TextAppearanceRenderer-Umbruchlogik (verallgemeinert), Font-Metriken.

M37 — Tabellen

Liefert:

- Table-API: Zeilen/Spalten, Spaltenbreiten (fest/auto/anteilig), Zellinhalt Text oder Bild, Zell-Ausrichtung (h/v), Rahmen (je Zelle/außen, Linienstärke/Farbe), Zell-Padding, Hintergrundfarbe, Kopfzeile, automatische Zeilenhöhe (aus umbrochenem Text), Seitenumbruch über Seiten (Kopfzeile wiederholt).
- Rendering über den M35-Canvas + M36-Textblock je Zelle.
- Test: 3×N-Tabelle mit Text-/Bildzellen, Rahmen, langer Text -> korrekte Zeilenhöhen; Umbruch über zwei Seiten mit wiederholter Kopfzeile.

Nutzt/erweitert: M35 (zeichnen), M36 (Zelltext).

M38 — Hoch-Level-Dokument-Fassade (Politur)

Liefert:

- Bequeme Autoren-API: Document mit Rändern/Kopf-/Fußzeilen, AddHeading/AddParagraph/AddImage/AddTable/AddPageBreak/AddSpacer, automatischer Seitenumbruch und fortlaufender Fluss (Fließdokument), Seitenzahlen (koppelt an M22).
- Optional: einfache Stile/Themes (Schrift, Farben, Abstände).
- Test/Demo: ein mehrseitiges Beispieldokument (Titel, Absätze, Bild, Tabelle, Seitenzahlen) end-to-end erzeugen und über die Extraktoren verifizieren.

Nutzt/erweitert: M34–M37, M22 (Seitennummern).

M39 — Zusammenführen (Buch) & Aufteilen (Split)

Warum: Häufigster Assembly-Anwendungsfall — mehrere PDFs zu einem Buch verbinden und ein PDF in mehrere Dokumente aufteilen. Arbeitet auf bestehenden PDFs; braucht nur den Voll-Writer (M33) und den vorhandenen Reader — unabhängig von M34–M38 (kann direkt nach M33 gebaut werden).

Liefert:
- PdfObjectGraphCopier (Kern, wiederverwendbar): kopiert die transitive Hülle aller von einer Seite erreichbaren Objekte (Resources, Fonts, Bilder, Content-Streams, Annotationen) in einen Ziel-Objektspeicher und nummeriert Referenzen um (Map alt->neu, zyklensicher). Fundament für Merge und Split.
- Zusammenführen (Merge / Buch):
- PdfMerger.Merge(IEnumerable<ReadOnlyMemory<byte>> documents) -> byte[]: alle Seiten aller Quell-PDFs in ein Dokument, Objekt-Hüllen kopiert+umnummeriert, gemeinsamer /Pages-Baum, via M33 geschrieben.
- Buch-Fassade darüber: optionales Deckblatt, durchgehende Seitenzahlen (koppelt M22), Lesezeichen/Outline je Quelldatei (Kapitel), optionale Kapiteltrennseiten.
- Bekannte Grenzen (dokumentiert, ggf. später): Zusammenführen von /AcroForm (Feldnamen-Kollisionen), benannten Zielen und Outlines der Quellen — erster Wurf: Seiten + Resources sauber, Formulare „flatten oder erste gewinnt".
- Aufteilen (Split):
- PdfSplitter.SplitAt(ReadOnlyMemory<byte> document, int pageIndex) -> (byte[] Teil1, byte[] Teil2)
- genau dein Fall: 1 PDF an einer Stelle zerschneiden -> 2 Dokumente.
- SplitByRanges(document, ranges) -> byte[][], ExtractPages(document, range) -> byte[], SplitEveryNPages(document, n) -> byte[][]. Jedes Ergebnis enthält nur die Objekt-Hülle seiner Seiten.
- Round-Trip-Tests: 3 PDFs mergen -> laden -> Seitenzahl/Reihenfolge/Inhalt (via PageTextExtractor) stimmen; ein 5-Seiten-PDF SplitAt(2) -> zwei Dokumente mit 2 bzw. 3 Seiten, Inhalte korrekt; Merge(Split) ergibt wieder das Original (inhaltlich).
Nutzt/erweitert: M33-Writer, Reader (PdfDocument), PageEditing-Seitenbaumlauf, M22 (Seitenzahlen), M23/M24 (Verifikation).

Abhängigkeiten (Reihenfolge zwingend)

M33 Voll-Writer

M34 Builder + Seitengrößen + Doppelseiten

M35 Zeichen-API (Text/Bild/Vektor)

M36 Text-Layout (Fluss)

M37 Tabellen

M38 Dokument-Fassade

M39 Zusammenführen (Buch) & Aufteilen (Split) <- nur M33 nötig, parallel zu M34-M38

Deine Punkte — wo abgedeckt

- Bilder: M35 (DrawImage, JPEG/PNG einbetten), Bildzellen in M37.
- Text: M35 (platzieren) + M36 (Absätze/Umbruch/Ausrichtung/Fluss).
- Tabellen: M37 (Spalten/Rahmen/Kopfzeile/Seitenumbruch).
- „alles": Vektorgrafik (M35), Layout (M36), Fassade (M38); alle bestehenden Werkzeuge (Formulare, Kommentare, Wasserzeichen, Ebenen) sind danach auf neu erzeugte Dokumente anwendbar.
- Seitengrößen wählen: M34 (PageSize: A-Reihe/US/frei in Punkten oder mm, Hoch-/Querformat).
- Doppelseiten: M34 (Spread-Ansicht /PageLayout + physische Doppelseite AddSpread).
- Buch aus mehreren PDFs / Zusammenführen: M39 (PdfMerger + Buch-Fassade).
- Aufteilen (1 PDF -> 2 Dokumente): M39 (PdfSplitter.SplitAt u.a.).

Absicherung

Jede Generierungs-Ebene wird über die eigene Extraktions-Schicht (M23–M27) round-trip-verifiziert — das Geschriebene wird mit dem eigenen Reader wieder ausgelesen und geprüft. Externe Validierung optional über einen echten Viewer/fonttools wie in früheren Meilensteinen.