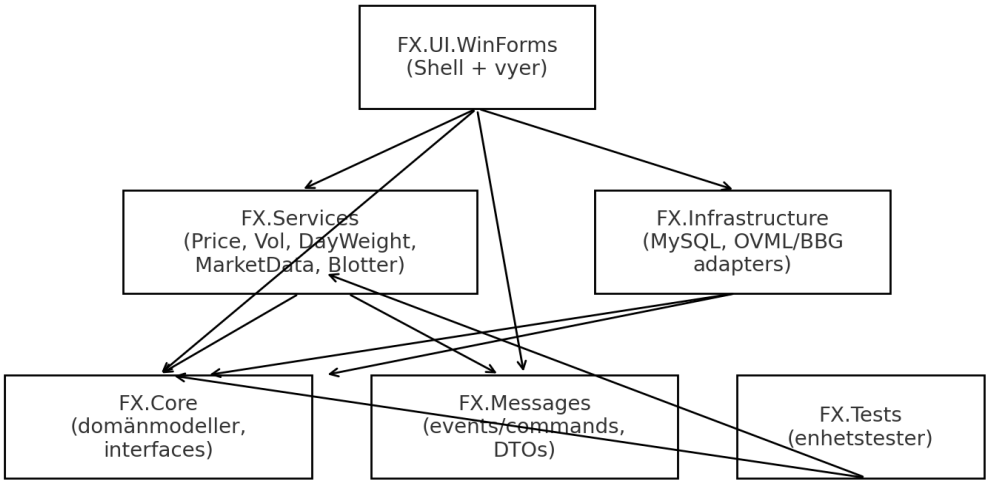
Option Suite – Sprint 1 (Steg 1–8)

Mål: ett körbart skelett med Pricer, Vol Surface och Blotter. Denna plan täcker steg 1–8, arkitektur, Gantt och RACI.

Arkitekturöversikt (lös koppling)

UI pratar via services och meddelanden. Core och Messages är stabila beroenden. Infrastruktur byts utan att påverka domänen.



Steg 1 – Skapa solution-skelett

- Skapa projekten: FX.Core, FX.Services, FX.Messages, FX.Infrastructure, FX.UI.WinForms.
- Sätt C#-version centralt (Directory.Build.props) och aktivera binding redirects.
- Lägg projekt-referenser: UI -> Core/Services/Messages/Infrastructure; Services -> Core/Messages; Infrastructure -> Core/Messages.
- Installera Microsoft.Extensions.DependencyInjection i Services och UI. Bygg en minimal composition root i Program.cs.

Klar när:

- Solution bygger utan fel.
- WinForms-fönstret startar.

Tips:

- Håll UI-projektet tomt på logik. Allt ska gå via services senare.

Steg 2 – Lyft ut domänen till FX.Core

- Skapa rena domänklasser: CurrencyPair, Expiry, Strike, OptionLeg, PricerResult, VolNode, VolSurface, DayWeightCurve.

- Lägg stateless helpers: ISO-currency whitelist, ATM-definition, delta/strike-inference.
- Inga beroenden till UI, DB eller I/O.

Klar när:

- UI kompilerar genom att referera Core-typerna.
- Core bygger utan externa beroenden.

Tips:

- Behåll allt immutabelt där det är möjligt för att förenkla trådsäkerhet.

Steg 3 – Definiera kontrakt (interfaces och messages)

- Interfaces: IPriceEngine, IVolService, IDayWeightService, IMarketDataService, IBlotterService, IMessageBus.
- Events/commands i FX.Messages: RequestPrice, RebuildVolSurface, ApplyDayWeights, BookTrade; PriceCalculated, SurfaceUpdated, TradeBooked, ErrorOccurred, SpotUpdated, RatesUpdated.

Klar när:

- Du kan skapa 'dummy presenters' i UI som bara skickar/lyssnar på events/commands.

Tips:

- Håll meddelanden små och seriella; undvik att skicka tunga objekt i events.

Steg 4 – Minimal MessageBus + AppState

- Implementera in-process pub/sub: Subscribe, Publish(evt). Gör den trådsäker.
- Inför AppState som ett immutabelt snapshot: aktivt par, spot/rd/rf, aktiv surface-id/version, senaste PricerResult, blotter-rader.
- UI lyssnar på AppState-uppdateringar och re-renderar.

Klar när:

- En vy uppdateras automatiskt när ett event påverkar state.

Tips:

- Använd en enkel version-counter i AppState för enkel jämförelse/trace.

Steg 5 – Tunna implementationer (MVP)

- IVolService: Dictionary pair -> VolSurface; API: GetVol(tenor, strike/delta), RebuildSurface(nodes, dayWeightCurve). Publicera SurfaceUpdated.
- IPriceEngine: ta PricingRequest, gör en iteration (ATM/vol lookup), returnera PricerResult. Publicera PriceCalculated.
- Hoppa sticky-delta/strike tills pipelinen fungerar.

Klar när:

- UI visar rimliga värden med platt vol (t.ex. 10%).

Tips:

- Logga inputs/outputs i debug-läge för snabb felsökning.

Steg 6 – UI Shell + tre vyer

- Shell (Form1): vänster navigering och huvudpanel med tabs eller dockade fönster.
- PricerView: inputs + Price-knapp -> RequestPrice; lyssna på PriceCalculated.
- SurfaceView: redigerbar tabell för tenors/ATM/RR/BF + Rebuild-knapp -> RebuildVolSurface.
- BlotterView: DataGridView (gärna VirtualMode) som lyssnar på TradeBooked.

Klar när:

- Du kan byta vy, prisa, se resultat och rebuilda dummy-yta.

Tips:

- Håll presenters tunna; all beräkningslogik i services.

Steg 7 – Koppla in verklig logik

- Ersätt dummy PriceEngine med din riktiga motor bakom IPriceEngine.
- Koppla din VolInterp under IVolService (lookup/interpolering).
- Lägg DayWeightService som först returnerar 1.0 för alla dagar.

Klar när:

- Ditt verkliga pris/greker visas och reagerar på surface-ändringar.

Tips:

- Mät latens från request till PriceCalculated-event (statusrad i UI).

Steg 8 – Persistens och MarketData

- IMarketDataService: starta med in-memory/fil; koppla OVML/BBG senare.
- IBlotterService: börja in-memory; byt till MySQL via Dapper. Lägg append-only historik + vy för 'senaste status'.
- Lägg kommandot 'Re-price with current surface' i blottern.

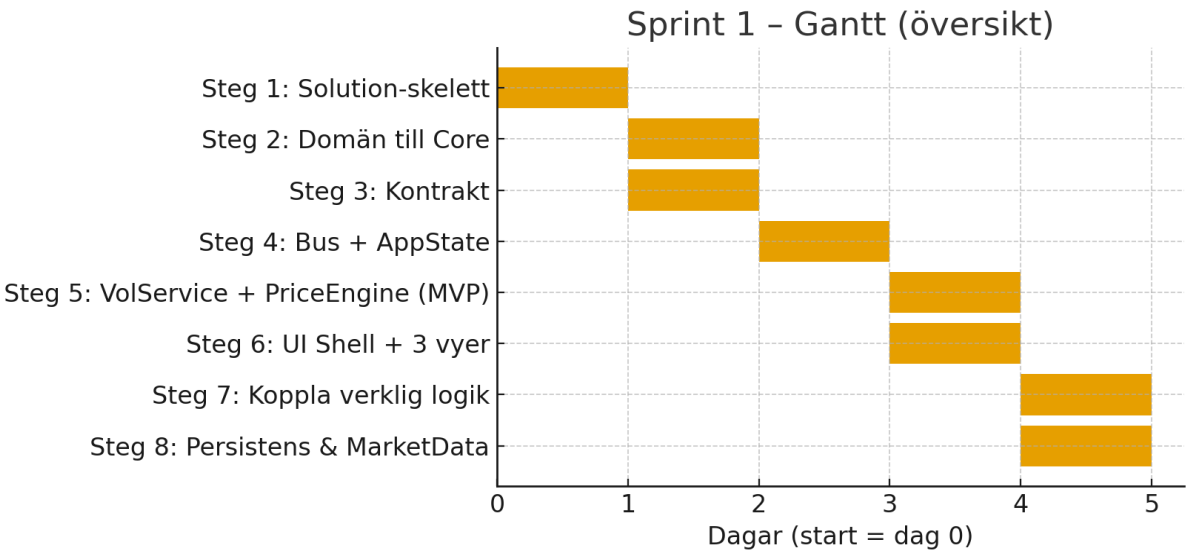
Klar när:

- Book Trade skapar rad i blottern (minne). Re-price fungerar.

Tips:

- Säkerställ transaktionsgränser och reconnect-logik för DB.

Gantt (översikt)



Antagande: 5 dagar. Vissa steg kan göras parallellt samma dag.

RACI – Roller och ansvar

Steg	Per	Assistent	Infra/DB	MarketData
1. Solution-skelett	R/A	C	I	I
2. Domän till Core	R/A	C	I	I
3. Kontrakt	A	R	I	I
4. Bus + AppState	A	R	I	I
5. VolService + PriceEngine (MVP)	A	R	I	I
6. UI + 3 vyer	R/A	C	I	I
7. Verklig logik	A	R	I	C
8. Persistens & MarketData	A	C	R	R/C

R = Responsible, A = Accountable, C = Consulted, I = Informed.

Acceptanskriterier

- Starta appen, välj EURSEK, tryck Price och få numeriskt pris och greker.
- Ändra en vol-nod i SurfaceView, kör Rebuild Surface och se att Price ändras.
- Book Trade skapar en rad i Blotter. Välj raden och kör Re-price with current surface.

Tips som sparar timmar

- Håll UI dumt; all logik i Services/Presenters.
- Använd async/await och CancellationToken i dyra anrop.
- Jobba med immutabla snapshots i AppState för att undvika race conditions.
- Lägg feature toggles: Auto-reprice on SurfaceUpdated, Sticky-delta on/off, Use day-weights.