
CADEC
– CALLISTA DEVELOPER CONFERENCE –
STOCKHOLM 2025.01.23 | GÖTEBORG 2025.01.29

CADEC 2025 - KONFERENSEN FÖR UTVECKLARE SOM VILL UTVECKLAS

Cadec är konferensen som ger dig de senaste trenderna inom arkitektur och utveckling. Här får du tillfälle att rivstarta det nya året med ett rejält kunskapslyft.

Vi håller till i Filmstadens biosalonger där vi kan utlova en presentationsupplevelse i toppklass.

Det blir naturligtvis också After Cadec på respektive kontor där vi kan varva ner efter konferensen med mat, dryck och mingel. En separat inbjudan till After Cadec kommer senare.

Konferensen är som vanligt kostnadsfri.

Cadec ges vid två tillfällen – ett i Stockholm och ett i Göteborg. Mer information om konferensen och länk till anmälan ser du i rutan till höger.

Antalet platser är begränsat - anmäl dig redan idag för att inte missa denna chans till gratis kompetensutveckling.

Stockholm

Datum	torsdag 23:e januari 2025
Tid	13.00 till 17.00 registrering från 12.30
Plats	Filmstaden Sergel, Hötorget 3, Stockholm
After Cadec	Mat, dryck och mingel 17.00 - Callistas kontor Drottninggatan 55, Stockholm
Anmälan	https://www.eventbrite.com/e/cadec-2025-stockholm-biljetter-1087016849659

Göteborg

Datum	onsdag 29:e januari 2025
Tid	13.00 till 17.00 registrering från 12.30
Plats	Filmstaden Bergakungen, Skånegatan 16 B, Göteborg
After Cadec	Mat, dryck och mingel 17.30 - Callistas kontor Fabriksgatan 13, Gårda
Anmälan	https://www.eventbrite.com/e/cadec-2025-goteborg-biljetter-1087105544949

Program Cadec 2025

Local-first - välkommen till en värld av responsiva appar som samarbetar sömlöst utan API:er

Stephen White

Föreställ dig för ett ögonblick en värld utan spinners – där appar startar omedelbart, samarbetet med kollegor sker utan avbrott, innehållet delas och synkroniseras sömlöst mellan enheter och där nätverkshastigheten aldrig sätter gränser.

Denna vision kan nu förverkligas. Välkommen till *local-first*-världen.

I den här presentationen utforskar vi hur local-first blir verklighet – från smarta lagringsstrategier och effektiva synkroniseringsmönster till den snabbt växande mängden bibliotek och ramverk som driver utvecklingen framåt. Vi visar också hur dessa principer kan omsättas i praktiken, med Cadec-appen som ett konkret exempel.

Snabbare uppstart av Java-applikationer med CRaC

Magnus Larsson

Med en ökad användning av mikrotjänster och molnbaserade arkitekturer har snabba uppstartstider för applikationer blivit allt viktigare för effektiv skalning och minskad nertid. För Java-applikationer kan dock krav på korta uppstartstider vara svåra att uppfylla.

I denna presentation introducerar vi CRaC (Coordinated Restore at Checkpoint), en teknik som kan minska uppstartstiden avsevärt för Java-applikationer. Vi går igenom hur man går tillväga för att använda CRaC och demonstrerar hur det fungerar för applikationer utvecklade med Spring Boot.

Presentationen jämför också CRaC med andra tekniker för att minska uppstartstider såsom GraalVM native compile, AppCDS och project Layden.

Säkra modulära applikationer i backend med WebAssembly och WASI

Peter Larsson

WebAssembly System Interface WASI-Preview 2 släpptes i början av 2024 och möjliggör utveckling av säkra, snabba och modulära applikationer på serversidan. Med stöd för Garbage Collection, Exceptions och trådar är det enklare att använda exempelvis JVM-baserade språk. Komponentmodellen möjliggör strukturerade monoliter med isolerade moduler som till stora delar möter arkitektmålen för mikrotjänster, och med ett utbrett stöd för att exekvera i lövtunna (OCI) containers.

Presentationen redovisar varför WASM är en viktig standardiserad teknik att bevaka, hur nuläget ser ut och vad vägen framåt bär med sig. Dessutom demonstreras hur man utvecklar och kör WASM tillämpningar där komponenterna kan vara skrivna i olika språk.

Domänen i fokus: Portar, Adaptrar och Hexagonal Arkitektur

Björn Beskow

Den till synes oundvikliga komplexiteten som plågar de flesta mjukvaruprojekt kommer ofta från beroenden mellan delar av lösningen, beroenden som över tid blir ohanterliga och leder till "legacy" (även känt som "big ball of mud"). Arkitekturarbetets kanske viktigaste uppgift är som bekant att bromsa denna ökande "mjukvaru-entropi" genom att begränsa och kontrollera beroenden med hjälp av abstraktioner, lagerindelning och arkitekturella principer som t.ex SOLID. De allra mest stabila och värdefulla delarna i en mjukvarulösning är de som också är viktigast att skydda mot osunda beroenden: domänen eller kärnverksamhetens regler och beteende.

Ironiskt nog kan en traditionell, "endimensionell" lager-indelning motverka detta syfte, genom att premiera det "understa" lagret. Det är skälet till att man i DDD-lägret allt oftare pratar om en annan, "flerdimensionell" lagerindelnig med domänen i centrum, omgiven av "portar" och "adaptrar" i en lökskalsmodell med det något kryptiska namnet Hexagonal Arkitektur.

I den här presentationen tittar vi närmare på denna arkitekturella stil, och finner ett användbart verktyg i DDD-arkitektens verktygslåda med (i vanlig ordning) ett antal avigsidor.

Structured Concurrency för enklare multitrådad programmering

Jesper Holmberg

Structured Concurrency är ett stöd för utvecklare att skriva multitrådad kod som är korrekt och enkel att resonera kring. Exekveringstrådar är resurser som måste hanteras på ett strukturerat sätt, på samma sätt som filer, nätverkskopplingar eller andra potentiellt dyra resurser. Med structured concurrency får utvecklaren hjälp att hantera den komplexitet som en multitrådad lösning innebär, och inte tappa bort skapade resurser.

Structured Concurrency är ett begrepp som började uppmärksammas för 5-6 år sedan och bland annat influerade designen av Kotlins coroutines. Därefter har det använts i populära bibliotek för Python och andra språk, och är sedan några år inlemmat i Swift. I Java finns nu Structured Concurrency som en naturlig överbyggnad på Project Looms virtuella trådar.

I den här presentationen tittar vi på vad Structured Concurreny egentligen innebär, och illustrerar med exempel från några olika programmeringsspråk.

Köra stora språkmodeller lokalt – hur gör man och vad får man?

Patrik Blommaskog

Sedan ChatGPT släpptes till allmänheten för två år sedan så har bruket av stora språkmodeller, LLM:er, blivit vardagsmat för de flesta av oss. Genom olika förpackningar använder vi dem för personligt bruk, som arbetsredskap, och ibland som aktiva delar av produkter och tjänster.

Idag finns det många modeller som går att köra på egna datorer utan att man behöver anförtro sig till leverantörer i molnet. Det är ett snabbt framåtskridande område där man ständigt lyckas klämma in bättre prestanda och mer funktionalitet i hårdvara som fortsätter att bli både billigare och kraftfullare.

Men - man klämmer knappast in ett helt moln i en laptop utan kompromisser. I detta föredrag tittar vi på vad det innebär att köra en språkmodell lokalt. När och varför skulle man vilja välja en lokal modell framför en molnbaserad kommersiell dito? Vi tittar på modellval, anpassningar, egna erfarenheter, och andra saker att ta i beaktande när man väljer och arbetar med sina modeller. Under presentationen kommer vi att använda modellerna både som utvecklingsverktyg och som lösningskomponent.