Profil

Dipl.-Inform. Jens Fransson

Senior Softwarearchitekt/-entwickler (Full Stack, Agentic Engineering)

Stand: August 2026

Jens Fransson konzipiert und entwickelt seit 1996 selbständig Java-Anwendungen – von modularen Monolithen bis zu Cloud-nativen Microservices. Besonderes Augenmerk legt er auf Agentic Engineering, Clean Code, SOLID-Prinzipien, Refactoring, Test-Driven Development und funktionalen Programmierstil.

Technische Schwerpunkte

- Java Experte, Erfahrung seit 1996
- Entwicklung mit KI-Assistenz (Agentic Engineering), automatisiert abgesichert
- Clean Code, SOLID-Prinzipien, funktionaler Programmierstil
- Refactoring, Test-Driven Development
- Continuous Integration & Delivery

Frameworks & Tools

- IntelliJ IDEA, Eclipse, Claude Code
- Spring Boot, Spring MVC, Spring Security, Spring Modulith; Google Guice; Jakarta EE
- JPA/Hibernate, Liquibase, PostgreSQL, Oracle
- JUnit, Mockito, Testcontainers, Playwright, PIT
- Maven, Gradle, Git, GitLab CI, Jenkins, Jira
- Docker, Kubernetes, Helm; AWS, Azure
- Swing, SWT, Eclipse RCP, Vaadin, Angular

Themen

- Agentic Engineering: MCP, A2A, agentensteuerbare Codebasis mit Vorgaben
- Full Stack, Backend, Frontend; Kanban
- Architekturen: Event Driven, BCE, modularer Monolith, Microservices
- Migration von Desktop (Swing, SWT, Eclipse RCP) auf Web (Angular, JxBrowser)
- Branchen: Versicherung, Logistik, Öffentlicher Sektor, Energie, Finanzen, IT-Sicherheit

Good programmers write code that humans can understand.

— Martin Fowler in *Refactoring*

Inhalt

Good programmers write code that humans can understand.

— Martin Fowler in *Refactoring*

Zur Person

Name	Jens Fransson
Verfügbar	auf Anfrage
Einsatzort	Remote; vor Ort im Raum Hamburg
Ausbildung	Universität, Hamburg Abschluss: Diplom-Informatiker
Berufserfahrung seit	1995
Schwerpunkte	Softwareanalyse, -design und -implementierung; Code-Reviews, Qualitätssicherung, Clean Code, Refactoring
Fremdsprachen	• Deutsch (Muttersprache) • Englisch (fließend) • Französisch (gut)
Veröffentlichung	BOGER, M., J. FRANSSON, M. JECKLE und S. MÜLLER: Diagram Interchange for UML. In: UML 2002 – The Unified Modeling Language, LNCS 2460, S. 398–411, Springer Verlag. ISBN 3-5404-4254-5.

IT-Kenntnisse

Java (Schwerpunkt, Erfahrung seit 1996)	JDK 1.0 bis Java 26; Jakarta EE, Java EE, J2EE, JSP, Servlets, JSF; Spring Boot, Spring Core, Spring MVC, Spring Security, Spring Social, Spring Modulith; Google Guice; JPA, Hibernate, EclipseLink, JDBC; AWT, Swing, SWT, JFace, Eclipse RCP, Tycho; Vaadin, GWT, CUBA; JUnit; Java Cryptography, JAXP, JDOM, Log4J, JFreeChart, Java 2D, Java Regular Expressions, JNI
Künstliche Intelligenz	Entwicklung mit Assistenz durch KI (Agentic Engineering): Claude Code mit Opus, GitHub Copilot, Kiro, Goose; Model Context Protocol (MCP), Agent-to-Agent-Kommunikation (A2A); agentensteuerbare Codebasis mit Architekturregeln und Kommandos als Vorgaben; LLM-gestützte Test- und Dokumentationsstrategien; Entwicklung agentenbasierter Systeme
Weitere Programmiersprachen	C/C++, STL, Microsoft C#, .NET Rahmenwerk, PHP, JavaScript, jQuery, Angular
Datenbanken	SQL, Oracle, PostgreSQL, MariaDB, MySQL, DB2, H2, HSQLDB
Markup-Sprachen	HTML 5, CSS 3, SASS, XHTML, XML, XSL, XSLT (XML Stylesheets), Bootstrap, Material
Kommunikation	REST, OpenAPI/Swagger, SOAP, JAX-WS; JMS (Java Message Service), Apache ActiveMQ Artemis, Apache Kafka; WebSockets (SignalR); OAuth 2.1, OpenID Connect, SSO/Kerberos; TCP/IP, RMI

IT-Kenntnisse

Test & Qualitätssicherung	JUnit, Mockito, Testcontainers, Playwright, HttpUnit, DBUnit; Mutationstests mit PIT; Checkstyle, SpotBugs/find-sec-bugs, Error Prone/NullAway, JSpecify, SonarQube; Mercury Quality Center, Test Track Pro
Cloud, Container & Betrieb	Docker, Kubernetes, Helm, k3s/k3d; Amazon Web Services (Lambda, S3, EventBridge, Step Functions, CDK); Microsoft Azure (SignalR, Event Hub); Heroku; Prometheus, Micrometer, Spring Boot Admin; Anwendungsserver: JBoss, GlassFish, Apache Tomcat, Oracle WebLogic, IBM WebSphere
Build, CI/CD & Werkzeuge	Maven, Gradle, Ant, Tycho; Jenkins, GitLab CI, TeamCity, CruiseControl; Git (GitLab, Bitbucket, Gerrit), Subversion, CVS; Nexus; Liquibase; Jira, ClearQuest, Kanban, Wiki
Entwicklungsumgebungen	IntelliJ IDEA, Eclipse, NetBeans, MS Visual Studio, Oxygen
Modellierungswerkzeuge	Together, Visual Paradigm for UML, Rational Rose, XDE, MagicDraw
Betriebssysteme	Windows 11, 10; Linux (Ubuntu, Red Hat), Unix (Solaris)
Weitere/historische Technologien	Object Pascal, Borland Delphi (VCL, BDE), Win32 API, Mumps, GCJ (Gnu Compiler for Java), Excelsior Jet, Struts, Taglibs, MDR (Meta Data Repository), Piccolo, Jazz, db4o, MS Access, SGML, serielle Schnittstelle, Windows 8/7/Vista/3.1, MS-DOS

Projekte/Tätigkeiten

IT-Sicherheit (Eigenentwicklung), 06/26 – laufend

Projektziel	Neuentwicklung von Keywarden, einem selbst gehosteten Passwort-Tresor als Web-Anwendung. Die Geheimnisse jedes Benutzers werden mit einem eigenen Datenschlüssel verschlüsselt, der nur innerhalb der angemeldeten Sitzung existiert: Er wird ausschließlich verpackt gespeichert, unter einem aus dem Anmeldepasswort abgeleiteten Schlüssel (PBKDF2), und erst bei der Anmeldung ausgepackt. Ohne angemeldete Sitzung kann der Server die Daten nicht lesen. Ein Passwortwechsel ersetzt lediglich die Verpackung des Datenschlüssels, sodass keine gespeicherten Daten neu verschlüsselt werden müssen. Funktionsumfang: hierarchisch organisierte und durchsuchbare Schlüssel, CSV-Export und -Import mit vorheriger Änderungsvorschau, Benutzerverwaltung mit Einladungsverfahren (auch der Administrator kennt kein fremdes Passwort), zweisprachige Oberfläche (Deutsch/Englisch) sowie eine Spring-Boot-Admin-Konsole für Betrieb und Monitoring. Umfang: rund 6.900 Zeilen Produktivcode in 90 Klassen und rund 16.700 Zeilen Testcode in 120 Klassen mit 620 Testfällen.
Aufgaben/Verantwortlichkeiten	• Konzeption, Architektur und Implementierung als alleiniger Entwickler (Full Stack). • Entwicklung mit KI-Assistenz (Agentic Engineering, Claude Code). Die Codebasis ist darauf ausgelegt: Architekturregeln, Kommandos und Entwurfsbegründungen sind als Vorgaben festgehalten, und jede Änderung läuft gegen automatisierte Regeln – statische Analyse, geprüfte Modulgrenzen, vollständige Testabdeckung und Mutationstests

Projekte/Tätigkeiten

	brechen den Build. • Architektur nach Boundary-Control-Entity in sechs fachlichen Modulen (identity, keys, vault, ops, backup, shared). Spring Modulith prüft die Modulgrenzen bei jedem Build – ein unerlaubter Zugriff über Modulgrenzen bricht den Build, statt erst im Review aufzufallen. • Entwurf der Schlüsselhierarchie und des Verschlüsselungsverfahrens: benutzereigener Datenschlüssel, Ableitung des Verpackungsschlüssels per PBKDF2, bewusst nicht serialisierbarer Sitzungshalter, damit ein Session-Store den Schlüssel nicht in die Datenbank tragen kann, die er schützt. • Statische Analyse in drei Stufen ohne Überschneidung: Checkstyle in der validate-Phase, Error Prone mit NullAway im Compiler (macht die JSpecify-Annotationen erzwingbar statt beratend), SpotBugs mit find-sec-bugs über den Bytecode in der verify-Phase. Der Maven Enforcer sichert zusätzlich die Dependency-Konvergenz. • Teststrategie mit echten Abhängigkeiten: Integrationstests gegen ein echtes PostgreSQL über Testcontainers einschließlich der Migrationen, Oberflächentests in echtem Chromium über Playwright, dazu ein Test, der das gebaute Container-Image gegen eine Datenbank im gemeinsamen Netz startet. • Testabdeckung als harte Build-Bedingung: 100 % der Instruktionen, Zeilen und Zweige. Mutationstests (PIT) messen zusätzlich die Güte der Tests statt nur ihre Reichweite: vom ersten Lauf mit 88 % auf 100 % Mutationsabdeckung gehoben – der Build lässt keinen überlebenden Mutanten zu. • Auslieferung als geschichtetes Docker-Image und Helm-Chart nach k3s/k3d, CI/CD in GitLab CI auf selbst betriebenem Runner. Ein einziges Maven-Kommando führt den Release durch (Version, Chart, Commit, Tag, Image, Veröffentlichung, Folge-Snapshot) mit Schutzregeln gegen SNAPSHOT-Versionen, unpassende Tags und falschen Branch.
Methoden/Technologien	Java 26, IntelliJ IDEA, Spring Boot 4.1, Spring MVC, Spring Modulith, Spring Security 7, OAuth2, Spring Data JPA, Liquibase, PostgreSQL, Maven, HTML/CSS/ES-Module, JUnit 5, Testcontainers, Playwright, PIT, Checkstyle, Error Prone, NullAway, SpotBugs/find-sec-bugs, JSpecify, Docker, Helm, k3s/k3d, GitLab CI, Git, Spring Boot Admin

Versicherung, 06/25 – 05/26

Projektziel	• **Altanwendung:** Event Driven Architecture mit Custom Framework. Behebung von Fehlern und Neuentwicklung von Features in einer existierenden Java 17 Jakarta EE Anwendung mit JPA/Hibernate/Microsoft SQL Server Backend und JSF/PrimeFaces Frontend. Die Anwendung ist als modularer Monolith in mehrere Macroservices aufgeteilt. Das Deployment erfolgt auf einem JBoss Server mittels Docker. Datenbankupdates werden mittels Liquibase umgesetzt. Die Anwendung dient zur Abbildung des Kerngeschäftes einer Versicherung. • **Neuanwendung:** Event Driven Architecture mit ActiveMQ Artemis. Neuentwicklung einer Java 25 Microservices Anwendung zur Abarbeitung von in Java implementierten Workflows (Zustandsautomaten) – eine Art Business Process Engine. Die Anwendung basiert auf Spring Boot 4, JPA, REST (OpenAPI), ActiveMQ Artemis, PostgreSQL und Docker. Die Java Workflows werden von einer anderen Anwendung nachrichtenbasiert gesteuert.

Projekte/Tätigkeiten

Aufgaben/Verantwortlichkeiten	• **Altanwendung:** Behebung von Fehlern und Neuentwicklung von Features in Front- und Backend (Full Stack). Erstellung von Liquibase Skripten zur Datenbankmigration. Entwicklung von automatisierten Tests in JUnit zur Überprüfung der Änderungen am Code. • **Neuanwendung:** Entwicklung eines Service Provider Interface (SPI) für Workflows in Java. Die SPI erlaubt die Neuentwicklung von Zustandsautomaten, die nachrichtenbasierte Zustandsübergänge ermöglichen und damit den Workflow implementieren. Ein Worker-Service arbeitet einen Workflow ab, indem er Nachrichten empfängt und Zustandsübergänge durchführt. • *Arbeitsprozess:* Aufgabenstellung und Dokumentation der Tätigkeiten erfolgen in Jira im Kanban-Prozess. Die Arbeitsergebnisse werden als Pull Request zum Review in GitLab zur Verfügung gestellt.
Methoden/Technologien	Java 17 + 25, IntelliJ IDEA, JPA, Hibernate, Microsoft SQL Server, Maven, Git, GitLab, Docker, Jira, JSF, Spring Boot 4, OpenAPI, REST, PostgreSQL, Apache ActiveMQ Artemis

Öffentlicher Sektor, 03/24 – 06/25

Projektziel	Performance-Optimierung und Refactoring einer existierenden Eclipse/ Equinox OSGi Anwendung mit JPA/Hibernate/Oracle Backend und Angular Frontend. Die Anwendung (ANBU) dient zur Buchhaltung von Anlagen und wird im Öffentlichen Dienst in mehreren Bundesländern für die Kommunikation zwischen verschiedenen Behörden eingesetzt und basiert auf der Plattform VOIS.
Aufgaben/Verantwortlichkeiten	• Performance-Optimierung einer Datenbankabfrage für einen Angular-Dialog. Erstellung eines Performance/Lasttests zur Überprüfung der Performance. Die Optimierung erfolgt durch das Erstellen einer Projektion mittels Criteria-API. Der Zeitverbrauch im Produktionsbetrieb sinkt durch die Optimierung von mehr als 3 Stunden auf unter 30 Sekunden. • Verbesserung der Paketstruktur, Umzug von Entitäten in andere packages. • Refactoring: Ersetzung von alten Entitäten durch neue. Schaffung und Verwendung von Convertern in Mapstruct zur schrittweisen Migration auf das neue Datenmodell. • Fehlerbehebung im Angular Frontend und Java Backend (Full Stack).
Methoden/Technologien	Java 17, Eclipse, VOIS, Equinox/OSGi, JPA, Hibernate, Criteria-API, Tycho, Maven, Git, GitLab, Jira, Docker, Oracle, Angular

Energiehandel, 01/24 – 03/24

Projektziel	Neuentwicklung einer Anwendung (Backend/Middleware, Event Driven Architecture mit Kafka) zur Verarbeitung von Orderbüchern, Kursdaten und weiteren handelsrelevanten Daten aus dem Energiehandel. Die Anwendung besteht aus mehreren Microservices. Die Daten werden über eine WebSocket-Schnittstelle empfangen (Microsoft SignalR/Volue), aggregiert und an einen Kafka-Service (Microsoft Azure Eventhub) weitergeleitet. Für den Fall des Ausfalls der WebSocket-Schnittstelle steht ein weiterer Service (REST-Proxy) bereit, der eine REST-API (Volue) zum Datenempfang verwendet und dann die Daten an ein Kafka Topic weiterleitet. Der Ausfall wird automatisch erkannt und durch den REST-Proxy ersetzt.

Projekte/Tätigkeiten	
	Für das Monitoring wird Prometheus und Micrometer eingesetzt.
Aufgaben/Verantwortlichkeiten	Komplette Neuentwicklung der Anwendung und Microservices – WebSocket-Proxy, WebSocket-Orderbook und REST-Proxy. Die Entwicklung beinhaltet JUnit- und Integrationstests. Alleinverantwortliche Entwicklung.
Methoden/Technologien	Java 21, IntelliJ IDEA, Spring Boot 3.2.2, Maven, Git, Jira, Bitbucket, Volue, WebSockets, REST, Swagger, Kafka, Microsoft Azure (SignalR, Eventhub), Docker, Prometheus, Micrometer
Logistik, 08/18 – 11/23	
Projektziel	Weiterentwicklung und Pflege einer Eclipse RCP Anwendung im Logistikumfeld. Die Anwendung dient als Plattform für per OSGi angebundene Kundenmodule (die wiederum eigene Anwendungen darstellen). Sie umfasst ca. 160.000 Codezeilen und läuft auf ca. 40.000 Client-Rechnern. Die Anwendung wird über eine weitere, eigens implementierte Service-Anwendung auf den Client-Rechnern ausgeliefert. Die Anwendung erfüllt die Anforderungen an Kritische Infrastruktur (KRITIS) gemäß der Spezifikation des Bundesamtes für Sicherheit in der Informationstechnik (BSI).
Aufgaben/Verantwortlichkeiten	• Erfassung von Änderungswünschen und Fehlerberichten für die Anwendungen in Atlassian Jira. • Dokumentation der erforderlichen Prozesse im Wiki. Dokumentation und Implementierung gemäß der KRITIS Spezifikation. • Implementation der Anforderungen in Java mit IntelliJ IDEA. • Überwachung, Wartung und Weiterentwicklung der Continuous Integration auf Jenkins mit Continuous Integration Pipelines (DevOps). • Unregelmäßige Erstellung und Veröffentlichung von Releases der RCP Anwendung. • Integration der Bibliothek JxBrowser zur Anbindung von Kundenmodulen, die als Web-Anwendung in Angular implementiert sind. • Migration der bestehenden SSO/Kerberos Authentifizierung auf eine MFA-fähige OAuth 2.1/OpenID Implementierung für den Desktop.
Methoden/Technologien	Java SE 1.8, IntelliJ IDEA, Spring 4, Swing, Angular, SWT, Eclipse RCP, OSGi, Maven, Tycho, Nexus, SonarQube, Jira, ClearQuest, Jenkins, Maven, Git/Bitbucket, OAuth 2.1, OpenID
Logistik, 08/16 – 11/23	
Projektziel	Weiterentwicklung einer umfangreichen Java/Spring Swing-Anwendung im Logistikumfeld. Die Anwendung umfasst ca. 3,5 Millionen Codezeilen und läuft auf ca. 40.000 Client-Rechnern. Das Backend läuft auf dynamisch skalierenden Server-Instanzen.
Aufgaben/Verantwortlichkeiten	• Die Anwendung ist in Unterprojekte gegliedert, für jedes Unterprojekt ist ein Team von ca. 10 Entwicklern zuständig. Insgesamt arbeiten ca. 100 Entwickler in dem Projekt. • Aufgaben: Full Stack – Entwicklung neuer Funktionen, sowohl im Frontend (UI: Swing, Angular) wie auch im Backend (Java Spring Services, JPA, Hibernate, Oracle, Tomcat 7), inklusive der erforderlichen Unit-Tests. Beheben von Fehlern.
Methoden/Technologien	Java SE 1.8, IntelliJ IDEA, Spring 4, Swing, Oracle, Hibernate, Maven, Nexus, Tomcat 7, SonarQube, Jira, ClearQuest, Jenkins, Maven,

Projekte/Tätigkeiten

	Git/Bitbucket

Börsenhandel (Eigenentwicklung, Nebentätigkeit), 09/15 – laufend

Projektziel	Entwicklung eines Backtesting-Rahmenwerks für Handelssysteme auf Finanzinstrumenten (Aktien, Futures, Forex). Das Rahmenwerk simuliert den Handel auf historischen Kurszeitreihen: Es verwaltet Orders, Signale, Positionen, Konten und Portfolios, schreibt Barbestände fort und wertet die Ergebnisse statistisch aus. Kursdaten werden als Ticks und CSV-Dateien eingelesen und zu Bars verdichtet. Eine Anbindung an die TWS-API von Interactive Brokers stellt Marktdaten und Kontozugriff bereit.
Aufgaben/Verantwortlichkeiten	• Alleinverantwortliche Entwicklung und Pflege seit 09/2015, rund 4.000 Commits. • Clean Code und funktionaler Programmierstil – nahezu alle Objektinstanzen sind unveränderlich. Durchgängige Einhaltung der SOLID-Prinzipien. • Fachliche Gliederung in einen Handelskern (Order, Signal, Position, Konto, Portfolio, Statistik) und einen Zeitreihenteil (Tick, Bar, CSV-Import, Zeitlogik). • Umfang: 908 Produktivklassen mit rund 72.500 Zeilen und 524 Testklassen mit rund 41.900 Zeilen; 1.820 Testfälle, Laufzeit aller Tests ca. 12 Sekunden. • Ablösung von Spring Boot durch Google Guice als Injektionsrahmenwerk. • Laufende Modernisierung: Migration auf Java 17 (2023) und Java 25 (2026) samt Aktualisierung aller Abhängigkeiten.
Methoden/Technologien	Java 25, IntelliJ IDEA, Google Guice, Guava, Apache Commons (Lang, Collections, IO, DBUtils, BeanUtils), Hibernate Validator, XStream, SLF4J/Logback, JUnit 5, Mockito, Hamcrest, Maven, Git, TeamCity, Interactive Brokers TWS API

Logistik, 02/15 – 06/16

Projektziel	Weiterentwicklung einer umfangreichen Java/Java EE Web-Anwendung im Logistikumfeld.
Aufgaben/Verantwortlichkeiten	• Die Anwendung ist in Unterprojekte gegliedert, für jedes Unterprojekt ist ein Team von ca. 5 – 10 Entwicklern, 2 – 5 Requirement Engineers und Business Stakeholdern zuständig. Insgesamt arbeiten ca. 100 Entwickler in dem Projekt. • Aufgaben: Full Stack – Entwicklung neuer Funktionen, sowohl im Frontend (UI: JSF, HTML, CSS) wie auch im Backend (Java EE Services, EJBs, JPA, EclipseLink, Oracle, GlassFish 3), inklusive der erforderlichen Unit-Tests. Beheben von Fehlern. Dokumentation der Implementierung.
Methoden/Technologien	Java SE 1.7, Java EE 6, JSF, HTML, CSS, Oracle, EclipseLink, GlassFish 3, Google Chrome, SonarQube, ClearQuest, Jenkins, Gradle, Git/Gerrit, IntelliJ IDEA, Eclipse

Mobile (Eigenentwicklung), 09/14 – 12/14

Projektziel	Evaluierung der Eignung aktueller Java-basierter Web-Rahmenwerke zur Realisierung einer mobilen Single Page App (SPA).

Projekte/Tätigkeiten

Aufgaben/Verantwortlichkeiten	Schwerpunkt der Untersuchung ist Vaadin als Rahmenwerk für eine vollständig in Java implementierte Single Page App.
Methoden/Technologien	Java SE 1.7, 1.8, GWT, Vaadin, Spring MVC, Spring Security, Spring Social, OAuth2, Gradle, Git, IntelliJ IDEA

Energieversorger, 11/12 – 06/14

Projektziel	Weiterentwicklung einer Client/Server Enterprise-Anwendung mit Swing GUI.
Aufgaben/Verantwortlichkeiten	• Die bestehende Enterprise-Anwendung zur Planung und Optimierung von Kraftwerkseinsätzen wird um ein neues Modul zur Planung von Stillständen (Revisionen, Störfälle etc.) erweitert. • Hierfür werden neue GUI-Ansichten (GUI-Schicht), Service-Methoden (Geschäftslogik) und Entitäten (Persistenzschicht) entwickelt. • Die Entwicklung der Anwendung erfolgt vor Ort in enger Abstimmung mit dem Anwender. • Die Unterstützung des Anwenders im laufenden Betrieb gehört ebenfalls zum Aufgabenfeld.
Methoden/Technologien	Java SE 1.6, Swing, Jide, Oracle Weblogic 12 Application-Server, Oracle, H2 SQL, Spring, iBatis, MyBatis Persistenz-Framework, RMI, JMS, Subversion V, Jenkins Build-Server, BoFiT, Gurobi, IntelliJ IDEA, Eclipse

Interbankenhandel (Eigenentwicklung), 04/09 – 11/12

Projektziel	Entwicklung und Ausführung eines Handelssystems zur Abwicklung von Devisengeschäften.
Aufgaben/Verantwortlichkeiten	• Ein Prototyp des Handelssystems wird auf der Plattform MultiCharts in der Programmiersprache PowerLanguage entworfen. Auf dieser Plattform findet auch die Überprüfung mittels historischer Kursdaten statt. • Die Validität des Handelsmodells und die fachliche Funktionalität wird im Prototypen sichergestellt. Zur Ausführung von Handelsgeschäften ist MultiCharts jedoch nicht ausreichend. Daher wird das Handelssystem auf der Zielplattform OpenQuant in der Programmiersprache C# neu entwickelt. Die Ausführung der Handelsgeschäfte wird in Echtzeit überwacht. • Zum Abruf der historischen Kursdaten wird ein Java-Programm entwickelt. Die Entwicklung erfolgt mit der Entwicklungsumgebung IntelliJ IDEA. Die Kursdaten werden als persistente Beans in der SQL-Datenbank H2 gespeichert. • Zur Vereinfachung des Umganges mit der Datenbank kommt das Apache Commons Framework DBUtils zum Einsatz. Die Daten liegen in sekundengenauer Auflösung vor. Aufgrund der hieraus resultierenden großen Mengen an Daten sind zahlreiche Maßnahmen zur Sicherstellung einer hohen Performance nötig. Das Datenmodell ist besonders platzsparend und effizient ausgelegt. Die Daten werden nicht nur über verschiedene Tabellen verteilt, sondern auch in nach Zeiträumen unterteilte Datenbanken abgelegt.
Methoden/Technologien	MultiCharts (PowerLanguage), OpenQuant (C#), IntelliJ IDEA, H2 SQL, DBUtils (Apache Commons Framework)

Projekte/Tätigkeiten

Finanzdienstleistungen, 10/08 – 03/09

Projektziel	Automatisierter Herstellertest für eine Web-Anwendung.
Aufgaben/Verantwortlichkeiten	• Neuentwicklung von automatisierten Herstellertests mit Java SE 5 für eine J2EE-Web-Anwendung (Plattform: IBM WebSphere, Rapid Application Developer (RAD)). Die Anwendung wird von Banken genutzt und dient der Bonitätsbewertung von Finanzierungen (Rating). • Zur Prüfung der Funktionalität wird das Rahmenwerk HttpUnit verwendet. Es werden Dialogabläufe sowie Feldinhalte und -zustände überprüft. Die Testdaten werden für den Herstellertest in XML angelegt. • Außerdem findet noch eine weitere Form des Tests statt, in dem die Ergebnisse von Berechnungen der Anwendung mit vom Kunden gelieferten Testdaten abgeglichen werden. Hierbei werden die Testdaten über eine Microsoft Access-Datenbank eingelesen und automatisch gegen die Rechenergebnisse der GUI geprüft.
Methoden/Technologien	Java SE 5, HttpUnit, JUnit, DBUnit, JavaDoc, Mercury Quality Center, Log4J, Apache Commons, Microsoft Access, Eclipse, Oxygen, CVS, Ant, Wiki

Logistik, 02/08 – 08/08

Projektziel	Architektur & Integration für eine Logistikanwendung.
Aufgaben/Verantwortlichkeiten	• Entwicklung einer Komponente zur Messung der Performance einer performancekritischen Logistikanwendung für Containerhäfen. Die Anwendung ist stark heterogen und verteilt. Die Kommunikation erfolgt über JMS (Java Message Service). • Zur Überwachung der Performance werden die JMS-Nachrichten zusammen mit ihren Sendezeiten mittels JPA (Java Persistence API) und Hibernate in einer Oracle-Datenbank gespeichert. Die primär eingesetzten Programmiersprachen sind Java 6 und C#. Das Projekt wird mit Maven 2 gebaut, geringfügig werden Ant-Skripte eingesetzt. • Das Projekt ist in Komponenten (Bundles, Services) im Sinne einer OSGi-Anwendung aufgeteilt. Die Komponenten werden mit Spring konfiguriert. • Die Builds werden mit CruiseControl gesteuert. Programmteile werden aus mit MagicDraw 15 erstellten UML-Modellen in Verbindung mit OpenArchitectureWare 4 generiert. Die OSGi-Komponenten werden mit Spring konfiguriert.
Methoden/Technologien	Maven 2, Java SE 6, MagicDraw 15, OpenArchitectureWare 4, Spring, Oracle, Hibernate, JPA Java Persistence API, JMS Java Message Service, Subversion, JUnit, Log4J, Apache Commons, IntelliJ IDEA 7, Eclipse 3.4, CruiseControl, Ant, Jira, Test Track Pro

Fortbildungen

Faster, Better, Cleaner Java Development with Agentic LLMs (Adam Bien, 26.02.2026)

Themen	• Agentische LLM-Werkzeuge im Java-Alltag: Claude Code, GitHub Copilot, Kiro, Goose • Wiederverwendbare Vorgaben: Instructions, AGENTS.md, spezialisierte Sub-Agenten • Test- und Dokumentationsstrategien mit LLMs (Unit-, Integrations-, Systemtests, JavaDoc)

Fortbildungen	
	• Architekturregeln, Guardrails und Hooks; BCE/ECB als wartbare Struktur • Modernisierung und Migration von Legacy-Anwendungen
MCP, A2A and AI Agents (Adam Bien, 18.12.2025)	
Themen	• Model Context Protocol (MCP) in Enterprise-Anwendungen • Agent-to-Agent-Kommunikation (A2A) • Agentische Workflows und Geschäftsprozessautomatisierung • Modernisierung bestehender Anwendungen über LLM-Anbindung
Hardcore Serverless Java on AWS (Adam Bien, 17.07.2025)	
Themen	• Serverless-Architekturen mit purem Java auf AWS Lambda • Infrastructure as Code mit CDK v2 (Java), wiederverwendbar als JAR-Abhängigkeit • Event Driven Architecture mit EventBridge; lang laufende Transaktionen mit Step Functions • S3 als Datalake und Datenbank; serverless Konfiguration • Trunk-based CI/CD mit CodeBuild und CodePipeline