CATALIN

Top 7 Entwurfs Entscheidungen

Verwaltungssoftware Flughafen

Liste der Entscheidungen

Entscheidung für eine Entwicklungsumgebung

Fragestellung	Welche Entwicklungsumgebung wird verwendet?
Beschreibung	
Datum:	10.04.2019
Entscheider, Entscheidungskreis:	Gesamtes Catalin-Team
Alternativen	Eclipse, IntelliJ, Netbeans, vim
Entscheidungs-kriterien	Usability, Integrationsmöglichkeiten in Build Systeme, Integration von VCS, Performance
Bewertung der Alternativen	Die Alternativen haben bekannte Probleme mit Performance und Usability oder sind allgemein langsamer. VIM fehlt es an Usability, vor allem im Bereich der Integrationen von Buildsystemen.
Entscheidung	Die Umsetzung erfolgt komplett mit IntelliJ
Begründung	IntelliJ hat als kommerzielles Produkt den besten Support. Außerdem findet man hier die besten Integrationen und die beste Performance.
Aspekte zur Umsetzung	Plugins: Gradle, Git, Guice, YAML... Codestyle: Standard Java

Entscheidung für Persistenz-Tool

Fragestellung	Wie wird Persistenz sichergestellt?
Beschreibung	Daten sollen auch nach Beenden des Programms gespeichert sein und bei Neustart fehlerfrei rekonstruiert werden können.
Datum:	10.04.2019
Entscheider, Entscheidungskreis:	Gesamtes Team
Alternativen	Hibernate, EclipseLink
Entscheidungs-kriterien	JPA muss implementiert sein, wenn möglich Integration in IDE, große Verbreitung
Bewertung der Alternativen	Alternativen zu unbekannt und aus Eclipse Umfeld
Entscheidung	Hibernate
Begründung	Branchenstandard, Beste Bekanntheit im Team
Aspekte zur Umsetzung	Standard JPA Implementierung, Guice Persistence Plugin

Entscheidung für eine Versionsverwaltung

Fragestellung	Welches Tool zur Versionierung wird verwendet?
Beschreibung	
Datum:	10.04.2019
Entscheider, Entscheidungskreis:	Gesamtes Catalin-Team
Alternativen	Git, SVN
Entscheidungs-kriterien	Einfaches Branching, integrierbar in Entwicklungsumgebung
Bewertung der Alternativen	SVN ist mittlerweile veraltet und wurde weitestgehend von Git abgelöst, Git wird bereits vom ganzen Team verwendet
Entscheidung	Git
Begründung	Beste Integration in IDE, Öffentlich auf GitHub hostbar, Sehr gute Plattform Integrationen auf GitHub
Aspekte zur Umsetzung	Repository wird auf GitHub gehosted, Master & Dev Branch, Vorerst keine Feature Branches

Entscheidung für ein Build Tool

Fragestellung	Welches Tool soll zum Bauen des Projekts benutzt werden
Beschreibung	
Datum:	10.04.2019
Entscheider, Entscheidungskreis:	Gesamtes Catalin-Team
Alternativen	Gradle, Maven, Ant, Bazel
Entscheidungs-kriterien	Schnelles Building, Gutes Caching, Breite Unterstützung von CI Plattformen, Sehr gut einstellbar
Bewertung der Alternativen	Alternativen wie Maven oder Ant sind entweder veraltet oder langsamer. Bazel ist noch zu unbekannt im Team, wenngleich das schnellste System, die Community aber recht klein und wenig allgemeine Akzeptanz, schlechte Unterstützung von vielen CI Plattformen.
Entscheidung	Gradle
Begründung	Gute IDE Integration, dank Daemon sehr schnell beim bauen, sehr gut einstellbar und individualisierbar
Aspekte zur Umsetzung	Gradle mit java und maven plugins, jar task erweitern zum Bauen von fat jar mit abhängigkeiten

Entscheidung für eine Dependency Injection Library

Fragestellung	Mit welchem Tool wird die Dependency Injection geregelt
Beschreibung	
Datum:	10.04.2019
Entscheider, Entscheidungskreis:	Gesamtes Catalin-Team
Alternativen	HK2, Guice, Governator, Spring
Entscheidungs-kriterien	Unterstützung für JSR-330, kompakte Module, Singleton Scope Unterstützung, gut testbar
Bewertung der Alternativen	Spring bring ein zu großes Ökosystem mit und ist schlecht integrierbar, Governator ist von Netflix, basiert auf Guice und wird nicht mehr weiterentwickelt, HK2 wird derzeit zu Eclipse transferiert und deshalb nur begrenzt maintained
Entscheidung	Guice
Begründung	Allgemein anerkannte und einfach zu verwendende Library, Weiterentwicklung durch Google, Gute Integration in IDE
Aspekte zur Umsetzung	JSR Annotations statt Guice Annotations verwenden, Module für verschiedene Persistenz Modi, JPA mit Guice JPA Extension

Entscheidung für ein Entwurfsmuster zur GUI Erzeugung

Fragestellung	Boostrapping der GUI Applikation
Beschreibung	Wie sollen die Hauptkomponenten der GUI erzeugt werden?
Datum:	10.06.2019
Entscheider, Entscheidungskreis:	Felix, Lena
Alternativen	Factory, Singleton
Entscheidungs-kriterien	Kapselung unserer Logik nach draußen, Dynamisch um bei Bedarf mehrere Implementierungen der GUI nutzen zu können
Bewertung der Alternativen	Singleton hinderlich bei Dependency Incjection, die sonst überall genutzt wird, Schlechter Stil und schlecht konfigurierbar
Entscheidung	Factory
Begründung	Passt sehr gut zur momentanen Infrastruktur und kapselt die Dependency Injection Logik nach außen ab
Aspekte zur Umsetzung	Wurde mit Guice umgesetzt CatalinGuiceGUIFactory.java

Entscheidung für ein Entwurfsmuster zur Erzeugung der CoreConfig

Fragestellung	Welche Entwurfsmuster soll zur Erzeugung der CoreConfig genutzt werden?
Beschreibung	
Datum:	10.06.2019
Entscheider, Entscheidungskreis:	Gesamtes Catalin-Team
Alternativen	Singleton, Builder, Factory
Entscheidungs-kriterien	Die Config selbst sollte immutable sein, wird aber ggf. aufwendig von mehreren Stellen erzeugt
Bewertung der Alternativen	Singleton schlecht, weil zu statisch und Immutable eher hinderlich Builder bietet sequenziellen Aufbau der Config
Entscheidung	Builder
Begründung	Die Config wird meist softwareseitig zusammengesetzt und durch verschiedene Programmteile gereicht. Sie soll zur Laufzeit immutable sein. Dies ist vor allem bei einem Builder gegeben, da man diesen (Im Gegensatz auch zur Factory) sehr gut weiterreichen kann.
Aspekte zur Umsetzung	CatalinCoreConfigBuilder